

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: October 10, 2016

G. Brown
CentralNic Group plc
J. Frakes
April 8, 2016

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	8
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:

- o "create" indicating a <create> command;
- o "renew" indicating a <renew> command;
- o "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-epext-launchphase]. It may also contain an OPTIONAL "subphase" attribute identifying the custom or sub-phase as described

in [draft-ietf-eppext-launchphase].

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

applicable to the transaction is the arithmetic sum of the values of each of the <fee:fee> and/or <fee:credit> elements. This amount applies to the total additional validity period applied to the object (where applicable) rather than to any incremental unit.

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-epext-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 (ie <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:

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

- o 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.
- o 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.
- o 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.
- o 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.
- o Zero or more `<fee:fee>` elements.
- o Zero or more `<fee:credit>` elements.
- o An OPTIONAL `<fee:class>` element.
- o 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:cd avail="0">
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:reason>
S:              minimum period is 2 years.
S:          </fee:reason>
S:      </fee:cd>
S:      </fee:chkData>
S:  </extension>
S:  <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:  </trID>
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:

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

```
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:period unit="y">1</fee:period>
S:         <fee:fee>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. 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:

- o An OPTIONAL <fee:currency> element;
- o 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:

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

```
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.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>jdl234</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:       <fee:create 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:create>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```

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="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>example.com</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:         <fee:creData 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:balance>-5.00</fee:balance>
S:           <fee:creditLimit>1000.00</fee:creditLimit>
S:         </fee:creData>
S:       </extension>
S:       <trID>
S:         <clTRID>ABC-12345</clTRID>
S:         <svTRID>54321-XYZ</svTRID>
S:       </trID>
S:     </response>
S: </epp>
```

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:

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

```
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:delData
S:         xmlns:fee="urn:ietf:params:xml:ns:fee-0.11">
S:         <fee:currency>USD</fee:currency>
S:         <fee:credit description="AGP Credit">-5.00</fee:credit>
S:         <fee:balance>1005.00</fee:balance>
S:       </fee:delData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
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:

- o An OPTIONAL `<fee:currency>` element;
- o 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:

- o A `<fee:currency>` element;

- o Zero or more <fee:fee> elements;
- o Zero or more <fee:credit> elements;
- o An OPTIONAL <fee:balance> element;
- o 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:

```
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:renData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:           <domain:name>example.com</domain:name>
S:           <domain:exDate>2005-04-03T22:00:00.0Z</domain:exDate>
S:         </domain:renData>
S:       </resData>
S:       <extension>
S:         <fee:renData 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:balance>1000.00</fee:balance>
S:         </fee:renData>
S:       </extension>
S:       <trID>
S:         <clTRID>ABC-12345</clTRID>
S:         <svTRID>54322-XYZ</svTRID>
S:       </trID>
S:     </response>
S: </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:

- o An OPTIONAL <fee:currency> element;
- o 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:

- o A <fee:currency> element;
- o Zero or more <fee:fee> elements;
- o Zero or more <fee:credit> elements;
- o An OPTIONAL <fee:balance> element;
- o 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:     <extension>
C:       <fee:transfer 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:transfer>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
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> has been selected during the <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:

- o An OPTIONAL <fee:currency> element;
- o 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:

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

```
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>example.com</domain:name>
C:           <domain:chg>
C:             <domain:registrant>sh8013</domain:registrant>
C:           </domain:chg>
C:         </domain:update>
C:       </update>
C:     <extension>
C:       <fee:update 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:update>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </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:

- o Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- o 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.
- o 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
      vl.0 extension schema for fee
      information.</documentation>
  </annotation>

  <!--
  Child elements found in EPP commands and responses
  -->
  <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" />

  <!--
  client <check> command
  -->
  <complexType name="checkType">
    <sequence>
```

```

    <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>

```

```
<element name="currency" type="fee:currencyType"
  minOccurs="0" />
<element name="fee" type="fee:feeType"
  minOccurs="0" 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" />

    <!-- only used op="query" responses -->
    <element name="period" type="domain:periodType"
      minOccurs="0" />

    <element name="fee" type="fee:feeType"
      minOccurs="0" 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>
```

```
<complexType name="feeType">
  <simpleContent>
    <extension base="fee:nonNegativeDecimal">
      <attribute name="description"/>
      <attribute name="refundable" type="boolean" />
      <attribute name="grace-period" type="duration" />
      <attribute name="applied" default="immediate">
        <simpleType>
          <restriction base="token">
            <enumeration value="immediate" />
            <enumeration value="delayed" />
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </simpleContent>
</complexType>

<complexType name="creditType">
  <simpleContent>
    <extension base="fee:negativeDecimal">
      <attribute name="description"/>
    </extension>
  </simpleContent>
</complexType>

<simpleType name="balanceType">
  <restriction base="decimal" />
</simpleType>

<simpleType name="creditLimitType">
  <restriction base="decimal" />
</simpleType>

</schema>

END
```

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:

- o James Gould of Verisign
- o Luis Munoz of ISC

- o Michael Young of Architelos
- o Ben Levac and Jeff Eckhaus of Demand Media
- o Seth Goldman of Google
- o Klaus Malorny and Michael Bauland of Knipp
- o Jody Kolker and Roger Carney of Go Daddy
- o Michael Holloway of Com Laude
- o Santosh Kalsangrah of Impetus Infotech
- o Alex Mayrhofer of Nic.at

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

- [ISO4217] International Organization for Standardization, "ISO 4217: 2008, Codes for the representation of currencies and funds", 2008.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC3915] Hollenbeck, S., "Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP)", RFC 3915, DOI 10.17487/RFC3915, September 2004, <<http://www.rfc-editor.org/info/rfc3915>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<http://www.rfc-editor.org/info/rfc5730>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <<http://www.rfc-editor.org/info/rfc5731>>.
- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013,

<<http://www.rfc-editor.org/info/rfc6982>>.

[RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <<http://www.rfc-editor.org/info/rfc7451>>.

[draft-ietf-eppext-launchphase]

Gould, J., Tan, W., and G. Brown, "Launch Phase Mapping for the Extensible Provisioning Protocol (EPP)", 2014.

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>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 24, 2016

J. Gould
VeriSign, Inc.
S. Wodjenski
Neustar
May 23, 2016

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

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	2
1.1. Conventions Used in This Document	3
2. Object Attributes	3
2.1. Allocation Token	3
3. EPP Command Mapping	4
3.1. EPP Query Commands	4
3.1.1. EPP <check> Command	4
3.1.2. EPP <info> Command	8
3.1.3. EPP <transfer> Command	10
3.2. EPP Transform Commands	11
3.2.1. EPP <create> Command	11
3.2.2. EPP <delete> Command	12
3.2.3. EPP <renew> Command	12
3.2.4. EPP <transfer> Command	12
3.2.5. EPP <update> Command	13
4. Formal Syntax	14
4.1. Allocation Token Extension Schema	15
5. IANA Considerations	15
5.1. XML Namespace	15
5.2. EPP Extension Registry	16
6. Security Considerations	16
7. Acknowledgements	16
8. Normative References	16
Appendix A. Change History	17
A.1. Change from 00 to 01	17
A.2. Change from 01 to 02	17
A.3. Change from 02 to 03	17
A.4. Change from 03 to 04	17
Authors' Addresses	17

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], 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":

```
<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
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: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:      <domain: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:      </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-DEF-12345</clTRID>
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:

```
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
C:        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:    <clTRID>ABC-12345</clTRID>
C:  </command>
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=
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>
```

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:

```
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>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:    </transfer>
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>
```

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:

```
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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC3915] Hollenbeck, S., "Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP)", RFC 3915, September 2004.

- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, August 2009.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, August 2009.
- [RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, February 2015.

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>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 2, 2017

J. Gould
VeriSign, Inc.
S. Wodjenski
Neustar
September 29, 2016

Change Poll Extension for the Extensible Provisioning Protocol (EPP)
draft-gould-change-poll-05

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 knowledgeable 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:

```

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 lang="en-US">
S:        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 update of domain.</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>jdl234</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: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:      state="before">
S:      <changePoll:operation>update</changePoll:operation>
S:      <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
S:      <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:      <changePoll:who>URS Admin</changePoll:who>
S:      <changePoll:caseId type="urs">urs123</changePoll:caseId>
S:      <changePoll:reason>URS Lock</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 a URS lock transaction on the domain.example domain name, with the "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 lang="en-US">
S:        Command completed successfully; ack to dequeue</msg>
S:      </result>
S:    <msgQ id="202" count="1">
S:      <qDate>2013-10-22T14:25:57.0Z</qDate>
S:      <msg>Registry initiated update of domain.</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="serverUpdateProhibited"/>
S:      <domain:status s="serverDeleteProhibited"/>
S:      <domain:status s="serverTransferProhibited"/>
S:      <domain:registrant>jdl234</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:      state="after">
S:      <changePoll:operation>update</changePoll:operation>
S:      <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
S:      <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:      <changePoll:who>URS Admin</changePoll:who>
S:      <changePoll:caseId type="urs">urs123</changePoll:caseId>
S:      <changePoll:reason>URS Lock</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 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
S:      </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
S:    </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 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
S:        domain resulting in immediate purge.</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: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 op="purge">
S:          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:    <msg>Registry purged domain with pendingDelete status.</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: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>
S:        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>
S:        Past pendingDelete 5 day period
S:      </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:

```
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 update of host.</msg>
S:    </msgQ>
S:    <resData>
S:      <host:infData
S:        xmlns:host="urn:ietf:params:xml:ns:host-1.0">
S:        <host:name>ns1.domain.example</host:name>
S:        <host:roid>NS1_EXAMPLE1-REP</host:roid>
S:        <host:status s="linked"/>
S:        <host:status s="serverUpdateProhibited"/>
S:        <host:status s="serverDeleteProhibited"/>
S:        <host:addr ip="v4">192.0.2.2</host:addr>
S:        <host:addr ip="v6">1080:0:0:0:8:800:200C:417A</host:addr>
S:        <host:clID>ClientX</host:clID>
S:        <host:crID>ClientY</host:crID>
S:        <host:crDate>2012-04-03T22:00:00.0Z</host:crDate>
S:        <host:upID>ClientY</host:upID>
S:        <host:upDate>2013-10-22T14:25:57.0Z</host:upDate>
S:      </host:infData>
S:    </resData>
S:    <extension>
S:      <changePoll:changeData
S:        xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0">
S:        <changePoll:operation>update</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>Host Lock</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>
```

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
        Change Poll Mapping Schema.
      </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".
-->
<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.
-->
<simpleType name="stateType">
  <restriction base="token">
    <enumeration value="before"/>
    <enumeration value="after"/>
  </restriction>
</simpleType>

<!--
  Transform operation type
-->
<complexType name="operationType">
  <simpleContent>
    <extension base="changePoll:operationEnum">
      <attribute name="op" type="token"/>
    </extension>
  </simpleContent>
</complexType>

<!--
  Case identifier type
-->
<complexType name="caseIdType">
  <simpleContent>
    <extension base="token">
      <attribute name="type" type="changePoll:caseTypeEnum"
        use="required"/>
      <attribute name="name" type="token">
```



```
        use="optional"/>
      </extension>
    </simpleContent>
  </complexType>

  <!--
    Enumerated list of case identifier types
  -->
  <simpleType name="caseTypeEnum">
    <restriction base="token">
      <enumeration value="udrp"/>
      <enumeration value="urs"/>
      <enumeration value="custom"/>
    </restriction>
  </simpleType>

  <!--
    Who type
  -->
  <simpleType name="whoType">
    <restriction base="normalizedString">
      <minLength value="1"/>
      <maxLength value="255"/>
    </restriction>
  </simpleType>

  <!--
    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: 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 Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

- [RFC3915] Hollenbeck, S., "Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP)", RFC 3915, September 2004.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, August 2009.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, August 2009.
- [RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, February 2015.

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

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>

Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 24, 2016

J. Gould
VeriSign, Inc.
May 23, 2016

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

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Conventions Used in This Document	2
2. EPP to RDAP Status Mapping	3
3. IANA Considerations	5
3.1. JSON Values Registry	5
4. Security Considerations	9
5. Normative References	9
Appendix A. Change History	10
A.1. Change from 00 to 01	10
A.2. Change from 01 to 02	10
A.3. Change from 02 to 03	10
Author's Address	10

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
redemptionPeriod =
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.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

Value: client transfer prohibited

Type: status

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

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

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.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

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.

Registrant Name: VeriSign Inc.

Registrant Contact Information: epp-registry@verisign.com

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 Name: VeriSign Inc.

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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3915] Hollenbeck, S., "Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP)", RFC 3915, September 2004.

- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, August 2009.
- [RFC5732] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Host Mapping", STD 69, RFC 5732, August 2009.
- [RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, August 2009.
- [RFC7483] Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", RFC 7483, March 2015.
- [rdap-json-values]
"RDAP JSON Values Registry",
<<https://www.iana.org/assignments/rdap-json-values/rdap-json-values.xhtml>>.

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>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 2, 2017

J. Gould
VeriSign, Inc.
September 29, 2016

Verification Code Extension for the Extensible Provisioning Protocol
(EPP)
draft-gould-eppext-verificationcode-04

Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension for including a verification code for marking the data for a transform command as being verified by a 3rd party, which is referred to as the Verification Service Provider (VSP). The verification code is digitally signed by the VSP using XML Signature and is "base64" encoded. The XML Signature includes the VSP signer certificate, so the server can verify that the verification code originated from the VSP.

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	2
1.1. Conventions Used in This Document	3
2. Object Attributes	4
2.1. Verification Code	4
2.1.1. Signed Code	4
2.1.2. Encoded Signed Code	6
2.2. Verification Profile	11
3. EPP Command Mapping	12
3.1. EPP Query Commands	12
3.1.1. EPP <check> Command	12
3.1.2. EPP <info> Command	12
3.1.3. EPP <transfer> Command	24
3.2. EPP Transform Commands	25
3.2.1. EPP <create> Command	25
3.2.2. EPP <delete> Command	27
3.2.3. EPP <renew> Command	28
3.2.4. EPP <transfer> Command	28
3.2.5. EPP <update> Command	28
4. Formal Syntax	28
4.1. Verification Code Extension Schema	28
5. IANA Considerations	32
5.1. XML Namespace	32
5.2. EPP Extension Registry	32
6. Security Considerations	33
7. Normative References	33
Appendix A. Acknowledgements	34
Appendix B. Change History	34
B.1. Change from 00 to 01	34
B.2. Change from 01 to 02	34
B.3. Change from 02 to 03	34
B.4. Change from 03 to 04	34
Author's Address	34

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], EPP host mapping [RFC5732], and EPP contact mapping [RFC5733], can be used to pass a verification code to one of the EPP transform commands. The domain name object is used for examples in

the document. The verification code is signed using XML Signature [W3C.CR-xmlsig-core2-20120124] and is "base64" encoded. The "base64" encoded text of the verification code MUST conform to [RFC2045]. The verification code demonstrates that verification was done by a Verification Service Provider (VSP).

The Verification Service Provider (VSP) is a certified party to verify that data is in compliance with the policies of a locality. A locality MAY require the client to have data verified in accordance with local regulations or laws utilizing data sources not available to the server. The VSP has access to the local data sources and is authorized to verify the data. Examples include verifying that the domain name is not prohibited and verifying that the domain name registrant is a valid individual, organization, or business in the locality. The data verified, and the objects and operations that require the verification code to be passed to the server is up to the policies of the locality. The verification code represents a marker that the verification was completed. The data verified by the VSP MUST be stored by the VSP along with the generated verification code to address any compliance issues. The signer certificate and the digital signature of the verification code MUST be verified by the server.

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.

"verificationCode-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:verificationCode-1.0". The XML namespace prefix "verificationCode" 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], EPP host mapping [RFC5732], and EPP contact mapping [RFC5733]. Only those new elements are described here.

2.1. Verification Code

The Verification Code is a formatted token, referred to as the Verification Code Token, that is digitally signed by a Verification Service Provider (VSP) using XML Signature [W3C.CR-xmlsig-core2-20120124], using the process described in Section 2.1.1, and is then "base64" encoded, as defined in Section 2.1.2. The Verification Code Token syntax is specified using Augmented Backus-Naur Form (ABNF) grammar [RFC5234] as follows:

Verification Code Token ABNF

```
token      = vsp-id "-" verification-id ; Verification Code Token
vsp-id     = 1*DIGIT                    ; VSP Identifier
verification-id = 1*(DIGIT / ALPHA)    ; Verification Identifier
```

For a VSP given VSP Identifier "1" and with a Verification Identifier of "abc123", the resulting Verification Code Token is "1-abc123". The Verification Identifier MUST be unique within a VSP and the VSP Identifier MUST be unique across supporting VSP's, so the Verification Code Token MUST be unique to an individual verification. The VSP Identifiers MAY require registration within an IANA registry.

2.1.1. Signed Code

The <verificationCode:signedCode> is the fragment of XML that is digitally signed using XML Signature [W3C.CR-xmlsig-core2-20120124]. The <verificationCode:signedCode> element includes a required "id" attribute of type XSD ID for use with an IDREF URI from the Signature element. The certificate of the issuer MUST be included with the Signature so it can be chained with the issuer's certificate by the validating client.

The <verificationCode:signedCode> element includes a REQUIRED "type" attribute for use in defining the type of the signed code. It is up to the VSP and the server to define the valid values for the "type" attribute. Examples of possible "type" attribute values include "domain" for verification of the domain name, "registrant" for verification of the registrant contact, or "domain-registrant" for verification of both the domain name and the registrant. The typed signed code is used to indicate the verifications that are done by

the VSP. The "type" attribute values MAY require registration within an IANA registry.

A `<verificationCode:signedCode>` element substitutes for the `<verificationCode:abstractSignedCode>` abstract element to define a concrete definition of a signed code. The `<verificationCode:abstractSignedCode>` element can be replaced by other signed code definitions using the XML schema substitution groups feature.

The child elements of the `<verificationCode:signedCode>` element include:

`<verificationCode:code>` Contains the Verification Code Token as defined by the ABNF in Section 2.1.
`<Signature>` XML Signature [W3C.CR-xmlsig-core2-20120124] for the `<verificationCode:signedCode>`. Use of a namespace prefix, like "dsig", is recommended for the XML Signature [W3C.CR-xmlsig-core2-20120124] elements.

Example of a "domain" typed signed code using the `<verificationCode:signedCode>` element and XML Signature [W3C.CR-xmlsig-core2-20120124]:

```
<verificationCode:signedCode
  xmlns:verificationCode=
    "urn:ietf:params:xml:ns:verificationCode-1.0"
  id="signedCode">
  <verificationCode:code type="domain">1-abc111
</verificationCode:code>
  <Signature xmlns="http://www.w3.org/2000/09/xmlsig#">
    <SignedInfo>
      <CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      <SignatureMethod
Algorithm="http://www.w3.org/2001/04/xmlsig-more#rsa-sha256" />
      <Reference URI="#signedCode">
        <Transforms>
          <Transform
Algorithm="http://www.w3.org/2000/09/xmlsig#enveloped-signature" />
        </Transforms>
        <DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256" />
        <DigestValue>wgyW3nZPoEfpptlhRILKnOQnbdU6ArM7ShrAfHgDFg=
</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>
```

```

jMu4PfYQGiJBfO2GWESEPFfCJjmywCEqR2h4LD+ge6XQ+JnmKFFCuCZS/3SLKAxOL1w
QDF02e0Y69k2G7/LGE37X3vOflobFMlOGwja8+GMVraoto5xAd4/AF7eHukgAymD
o9toxo2h0yV4A4PmXzsU6S86XtCcUE+S/WM72nyn47zoUCzzPKHZBRyeWehVFQ+
jYRMIAmZm57HHQA+6eaXefRvtPETgUO4aVIVSugc4OUAZZwbYcZrC6wOaQqqqAZi
30aPOBYbAvHMSmWSS+hFkbshomJfHxb97TD2grlYnRqIzqXk7WbHWy2SYdA+sI/Z
ipJsXNa6osTUw1CzA7jfwA==
  </SignatureValue>
  <KeyInfo>
    <X509Data>
      <X509Certificate>
MIIESTCCAzGgAwIBAgIBAgjANBgkqhkiG9w0BAQsFAADBiMQswCQYDVQQGEwJVUzEL
MAkGALUECBMCQ0ExFDASBgNVBACTC0xvcyBBbmdlbGVzMRMwEQYDVQQKEwppJQ0FO
TiBUTUNIMRswGQYDVQQDEExJJQ0FOTiBUTUNIIFRFUlQgQ0EwHhcNMjMwMDAw
MDAwWhcNMjMwMDAwMDAwWjBsmQswCQYDVQQGEwJVUzELMAkGALUECBMCQ0Ex
FDASBgNVBACTC0xvcyBBbmdlbGVzMRcwFQYDVQQKEw5WYXpZGF0b3IgVE1DSDEh
MB8GALUEAAMyVmfFsaWRhdG9yIFRnQ0ggVEVTVCBDRVJUMIIBIjANBgkqhkiG9w0B
AQEFAAOCAQ8AMIIBCgKCAQEAO/cwvXhbVYl0RDWWvOyeZpETVZVVCMCovUVNg/sw
WinuMgEWgVQFrz0xA04pEhXCFVv4evbUpekj5buqU1gmQyOsCKQ1hOHTdPjvkC5u
pDqa51Flk0TMaMkIQjs7aUKCmA4RG4tTTGK/EjRlIx8/D0gHYVRldy1YPrMP+ou7
5bOVnIos+HifrAtrIv4qEqwLL4FTZAUPaCa2BmgXfy2CSRQbxD5OrlgcSa3vurh5
sPMCNxqaXmIXmQipS+DuEBqMM8tldan7RYojuUEKrGVsNk5i9y2/7sjnlzzyUPf7v
L4GgDYqhJYWV61DnXgx/Jd6CWxvsndf6scscQzUTel+hyWIDAQABO4H/MIH8MAwG
AlUdEwEB/wCMAAwHQQYDVR0OBBYEFPZecIQcD/Bj2IFz/LErUo2ADJviMIGMBGnV
HSMegYQwqYGAF00/7kEh3FueKS+q/kyHaD/W6wihoWakZDBiMQswCQYDVQQGEwJV
UzELMAkGALUECBMCQ0ExFDASBgNVBACTC0xvcyBBbmdlbGVzMRMwEQYDVQQKEwpp
JQ0FOTiBUTUNIMRswGQYDVQQDEExJJQ0FOTiBUTUNIIFRFUlQgQ0GCAQEwDgYDVR0P
AQH/BAQDAgeAMC4GALUdHwQnMCUwI6AhoB+GHWh0dHA6Ly9jcmwuaWNhbm4ub3Jn
L3RtY2guY3J5MA0GCSqGSIb3DQEBEwUAA4IBAQB2qSy7ui+43cebKUKwWPrzz9y/
IkrMeJGKjo40n+9uekaw3Dj5EqiOf/qZ4pJBD++oR6BJCb6NQuQKwnoAz51E4Ssu
y5+i93oT3HfyVc4gNMioHm1PS1917DBKrbwbzAea/0jKWVzrmv77TBfjxD3AQo1R
bu5dBr6IjbdLFlno5x0G0mrG7x50UPuurihyIURpFDPwH8KAHlwMcCpXGXFRtGKk
wydgyVYAty7otkl/z3bZkCVT34gPvF70sR6+QxUy8u0LzF5A/beYaZpxSYG31amL
AdXitTWFipaIGea9lEGFM0L9+Bg7XzNn4nVLXokyEB3bgS4scG6QznX23FGk
      </X509Certificate>
    </X509Data>
  </KeyInfo>
  </Signature>
</verificationCode:signedCode>

```

2.1.2. Encoded Signed Code

The <verificationCode:encodedSignedCode> element contains one or more encoded form of the digitally signed <verificationCode:signedCode> element, described in Section 2.1.1.

The child elements of the `<verificationCode:encodedSignedCode>` element include:

`<verificationCode:code>` One or more `<verificationCode:code>` elements that is an encoded form of the digitally signed `<verificationCode:signedCode>` element, described in Section 2.1.1, with the encoding defined by the "encoding" attribute with the default "encoding" value of "base64". The "base64" encoded text of the `<verificationCode:code>` element MUST conform to [RFC2045].

Example `<verificationCode:encodedSignedCode>` element that contains one "base64" encoded `<verificationCode:signedCode>` contained in the `<verificationCode:code>` element:

```
<verificationCode:encodedSignedCode>
  xmlns:verificationCode=
    "urn:ietf:params:xml:ns:verificationCode-1.0">
  <verificationCode:code>
ICAgICAgPHZlcmllmaWNhdGlvbkNvZGU6c2lnbmVkJ29kZQogICAgICAgIHhtbG5z
OnZlcmllmaWNhdGlvbkNvZGU9CiAgICAgICAgICAidXJuOmlldGY6cGFyYWlzOnht
bDpuczp2ZXJpZmljYXRpb25Db2RlLTEuMCiKICAgICAgICAgIGlkPSJzaWduZWRD
b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTpb2RlPjEtYWJjMTIzPC92ZXJp
ZmljYXRpb25Db2RlOmNvZGU+CiAgPFNpZ25hdHVyZSB4bWxuc20iaHR0cDovL3d3
dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CiAgIDxTaWduZWRJbmZvPgogICAg
PENhbm9uacWnhbG16YXRpb25NZXRob2QKIeFzS29yaXRobT0iaHR0cDovL3d3dy53
My5vcmcvMjAwMS8xMC94bWwtZXhjLWwXNG4jIi8+CiAgICA8U2lnbmF0dXJlTWVz
aG9kCiBBbGdvcmll0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveGlsZHNp
Zy1tb3JlI3JzYS1zaGEyNTYiLz4KICAgIDxSZWZlcmVuY2UgVVJJPSIjc2lnbmVkJ2
9kZSI+CiAgICAgPFryYW5zZm9ybXM+CiAgICAgIDxUcmFuc2ZvcmlkIEFzS29y
aXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3Bl
ZC1zaWduYXRlcmUiLz4KICAgICA8L1RyYW5zZm9ybXM+CiAgICAgPERpZ2VzdE1l
dGhvZAogQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVu
YyNzaGEyNTYiLz4KIDxEaWdlc3RlYXNlZT53Z3JlXm25aUG9FZnBwdGxoUk1MS25P
UW5iZHRVnKfYtTdTaHJBZkhNREZnPTwvRGlNzXN0VmFsdWU+CiAgICA8L1JlZmVy
ZW5jZT4KICAgPC9TaWduZWRJbmZvPgogICAgICA8U2lnbmF0dXJlVmFsdWU+CiBqTXU0
ZG55U0dpSkJGMEdeXU0VQRkNkMl5d0NFcVIAuDRMRcTnZTYZUStKbm1LRkZdDUNa
Uy8zUUxlQXgwTDF3CiBRREZkPmUWWTY5azJHNY9MR0UzNlZgZkd9mbG9iRk0xb0d3
amE4K0dNvNjhb3RvNXhBZDQvQUY3ZUhla2dBewlECiBvOXRveG9hMmgweVY0QTRQ
bVh6c1U2Uzg2WHRDY1VfK1MvV003Mm55bjQ3em9VQ3p6UEtIWkJSeWVXZWhWrlEr
CiBqVWJNSUFNek01N0hIUUErNmVhWGVMUnZ0UEVUZlVPNGFWSVZTdWdjNE9VQVpa
d2JZYlpyQzZ3T2FRcXFXvQVppCiAzMGFQT0JZYkF2SE1TbVdTUytoRmtic2hvbUpm
SHhiOTdURDJncmxZTnJRSXpxWGS3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z
VFV3MUN6QTdqZndBPT0KICAgPC9TaWduYXRlcmVWYXNlZT4KICAgPETleUluZm8+
CiAgICA8WDUwOURhdGE+CiAgICA8WDUwOUNlcnRpZmljYXRlPgogTULjRVNUQ0NB
ekdnQXJkFjNSUJBakFOQmdrcWhraUc5dzBCQVZzRkFEQmlNUXN3Q1FZRFZRUUdF
d0pWVXpFTaogTUFRrR0ExVUVDQk1DUTBFEeZEQVNCZ05WQkFjVEMweHZeUJCYmlk
bGJHVNpNuk13RVFZRFZRUUFTd3BkUTBGTogVGlCVVRVTklNUN3R1FZRFZRUURF
eEpKUTBGTlRpQlVUUV5JSUZSRlUxUWdRMEV3SGhjTklUTXDNake0TURBdwogTURB
dlDoY05NVGd3TwpBM01qTTFPFVU1V2pCc01Rc3dUdU1EVlFRR0V3S1ZVekVMTUFR
R0ExVUVDQk1DUTBFEeAogRkRBU0JnTlZCOWNUQzB4dmN5OkJibWRsYkdWek1SY3dG
```

```

UV1EVlFRS0V3NVdZV3hwWkdGMGIzSWdWRTFEU0RFaAogTUI4R0ExVUVBeE1ZVm1G
c2FXUmhkRz15SUZSTlEwZ2dWRVZUVkNCRFJWSlVNSUlCSWpBTkNa3Foa2lHOXcw
QgogQVFFRkFBT0NBUThtBTUlJQkNnS0NBuUVBby9jd3ZYaGJWWwwUkRXV3ZveWVa
cEVUVlpWVmNNQ292VVZOZy9zdWogV2ludUlnRVdnVlFGcnoweEEwNHBFaFhDR1Z2
NGV2YlVwZWtKNWJ1cVUxZ21ReU9zQ0tRbGhPSFRkUGp2a0MldQogcERxYTUxRmxr
MFRNYUlRlSVFqcZdhVUtDbUE0Ukc0dFRUR0svRWpSMWl4OC9EMGdIWVZSbGR5MVLQ
ck1QK291NwogNWJpVm5Jb3MrSGlmcKf0ck12NHFFcXdMTDRGVFpBVXBhQ2EyQm1n
WGZ5MkNTUlFieEQ1T3IxZ2NTYTN2dXJONQogc1BNQ054cWFYbUlYbVFpCFMRHVF
QnFNTTh0bGRhTjdSWW9qVUVLckdWc05rNwK5eTivN3NqbJf6eXlVUGY3dgogTDRH
Z0RzcWhKWVdWNjFEB1hneC9KZDZDV3h2c25ERjZyY3NjUXpVVEVsK2h5d0lEQVFB
Qm80SC9NSUG4TUF3RwogQTFVZEV3RUlvd1FDTUFBd0hRWURWUjBPQkJZRUZQWkvj
SVFjRC9CaJjJRnovTEVSdW8yQURKdmlNSUdNQmdOVgogSFNNRwdZUXdnWUdBRk8w
LzdrRWgzRnVFS1MrUS9rWUhhRC9XNndpaG9XYWtaREJpTVFzd0NRWURWUUVFHRXdk
VgogVXpFTE1Ba0dBmVVFQ0JNQ1EwRXhGREFTQmdOVk1RDMHh2Y3lCQmJtZGxi
R1Z6TVJnd0VRWURWUUVFLRXdwSgogUTBGT1RpQ1VUVU5JTUVJzd0dRWURWUUVFERXhK
SlEwRk9UaUJVVfVOSUlGUkZVMVFnUTBHQ0FRRXdeZ1lEVlIwUAogQVFIL0JBUURB
Z2VTUM0R0ExVVRId1FutUNvd0k2QWhvQitHSFdoMGRIQTZMeTlqY213dWFXtMhi
bTR1YjNkBgogTDNSdFkyZ3VZM0pzTUEwR0NTcUdTSWlZrFFFQkN3VUFBNelCQVFC
MnFTeTdlaSs0M2NlYktVS3dXUHJ6ejl5LwogSWtyTWVKR0tqbzQwbis5dWVrYXcz
REolRXFpT2YvcVo0cGpCRCSrb1I2QkpDYjZOUXVRS3dub0F6NWxFNFNzdQogeTUR
aTkzblQzSGZ5VmM0Z05NSW9IbTFQUZe5bDdeQktyYndiekF1YS8waktXVnpydm1W
NlRCZmp4RDNBuW8xUgogYlU1ZEJyNklqYmRMRmxuTzV4MEcwbXJHN3g1T1VQdXVy
aWh5aVVSceZEchdIOEtBSDF3TWNDcFhHWEZSdEdLawogd3lkZ3lWWUF0eTdvdGts
L3ozYlprQ1ZUMzRnUHZGNzBzUjYrUXhVeThlMEX6RjVBL2JlWWFacHhTWUczMWft
TAogQWRyYXRUV0ZpcGFJR2VhOWxFR0ZNMEw5K0JnN1h6Tm40blZMWG9reUVCM2Jn
UzRzY0c2UXpuWDIzRkdrCiAgIDwvWUdUOUNlcnRpZmljYXRlPgogICA8LglMDlE
YXRhPgogICA8L0tleUluZm8+CiAgPC9TaWduYXRlcUUCgkJPc92ZXJpZmljYXRp
b25Db2RlOnNpZ25lZENvZGU+Cg==
</verificationCode:code>
</verificationCode:encodedSignedCode>

```

Example <verificationCode:encodedSignedCode> element that contains two <verificationCode:code> elements ;.

```

<?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>
    </command>
  </epp>

```

[illegible]

Qm80SC9NSUG4TUF3RwogQTFVZEV3RUIvd1FDTUFbd0hRWURWUjBPQkJZRZQWkVj
SVFjRC9CaJJrNovTEVSdW8yQURKdmlNSUdNQmdOVgogSFNNRwdZUXdnWUdBRk8w
LzdrRWgzRnVFS1MrUS9rWUhhRC9XNndpaG9XYWtaREJpTVFzd0NRWURWUVFHRXdk
VgogVXpFTE1Ba0dBmVVFQ0JNQ1EwRKhGREFTQmdOVkJBY1RDMHh2Y3lCQmJtZGxi
R1Z6TVJnd0VRWURWUVFLRXdwSgogUTBGT1RpQ1VUVU5JTVJzd0dRWURWUVFERXhK
SlEwRk9UaUJVVVFVOSU1GUkZVMVFnUTBHQ0FRRXdEZ1lEVlIwUAogQVFIL0JBUURB
Z2VBTUM0R0ExVWRIdlFuTUNvd0k2QWhvQitHSFdoMGRIQTZMeTlqY213dWFXtMhi
bTR1YjNkBgogTDNSdFkyZ3VZM0pzTUEwR0NTcUdTSWIZRFFFQkN3VUFBNELCQVFC
MnFTeTd1aSs0M2N1YktVS3dXUHJ6ejl5LwogSWtyTWVKR0tqbzQwbis5dWVrYXcz
REolRXFpT2YvcVo0cGpCRCSrb1I2QkpDYjZOUXVRS3dub0F6NWxFNFnZdQogeTur
aTkzb1QzSGZ5VmM0Z05NSW9IbTFQUZe5bDdEQktyYndiekFlYS8waktXVnpydmlW
N1RCZmp4RDNBuW8xUgogYlU1ZEJyNklqYmRmRmxuTzV4MEcwbXJHN3g1T1VQdXVY
aWh5aVVSceZEeChdIOEtBSDF3TWNDcFhHWEZSdEdLawogd3lkZ3lWWUF0eTdvdGts
L3ozYlprQ1ZUMzRnUHZGNzBzUjYrUXhVeThlMEx6RjVBL2JlWWFacHhTWUczMWFt
TAogQWRYaXRUV0ZpcGFJR2VhOWxFR0ZNMew5K0JnN1h6Tm40blZMWG9reUVCm2Jn
UzRzY0c2UXpuWDIzRkdrCiAgIDwvWUWOUNlcnRpZmljYXRlPgogICA8L1g1MD1E
YXRhPgogICA8L0tleUluZm8+CiAgPC9TaWduYXRlcmlU+CgkJPc92ZXJpZmljYXRp
b25Db2RlOnNpZ25lZENvZGU+Cg==

</verificationCode:code>

<verificationCode:code>

PD94bWwgdmVyc2lvcj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz48dmVyaWZpY2F0
aW9uQ29kZTpzaWduZWRDb2RlIHhtbG5zOnZlcmlmaWNhdGlvbkNvZGU9InVybJpp
ZXRMOnBhcmFtcz4bWw6bnM6dmVyaWZpY2F0aW9uQ29kZS0xLjAiIGlkPSJzaWdu
ZWRDb2RlIiB0eXB1PSJyZWdpc3RyYW50Ij48dmVyaWZpY2F0aW9uQ29kZTpjb2Rl
PjEtYWJjMjIyPC92ZXJpZmljYXRpb25Db2RlOmNvZGU+PGRzaWc6U2lnbmF0dXJl
IHhtbG5zOmRzaWc6Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyMi
Pjxkc2lnOlNpZ25lZEluZm8+PGRzaWc6Q2Fub25pY2FsaXphdGlvbk1ldGhvZCBB
bGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnL1RSLzIwMDEvUkVdLXhtbC1jMTRu
LTlwMDewMzEl1ldpdGhDb21tZW50cyIvPjxkc2lnOlNpZ25hdHVyZU1ldGhvZCBB
bGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyNyc2Et
c2hhMSIvPjxkc2lnOlJlZmVyZW5jZSBVUkk9IiNzaWduZWRDb2RlIj48ZHNpZzpU
cmFuc2ZvcmlzPjxkc2lnOlRyYW5zZm9ybSBbbGdvcml0aG09Imh0dHA6Ly93d3cu
dzMub3JnLzIwMDAvMDkveG1sZHNpZyNlbnZlbG9wZWQtc2lnbmF0dXJlIi8+PC9k
c2lnOlRyYW5zZm9ybXM+PGRzaWc6RGlnZXN0TWV0aG9kIEFsZ29yaXRobT0iaHR0
cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxlbmMjc2hhMjU2Ii8+PGRzaWc6RGln
ZXN0VmFsdWU+SfG2TU1WUWdnSStzNG9tT3haYjBGTW1VS1BRdk15WmUybdVEdeHh
QLZMND08L2RzaWc6RGlnZXN0VmFsdWU+PC9kc2lnOlJlZmVyZW5jZT48L2RzaWc6
U2lnbmVksW5mbz48ZHNpZzpTaWduYXRlcmlVWYwX1ZT5VOUhPNVlYVWE0ZUsyYXRz
U1RuQk1DU3dXM0dWUzZnUetkaDBZTlZicERudld4b1BtYlR2YkVsNDE4NF1KZ3Uw
WXB3RkROMmZLY3JVCK1YV0hnce56K0oycTh6MWpTcVJMUEw0UmpnRWw0eGhiOX15
cExOZC8xQXJXRv1hWWZEdUc1S3FYV05MRG5YVzJoQkEzK0R5Wk82MFQKcTVPd0R5
ZVFSVlNPVWNXVE9FOTJsSlZ4M014Q1V6dlhoL0ZOSTlPbGtXK0ZPNVZNNTZlTmZq
UEhkU1JVdjdzQzRmM0NnWmFaSWFXNQP2RmJnTmJodFJVa0hsSVhnYVNGWDgvcFdV
RXFIY0dLTUxnRU1nbHBnQ3RtOFlIcXVqb0tXUk0yUDNiK2h3ZTRsU0hSWVRjK0pB
eEluClU4RDclWnliWThnSWFuZUpRS2dwVTk2T0tJTGQ5L0l0UVhaeHZNPT08L2Rz
aWc6U2lnbmF0dXJlVmFsdWU+PGRzaWc6S2V5SW5mbz48ZHNpZzpYNTA5RGF0YT48
ZHNpZzpYNTA5Q2VydG1maWNhdGU+TUlJRG1UQ0NBbkdnQXdkQkFnSUVmcXE2SFRB
TkJna3Foa2lHOXcwQkFRc0ZBREIxtVJBd0RnWURWUVFHRXdkVmJtdHVIM2R1TVJB


```

dwpeZ11EVlFRSUV3ZFZibXR1YjNkdU1SQXdEZ11EVlFRSEV3ZFZibXR1YjNkdU1S
QXdEZ11EVlFRS0V3ZFZibXR1YjNkdU1SQXdEZ11EC1ZR0UxXd2RWYm10dW1zZHVN
Umt3RndZRFZRUURFeEiYwLhKcFptbGpZWfJwYji1RGiYUmxNQjRYRFRFMU1EWXhO
VEl4TURBeU1sb1gKRFRNMU1EWXhNREl4TURBeU1sb3dkVEVRTUE0R0ExVUVCaE1I
Vlc1cmJtOTNiakVRTUE0R0ExVUVDQk1IVlc1cmJtOTNiakVRTUE0RwpBMVVFQnhN
SFZXNXJibTkzYmpFUU1BNEdBMVVFQ2hNSFZXNXJibTkzYmpFUU1BNEdBMVVFQ3hN
SFZXNXJibTkzYmpFWk1CY0dBMVVFQkF4TVFkbVZ5YVdacFkyRjBhVz11UTI5a1pU
Q0NBU013RFFZSkTvWklodmNOQVFFQkJRQURnZ0VQQURDQ0FRb0NnZ0VCQUpjY2pY
cmsKUWFJL2lHUEZ3WmVITjFnRfVhcTltVnJmQis2eWR5Qmdoc2FHVfZoaERIOFNO
TmtpamxIMkxQ3J3TjhjVjhQZ1BPOXRwbG9rR2F5UwpxNktFaHZtTk03bldsZk5L
SkdSdGNidGMzTnJuYzhiUUJacU1xcFo0UlnRTmh5QWh6Ri85UmErd3Rfc0JWeGF3
VdclL2J0SDZ1YytmClJOde5FcmhJdVlJUmN0WTZIRmRa3BlS3cxYnlYK0RsNkJP
L3ZLdnQ4ND1lY1R3aEzIcDUwWGH2NFVTL0Z5aWVLaGs3dDdHRnJGRlQKL2NCTGsy
WmxFallLcFlEU2dlc2lseFg2QkptZVdCbXZLQz1TL2pBZDhNWmRHVUg2aHNHRXB1
U1BmZkZQV3FWcXl6V0p5bG9lOXF4ZQpnUTZjOFo2SVpXZkUzakxSOUVySDhzOTFD
MmlpTFZrQ0F3RUFBYU1oTUI4d0hRWURWUjBPQkJZRUZiY0JLdk03dmk3dUZNTUx5
ZE43CmVGVXF2YzVVTUEwR0NTcUdTSW1zRFFFQkN3VUFBNELCQVFBVjB2cm1rSWRB
d2l4THZ0NUx5eXpTNFdTUld0dVlWL2JQMVG3NzVMRmYKSWH3a2xoMENidk5rYXlK
Tms2Tnp0eDlSc1AwNWZndkxrZERlN0V5cnRzY3I1ZVdETG1WMGTkMWE1N1Z4bnJh
aEdLTmM2wit1Ui9pSApMatJXb3liWEpFT2N0NWtJSjFzL05CeUUrkdGdJFoTmJz
dVvVUEVCYwVtaWpYUFR0OWxxZE9uM1FibktobXhsalcZYS9KbmhtT20vCkRWYTE0
NDJXTVVUS1UyVFlwVldtdUs2NFkwQXFrN2FldzkvVzIzZEcrT2xhOW9VYnBrSXJr
dDRDN3hRa0d5SXN2eUo3bi9lOFhBRDIKbno1Tlcvek5GwnlrZDAzT2N3M240NkZx
clIwVDlBbFBFWHQxUjlmMjZMdlldj3dWtVNEcrMVRJNHOrV0F2TctVRk9FVnNu
PC9kc2lnOlglMDlDZXJ0aWZpY2F0ZT48L2RzaWc6WDUwOURhdGE+PC9kc2lnOktl
eUluZm8+PC9kc2lnOlNpZ25hdHVyZT48L3Zlcm1maWNhdGlvbkNvZGU6c2lnbmVk
Q29kZT4=

```

```

    </verificationCode:code>
  </verificationCode:encodedSignedCode>
</extension>
<clTRID>ABC-12345</clTRID>
</command>
</epp>

```

2.2. Verification Profile

A Verification Profile defines the set of verification code types, the commands that the verification code types are required, supported, or not supported, and the grace period by which the verification code types MUST be set. A server MAY support many verification profiles, each with a unique name and a unique verification policy that is implemented by the server. Each client MAY have zero or more server assigned verification profiles that will enforce the required verification policies. Most likely a client will be assigned zero or one server assigned verification profile, but overlapping profiles is possible. Overlapping verification profiles MUST be treated as a logical "and" of the policies by the

server. If no verification profile is assigned to the client, no additional verification is required by the client.

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 defines additional elements to extend the EPP <info> command of an object mapping like the EPP domain name mapping [RFC5731], EPP host mapping [RFC5732], and EPP contact mapping [RFC5733].

The EPP <info> command is used to retrieve the verification information. The verification information is based on the verification profile, as defined in Section 2.2, set in the server for the client. The <verificationCode:info> element is an empty element that indicates that the client requests the verification information. The OPTIONAL "profile" attribute can be used by the client to explicitly specify a verification profile, as defined in Section 2.2, to base the verification information on. It is up to server policy on the set of verification profiles that the client is allowed to explicitly specify, and if the client is not allowed, the server MUST return the 2201 error response.

Example <info> domain command with the <verificationCode:info> extension to retrieve the verification information for the domain "domain.example", using the profiles associated with the client:

```
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:      <verificationCode:info
C:        xmlns:verificationCode=
C:          "urn:ietf:params:xml:ns:verificationCode-1.0"/>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <info> domain command with the <verificationCode:info> extension to retrieve the verification information for the domain "domain.example", using the profiles associated with the client and with the authorization information to retrieve the verification codes from the non-sponsoring client:

```
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:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:info>
C:    </info>
C:    <extension>
C:      <verificationCode:info
C:        xmlns:verificationCode=
C:          "urn:ietf:params:xml:ns:verificationCode-1.0"/>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <info> domain command with the <verificationCode:info> extension to retrieve the verification information for the domain "domain.example", using the the "sample" profile:

```
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:      <verificationCode:info
C:        xmlns:verificationCode=
C:          "urn:ietf:params:xml:ns:verificationCode-1.0"
C:        profile="sample"/>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

<verificationCode:status> The status of the verification for the object, using all of the verification profiles assigned to the client. There are four possible values for the status:

- notApplicable The status is not applicable to the client since there is no assigned verification profile.
- nonCompliant The object is non-compliant according to the verification profiles. If at least one of the profiles is "nonCompliant", the object is "nonCompliant".
- pendingCompliance The object is not in compliance with the verification profiles, but has a grace period to set the required set of verification codes, as reflected by the due date of the verification code type. If at least one of the profiles is "pendingCompliance" and none of the profiles is "nonCompliant", the object is "pendingCompliance".
- compliant The object is compliant with the verification profiles. If All of the profiles for the object are "compliant" or if the object has no assigned profiles, the object is "compliant".

`<verificationCode:profile>` Zero or more OPTIONAL `<verificationCode:profile>` elements that defines the verification status of the object based on the profile. The required "name" attribute defines the name of the profile. The `<verificationCode:profile>` element contains the following child elements:

`<verificationCode:status>` The status of the verification for the object and the profile. There are four possible values for the status:

`notApplicable` The profile status is not applicable to the client based on the assigned verification profiles or the profile specified.

`nonCompliant` The object is non-compliant according to the verification profile.

`pendingCompliance` The object is not in compliance with the verification profile, but has a grace period to set the required set of verification codes, as reflected by the due date of the verification code type.

`compliant` The object is compliant with the verification profile.

`<verificationCode:missing>` OPTIONAL list of missing verification code types. The `<verificationCode:missing>` element is returned only if there is at least one missing verification code type and based on server policy. The `<verificationCode:missing>` element contains the following child elements:

`<verificationCode:code>` One or more `<verificationCode:code>` elements that is empty with the REQUIRED "type" attribute that indicates the verification code type and the REQUIRED "due" attribute that indicates when the verification code type was or is due. Past due verification code types will result in the `<verificationCode:status>` element being set to "nonCompliant".

`<verificationCode:set>` OPTIONAL list of set verification codes. The `<verificationCode:set>` element is returned only if there is at least one set verification code. The `<verificationCode:set>` element contains the following child elements:

<verificationCode:code> One or more <verificationCode:code> elements containing the verification code with a REQUIRED "type" attribute that indicates the code type and a REQUIRED "date" attribute that indicates when the verification code was set. The inclusion of the code value is up server policy, so if the server determines that the code value cannot be exposed to a non-sponsoring client, the <verificationCode:code> element MUST be empty.

Example <info> domain response using the <verificationCode:infData> extension for a compliant domain using the "sample" profile, and with the two verification codes, from the sponsoring or 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>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>compliant
S:      </verificationCode:status>
S:      <verificationCode:profile name="sample">
S:        <verificationCode:status>compliant
S:      </verificationCode:status>
S:      <verificationCode:set>
S:        <verificationCode:code type="domain"
```

```

S:         date="2010-04-03T22:00:00.0Z">1-abc333
S:         </verificationCode:code>
S:         <verificationCode:code type="registrant"
S:         date="2010-04-03T22:00:00.0Z">1-abc444
S:         </verificationCode:code>
S:         </verificationCode:set>
S:         </verificationCode:profile>
S:         </verificationCode:infData>
S:     </extension>
S:     <trID>
S:         <clTRID>ABC-12345</clTRID>
S:         <svTRID>54322-XYZ</svTRID>
S:     </trID>
S: </response>
S:</epp>

```

Example <info> domain response using the <verificationCode:infData> extension for a compliant domain using the "sample" profile, and with the two verification codes, from the sponsoring or authorized client that also includes codes set for the "sample2" profile:

```

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>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">

```

```
S:      <verificationCode:status>compliant
S:      </verificationCode:status>
S:      <verificationCode:profile name="sample">
S:          <verificationCode:status>compliant
S:          </verificationCode:status>
S:          <verificationCode:set>
S:              <verificationCode:code type="domain"
S:                  date="2010-04-03T22:00:00.0Z">1-abc333
S:              </verificationCode:code>
S:              <verificationCode:code type="registrant"
S:                  date="2010-04-03T22:00:00.0Z">1-abc444
S:              </verificationCode:code>
S:          </verificationCode:set>
S:      </verificationCode:profile>
S:      <verificationCode:profile name="sample2">
S:          <verificationCode:status>notApplicable
S:          </verificationCode:status>
S:          <verificationCode:set>
S:              <verificationCode:code type="domain"
S:                  date="2010-04-03T22:00:00.0Z">2-abc555
S:              </verificationCode:code>
S:          </verificationCode:set>
S:      </verificationCode:profile>
S:      </verificationCode:infData>
S:  </extension>
S:  <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:  </trID>
S: </response>
S: </epp>
```


Example <info> domain response using the <verificationCode:infData> extension for a compliant domain using the "sample" profile, and with the two verification code types, from the non-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:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>compliant
S:        </verificationCode:status>
S:        <verificationCode:profile name="sample">
S:          <verificationCode:status>compliant
S:          </verificationCode:status>
S:          <verificationCode:set>
S:            <verificationCode:code type="domain"
S:              date="2010-04-03T22:00:00.0Z"/>
S:            <verificationCode:code type="registrant"
S:              date="2010-04-03T22:00:00.0Z"/>
S:          </verificationCode:set>
S:        </verificationCode:profile>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> domain response using the <verificationCode:infData> extension for a non-compliant domain using the "sample" profile, and with the verification code types missing along with their due dates:

```
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>DOMAIN-REP</domain:roid>
S:        <domain:status s="serverHold"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:          "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>nonCompliant
S:        </verificationCode:status>
S:        <verificationCode:profile name="sample">
S:          <verificationCode:status>nonCompliant
S:          </verificationCode:status>
S:          <verificationCode:missing>
S:            <verificationCode:code
S:              type="domain"
S:              due="2010-04-03T22:00:00.0Z"/>
S:            <verificationCode:code
S:              type="registrant"
S:              due="2010-04-08T22:00:00.0Z"/>
S:          </verificationCode:missing>
S:        </verificationCode:profile>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> domain response using the <verificationCode:infData>

extension for a pending compliance domain using the "sample" profile, with the verification code type missing along with the due date, and with set verification code:

```
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>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>pendingCompliance
S:        </verificationCode:status>
S:        <verificationCode:profile name="sample">
S:          <verificationCode:status>pendingCompliance
S:          </verificationCode:status>
S:          <verificationCode:missing>
S:            <verificationCode:code
S:              type="registrant"
S:              due="2010-04-08T22:00:00.0Z"/>
S:          </verificationCode:missing>
S:          <verificationCode:set>
S:            <verificationCode:code type="domain"
S:              date="2010-04-03T22:00:00.0Z">1-abc333
S:            </verificationCode:code>
S:          </verificationCode:set>
S:        </verificationCode:profile>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> domain response using the <verificationCode:infData> extension for a client that does not have a verification profile assigned:

```
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>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>notApplicable
S:        </verificationCode:status>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-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 the EPP domain name mapping [RFC5731], EPP host mapping [RFC5732], and EPP contact mapping [RFC5733].

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 MAY contain a child <verificationCode:encodedSignedCode> element, as defined in Section 2.1.2, that identifies the extension namespace for the client to provide proof of verification by a Verification Service Provider (VSP). The server MAY support multiple policies for the passing of the <verificationCode:encodedSignedCode> element based on the client profile, which include:

required The client MUST pass a valid
 <verificationCode:encodedSignedCode> element containing the
 required set of verification codes. If a
 <verificationCode:encodedSignedCode> element is not passed or the
 required set of verification codes is not included, the server
 MUST return an EPP error result code of 2306. If an invalid
 <verificationCode:encodedSignedCode> element is passed, the
 server MUST return an EPP error result code of 2005.
optional The client MAY pass a valid
 <verificationCode:encodedSignedCode> element. If an invalid
 <verificationCode:encodedSignedCode> element is passed, the
 server MUST return an EPP error result code of 2005.
not supported The client MUST NOT pass a
 <verificationCode:encodedSignedCode> element. If a
 <verificationCode:encodedSignedCode> element is passed, the
 server MUST return an EPP error result code of 2102.

Example <create> command to create a domain object with a verification code:

```
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>jdl234</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:          <verificationCode:encodedSignedCode
C:              xmlns:verificationCode=
C:                  "urn:ietf:params:xml:ns:verificationCode-1.0">
C:              <verificationCode:code>
C:ICAgICAgPHZlcmlmaWNhdGlvbkNvZGU6c2lnbmVkQ29kZQogICAgICAgIHhtbG5z
C:OnZlcmlmaWNhdGlvbkNvZGU9CiAgICAgICAgICAidXJuOmlldGY6cGFyYW1zOnht
C:bDpuczp2ZXJpZmljYXRpb25Db2RlLTEuMCiKICAgICAgICAgIGlkPSJzaWduZWRD
C:b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMTIzPC92ZXJp
C:ZmljYXRpb25Db2RlOmNvZGU+CiAgPFNpZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3
C:dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CiAgIDxTaWduZWRJbmZvPgogICAg
C:PENhbm9uaWNhbGl6YXRpb25NZXRob2QKIEFsZ29yaXRobT0iaHR0cDovL3d3dy53
C:My5vcmcvMjAwMS8xMC94bWwtZXhjLWwNG4jIi8+CiAgICA8U2lnbmF0dXJlTWV0
C:aG9kCiBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveGlsZHNp
C:Zy1tb3JlIj4KICAgIDxSZWZlcmluY2UgVVJJPStjc2lnbmVk
C:Q29kZSI+CiAgICAgPFYyZW5zZm9ybXh0dXJlTWV0dXJlTWV0dXJlTWV0dXJlTWV0
C:aXR0bT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CiAgIDxTaWduZWRJbmZvPgogICAg
C:UGZ5U0UpdSkJGMEUxU0VQRkNkaml5d0NfCVIyaDRMRcTnZTZYUStKbmlLRkZDdUNa
C:Uy8zU0xLQXgwTDF3CiBRREZPMmUwWTY5azJHNY9MR0UzNlgzdK9mbG9iRk0xb0d3
C:amE4K0dNvNjhb3RvNXhBZDQvQUY3ZUhlal2dBWlECiBvOXRveG9hMmgweVY0QTRQ
C:bVh6c1U2Uzg2WHRDYlVfK1MvV003Mm55bjQ3em9VQ3p6UEtIwkJSeWVXZWhWRLer
C:CbQWVJNSUFNek01N0hIUUERnmVhWGVmUnZ0UEVUZlVPNGFWSVZTdWdjNE9VQVpa
C:d2JZYlpyQzZ3T2FRcXFxQVppCiAzMGFQT0JZYkF2SE1TbVdTUytoRmtic2hvbUpm
C:SHhiOTdURDJncmZTnJRSXpxWGs3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z
C:VFV3MUN6QTdqZndBPT0KICAgPC9TaWduYXRlcmluVWYwLzI0ZDQKICAgPETleUluZm8+
C:CiAgICA8WDUwOURhdGE+CiAgICA8WDUwOUNlcnRpZmljYXRlPgogTUlJRvNUQ0NB
C:ekdnQXJkQkFnSUJBakFOQmdrcWhraUc5dzBCQVZzRkFEQmlNUXN3Q1FZRFZRUUdf
C:d0pWVXpFTaogTUFRrR0ExVUVDQk1DUTBFeEZEQVNCZ05WQkFjVEMweHZeUJCYm1k
C:bGJHVnpNUk13RVFZRFZRUUtdf3BKUTBGTwogVGLCVVRVtKlNUN3RlFZRFZRUURF
C:eEpKUvBTGT1RqP1VUVU5JUSZSRlUxUWdRMEV3SGhjTklUTXhNake0TURBdwZURRB
C:dlldY05NVGd3TWBmBM01qTTFPVFU1V2pCc01Rc3dDUVlEVlFRR0V3SlZVekVMTUFr
```



```
C:ROEXVUVDQk1DUTBFeAogRkrBU0JnTlZCQWNUQzB4dmN5QkJibWRsYkdWek1SY3dG
C:UV1EV1FRS0V3NVdZV3hwWkdGMGIzSWdWRTFEUORFaAogTUI4R0ExVUVBeE1ZVm1G
C:c2FXUmhkRz15SUZSTLEwZ2dWRVZUVkNCRFJWS1VNSULCSWPBTkJna3Foa2lHOXcw
C:QgogQVFfRkFBT0NBUTHtBUlJQkNnS0NBUUVBby9jd3ZYaGJWWwwwUkRXV3ZveWVa
C:cEVUVlpWmMNQ292VVZOzy9zdwoGV2ludUl nRVdnVlFGcnoweEEwNHBFafHdRLZ2
C:NGV2YlVwZWtKNWJ1cVUxZ21ReU9zQ0trBgHPsFRkUGp2a0MldQogcERxyTUxRmxj
C:MFRNYUlrSVFqczdhVUtDbUE0Ukc0dFRUR0svRWpSMWL4OC9EMGDIVWZSbsGR5MVlQ
C:ck1QK291NwogNWJPVm5Jb3MrSGlmckf0cck12NHFFcXdmTDRGVFPBVXBhQ2EyQmln
C:WGZ5MKNTUFlieEQ1T3IxZ2NTYTNDxJONQogc1BNQ054cWFYbulYbVFpcFMrRHVF
C:CqnFNtTh0bGRhtjdjSSW9qVUVLckdwC05rNwk5eTiVN3Nqbjf6ex1LVUGY3dgogTDRH
C:Z0RZcWhKWvdWNjFEblhneC9KZDZDV3h2c25ERjZzY3NjUXpvVEVsK2h5d0leQVFB
C:Qm80SC9NSUG4TUF3RwogQTFVZE3RU1vdlFDtUFBd0hRWURWUjBPQkJZRUZQWkvj
C:SVFjRC9CaJJJRnovTEVSdW8yQURKdmlNSUDnQmdOVgogSFNNRwdZUXdnWUDBRk8w
C:LzdrRWgzRnVFS1MrUS9rWUhhRC9XNdpaG9XYWtaREJpTVFzd0NRWURWUVFHRXdK
C:VgogVXpFTelBa0dBMMVVFQ0JNQ1EWrxHgREFTQmdOVkjbY1RDMHH2Y3lCQmJtZGxi
C:R1Z6TVJNd0VRWURWUVFLRXdwSgogUTBGt1RpQ1VUVU5JTJVJzd0dRWURWUVFERXHk
C:S1EWrk9UaUJVVVFOSULGUkZVMVFnUTBHQ0FRRXdeZ1lEVlIiwUAogQVFIL0JBuurB
C:Z2VBtUM0R0ExVWRIdlFuTUNvd0k2QwhvQi tHSFdoMGRIQTZMeTlqY2l3dWFXtmhi
C:bTR1YjNkbGogTDNSdfkyZ3VZM0pzTUEwr0NTcUdTswiZRFFFQkn3VUFBNElcQVFC
C:MnFTetdlaSs0M2NlyktVS3dXUHJ6ejl5LwogSWtyTWVKR0tgzbzQwbis5dWVrYxcz
C:REolRXfpT2YvcVo0cGpCRCsrbl12QkpDYjZOUXVRS3dub0F6NwxNFNnzDQogeTUR
C:aTkzb1QzSGZ5VmM0Z05NSW9lbTFQUZe5bdDEQktyYndiekFlYS8waktXVnpdymlW
C:NlRCZmp4RDNBuw8xUgogYlU1ZEJyNklqYmRMrmxuTzV4MEcwbXJHN3glTlVQdxVy
C:aWh5aVVSceZEchdioEtBSDF3TwNDcfHWEZSDedLawogd3lkZ3lWWUF0etdvdtGs
C:L3ozYlprQ1ZUMzRnUHZGNzBzUjYrUXhVeThlMex6RjVBL2JlWWFacHhTWUczMWft
C:TAogQWRyaXRUV0ZpcGFJR2VhOWxfR0ZNMEw5K0JnNlh6Tm40blZMWG9reUVCm2Jn
C:UzRzY0c2UXpuWDIzRkdrCiAgIDwvWduOUnlcnRpZmljYXRlPgogICA8Llg1MDIE
C:YXRhPgogICA8L0tleUluZm8+CiAgPC9TaWduYXR1cmU+CgkJPc92ZXJpZmljYXRp
C:b25Db2RlOnNpZ25lZENvZGU+Cg==
C:
C:      </verificationCode:code>
C:      </verificationCode:encodedSignedCode>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
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 defines additional elements to extend the EPP <delete> command and response in the same fashion as defined for the EPP <create> Command (Section 3.2.1).

3.2.3. EPP <renew> Command

This extension defines additional elements to extend the EPP <renew> command and response in the same fashion as defined for the EPP <create> Command (Section 3.2.1).

3.2.4. EPP <transfer> Command

This extension defines additional elements to extend the EPP <transfer> command and response in the same fashion as defined for the EPP <create> Command (Section 3.2.1).

3.2.5. EPP <update> Command

This extension defines additional elements to extend the EPP <update> command and response in the same fashion as defined for the EPP <create> Command (Section 3.2.1).

4. Formal Syntax

One schema is presented here that is the EPP Verification Code 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. Verification Code Extension Schema

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace=
    "urn:ietf:params:xml:ns:verificationCode-1.0"
  xmlns:verificationCode=
    "urn:ietf:params:xml:ns:verificationCode-1.0"
  xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
      Verification Code Extension.
    </documentation>
  </annotation>
```

```
<import namespace="http://www.w3.org/2000/09/xmldsig#"
  schemaLocation="xmldsig-core-schema.xsd"/>

<!-- Abstract signed code for substitution -->
<element name="abstractSignedCode"
  type="verificationCode:abstractSignedCodeType"
  abstract="true"/>

<!-- Empty type for use in extending for a signed code -->
<complexType name="abstractSignedCodeType"/>

<!-- Definition of concrete signed code -->
<element name="signedCode"
  type="verificationCode:signedCodeType"
  substitutionGroup="verificationCode:abstractSignedCode"/>

<complexType name="signedCodeType">
  <complexContent>
    <extension base="verificationCode:abstractSignedCodeType">
      <sequence>
        <element name="code"
          type="verificationCode:verificationCodeType"/>
        <element ref="dsig:Signature"/>
      </sequence>
      <attribute name="id" type="ID" use="required"/>
    </extension>
  </complexContent>
</complexType>

<simpleType name="verificationCodeValueType">
  <restriction base="token">
    <pattern value="\d+-[A-Za-z0-9]+"/>
  </restriction>
</simpleType>

<complexType name="verificationCodeType">
  <simpleContent>
    <extension base=
      "verificationCode:verificationCodeValueType">
      <attribute name="type" type="token"
        use="required"/>
    </extension>
  </simpleContent>
</complexType>

<!-- Definition of an encoded signed code -->
<element name="encodedSignedCode"
  type="verificationCode:encodedSignedCodeListType"/>
```

```
<complexType name="encodedSignedCodeListType">
  <sequence>
    <element name="code"
      type="verificationCode:encodedSignedCodeType"
      minOccurs="1" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<complexType name="encodedSignedCodeType">
  <simpleContent>
    <extension base="token">
      <attribute name="encoding"
        type="token" default="base64"/>
    </extension>
  </simpleContent>
</complexType>

<!-- info command extension elements -->
<element name="info" type="verificationCode:infoType"/>

<complexType name="infoType">
  <simpleContent>
    <extension base="token">
      <attribute name="profile" type="token"/>
    </extension>
  </simpleContent>
</complexType>

<!-- info response extension elements -->
<element name="infData" type="verificationCode:infDataType"/>

<complexType name="infDataType">
  <sequence>
    <element name="status"
      type="verificationCode:statusEnum"/>
    <element name="profile"
      type="verificationCode:profileDataType"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<complexType name="profileDataType">
  <sequence>
    <element name="status"
      type="verificationCode:statusEnum"/>
    <element name="missing"
      type="verificationCode:missingCodes"/>
  </sequence>
</complexType>
```

```
        minOccurs="0"/>
      <element name="set"
        type="verificationCode:codesType"
        minOccurs="0"/>
    </sequence>
    <attribute name="name" type="token"/>
  </complexType>

  <simpleType name="statusEnum">
    <restriction base="token">
      <enumeration value="notApplicable"/>
      <enumeration value="nonCompliant"/>
      <enumeration value="pendingCompliance"/>
      <enumeration value="compliant"/>
    </restriction>
  </simpleType>

  <complexType name="missingVerificationCode">
    <simpleContent>
      <extension base="token">
        <attribute name="type" type="token"
          use="required"/>
        <attribute name="due" type="dateTime"
          use="required"/>
      </extension>
    </simpleContent>
  </complexType>

  <complexType name="missingCodes">
    <sequence>
      <element name="code"
        type="verificationCode:missingVerificationCode"
        minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="infoVerificationCodeType">
    <simpleContent>
      <extension base="token">
        <attribute name="type" type="token"
          use="required"/>
        <attribute name="date" type="dateTime"
          use="required"/>
      </extension>
    </simpleContent>
  </complexType>

  <complexType name="codesType">
```

```
<sequence>
  <element name="code"
    type="verificationCode:infoVerificationCodeType"
    minOccurs="1" maxOccurs="unbounded"/>
</sequence>
</complexType>

</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].

Registration request for the verificationCode namespace:

URI: ietf:params:xml:ns:verificationCode-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 verificationCode XML schema:

URI: ietf:params:xml:ns:verificationCode-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: "Verification Code 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 extension described in this document is based on the security services described by EPP [RFC5730] and protocol layers used by EPP. The security considerations described in these other specifications apply to this specification as well.

XML Signature [W3C.CR-xmlsig-core2-20120124] is used in this extension to verify that the Verification Code originated from a trusted Verification Service Provider (VSP) and that it wasn't tampered with in transit from the VSP to the client to the server. To support multiple VSP keys, the VSP certificate chain MUST be included in the <X509Certificate> elements of the Signed Code (Section 2.1.1) and MUST chain up and be verified by the server against a set of trusted certificates.

It is RECOMMENDED that signed codes do not include white-spaces between the XML elements in order to mitigate risks of invalidating the digital signature when transferring of signed codes between applications takes place.

Use of XML canonicalization SHOULD be used when generating the signed code. SHA256/RSA-SHA256 SHOULD be used for digesting and signing. The size of the RSA key SHOULD be at least 2048 bits.

7. Normative References

- [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, August 2009.

- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, August 2009.
- [RFC5732] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Host Mapping", STD 69, RFC 5732, August 2009.
- [RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, August 2009.
- [RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, February 2015.
- [W3C.CR-xmlsig-core2-20120124]
Cantor, S., Roessler, T., Eastlake, D., Yiu, K., Reagle, J., Solo, D., Datta, P., and F. Hirsch, "XML Signature Syntax and Processing Version 2.0", World Wide Web Consortium CR CR-xmlsig-core2-20120124, January 2012, <<http://www.w3.org/TR/2012/CR-xmlsig-core2-20120124>>.

Appendix A. Acknowledgements

Appendix B. Change History

B.1. Change from 00 to 01

1. Fixed pendingCompliance and complaint to pendingCompliance and compliant in text.
2. Fixed verificaton to verification.

B.2. Change from 01 to 02

1. Added support for the notApplicable status value.

B.3. Change from 02 to 03

1. Added regular expression pattern for the format of the verification code token value in the XML schema.

B.4. Change from 03 to 04

1. Ping update.

Author's Address

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

Email: jgould@verisign.com
URI: <http://www.verisign.com>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 18, 2019

J. Gould
VeriSign, Inc.
F. Obispo
L. Munoz
Uniregistry Corp.
October 15, 2018

Extensible Provisioning Protocol (EPP) Internationalized Domain Name
(IDN) Table Mapping
draft-gould-idn-table-07

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

8.2. Informative References	28
Appendix A. Change History	28
A.1. Change from 00 to 01	28
A.2. Change from 01 to 02	29
A.3. Change from 02 to 03	29
A.4. Change from 03 to 04	29
A.5. Change from 04 to 05	29
A.6. Change from 05 to 06	29
A.7. Change from 06 to 07	29
Authors' Addresses	29

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:    <clTRID>ABC-12345</clTRID>
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:

```
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:domain>
S:            <idnTable:name valid="true">
S:              idn1.example</idnTable:name>
S:            <idnTable:table>CHI</idnTable:table>
S:          </idnTable:domain>
S:          <idnTable:domain>
S:            <idnTable:name valid="true" idnmap="true">
S:              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">
S:              idn3.example</idnTable:name>
S:            <idnTable:reason>Commingled scripts
S:          </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:

```
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:table>CHI</idnTable:table>
C:          <idnTable:table>JPN</idnTable:table>
C:          <idnTable:table>INVALID</idnTable:table>
C:        </idnTable: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 `<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:

```
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
S:        </idnTable:table>
S:        <idnTable:table exists="true">JPN
S:        </idnTable:table>
S:        <idnTable:table exists="false">INVALID
S:        </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:

```
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:

```
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
S:            </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)
S:          </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)
S:          </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:

```
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:table>CHI</idnTable:table>
C:        </idnTable:info>
C:      </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

```
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:table>THAI</idnTable:table>
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 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)
S:          </idnTable:description>
S:          <idnTable:upDate>2015-02-04T09:30:00.0Z
S:          </idnTable:upDate>
S:          <idnTable:version>1.0</idnTable:version>
S:          <idnTable:effectiveDate>2014-11-24
S:          </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
C:        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
S:            </idnTable:upDate>
S:          </idnTable:table>
S:          <idnTable:table>
S:            <idnTable:name>JPN</idnTable:name>
S:            <idnTable:upDate>2015-01-01T09:40:00.0Z
S:            </idnTable:upDate>
S:          </idnTable:table>
S:          <idnTable:table>
S:            <idnTable:name>THAI</idnTable:name>
S:            <idnTable:upDate>2014-08-16T09:20:00.0Z
S:            </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>
```

```
<!--
Child elements of check command.
-->
<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>
```

```
<!--
Table check response element
-->
<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>
```



```
        </choice>
      </sequence>
    </complexType>

    <!--
    Table types
    -->
    <simpleType name="tableTypeEnumType">
      <restriction base="token">
        <enumeration value="language"/>
        <enumeration value="script"/>
      </restriction>
    </simpleType>

    <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>
```

```
        </choice>
        <element name="table" type="idnTable:infDomainTableType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </complexType>

    <!--
    Domain table info response information
    -->
    <complexType name="infDomainTableType">
      <sequence>
        <element name="name" type="eppcom:minTokenType"/>
        <element name="type" type="idnTable:tableTypeEnumType"/>
        <element name="description" type="idnTable:descriptionType"/>
        <element name="variantGen" type="boolean" minOccurs="0"/>
      </sequence>
    </complexType>

    <!--
    Table list info response information.
    -->
    <complexType name="infListType">
      <sequence>
        <element name="table" type="idnTable:infListTableType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </complexType>

    <!--
    Table elements in list info form.
    -->
    <complexType name="infListTableType">
      <sequence>
        <element name="name" type="eppcom:minTokenType"/>
        <element name="upDate" type="dateTime"/>
      </sequence>
    </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: 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

URL: https://www.verisign.com/en_US/channel-resources/domain-registry-products/epp-sdks

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

- [I-D.ietf-eppext-idnmap]
Obispo, F., "Internationalized Domain Name Mapping Extension for the Extensible Provisioning Protocol (EPP)", draft-ietf-eppext-idnmap-02 (work in progress), January 2015.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<https://www.rfc-editor.org/info/rfc5730>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <<https://www.rfc-editor.org/info/rfc5731>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<https://www.rfc-editor.org/info/rfc5890>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.

8.2. Informative References

- [RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <<https://www.rfc-editor.org/info/rfc7451>>.

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/>

Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: July 9, 2015

F. Obispo
L. Munoz
Uniregistry
January 5, 2015

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 characters 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

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions Used in This Document	3
3. EPP Command Mapping	3
3.1. EPP Query Commands	3
3.1.1. EPP <info> Command	3
3.2. EPP Transform Commands	6
3.2.1. EPP <create> Command	6
3.3. Formal Syntax	7
4. IANA Considerations	8
5. Security Considerations	9
6. References	9
6.1. Normative References	9
6.2. Informational References	9
6.3. URIs	9
Authors' Addresses	9

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:

```
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>xn--espaol-zwa.example.com</domain:name>
C:          <domain:authInfo>
C:            <domain:pw>2fooBAR</domain:pw>
C:          </domain:authInfo>
C:        </domain:info>
C:      </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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:

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

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--espaol-zwa.example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrar>jdl1234</domain:registrar>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
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&#xF1;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:

- o A <idn:table> element that contains the IDN table identifier as provided by the server.
- o 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>jdl234</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:      <idn:data xmlns:idn="urn:ietf:params:xml:ns:idn-1.0">
C:        <idn:table>es</idn:table>
C:        <idn:uname>espa&#xF1;ol.example.com</idn:uname>
C:      </idn:data>
C:    </extension>
C:    <clTRID>123456</clTRID>
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.

XML: See the "Formal Syntax" 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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, August 2009.
- [RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.

6.2. Informational References

- [RFC3735] Hollenbeck, S., "Guidelines for Extending the Extensible Provisioning Protocol (EPP)", RFC 3735, March 2004.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, August 2009.

6.3. URIs

- [1] <http://www.icann.org/en/topics/idn/idn-guidelines-26apr07.pdf>
- [2] <http://www.unicode.org/reports/tr15/>

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/>

eppext
Internet-Draft
Intended status: Standards Track
Expires: December 2, 2016

H.W. Ribbers
M.W. Groeneweg
SIDN
R. Gieben

A.L.J Verschuren

May 31, 2016

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

A.13. draft-ietf-eppext-keyrelay-08	18
A.14. draft-ietf-eppext-keyrelay-09	18
A.15. draft-ietf-eppext-keyrelay-10	18
A.16. draft-ietf-eppext-keyrelay-11	18
A.17. draft-ietf-regext-keyrelay-00	18
Authors' Addresses	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, reliable 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.

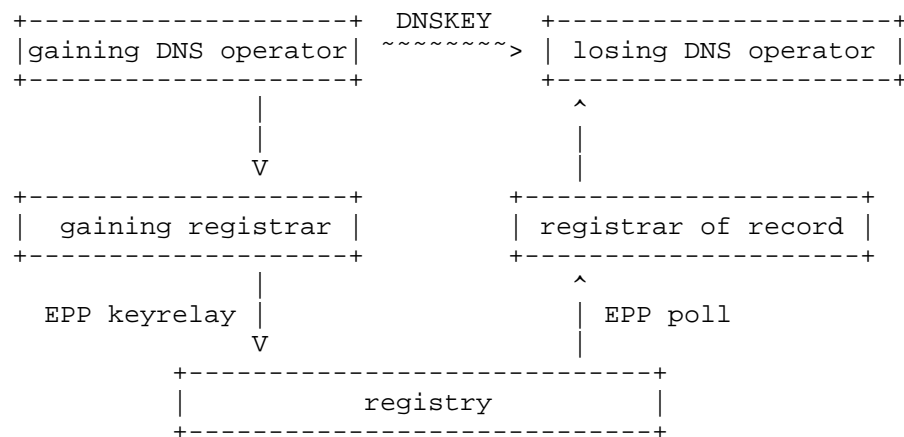


Figure 1: Transfer of DNSSEC key material.

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:

- o One REQUIRED <keyData> element that contains the DNSSEC key material as described in [RFC5910], Section 4
- o 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.

3.1.1.1. 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.

3.1.1.2. 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:

- o A REQUIRED <name> element containing the domain name for which the DNSSEC key material is relayed.
- o A REQUIRED <authInfo> element that contains authorization information associated with the domain object ([RFC5731], Section 3.2.1).
- o 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".
- o An OPTIONAL <crDate> element that contains the date and time of the submitted <create> command.
- o 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:


```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
S:  xmlns:keyrelay="urn:ietf:params:xml:ns:keyrelay-1.0"
S:  xmlns:s="urn:ietf:params:xml:ns:secDNS-1.1"
S:  xmlns:d="urn:ietf:params:xml:ns:domain-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>1999-04-04T22:01:00.0Z</qDate>
S:      <msg>Keyrelay action completed successfully.</msg>
S:    </msgQ>
S:    <resData>
S:      <keyrelay:infData>
S:        <keyrelay:name>example.org</keyrelay:name>
S:        <keyrelay:authInfo>
S:          <d:pw>JnSdBAZSxxzJ</d:pw>
S:        </keyrelay:authInfo>
S:        <keyrelay:keyRelayData>
S:          <keyrelay:keyData>
S:            <s:flags>256</s:flags>
S:            <s:protocol>3</s:protocol>
S:            <s:alg>8</s:alg>
S:            <s:pubKey>cmlraXN0aGVlZXN0</s:pubKey>
S:          </keyrelay:keyData>
S:          <keyrelay:expiry>
S:            <keyrelay:relative>P1M13D</keyrelay:relative>
S:          </keyrelay:expiry>
S:        </keyrelay:keyRelayData>
S:        <keyrelay:crDate>
S:          1999-04-04T22:01:00.0Z
S:        </keyrelay:crDate>
S:        <keyrelay:reID>
S:          ClientX
S:        </keyrelay:reID>
S:        <keyrelay:acID>
S:          ClientY
S:        </keyrelay:acID>
S:      </keyrelay:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-ZYX</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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:

- o A REQUIRED <keyrelay:name> element containing the domain name for which the DNSSEC key material is relayed.
- o A REQUIRED <authInfo> element that contains authorization information associated with the domain object ([RFC5731], Section 3.2.1).
- o 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).

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
C:  xmlns:keyrelay="urn:ietf:params:xml:ns:keyrelay-1.0"
C:  xmlns:s="urn:ietf:params:xml:ns:secDNS-1.1"
C:  xmlns:d="urn:ietf:params:xml:ns:domain-1.0">
C:  <command>
C:    <create>
C:      <keyrelay:create>
C:        <keyrelay:name>example.org</keyrelay:name>
C:        <keyrelay:authInfo>
C:          <d:pw>JnSdBAZSxxzJ</d:pw>
C:        </keyrelay:authInfo>
C:        <keyrelay:keyRelayData>
C:          <keyrelay:keyData>
C:            <s:flags>256</s:flags>
C:            <s:protocol>3</s:protocol>
C:            <s:alg>8</s:alg>
C:            <s:pubKey>cmlraXN0aGVlZXN0</s:pubKey>
C:          </keyrelay:keyData>
C:          <keyrelay:expiry>
C:            <keyrelay:relative>P1M13D</keyrelay:relative>
C:          </keyrelay:expiry>
C:        </keyrelay:keyRelayData>
C:        <keyrelay:keyRelayData>
C:          <keyrelay:keyData>
C:            <s:flags>256</s:flags>
C:            <s:protocol>3</s:protocol>
C:            <s:alg>8</s:alg>
C:            <s:pubKey>bWFyY2lzdGhlYmVzdA==</s:pubKey>
C:          </keyrelay:keyData>
C:          <keyrelay:expiry>
C:            <keyrelay:relative>P0D</keyrelay:relative>
C:          </keyrelay:expiry>
C:        </keyrelay:keyRelayData>
C:      </keyrelay:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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:

```
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-ZYX</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

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:

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

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

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 succesful 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:

Maarten Wullink, Marco Davids, Ed Lewis, James Mitchell, David Peal, Patrik Faltstrom, Klaus Malorny, James Gould, Patrick Mevzek, Seth Goldman, Maarten Bosteels, Ulrich Wisser, Kees Monshouwer and Scott Hollenbeck.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, August 2009.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, August 2009.
- [RFC5910] Gould, J. and S. Hollenbeck, "Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)", RFC 5910, May 2010.

8.2. Informative References

[I-D.koch-dnsop-dnssec-operator-change]

Koch, P., Sanz, M., and A. Verschuren, "Changing DNS Operators for DNSSEC signed Zones", draft-koch-dnsop-dnssec-operator-change-06 (work in progress), February 2014.

[RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, February 2015.

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 Mevzeck;
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-eppeext-keyrelay-00

1. Added feedback from Seth Goldman and put him in the acknowledgement section.
2. IDnits formatting adjustments

A.6. draft-ietf-eppeext-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-eppeext-keyrelay-02

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

A.8. draft-ietf-eppeext-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-eppeext-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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: May 5, 2016

J. Gould
VeriSign, Inc.
W. Tan
Cloud Registry
G. Brown
CentralNic Ltd
November 2, 2015

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

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. REngin v3.7	52
6.5. RegistryEngine EPP Service	52
6.6. Neustar EPP SDK	53

6.7. gTLD Shared Registry System	53
7. Security Considerations	54
8. Acknowledgements	54
9. Normative References	54
Appendix A. Change History	55
A.1. Change from 00 to 01	55
A.2. Change from 01 to 02	55
A.3. Change from 02 to 03	56
A.4. Change from 03 to 04	56
A.5. Change from 04 to 05	56
A.6. Change from 05 to 06	57
A.7. Change from 06 to 07	57
A.8. Change from 07 to 08	57
A.9. Change from 08 to 09	57
A.10. Change from 09 to 10	58
A.11. Change from 10 to 11	59
A.12. Change from 11 to 12	59
A.13. Change from 12 to WG 00	59
A.14. Change WG 00 to WG 01	59
A.15. Change WG 01 to WG 02	59
A.16. Change WG 02 to WG 03	60
A.17. Change WG 03 to WG 04	60
A.18. Change WG 04 to WG 05	60
A.19. Change WG 05 to WG 06	60
A.20. Change WG 06 to WG 07	60
Authors' Addresses	61

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

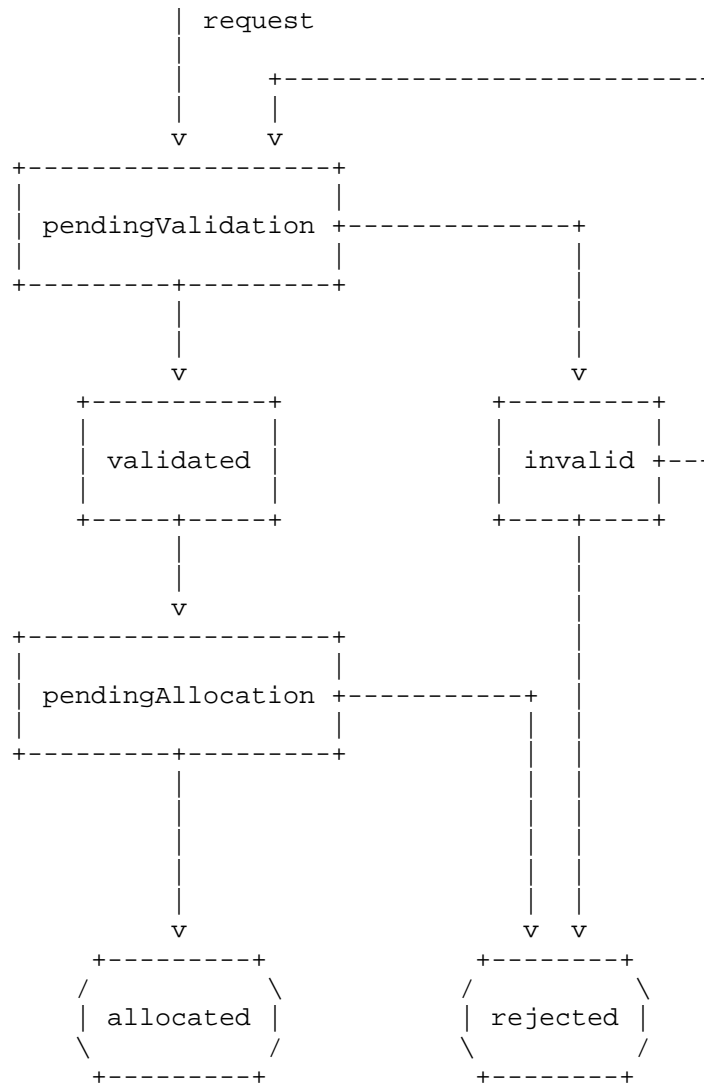


Figure 1

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.

```
<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>
```

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:cd>
S:          <launch:name exists="1">domain2.example</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJlNrDO92vDsAzf7EQzgjX4R0000000001
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/rJlNrDO92vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:          <launch:claimKey validatorID="custom-tmch">
S:            20140423200/1/2/3/rJlNr2vDsAzasdff7EasdfgjX4R0000000002
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>

response. This form MUST be identified by setting the <launch:check> "type" attribute to "avail".

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:

```
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:

```
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="trademark"/>
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: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:

```
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: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/rJlNrDO92vDsAzf7EQzgJX4R0000000001
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/rJlNrDO92vDsAzf7EQzgJX4R0000000001
S:          </launch:claimKey>
S:          <launch:claimKey validatorID="custom-tmch">
S:            20140423200/1/2/3/rJlNr2vDsAzsdff7EasdfgjX4R0000000002
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>
```

3.2. 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":

```
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:        includeMark="true">
C:          <launch:phase>sunrise</launch:phase>
C:          <launch:applicationID>abc123</launch:applicationID>
C:        </launch:info>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</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:

```
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>jdl234</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">
C:            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">
C:              49FD46E6C4B45C55D4AE</launch:code>
C:            </launch:codeMark>
C:          </launch:codeMark>
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 "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
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:        <launch:phase>sunrise</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: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 "code with mark" validation model, with a code and 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
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domain.example</domain:name>
C:        <domain:registrant>jdl1234</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="sample">
C:            49FD46E6C4B45C55D4AC</launch:code>
C:          <mark:mark
C:            xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
C:            ...
C:          </mark:mark>
C:        </launch:codeMark>
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 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>jdl234</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">
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>jdl234</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:        <smd:encodedSignedMark
C:          xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
C:          ...
C:        </smd:encodedSignedMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</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:registrar>jdl1234</domain:registrar>
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>claims</launch:phase>
C:        <launch:notice>
C:          <launch:noticeID validatorID="tmch">
C:            370d0b7c9223372036854775807</launch:noticeID>
C:          <launch:notAfter>2014-06-19T10:00:00.0Z
C:          </launch:notAfter>
C:          <launch:acceptedDate>2014-06-19T09:00:00.0Z
C:          </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
C:          </launch:notAfter>
C:          <launch:acceptedDate>2014-06-19T09:00:30.0Z
C:          </launch:acceptedDate>
C:        </launch:notice>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
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:

```
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:        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
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domainone.example</domain:name>
C:        <domain:registrar>jdl1234</domain:registrar>
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 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:notice>
C:          <launch:noticeID validatorID="tmch">
C:            49FD46E6C4B45C55D4AC
C:          </launch:noticeID>
C:          <launch:notAfter>2012-06-19T10:00:10.0Z
C:          </launch:notAfter>
C:          <launch:acceptedDate>2012-06-19T09:01:30.0Z
C:          </launch:acceptedDate>
C:        </launch:notice>
C:      </launch:create>
C:    </extension>
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:

```
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
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:ns>
C:          </domain:add>
C:          <domain:rem>
C:            <domain:ns>
C:              <domain:hostObj>ns1.domain.example</domain:hostObj>
C:            </domain:ns>
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:

```
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
C:        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
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: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:

- o Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- o 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.
- o 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 &quot;custom&quot;, 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>
      </simpleContent>
    </complexType>

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

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

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

    <!--
    Possible status values for sunrise application
    -->
    <simpleType name="statusValueType">
      <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>
    </simpleType>

    <!--
    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>

<!--
codeMark Type that contains an optional code
with mark information.
-->
<complexType name="codeMarkType">
  <sequence>
    <element name="code" type="launch:codeType"
      minOccurs="0"/>
    <element ref="mark:abstractMark"
      minOccurs="0"/>
  </sequence>
</complexType>

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

<!--
```

```
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)
-->
<simpleType name="checkFormType">
  <restriction base="token">
    <enumeration value="claims"/>
    <enumeration value="avail"/>
    <enumeration value="trademark"/>
  </restriction>
</simpleType>
```

```
<!--
Child elements of info command.
-->
<complexType name="infoType">
  <sequence>
    <element name="phase" type="launch:phaseType"/>
    <element name="applicationID"
      type="launch:applicationIDType"
      minOccurs="0"/>
  </sequence>
  <attribute name="includeMark" type="boolean"
    default="false"/>
</complexType>

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

<!--
<check> response elements.
-->
<complexType name="chkDataType">
  <sequence>
    <element name="phase" type="launch:phaseType"
      minOccurs="0"/>
    <element name="cd" type="launch:cdType"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>

<complexType name="cdType">
  <sequence>
    <element name="name" type="launch:cdNameType"/>
    <element name="claimKey" type="launch:claimKeyType"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<complexType name="cdNameType">
  <simpleContent>
    <extension base="eppcom:labelType">
      <attribute name="exists" type="boolean"
        use="required"/>
    </extension>
  </simpleContent>
```

```
</complexType>

<complexType name="claimKeyType">
  <simpleContent>
    <extension base="token">
      <attribute name="validatorID"
        type="launch:validatorIDType" use="optional"/>
    </extension>
  </simpleContent>
</complexType>

<!--
<info> response elements
-->
<complexType name="infDataType">
  <sequence>
    <element name="phase" type="launch:phaseType"/>
    <element name="applicationID"
      type="launch:applicationIDType"
      minOccurs="0"/>
    <element name="status" type="launch:statusType"
      minOccurs="0"/>
    <element ref="mark:abstractMark"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

</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].

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

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: "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-epext-launchphase.

Level of maturity: Production

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

URL: http://www.verisigninc.com/en_US/channel-resources/domain-registry-products/epp-sdks

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-epext-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-epext-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]
Lozano, G., "TMCH functional specifications", draft-ietf-eppext-tmch-func-spec-00 (work in progress), October 2015.

- [I-D.ietf-eppext-tmch-smd]
Lozano, G., "Mark and Signed Mark Objects Mapping", draft-ietf-eppext-tmch-smd-03 (work in progress), September 2015.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<http://www.rfc-editor.org/info/rfc5730>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <<http://www.rfc-editor.org/info/rfc5731>>.
- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <<http://www.rfc-editor.org/info/rfc6982>>.
- [RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <<http://www.rfc-editor.org/info/rfc7451>>.

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.
4. Added missing description of the <mark:pc> element in the <mark:addr> element.
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 poll 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 "extended" 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

1. Changed reference of lozano-tmch-func-spec to ietf-eppext-tmch-func-spec.

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>

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016

G. Lozano
ICANN
March 21, 2016

ICANN TMCH functional specifications
draft-ietf-eppext-tmch-func-spec-01

Abstract

This document describes the requirements, the architecture and the interfaces between the ICANN Trademark Clearinghouse (TMCH) and Domain Name Registries as well as between the ICANN TMCH and Domain Name Registrars for the provisioning and management of domain names during Sunrise and Trademark Claims Periods.

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 22, 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
2. Terminology	4
3. Glossary	4
4. Architecture	9
4.1. Sunrise Period	9
4.2. Trademark Claims Period	10
4.3. Interfaces	10
4.3.1. hv	10
4.3.2. vd	11
4.3.3. dy	11
4.3.4. tr	11
4.3.5. ry	11
4.3.6. dr	11
4.3.7. yd	11
4.3.8. dv	12
4.3.9. vh	12
4.3.10. vs	12
4.3.11. sy	12
4.3.12. sr	12
4.3.13. vc	13
4.3.14. cy	13
4.3.15. cr	13
5. Process Descriptions	13
5.1. Bootstrapping	13
5.1.1. Bootstrapping for Registries	13
5.1.1.1. Credentials	13
5.1.1.2. IP Addresses for Access Control	14
5.1.1.3. ICANN TMCH Trust Anchor	14
5.1.1.4. TMDB PGP Key	14
5.1.2. Bootstrapping for Registrars	15
5.1.2.1. Credentials	15
5.1.2.2. IP Addresses for Access Control	15
5.1.2.3. ICANN TMCH Trust Anchor	15
5.1.2.4. TMDB PGP Key	15
5.2. Sunrise Period	16
5.2.1. Domain Name registration	16
5.2.2. Sunrise Domain Name registration by Registries	17
5.2.3. TMDB Sunrise Services for Registries	18
5.2.3.1. SMD Revocation List	18
5.2.3.2. TMV Certificate Revocation List (CRL)	18
5.2.3.3. Notice of Registered Domain Names (NORN)	19
5.2.4. Sunrise Domain Name registration by Registrars	22
5.2.5. TMDB Sunrise Services for Registrars	22
5.3. Trademark Claims Period	23
5.3.1. Domain Registration	23
5.3.2. Trademark Claims Domain Name registration by	

Registries	24
5.3.3. TMBD Trademark Claims Services for Registries	25
5.3.3.1. Domain Name Label (DNL) List	25
5.3.3.2. Notice of Registered Domain Names (NORN)	26
5.3.4. Trademark Claims Domain Name registration by Registrars	26
5.3.5. TMBD Trademark Claims Services for Registrars	27
5.3.5.1. Claims Notice Information Service (CNIS)	27
5.4. Qualified Launch Program (QLP) Period	28
5.4.1. Domain Registration	28
5.4.2. TMBD QLP Services for Registries	30
5.4.2.1. Sunrise List (SURL)	30
6. Data Format Descriptions	30
6.1. Domain Name Label (DNL) List	30
6.2. SMD Revocation List	32
6.3. List of Registered Domain Names (LORDN) file	34
6.3.1. LORDN Log file	39
6.3.1.1. LORDN Log Result Codes	41
6.4. Signed Mark Data (SMD) File	45
6.5. Trademark Claims Notice (TCN)	46
6.6. Sunrise List (SURL)	53
7. Formal Syntax	54
7.1. Trademark Claims Notice (TCN)	54
8. Acknowledgements	57
9. IANA Considerations	57
10. Security Considerations	57
11. References	58
11.1. Normative References	58
11.2. Informative References	58
Author's Address	60

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).

Along with the introduction of new generic TLDs (gTLD), two special modes came into effect:

- o Sunrise Period, the Sunrise Period allows trademark holders an advance opportunity to register domain names corresponding to their marks before names are generally available to the public.
- o Trademark Claims Period, the Trademark Claims Period follows the Sunrise Period and runs for at least the first 90 days of an initial operating period of general registration. During the Trademark Claims Period, anyone attempting to register a domain

name matching a mark that is recorded in the ICANN Trademark Clearinghouse (TMCH) will receive a notification displaying the relevant mark information.

This document describes the requirements, the architecture and the interfaces between the ICANN TMCH and Domain Name Registries (called Registries in the rest of the document) as well as between the ICANN TMCH and Domain Name Registrars (called Registrars in the rest of the document) for the provisioning and management of domain names during the Sunrise and Trademark Claims Periods.

For any date and/or time indications, Coordinated Universal Time (UTC) applies.

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].

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.

"tmNotice-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:tmNotice-1.0". The XML namespace prefix "tmNotice" 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. Glossary

In the following section, the most common terms are briefly explained:

- o Backend Registry Operator: Entity that manages (a part of) the technical infrastructure for a Registry Operator. The Registry Operator may also be the Backend Registry Operator.
- o CA: Certificate Authority, see [RFC5280].
- o CNIS, Claims Notice Information Service: This service provides Trademark Claims Notices (TCN) to Registrars.
- o CRC32, Cyclic Redundancy Check: algorithm used in the ISO 3309 standard and in section 8.1.1.6.2 of ITU-T recommendation V.42.

- o CRL: Certificate Revocation List, see [RFC5280].
- o CSV: Comma-Separated Values, see [RFC4180]
- o Date and time, datetime: Date and time are specified following the standard "Date and Time on the Internet specification", see [RFC3339].
- o DN, Domain Name, domain name: see definition of Domain name in [RFC7719].
- o DNROID, DN Repository Object IDentifier: an identifier assigned by the Registry to each DN object that unequivocally identifies said DN object. For example, if a new DN object is created for a name that existed in the past, the DN objects will have different DNRoids.
- o DNL, Domain Name Label, see definition of Label in [RFC7719].
- o DNL List: A list of DNLs that are covered by a PRM.
- o DNS: Domain Name System, see [RFC7719].
- o Effective allocation: A DN is considered effectively allocated when the DN object for the DN has been created in the SRS of the Registry and has been assigned to the effective user. A DN object in status "pendingCreate" or any other status that precedes the first time a DN is assigned to an end-user is not considered an effective allocation. A DN object created internally by the Registry for subsequent delegation to another Registrant is not considered an effective allocation.
- o EPP: The Extensible Provisioning Protocol, see definition of EPP in [RFC7719].
- o FQDN: Fully-Qualified Domain Name, see definition of FQDN in [RFC7719].
- o HTTP: Hypertext Transfer Protocol, see [RFC2616]
- o HTTPS: HTTP over TLS (Transport Layer Security), [RFC2818].
- o IDN: Internationalized Domain Name, see definition of IDN in [RFC7719].
- o Lookup Key: A random string of up to 51 chars from the set [a-zA-Z0-9/] to be used as the lookup key by Registrars to obtain the

TCN using the CNIS. Lookup Keys are unique and are related to one DNL only.

- o LORDN, List of Registered Domain Names: This is the list of effectively allocated DNS matching a DNL of a PRM. Registries will upload this list to the TMDB (during the NORDN process).
- o NORDN, Notification of Registered Domain Names: The process by which Registries upload their recent LORDN to the TMDB.
- o PGP: Pretty Good Privacy, see [RFC4880]
- o PKI: Public Key Infrastructure, see [RFC5280].
- o PRM, Pre-registered mark: Mark that has been pre-registered with the ICANN TMCH.
- o QLP Period, Qualified Launch Program Period: During this optional period, a special process applies to DNS matching the Sunrise List (SURL) and/or the DNL List, to ensure that TMHs are informed of a DN matching their PRM.
- o Registrant: see definition of Registrant in [RFC7719].
- o Registrar, Domain Name Registrar: see definition of Registrar in [RFC7719].
- o Registry, Domain Name Registry, Registry Operator: see definition of Registry in [RFC7719]. A Registry Operator is the contracting party with ICANN for the TLD.
- o SMD, Signed Mark Data: A cryptographically signed token issued by the TMV to the TMH to be used in the Sunrise Period to apply for a DN that matches a DNL of a PRM; see also [I-D.ietf-epext-tmch-smd]. An SMD generated by an ICANN-approved trademark validator (TMV) contains both the signed token and the TMV's PKIX certificate.
- o SMD File: A file containing the SMD (see above) and some human readable data. The latter is usually ignored in the processing of the SMD File. See also Section 6.4.
- o SMD Revocation List: The SMD Revocation List is used by Registries (and optionally by Registrars) during the Sunrise Period to ensure that an SMD is still valid (i.e. not revoked). The SMD Revocation List has a similar function as CRLs used in PKI.

- o SRS: Shared Registration System, see also [ICANN-GTLD-AGB-20120604].
- o SURL, Sunrise List: The list of DNLs that are covered by a PRM and eligible for Sunrise.
- o Sunrise Period: During this period DNs matching a DNL of a PRM can be exclusively obtained by the respective TMHs. For DNs matching a PRM, a special process applies to ensure that TMHs are informed on the effective allocation of a DN matching their PRM.
- o TLD: Top-Level Domain Name, see definition of TLD in [RFC7719].
- o ICANN TMCH: a central repository for information to be authenticated, stored, and disseminated, pertaining to the rights of TMHs. The ICANN TMCH is split into two functions TMV and TMDB (see below). There could be several entities performing the TMV function, but only one entity performing the TMDB function.
- o ICANN TMCH-CA: The Certificate Authority (CA) for the ICANN TMCH. This CA is operated by ICANN. The public key for this CA is the trust anchor used to validate the identity of each TMV.
- o TMDB, Trademark Clearinghouse Database: Serves as a database of the ICANN TMCH to provide information to the gTLD Registries and Registrars to support Sunrise or Trademark Claims services. There is only one TMDB in the ICANN TMCH that concentrates the information about the "verified" Trademark records from the TMVs.
- o TMH, Trademark Holder: The person or organization owning rights on a mark.
- o TMV, Trademark Validator, Trademark validation organization: An entity authorized by ICANN to authenticate and validate registrations in the TMDB ensuring the marks qualify as registered or are court-validated marks or marks that are protected by statute or treaty. This entity would also be asked to ensure that proof of use of marks is provided, which can be demonstrated by furnishing a signed declaration and one specimen of current use.
- o Trademark, mark: Marks are used to claim exclusive properties of products or services. A mark is typically a name, word, phrase, logo, symbol, design, image, or a combination of these elements. For the scope of this document only textual marks are relevant.
- o Trademark Claims, Claims: Provides information to enhance the understanding of the Trademark rights being claimed by the TMH.

- o TCN, Trademark Claims Notice, Claims Notice, Trademark Notice: A Trademark Claims Notice consist of one or more Trademark Claims and are provided to prospective Registrants of DNs.
- o TCNID, Trademark Claims Notice Identifier: An element of the Trademark Claims Notice (see above), identifying said TCN. The Trademark Claims Notice Identifier is specified in the element <tmNotice:id>.
- o Trademark Claims Period: During this period, a special process applies to DNs matching the DNL List, to ensure that TMHs are informed of a DN matching their PRM. For DNs matching the DNL List, Registrars show a TCN to prospective Registrants that has to be acknowledged before effective allocation of the DN.
- o UTC: Coordinated Universal Time, as maintained by the Bureau International des Poids et Mesures (BIPM); see also [RFC3339].

4. Architecture

4.1. Sunrise Period

Architecture of the Sunrise Period

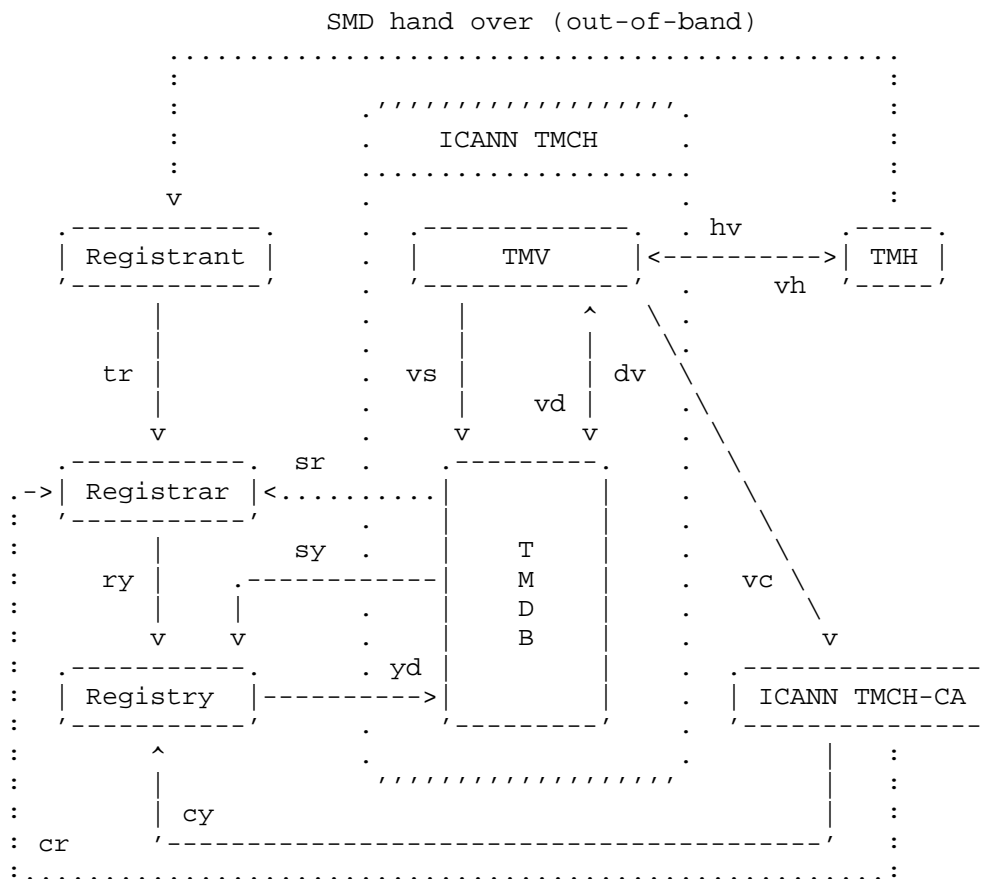


Figure 1

4.2. Trademark Claims Period

Architecture of the Trademark Claims Period

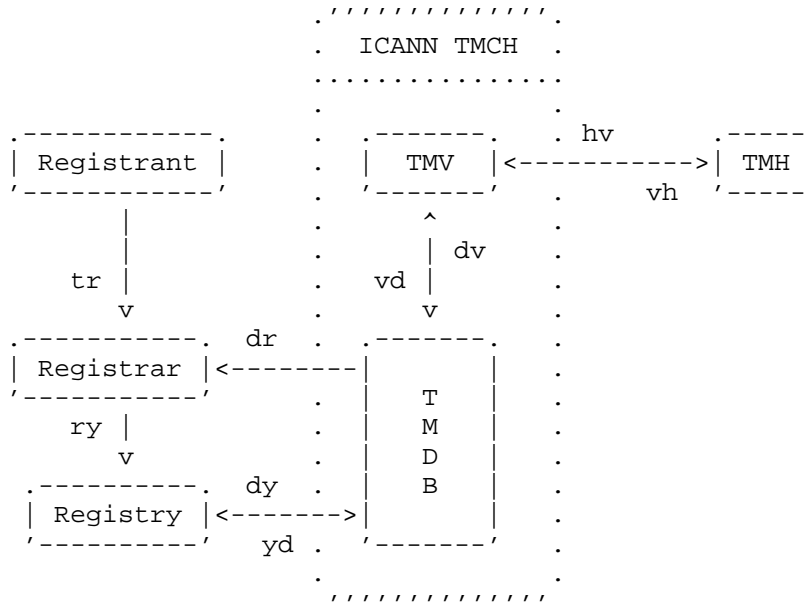


Figure 2

4.3. Interfaces

In the sub-sections below follows a short description of each interface to provide an overview of the architecture. More detailed descriptions of the relevant interfaces follow further below (Section 5).

4.3.1. hv

The TMH registers a mark with a TMV via the hv interface.

After the successful registration of the mark, the TMV makes available a SMD File (see also Section 6.4) to the TMH to be used during the Sunrise Period.

The specifics of the hv interface are beyond the scope of this document.

4.3.2. vd

After successful mark registration, the TMV ensures the TMDB inserts the corresponding DNLs and mark information into the database via the vd interface.

The specifics of the vd interface are beyond the scope of this document.

4.3.3. dy

During the Trademark Claims Period the Registry fetches the latest DNL List from the TMDB via the dy interface at regular intervals. The protocol used on the dy interface is HTTPS.

Not relevant during the Sunrise Period.

4.3.4. tr

The Registrant communicates with the Registrar via the tr interface.

The specifics of the tr interface are beyond the scope of this document.

4.3.5. ry

The Registrar communicate with the Registry via the ry interface. The ry interfaces is typically implemented in EPP.

4.3.6. dr

During the Trademark Claims Period, the Registrar fetches the TCN from the TMDB (to be displayed to the Registrant via the tr interface) via the dr interface. The protocol used for fetching the TCN is HTTPS.

Not relevant during the Sunrise Period.

4.3.7. yd

During the Sunrise Period the Registry notifies the TMDB via the yd interface of all DNS effectively allocated.

During the Trademark Claims Period, the Registry notifies the TMDB via the yd interface of all DNS effectively allocated that matched an entry in the Registry previously downloaded DNL List during the creation of the DN.

The protocol used on the yd interface is HTTPS.

4.3.8. dv

The TMDB notifies via the dv interface to the TMV of all DNSs effectively allocated that match a mark registered by that TMV.

The specifics of the dv interface are beyond the scope of this document.

4.3.9. vh

The TMV notifies the TMH via the vh interface after a DN has been effectively allocated that matches a PRM of this TMH.

The specifics of the vh interface are beyond the scope of this document.

4.3.10. vs

The TMV requests to add a revoked SMD to the SMD Revocation List at the TMDB.

The specifics of the vs interface are beyond the scope of this document.

Not relevant during the Trademark Claims Period.

4.3.11. sy

During the Sunrise Period the Registry fetches the most recent SMD Revocation List from the TMDB via the sy interface in regular intervals. The protocol used on the sy interface is HTTPS.

Not relevant during the Trademark Claims Period.

4.3.12. sr

During the Sunrise Period the Registrar may fetch the most recent SMD Revocation List from the TMDB via the sr interface. The protocol used on the sr interface is the same as on the sy interface (s. above), i.e. HTTPS.

Not relevant during the Trademark Claims Period.

4.3.13. vc

The TMV registers its public key, and requests to revoke an existing key, with the ICANN TMCH-CA over the vc interface.

The specifics of the vc interface are beyond the scope of this document, but it involves personal communication between the operators of the TMV and the operators of the ICANN TMCH-CA.

Not relevant during the Trademark Claims Period.

4.3.14. cy

During the Sunrise Period the Registry fetches the most recent TMV CRL file from the ICANN TMCH-CA via the cy interface at regular intervals. The TMV CRL is used for validation of TMV certificates. The protocol used on the cy interface is HTTPS.

Not relevant during the Trademark Claims Period.

4.3.15. cr

During the Sunrise Period the Registrar optionally fetches the most recent TMV CRL file from the ICANN TMCH-CA via the cr interface at regular intervals. The TMV CRL is used for validation of TMV certificates. The protocol used on the cr interface is HTTPS.

Not relevant during the Trademark Claims Period.

5. Process Descriptions

5.1. Bootstrapping

5.1.1. Bootstrapping for Registries

5.1.1.1. Credentials

Each Registry Operator will receive authentication credentials from the TMDB to be used:

- o During the Sunrise Period to fetch the SMD Revocation List from the TMDB via the sy interface (Section 4.3.11).
- o During the Trademark Claims Period to fetch the DNL List from the TMDB via the dy interface (Section 4.3.3).
- o During the NORDN process to notify the LORDN to the TMDB via the yd interface (Section 4.3.7).

Note: credentials are created per TLD and provided to the Registry Operator.

5.1.1.2. IP Addresses for Access Control

Each Registry Operator MUST provide to the TMDB all IP addresses that will be used to:

- o Fetch the SMD Revocation List via the sy interface (Section 4.3.11).
- o Fetch the DNL List from the TMDB via the dy interface (Section 4.3.3).
- o Upload the LORDN to the TMDB via the yd interface (Section 4.3.7).

This access restriction MAY be applied by the TMDB in addition to HTTP Basic access authentication (for credentials to be used, see Section 5.1.1.1).

The TMDB MAY limit the number of IP addresses to be accepted per Registry Operator.

5.1.1.3. ICANN TMCH Trust Anchor

Each Registry Operator MUST fetch the PKIX certificate ([RFC5280]) of the ICANN TMCH-CA (Trust Anchor) from < <https://ca.icann.org/tmch.crt> > to be used:

- o During the Sunrise Period to validate the TMV certificates and the TMV CRL.

5.1.1.4. TMDB PGP Key

The TMDB MUST provide each Registry Operator with the public portion of the PGP Key used by TMDB, to be used:

- o During the Sunrise Period to perform integrity checking of the SMD Revocation List fetched from the TMDB via the sy interface (Section 4.3.11).
- o During the Trademark Claims Period to perform integrity checking of the DNL List fetched from the TMDB via the dy interface (Section 4.3.3).

5.1.2. Bootstrapping for Registrars

5.1.2.1. Credentials

Each ICANN-accredited Registrar will receive authentication credentials from the TMDB to be used:

- o During the Sunrise Period to (optionally) fetch the SMD Revocation List from the TMDB via the sr interface (Section 4.3.12).
- o During the Trademark Claims Period to fetch TCNs from the TMDB via the dr interface (Section 4.3.6).

5.1.2.2. IP Addresses for Access Control

Each Registrar MUST provide to the TMDB all IP addresses, which will be used to:

- o Fetch the SMD Revocation List via the sr interface (Section 4.3.12).
- o Fetch TCNs via the dr interface (Section 4.3.6).

This access restriction MAY be applied by the TMDB in addition to HTTP Basic access authentication (for credentials to be used, see Section 5.1.2.1).

The TMDB MAY limit the number of IP addresses to be accepted per Registrar.

5.1.2.3. ICANN TMCH Trust Anchor

Registrars MAY fetch the PKIX certificate of the ICANN TMCH-CA (Trust Anchor) from < <https://ca.icann.org/tmch.crt> > to be used:

- o During the Sunrise Period to (optionally) validate the TMV certificates and TMV CRL.

5.1.2.4. TMDB PGP Key

Registrars MUST receive the public portion of the PGP Key used by TMDB from the TMDB administrator to be used:

- o During the Sunrise Period to (optionally) perform integrity checking of the SMD Revocation List fetched from the TMDB via the sr interface (Section 4.3.12).

5.2. Sunrise Period

5.2.1. Domain Name registration

Domain Name registration during the Sunrise Period

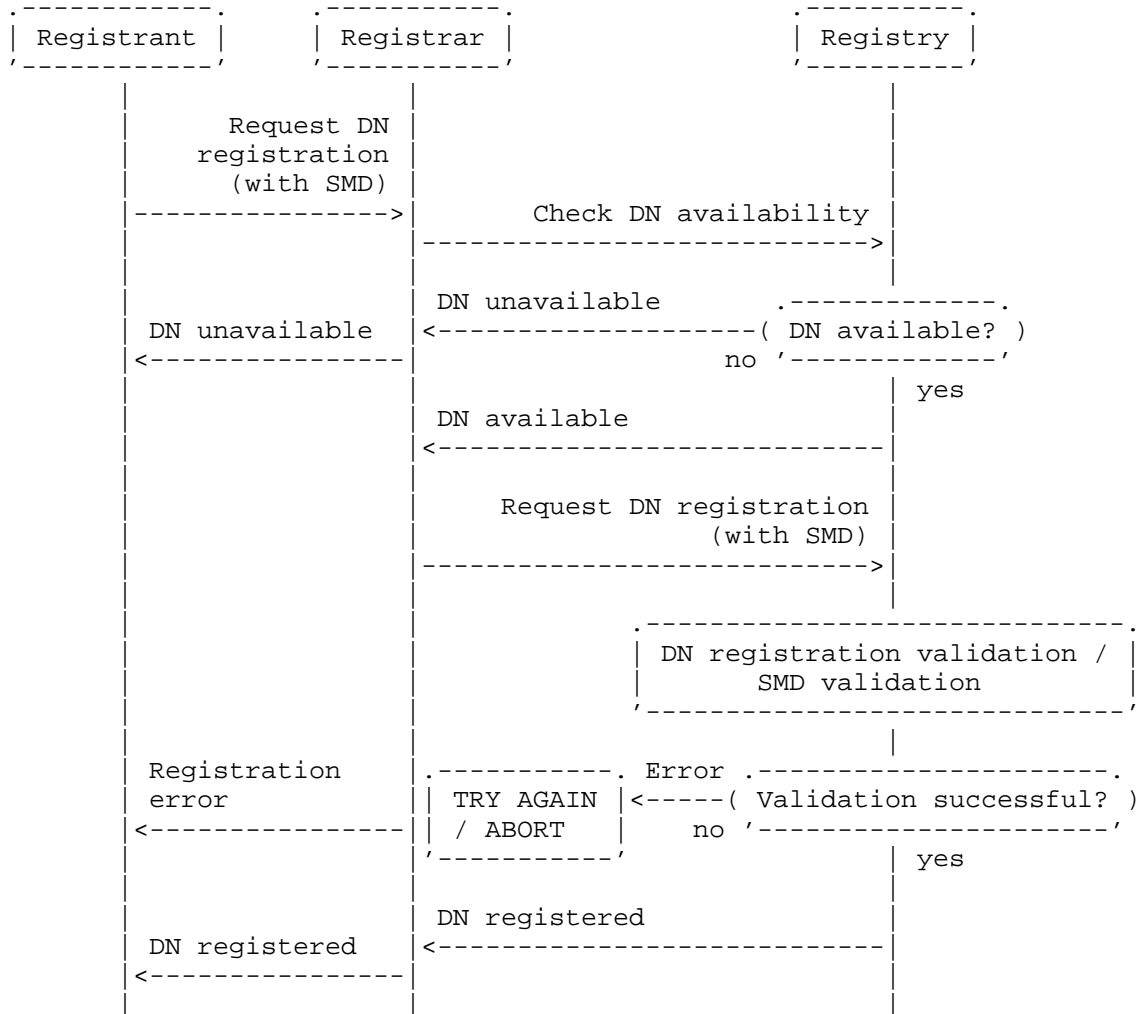


Figure 3

Note: the figure depicted above represents a synchronous DN registration workflow (usually called first come first served).

5.2.2. Sunrise Domain Name registration by Registries

Registries MUST perform a minimum set of checks for verifying each DN registration during the Sunrise Period upon reception of a registration request over the ry interface (Section 4.3.5). If any of these checks fails the Registry MUST abort the registration. Each of these checks MUST be performed before the DN is effectively allocated.

In case of asynchronous registrations (e.g. auctions), the minimum set of checks MAY be performed when creating the intermediate object (e.g. a DN application) used for DN registration. If the minimum set of checks is performed when creating the intermediate object (e.g. a DN application) a Registry MAY effectively allocate the DN without performing the minimum set of checks again.

Performing the minimum set of checks Registries MUST verify that:

1. An SMD has been received from the Registrar along with the DN registration.
2. The certificate of the TMV has been correctly signed by the ICANN TMCH-CA. (The certificate of the TMV is contained within the SMD.)
3. The datetime when the validation is done is within the validity period of the TMV certificate.
4. The certificate of the TMV is not listed in the TMV CRL file specified in the CRL distribution point of the TMV certificate.
5. The signature of the SMD (signed with the TMV certificate) is valid.
6. The datetime when the validation is done is within the validity period of the SMD based on <smd:notBefore> and <smd:notAfter> elements.
7. The SMD has not been revoked, i.e., is not contained in the SMD Revocation List.
8. The leftmost DNL (A-label in case of IDNs) of the DN being effectively allocated matches one of the labels (<mark:label>) elements in the SMD.

These procedure apply to all DN effective allocations at the second level as well as to all other levels subordinate to the TLD that the Registry accepts registrations for.

5.2.3. TMDB Sunrise Services for Registries

5.2.3.1. SMD Revocation List

A new SMD Revocation List MUST be published by the TMDB twice a day, by 00:00:00 and 12:00:00 UTC.

Registries MUST refresh the latest version of the SMD Revocation List at least once every 24 hours.

Note: the SMD Revocation List will be the same regardless of the TLD. If a Backend Registry Operator manages the infrastructure of several TLDs, the Backend Registry Operator could refresh the SMD Revocation List once every 24 hours, the SMD Revocation List could be used for all the TLDs managed by the Backend Registry Operator.

Update of the SMD Revocation List

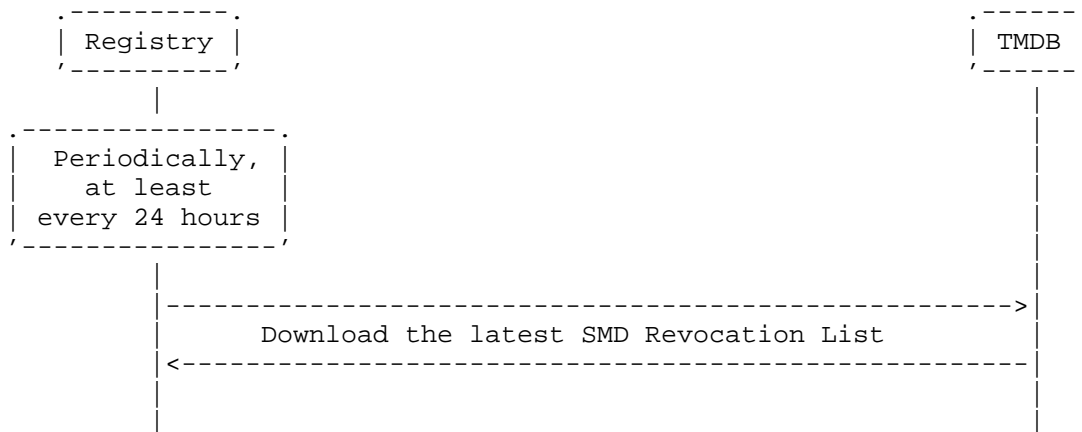


Figure 4

5.2.3.2. TMV Certificate Revocation List (CRL)

Registries MUST refresh their local copy of the TMV CRL file at least every 24 hours using the CRL distribution point specified in the TMV certificate.

Operationally, the TMV CRL file and CRL distribution point is the same for all TMVs and (at publication of this document) located at < <http://crl.icann.org/tmch.crl> >.

Note: the TMV CRL file will be the same regardless of the TLD. If a Backend Registry Operator manages the infrastructure of several TLDs, the Backend Registry Operator could refresh the TMV CRL file once every 24 hours, the TMV CRL file could be used for all the TLDs managed by the Backend Registry Operator.

Update of the TMV CRL file

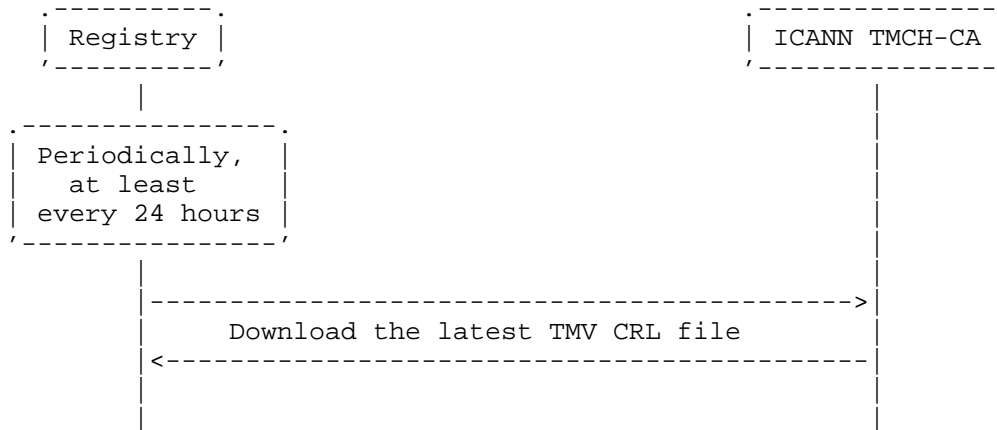


Figure 5

5.2.3.3. Notice of Registered Domain Names (NORN)

The Registry MUST send a LORDN file containing DNS effectively allocated to the TMDB (over the yd interface, Section 4.3.7).

The effective allocation of a DN MUST be reported by the Registry to the TMDB within 26 hours of the effective allocation of such DN.

The Registry MUST create and upload a LORDN file in case there are effective allocations in the SRS, that have not been successfully reported to the TMDB in a previous LORDN file.

Based on the timers used by TMVs and the TMDB, the RECOMMENDED maximum frequency to upload LORDN files from the Registries to the TMDB is every 3 hours.

It is RECOMMENDED that Registries try to upload at least two LORDN files per day to the TMDB with enough time in between, in order to have time to fix problems reported in the LORDN file.

The Registry SHOULD upload a LORDN file only when the previous LORDN file has been processed by the TMDB and the related LORDN Log file has been downloaded and processed by the Registry.

The Registry MUST upload LORDN files for DNSs effectively allocated during the Sunrise or Trademark Claims Period (same applies to DNSs effectively allocated using applications created during the Sunrise or Trademark Claims Period in case of using asynchronous registrations).

The yd interface (Section 4.3.7) MUST support at least one (1) and MAY support up to ten (10) concurrent connections from each IP address registered by a Registry Operator to access the service.

The TMDB MUST process each uploaded LORDN file and make the related log file available for Registry download within 30 minutes of the finalization of the upload.

Notification of Registered Domain Name

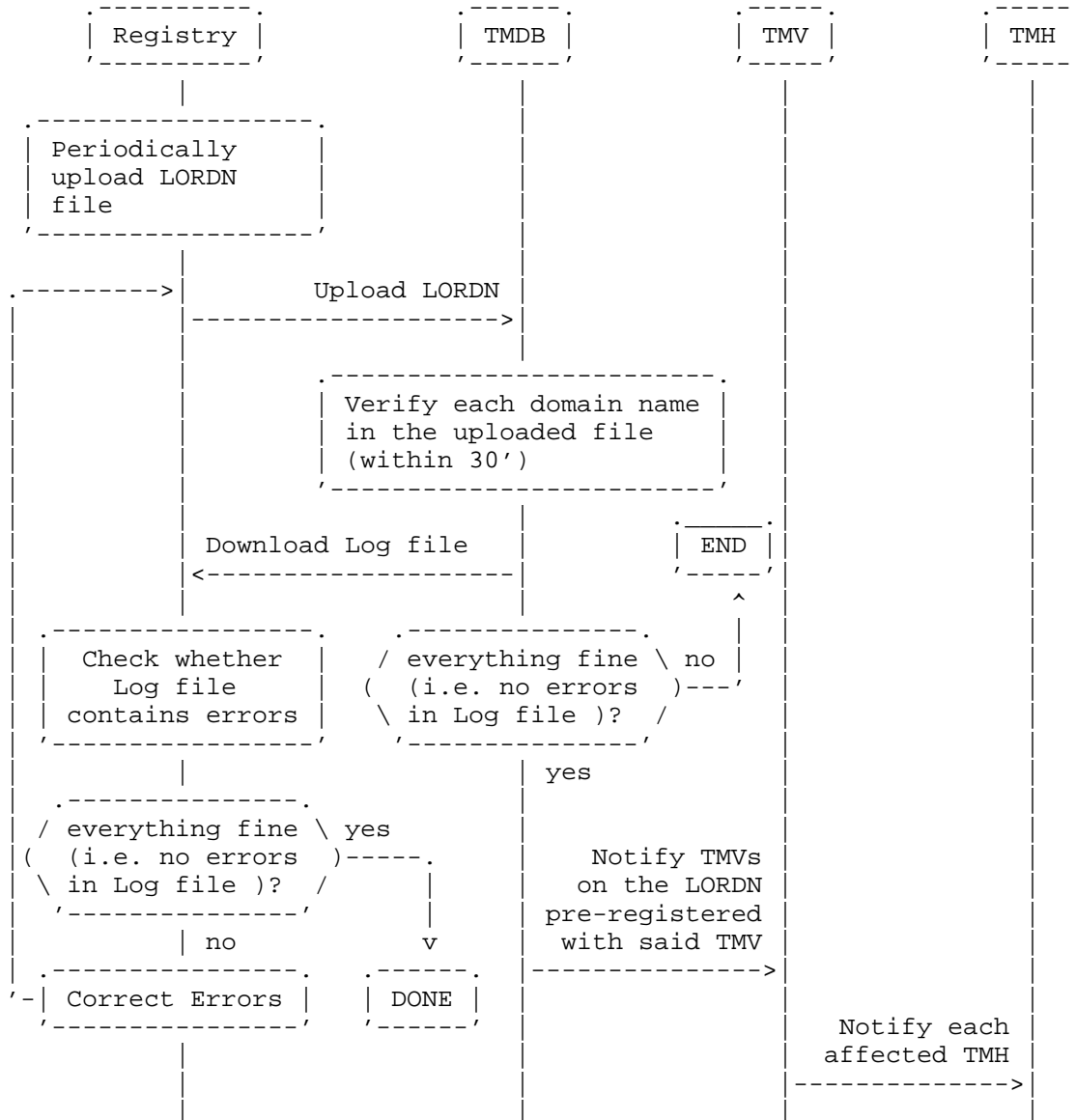


Figure 6

The format used for the LORDN is described in Section 6.3

5.2.4. Sunrise Domain Name registration by Registrars

Registrars MAY choose to perform the checks for verifying DN registrations as performed by the Registries (see Section 5.2.2) before sending the command to register a DN.

5.2.5. TMDB Sunrise Services for Registrars

The processes described in Section 5.2.3.1 and Section 5.2.3.2 are also available for Registrars to optionally validate the SMDs received.

5.3. Trademark Claims Period

5.3.1. Domain Registration

Domain Name registration during the Trademark Claims Period

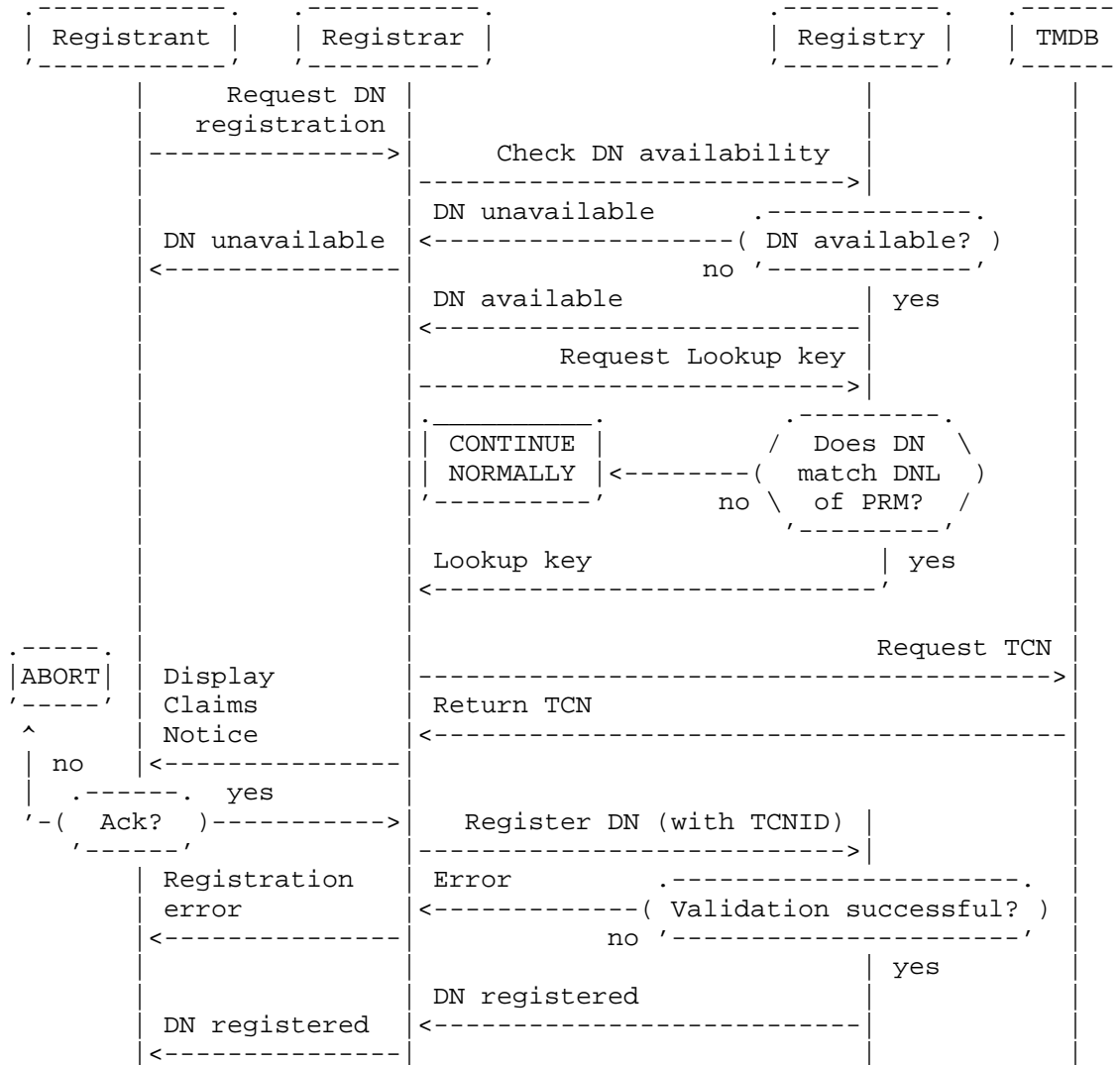


Figure 7

Note: the figure depicted above represents a synchronous DN registration workflow (usually called first come first served).

5.3.2. Trademark Claims Domain Name registration by Registries

During the Trademark Claims Period, Registries perform two main functions:

- o Registries MUST provide Registrars (over the ry interface, Section 4.3.5) the Lookup Key used to retrieve the TCNs for DNs that match the DNL List.
- o Registries MUST provide the Lookup Key only when queried about a specific DN.
- o For each DN matching a DNL of a PRM, Registries MUST perform a minimum set of checks for verifying DN registrations during the Trademark Claims Period upon reception of a registration request over the ry interface (Section 4.3.5). If any of these checks fails the Registry MUST abort the registration. Each of these checks MUST be performed before the DN is effectively allocated.
- o In case of asynchronous registrations (e.g. auctions), the minimum set of checks MAY be performed when creating the intermediate object (e.g. a DN application) used for DN effective allocation. If the minimum set of checks is performed when creating the intermediate object (e.g. a DN application) a Registry MAY effectively allocate the DN without performing the minimum set of checks again.
- o Performing the minimum set of checks Registries MUST verify that:
 1. The TCNID (<tmNotice:id>), expiration datetime (<tmNotice:notAfter>) and acceptance datetime of the TCN, have been received from the Registrar along with the DN registration.

If the three elements mentioned above are not provided by the Registrar for a DN matching a DNL of a PRM, but the DNL was inserted (or re-inserted) for the first time into DNL List less than 24 hours ago, the registration MAY continue without this data and the tests listed below are not required to be performed.

2. The TCN has not expired (according to the expiration datetime sent by the Registrar).
3. The acceptance datetime is no more than 48 hours in the past.

4. Using the leftmost DNL (A-label in the case of IDNs) of the DN being registered, the expiration datetime provided by the Registrar, and the TMDB Notice Identifier extracted from the TCNID provided by the Registrar compute the TCN Checksum. Verify that the computed TCN Checksum match the TCN Checksum present in the TCNID.

These procedures apply to all DN registrations at the second level as well as to all other levels subordinate to the TLD that the Registry accepts registrations for.

5.3.3. TMDB Trademark Claims Services for Registries

5.3.3.1. Domain Name Label (DNL) List

A new DNL List MUST be published by the TMDB twice a day, by 00:00:00 and 12:00:00 UTC.

Registries MUST refresh the latest version of the DNL List at least once every 24 hours.

Update of the DNL List

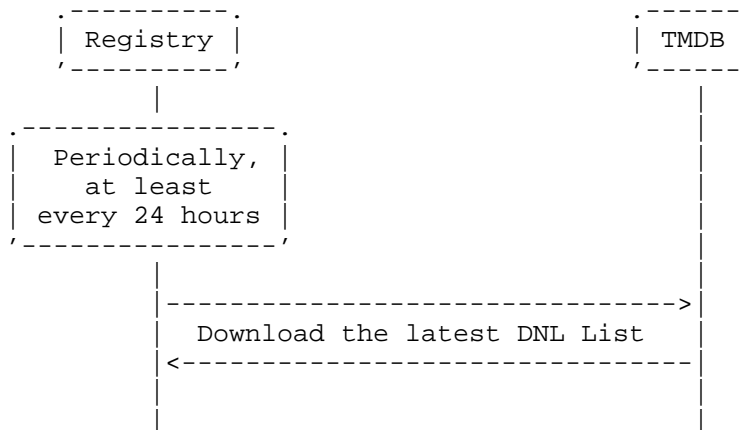


Figure 8

Note: the DNL List will be the same regardless of the TLD. If a Backend Registry Operator manages the infrastructure of several TLDs, the Backend Registry Operator could refresh the DNL List once every 24 hours, the DNL List could be used for all the TLDs managed by the Backend Registry Operator.

5.3.3.2. Notice of Registered Domain Names (NORN)

The NORDN process during the Trademark Claims Period is almost the same as during Sunrise Period as defined in Section 5.2.3.3 with the difference that only registrations subject to a Trademark Claim (i.e., at registration time the name appeared in the current DNL List downloaded by the Registry Operator) are included in the LORDN.

5.3.4. Trademark Claims Domain Name registration by Registrars

For each DN matching a DNL of a PRM, Registrars MUST perform the following steps:

1. Use the Lookup Key received from the Registry to obtain the TCN from the TMDB using the dr interface (Section 4.3.6) Registrars MUST only query for the Lookup Key of a DN that is available for registration.
2. Present the TCN to the Registrant as described in Exhibit A, [RPM-Requirements].
3. Ask Registrant for acknowledgement, i.e. the Registrant MUST consent with the TCN, before any further processing. (The transmission of a TCNID to the Registry over the ry interface, Section 4.3.5 implies that the Registrant has expressed his/her consent with the TCN.)
4. Perform the minimum set of checks for verifying DN registrations. If any of these checks fails the Registrar MUST abort the DN registration. Each of these checks MUST be performed before the registration is sent to the Registry. Performing the minimum set of checks Registrars MUST verify that:
 1. The datetime when the validation is done is within the TCN validity based on the <tmNotice:notBefore> and <tmNotice:notAfter> elements.
 2. The leftmost DNL (A-label in case of IDNs) of the DN being effectively allocated matches the label (<tmNotice:label>) element in the TCN.
 3. The Registrant has acknowledged (expressed his/her consent with) the TCN.
5. Record the date and time when the registrant acknowledged the TCN.

6. Send the registration to the Registry (ry interface, Section 4.3.5) and include the following information:

- * TCNID (<tmNotice:id>)
- * Expiration date of the TCN (<tmNotice:notAfter>)
- * Acceptance datetime of the TCN.

TCN are generated twice a day. The expiration date (<tmNotice:notAfter>) of each TCN MUST be set to 48 hours in the future by the TMDB, allowing the implementation of a cache by Registrars and enough time for acknowledging the TCN. Registrars SHOULD implement a cache of TCNs to minimize the number of queries sent to the TMDB. A cached TCN MUST be removed from the cache after the expiration date of the TCN as defined by <tmNotice:notAfter>. The TMDB MAY implement rate-limiting as one of the protection mechanisms to mitigate the risk of performance degradation.

5.3.5. TMDB Trademark Claims Services for Registrars

5.3.5.1. Claims Notice Information Service (CNIS)

The TCNs are provided by the TMDB online and are fetched by the Registrar via the dr interface (Section 4.3.6).

To get access to the TCNs, the Registrar needs the credentials provided by the TMDB (Section 5.1.2.1) and the Lookup Key received from the Registry via the ry interface (Section 4.3.5). The dr interface (Section 4.3.6) uses HTTPS with Basic access authentication.

The dr interface (Section 4.3.6) MAY support up to ten (10) concurrent connections from each Registrar.

The URL of the dr interface (Section 4.3.6) is:

< https://<tmdb-domain-name>/cnis/<lookupkey>.xml >

Note that the "lookupkey" may contain SLASH characters ("/"). The SLASH character is part of the URL path and MUST NOT be escaped when requesting the TCN.

The TLS certificate (HTTPS) used on the dr interface (Section 4.3.6) MUST be signed by a well-know public CA. Registrars MUST perform the Certification Path Validation described in Section 6 of [RFC5280]. Registrars will be authenticated in the dr interface using HTTP Basic access authentication. The dr (Section 4.3.6) interface MUST support

HTTPS keep-alive and MUST maintain the connection for up to 30 minutes.

5.4. Qualified Launch Program (QLP) Period

5.4.1. Domain Registration

During the OPTIONAL (see [QLP-Addendum]) Qualified Launch Program (QLP) Period effective allocations of DNS to third parties could require that Registries and Registrars provide Sunrise and/or Trademark Claims services. If required, Registries and Registrars MUST provide Sunrise and/or Trademark Claims services as described in Section 5.2 and Section 5.3.

The effective allocation scenarios are:

- o If the leftmost DNL (A-label in case of IDNs) of the DN being effectively allocated (QLP Name in this section) matches a DNL in the SURL, and an SMD is provided, then Registries MUST provide Sunrise Services (see Section 5.2) and the DN MUST be reported in a Sunrise LORDN file during the QLP Period.
- o If the QLP Name matches a DNL in the SURL but does not match a DNL in the DNL List, and an SMD is NOT provided (see section 2.2 of [QLP-Addendum]), then the DN MUST be reported in a Sunrise LORDN file using the special SMD-id "99999-99999" during the QLP Period.
- o If the QLP Name matches a DNL in the SURL and also matches a DNL in the DNL List, and an SMD is NOT provided (see section 2.2 of [QLP-Addendum]), then Registries MUST provide Trademark Claims services (see Section 5.3) and the DN MUST be reported in a Trademark Claims LORDN file during the QLP Period.
- o If the QLP Name matches a DNL in the DNL List but does not match a DNL in the SURL, then Registries MUST provide Trademark Claims services (see Section 5.2) and the DN MUST be reported in a Trademark Claims LORDN file during the QLP Period.

The following table lists all the effective allocation scenarios during a QLP Period:

QLP Name match in the SURL	QLP Name match in the DNL List	SMD was provided by the potential Registrant	Registry MUST provide Sunrise or Trademark Claims Services	Registry MUST report DN registration in <type> LORDN file
Y	Y	Y	Sunrise	Sunrise
Y	N	Y	Sunrise	Sunrise
N	Y	--	Trademark Claims	Trademark Claims
N	N	--	--	--
Y	Y	N (see section 2.2 of [QLP-Addendum])	Trademark Claims	Trademark Claims
Y	N	N (see section 2.2 of [QLP-Addendum])	--	Sunrise (using special SMD-id)

QLP Effective Allocation Scenarios

The TMDB MUST provide the following services to Registries during a QLP Period:

- o SMD Revocation List (see Section 5.2.3.1)
- o NORN (see Section 5.2.3.3)
- o DNL List (see Section 5.3.3.1)
- o Sunrise List (SURL) (see Section 5.4.2.1)

The TMDB MUST provide the following services to Registrars during a QLP Period:

- o SMD Revocation List (see Section 5.2.3.1)

- o CNIS (see Section 5.3.5.1)

5.4.2. TMDB QLP Services for Registries

5.4.2.1. Sunrise List (SURL)

A new Sunrise List (SURL) MUST be published by the TMDB twice a day, by 00:00:00 and 12:00:00 UTC.

Registries offering the OPTIONAL QLP Period MUST refresh the latest version of the SURL at least once every 24 hours.

Update of the SURL

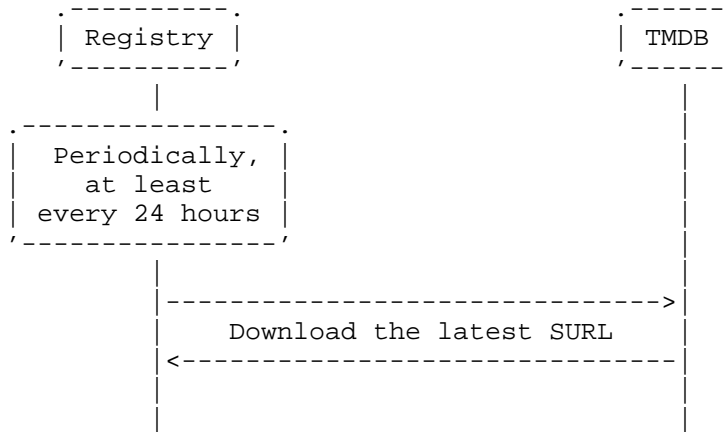


Figure 9

Note: the SURL will be the same regardless of the TLD. If a Backend Registry Operator manages the infrastructure of several TLDs, the Backend Registry Operator could refresh the SURL once every 24 hours, the SURL could be used for all the TLDs managed by the Backend Registry Operator.

6. Data Format Descriptions

6.1. Domain Name Label (DNL) List

This section defines the format of the list containing every Domain Name Label (DNL) that matches a Pre-Registered Mark (PRM). The list is maintained by the TMDB and downloaded by Registries in regular intervals (see Section 5.3.3.1). The Registries use the DNL List

during the Trademark Claims Period to check whether a requested DN matches a DNL of a PRM.

The DNL List contains all the DNLs covered by a PRM present in the TMDB at the datetime it is generated.

The DNL List is contained in a CSV formatted file that has the following structure:

- o first line: <version>,<DNL List creation datetime>

Where:

- + <version>, version of the file, this field MUST be 1.
- + <DNL List creation datetime>, date and time in UTC that the DNL List was created.

- o second line: a header line as specified in [RFC4180]

With the header names as follows:

DNL,lookup-key,insertion-datetime

- o One or more lines with: <DNL>,<lookup key>,<DNL insertion datetime>

Where:

- + <DNL>, a Domain Name Label covered by a PRM.
- + <lookup key>, lookup key that the Registry MUST provide to the Registrar. The lookup key has the following format: <YYYY><MM><DD><vv>/<X>/<X>/<X>/<Random bits><Sequential number>, where:
 - YYYY: year that the TCN was generated.
 - MM: zero-padded month that the TCN was generated.
 - DD: zero-padded day that the TCN was generated.
 - vv: version of the TCN, possible values are 00 and 01.
 - X: one hexadecimal digit [0-9A-F]. This is the first, second and third hexadecimal digit of encoding the <Random bits> in base16 as specified in [RFC4648].

- Random bits: 144 random bits encoded in base64url as specified in [RFC4648].
 - Sequential number: zero-padded natural number in the range 0000000001 to 2147483647.
- + <DNL insertion datetime>, datetime in UTC that the DNL was first inserted into the DNL List. The possible two values of time for inserting a DNL to the DNL List are 00:00:00 and 12:00:00 UTC.

Example of a DNL List

```
1,2012-08-16T00:00:00.0Z
DNL,lookup-key,insertion-datetime
example,2013041500/2/6/9/rJlNrDO92vDsAzf7EQzgJX4R0000000001,\
  2010-07-14T00:00:00.0Z
another-example,2013041500/6/A/5/alJAqG2vI2BmCv5PfUvuDkf40000000002,\
  2012-08-16T00:00:00.0Z
anotherexample,2013041500/A/C/7/rHdC4wnrWRvPY6nneCVtQhFj0000000003,\
  2011-08-16T12:00:00.0Z
```

Figure 10

To provide authentication and integrity protection, the DNL List will be PGP [RFC4880] signed by the TMDB (see also Section 5.1.1.4). The PGP signature of the DNL List can be found in the similar URI but with extension .sig as shown below.

The URL of the dy interface (Section 4.3.3) is:

- o < https://<tmdb-domain-name>/dnl/dnl-latest.csv >
- o < https://<tmdb-domain-name>/dnl/dnl-latest.sig >

6.2. SMD Revocation List

This section defines the format of the list of SMDs that have been revoked. The list is maintained by the TMDB and downloaded by Registries (and optionally by Registrars) in regular intervals (see Section 5.2.3.1). The SMD Revocation List is used during the Sunrise Period to validate SMDs received. The SMD Revocation List has a similar function as CRLs used in PKI [RFC5280].

The SMD Revocation List contains all the revoked SMDs present in the TMDB at the datetime it is generated.

The SMD Revocation List is contained in a CSV formatted file that has the following structure:

- o first line: <version>,<SMD Revocation List creation datetime>

Where:

- + <version>, version of the file, this field MUST be 1.
- + <SMD Revocation List creation datetime>, datetime in UTC that the SMD Revocation List was created.

- o second line: a header line as specified in [RFC4180]

With the header names as follows:

smd-id,insertion-datetime

- o One or more lines with: <smd-id>,<revoked SMD datetime>

Where:

- + <smd-id>, identifier of the SMD that was revoked.
- + <revoked SMD datetime>, revocation datetime in UTC of the SMD. The possible two values of time for inserting an SMD to the SMD Revocation List are 00:00:00 and 12:00:00 UTC.

To provide integrity protection, the SMD Revocation List is PGP signed by the TMDB (see also Section 5.1.1.4). The SMD Revocation List is provided by the TMDB with extension .csv. The PGP signature of the SMD Revocation List can be found in the similar URI but with extension .sig as shown below.

The URL of the sr interface (Section 4.3.12) and sy interface (Section 4.3.11) is:

- o < https://<tmdb-domain-name>/smdrl/smdrl-latest.csv >
- o < https://<tmdb-domain-name>/smdrl/smdrl-latest.sig >

Example of an SMD Revocation List

```
1,2012-08-16T00:00:00.0Z  
smd-id,insertion-datetime  
2-2,2012-08-15T00:00:00.0Z  
3-2,2012-08-15T00:00:00.0Z  
1-2,2012-08-15T00:00:00.0Z
```

Figure 11

6.3. List of Registered Domain Names (LORDN) file

This section defines the format of the List of Registered Domain Names (LORDN), which is maintained by each Registry and uploaded at least daily to the TMDB. Every time a DN matching a DNL of a PRM said DN is added to the LORDN along with further information related to its registration.

The URIs of the yd interface (Section 4.3.7) used to upload the LORDN file is:

- o Sunrise LORDN file:

< https://<tmdb-domain-name>/LORDN/<TLD>/sunrise >

- o Trademark Claims LORDN file:

< https://<tmdb-domain-name>/LORDN/<TLD>/claims >

During a QLP Period, Registries MAY be required to upload Sunrise or Trademark Claims LORDN files. The URIs of the yd interface used to upload LORDN files during a QLP Period is:

- o Sunrise LORDN file (during QLP Period):

< https://<tmdb-domain-name>/LORDN/<TLD>/sunrise/qlp >

- o Trademark Claims LORDN file (during a QLP Period):

< https://<tmdb-domain-name>/LORDN/<TLD>/claims/qlp >

The yd interface (Section 4.3.7) returns the following HTTP status codes after a HTTP POST request method is received:

- o The interface provides a HTTP/202 status code if the interface was able to receive the LORDN file and the syntax of the LORDN file is correct.

The interface provides the LORDN Transaction Identifier in the HTTP Entity-body that would be used by the Registry to download the LORDN Log file. The LORDN Transaction Identifier is a natural number zero-padded in the range 00000000000000000001 to 9223372036854775807.

The TMDB uses the <LORDN creation datetime> element of the LORDN file as a unique client-side identifier. If a LORDN file with the same <LORDN creation datetime> of a previously sent LORDN file is received by the TMDB, the LORDN Transaction Identifier of the previously sent LORDN file MUST be provided to the Registry. The TMDB MUST ignore the DN Lines present in the LORDN file if a LORDN file with the same <LORDN creation datetime> was previously sent.

The HTTP Location header field contains the URI where the LORDN Log file could be retrieved later, for example:

202 Accepted

Location: https://<tmdb-domain-name>/LORDN/example/sunrise/00000000000000000001/result

- o The interface provides a HTTP/400 if the request is incorrect or the syntax of the LORDN file is incorrect. The TMDB MUST return a human readable message in the HTTP Entity-body regarding the incorrect syntax of the LORDN file.
- o The interface provides a HTTP/401 status code if the credentials provided does not authorize the Registry Operator to upload a LORDN file.
- o The TMDB MUST return a HTTP/404 status code when trying to upload a LORDN file using the https://<tmdb-domain-name>/LORDN/<TLD>/sunrise/qlp or https://<tmdb-domain-name>/LORDN/<TLD>/claims/qlp interface outside of a QLP Period plus 26 hours.
- o The interface provides a HTTP/500 status code if the system is experiencing a general failure.

For example, to upload the Sunrise LORDN file for TLD "example", the URI would be:

< https://<tmdb-domain-name>/LORDN/example/sunrise >

The LORDN is contained in a CSV formatted file that has the following structure:

o For Sunrise Period:

- * first line: <version>,<LORDN creation datetime>,<Number of DN Lines>

Where:

- <version>, version of the file, this field MUST be 1.
- <LORDN creation datetime>, date and time in UTC that the LORDN was created.
- <Number of DN Lines>, number of DN Lines present in the LORDN file.

- * second line: a header line as specified in [RFC4180]

With the header names as follows:

roid, domain-name, SMD-id, registrar-id, registration-datetime, application-datetime

- * One or more lines with: <roid>,<DN registered>,<SMD-id>,<IANA Registrar id>,<datetime of registration>,<datetime of application creation>

Where:

- <roid>, DN Repository Object Identifier (DNROID) in the SRS.
- <DN registered>, DN that was effectively allocated. For IDNs the A-Label (see [RFC5890]) is used.
- <SMD-id>, SMD ID used for registration.
- <IANA Registrar ID>, IANA Registrar ID.
- <datetime of registration>, date and time in UTC that the domain was effectively allocated.
- OPTIONAL <datetime of application creation>, date and time in UTC that the application was created. The

<datetime of application creation> MUST be provided in case of a DN effective allocation based on an asynchronous registration (e.g., when using auctions).

Example of a Sunrise LORDN file

```
1,2012-08-16T00:00:00.0Z,3
roid, domain-name, SMD-id, registrar-id, registration-datetime, \
  application-datetime
SH8013-REP, example1.gtld, 1-2, 9999, 2012-08-15T13:20:00.0Z, \
  2012-07-15T00:50:00.0Z
EK77-REP, example2.gtld, 2-2, 9999, 2012-08-15T14:00:03.0Z
HB800-REP, example3.gtld, 3-2, 9999, 2012-08-15T15:40:00.0Z
```

Figure 12

o For Trademark Claims Period:

- * first line: <version>, <LORDN creation datetime>, <Number of DN Lines>

Where:

- <version>, version of the file, this field MUST be 1.
- <LORDN creation datetime>, date and time in UTC that the LORDN was created.
- <Number of DN Lines>, number of DN Lines present in the LORDN file.

- * second line: a header line as specified in [RFC4180]

With the header names as follows:

```
roid, domain-name, notice-id, registrar-id, registration-
datetime, ack-datetime, application-datetime
```

- * One or more lines with: <roid>, <DN registered>, <TCNID>, <IANA Registrar id>, <datetime of registration>, <datetime of acceptance of the TCN>, <datetime of application creation>

Where:

- <roid>, DN Repository Object Identifier (DNROID) in the SRS.

- <DN registered>, DN that was effectively allocated. For IDNs the A-Label is used.
- <TCNID>, Trademark Claims Notice Identifier as specified in <tmNotice:id>.
- <IANA Registrar ID>, IANA Registrar ID.
- <datetime of registration>, date and time in UTC that the domain was effectively allocated.
- <datetime of acceptance of the TCN>, date and time in UTC that the TCN was acknowledged.
- OPTIONAL <datetime of application creation>, date and time in UTC that the application was created. The <datetime of application creation> MUST be provided in case of a DN effective allocation based on an asynchronous registration (e.g., when using auctions).

For a DN matching a DNL of a PRM at the moment of registration, created without the TCNID, expiration datetime and acceptance datetime, because DNL was inserted (or re-inserted) for the first time into DNL List less than 24 hours ago, the string "recent-dnl-insertion" MAY be specified in <TCNID> and <datetime of acceptance of the TCN>.

Example of a Trademark Claims LORDN file

```
1,2012-08-16T00:00:00.0Z,3
roid, domain-name, notice-id, registrar-id, registration-datetime, \
  ack-datetime, application-datetime
SH8013-REP, example1.gtld, a76716ed9223352036854775808, \
  9999, 2012-08-15T14:20:00.0Z, 2012-08-15T13:20:00.0Z
EK77-REP, example2.gtld, a7b786ed9223372036856775808, \
  9999, 2012-08-15T11:20:00.0Z, 2012-08-15T11:19:00.0Z
HB800-REP, example3.gtld, recent-dnl-insertion, \
  9999, 2012-08-15T13:20:00.0Z, recent-dnl-insertion
```

Figure 13

6.3.1. LORDN Log file

After reception of the LORDN file, the TMDB verifies its content for syntactical and semantical correctness. The output of the LORDN file verification is retrieved using the yd interface (Section 4.3.7).

The URI of the yd interface (Section 4.3.7) used to retrieve the LORDN Log file is:

- o Sunrise LORDN Log file:

```
< https://<tmdb-domain-name>/LORDN/<TLD>/sunrise/<lordn-transaction-identifier>/result >
```

- o Trademark Claims LORDN Log file:

```
< https://<tmdb-domain-name>/LORDN/<TLD>/claims/<lordn-transaction-identifier>/result >
```

A Registry Operator MUST NOT send more than one request per minute per TLD to download a LORDN Log file.

The yd interface (Section 4.3.7) returns the following HTTP status codes after a HTTP GET request method is received:

- o The interface provides a HTTP/200 status code if the interface was able to provide the LORDN Log file. The LORDN Log file is contained in the HTTP Entity-body.
- o The interface provides a HTTP/204 status code if the LORDN Transaction Identifier is correct, but the server has not finalized processing the LORDN file.
- o The interface provides a HTTP/400 status code if the request is incorrect.
- o The interface provides a HTTP/401 status code if the credentials provided does not authorize the Registry Operator to download the LORDN Log file.
- o The interface provides a HTTP/404 status code if the LORDN Transaction Identifier is incorrect.
- o The interface provides a HTTP/500 status code if the system is experiencing a general failure.

For example, to obtain the LORDN Log file in case of a Sunrise LORDN file with LORDN Transaction Identifier 000000000000000001 and TLD "example" the URI would be:

```
< https://<tmdb-domain-  
name>/LORDN/example/sunrise/000000000000000001/result >
```

The LORDN Log file is contained in a CSV formatted file that has the following structure:

- o first line: <version>,<LORDN Log creation datetime>,<LORDN file creation datetime>,<LORDN Log Identifier>,<Status flag>,<Warning flag>,<Number of DN Lines>

Where:

- + <version>, version of the file, this field MUST be 1.
- + <LORDN Log creation datetime>, date and time in UTC that the LORDN Log was created.
- + <LORDN file creation datetime>, date and time in UTC of creation for the LORDN file that this log file is referring to.
- + <LORDN Log Identifier>, unique identifier of the LORDN Log provided by the TMDB. This identifier could be used by the Registry Operator to unequivocally identify the LORDN Log. The identifier will be a string of a maximum LENGTH of 60 characters from the Base 64 alphabet.
- + <Status flag>, whether the LORDN file has been accepted for processing by the TMDB. Possible values are "accepted" or "rejected".
- + <Warning flag>, whether the LORDN Log has any warning result codes. Possible values are "no-warnings" or "warnings-present".
- + <Number of DN Lines>, number of DNS effective allocations processed in the LORDN file.

A Registry Operator is NOT REQUIRED to process a LORDN Log with a <Status flag>="accepted" and <Warning flag>="no-warnings".

- o second line: a header line as specified in [RFC4180]

With the header names as follows:

- roid,result-code
- o One or more lines with: <roid>,<result code>
- Where:
- + <roid>, DN Repository Object IDentifier (DNROID) in the SRS.
 - + <result code>, result code as described in Section 6.3.1.1.

Example of a LORDN Log file

```
1,2012-08-16T02:15:00.0Z,2012-08-16T00:00:00.0Z,\
  0000000000000478Nzs+3VMkR8ckuUynOLmyeqTmZQSbzDuf/R50n2n5QX4=,\
  accepted,no-warnings,1
roid,result-code
SH8013-REP,2000
```

Figure 14

6.3.1.1. LORDN Log Result Codes

In Figure 15 the classes of result codes (rc) are listed. Those classes in square brackets are not used at this time, but may come into use at some later stage. The first two digits of a result code denote the result code class, which defines the outcome at the TMDB:

- o ok: Success, DN Line accepted by the TMDB.
- o warn: a warning is issued, DN Line accepted by the TMDB.
- o err: an error is issued, LORDN file rejected by the TMDB.

In case that after processing a DN Line, the error result code is 45xx or 46xx for that DN Line, the LORDN file MUST be rejected by the TMDB. If the LORDN file is rejected, DN Lines that are syntactically valid will be reported with a 2001 result code. A 2001 result code means that the DN Line is syntactically valid, however the DN Line was not processed because the LORDN file was rejected. All DNS reported in a rejected LORDN file MUST be reported again by the Registry because none of the DN Lines present in the LORDN file have been processed by the TMDB.

LORDN Log Result Code Classes

code	Class	outcome
----	-----	-----
20xx	Success	ok
35xx	[DN Line syntax warning]	warn
36xx	DN Line semantic warning	warn
45xx	DN Line syntax error	err
46xx	DN Line semantic error	err

Figure 15

In the following, the LORDN Log result codes used by the TMDB are described:

LORDN Log result Codes

rc	Short Description	Long Description
----	-----	-----
2000	OK	DN Line successfully processed.
2001	OK but not processed	DN Line is syntactically correct but was not processed because the LORDN file was rejected.
3601	TCN Acceptance Date after Registration Date	TCN Acceptance Date in DN Line is newer than the Registration Date.
3602	Duplicate DN Line	This DN Line is an exact duplicate of another DN Line in same file, DN Line ignored.
3603	DNROID Notified Earlier	Same DNROID has been notified earlier, DN Line ignored.
3604	TCN Checksum invalid	Based on the DN effectively allocated, the TCNID and the expiration date of the linked TCN, the TCN Checksum is

invalid.

- 3605 TCN Expired
The TCN was already expired (based on the <tmNotice:notAfter> field of the TCN) at the datetime of acknowledgement.
- 3606 Wrong TCNID used
The TCNID used for the registration does not match the related DN.
- 3609 Invalid SMD used
The SMD used for registration was not valid at the moment of registration based on the <smd:notBefore> and <smd:notAfter> elements.
In case of an asynchronous registration, this refer to the <datetime of application creation>.
- 3610 DN reported outside of the time window
The DN was reported outside of the required 26 hours reporting window.
- 3611 DN does not match the labels in SMD
The DN does not match the labels included in the SMD.
- 3612 SMDID does not exist
The SMDID has never existed in the central repository.
- 3613 SMD was revoked when used
The SMD used for registration was revoked more than 24 hours ago of the <datetime of registration>.
In case of an asynchronous registration, the <datetime of application creation> is used when validating the DN Line.
- 3614 TCNID does not exist
The TCNID has never existed in the central repository.
- 3615 Recent-dnl-insertion outside of the time window
The DN registration is reported as a recent-dnl-insertion, but the (re) insertion into the DNL occurred more than 24 hours ago.
- 3616 Registration Date of DN in Claims before the end of Sunrise Period
The registration date of the DN is before the end of Sunrise Period and the DN was reported in a Trademark Claims LORDN file.
- 3617 Registrar has not been approved by the TMDB
Registrar ID in DN Line has not completed Trademark Claims integration

testing with the TMDB.

- 3618 Registration Date of DN in a QLP LORDN file outside of the QLP Period
The registration date of the DN in a QLP LORDN file is outside of the QLP Period.
- 3619 TCN was not valid
The TCN was not valid (based on the <tmNotice:notBefore> field of the TCN) at the datetime of acknowledgement.
- 4501 Syntax Error in DN Line
Syntax Error in DN Line.
- 4601 Invalid TLD used
The TLD in the DN Line does not match what is expected for this LORDN.
- 4602 Registrar ID Invalid
Registrar ID in DN Line is not a valid ICANN-Accredited Registrar.
- 4603 Registration Date in the future
The <datetime of registration> in the DN Line is in the future.
- 4606 TLD not in Sunrise or Trademark Claims Period
The <datetime of registration> was reported when the TLD was not in Sunrise or Trademark Claims Periods.
In case of an asynchronous registration, the <datetime of application creation> is used when validating the DN Line.
- 4607 Application Date in the future
The <datetime of application creation> in the DN Line is in the future.
- 4608 Application Date is later than Registration Date
The <datetime of application creation> in the DN Line is later than the <datetime of registration>.
- 4609 TCNID wrong syntax
The syntax of the TCNID is invalid.
- 4610 TCN Acceptance Date is in the future
The <datetime of acceptance of the TCN> is in the future.
- 4611 Label has never existed in the TMDB
The label in the registered DN has never existed in the TMDB.

Figure 16

6.4. Signed Mark Data (SMD) File

This section defines the format of the Signed Mark Data (SMD) File. After a successful registration of a mark, the TMV returns an SMD File to the TMH. The SMD File can then be used for registration of one or more DNS covered by the PRM during the Sunrise Period of a TLD.

Two encapsulation boundaries are defined for delimiting the encapsulated base64 encoded SMD: i.e. "-----BEGIN ENCODED SMD-----" and "-----END ENCODED SMD-----". Only data inside the encapsulation boundaries MUST be used by Registries and Registrars for validation purposes, i.e. any data outside these boundaries as well as the boundaries themselves MUST be ignored for validation purposes.

The structure of the SMD File is as follows, all the elements are REQUIRED, and MUST appear in the specified order.

1. Marks: <marks>
2. smdID: <SMD-ID>
3. U-labels: <comma separated list of labels in U-label (see [RFC5890]) form (i.e., U-labels or LDH as the case may be)>
4. notBefore: <begin validity>
5. notAfter: <end validity>
6. -----BEGIN ENCODED SMD-----
7. <encoded SMD (see [I-D.ietf-eppext-tmch-smd])>
8. -----END ENCODED SMD-----

Example of an SMD File:

```
Marks: Example One
smdID: 1-2
U-labels: example-one, exampleone
notBefore: 2011-08-16 09:00
notAfter: 2012-08-16 09:00
-----BEGIN ENCODED SMD-----
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPHNtZDpzaWdu
ZWRNYXJrIHhtbG5zOnNtZD0idXJuOmlldGY6cGFyYW1zOnhtbDpuczpzaWduZWRN
... (base64 data elided for brevity) ...
dXJlPgo8L3NtZDpzaWduZWRNYXJrPgo=
-----END ENCODED SMD-----
```

Figure 17

6.5. Trademark Claims Notice (TCN)

The TMDB MUST provide the TCN to Registrars in XML format as specified below.

An enclosing element `<tmNotice:notice>` that describes the Trademark Notice to a given label.

The child elements of the `<tmNotice:notice>` element include:

- o A `<tmNotice:id>` element that contains the unique identifier of the Trademark Notice. This element contains the the TCNID.

The TCNID is a string concatenation of a TCN Checksum and the TMDB Notice Identifier. The first 8 characters of the TCNID is a TCN Checksum. The rest is the TMDB Notice Identifier, which is a zero-padded natural number in the range of 00000000000000000001 to 9223372036854775807.

Example of a TCNID:

370d0b7c9223372036854775807.

Where:

- + TCN Checksum=370d0b7c
- + TMDB Notice Identifier=9223372036854775807

The TCN Checksum is a 8 characters long Base16 encoded output of computing the CRC32 of the string concatenation of: label + unix_timestamp(<tmNotice:notAfter>) + TMDB Notice Identifier

TMDB MUST use the Unix time conversion of the <tmNotice:notAfter> in UTC to calculate the TCN Checksum. Unix time is defined as the number of seconds that have elapsed since 1970-01-01T00:00:00Z not counting leap seconds. For example, the conversion to Unix time of 2010-08-16T09:00:00.0Z is shown:

```
unix_time(2010-08-16T09:00:00.0Z)=1281949200
```

The TMDB uses the <tmNotice:label> and <tmNotice:notAfter> elements from the TCN along with the TMDB Notice Identifier to compute the TCN Checksum.

A Registry MUST use the leftmost DNL (A-label in case of IDNs) of the DN being registered, the expiration datetime of the TCN (provided by the Registrar) and the TMDB Notice Identifier extracted from the TCNID (provided by the Registrar) to compute the TCN Checksum. For example the DN "foo.bar.example" being effectively allocated, the left most label would be "foo".

Example of computation of the TCN Checksum:

```
CRC32(example-one12819492009223372036854775807)=370d0b7c
```

- o A <tmNotice:notBefore> element that contains the start of the validity date and time of the TCN.
- o A <tmNotice:notAfter> element that contains the expiration date and time of the TCN.
- o A <tmNotice:label> elements that contain the DNL (A-label in case of IDNs) form of the label that correspond to the DN covered by a PRM.
- o One or more <tmNotice:claim> elements that contain the Trademark Claim. The <tmNotice:claim> element contains the following child elements:
 - * A <tmNotice:markName> element that contains the mark text string.
 - * One or more <tmNotice: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 or licensee. The child elements of <tmNotice:holder> include:

- + An OPTIONAL <tmNotice:name> element that contains the name of the holder. A <tmNotice:name> MUST be specified if <tmNotice:org> is not specified.
- + An OPTIONAL <tmNotice:org> element that contains the name of the organization holder of the mark. A <tmNotice:org> MUST be specified if <tmNotice:name> is not specified.
- + A <tmNotice:addr> element that contains the address information of the holder of a mark. A <tmNotice:addr> contains the following child elements:
 - One, two or three OPTIONAL <tmNotice:street> elements that contains the organization's street address.
 - A <tmNotice:city> element that contains the organization's city.
 - An OPTIONAL <tmNotice:sp> element that contains the organization's state or province.
 - An OPTIONAL <tmNotice:pc> element that contains the organization's postal code.
 - A <tmNotice:cc> element that contains the organization's country code. This a two-character code from [ISO3166-2].
- + An OPTIONAL <tmNotice:voice> element that contains the organization's voice telephone number.
- + An OPTIONAL <tmNotice:fax> element that contains the organization's facsimile telephone number.
- + An OPTIONAL <tmNotice:email> element that contains the email address of the holder.
- * Zero or more OPTIONAL <tmNotice: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. The child elements of <tmNotice:contact> include:
 - + A <tmNotice:name> element that contains name of the responsible person.

- + An OPTIONAL <tmNotice:org> element that contains the name of the organization of the contact.
- + A <tmNotice:addr> element that contains the address information of the contact. A <tmNotice:addr> contains the following child elements:
 - One, two or three OPTIONAL <tmNotice:street> elements that contains the contact's street address.
 - A <tmNotice:city> element that contains the contact's city.
 - An OPTIONAL <tmNotice:sp> element that contains the contact's state or province.
 - An OPTIONAL <tmNotice:pc> element that contains the contact's postal code.
 - A <tmNotice:cc> element that contains the contact's country code. This a two-character code from [ISO3166-2].
- + A <tmNotice:voice> element that contains the contact's voice telephone number.
- + An OPTIONAL <tmNotice:fax> element that contains the contact's facsimile telephone number.
- + A <tmNotice:email> element that contains the contact's email address.
- * A <tmNotice:jurDesc> element that contains the name (in English) of the jurisdiction where the mark is protected. A jurCC attribute contains the two-character code of the jurisdiction where the mark was registered. This is a two-character code from [WIPO.ST3].
- * Zero or more OPTIONAL <tmNotice:classDesc> element that contains the description (in English) of the Nice Classification as defined in [WIPO-NICE-CLASSES]. A classNum attribute contains the class number.
- * A <tmNotice:goodsAndServices> element that contains the full description of the goods and services mentioned in the mark registration document.

- * An OPTIONAL `<tmNotice:notExactMatch>` element signals that the claim notice was added to the TCN based on other rule (e.g. [Claims50]) than exact match (defined in [MatchingRules]). The `<tmNotice:notExactMatch>` contains one or more:
 - + An OPTIONAL `<tmNotice:udrp>` element that signals that the claim notice was added because of a previously abused name included in an UDRP case. The `<tmNotice:udrp>` contains:
 - A `<tmNotice:caseNo>` element that contains the UDRP case number used to validate the previously abused name.
 - A `<tmNotice:udrpProvider>` element that contains the name of the UDRP provider.
 - + An OPTIONAL `<tmNotice:court>` element that signals that the claim notice was added because of a previously abused name included in a court's resolution. The `<tmNotice:court>` contains:
 - A `<tmNotice:refNum>` element that contains the reference number of the court's resolution used to validate the previously abused name.
 - A `<tmNotice:cc>` element that contains the two-character code from [ISO3166-2] of the jurisdiction of the court.
 - A `<tmNotice:courtName>` element that contains the name of the court.

Example of a `<tmNotice:notice>` object:

```
<?xml version="1.0" encoding="UTF-8"?>
<tmNotice:notice xmlns:tmNotice="urn:ietf:params:xml:ns:tmNotice-1.0">
  <tmNotice:id>370d0b7c9223372036854775807</tmNotice:id>
  <tmNotice:notBefore>2010-08-14T09:00:00.0Z</tmNotice:notBefore>
  <tmNotice:notAfter>2010-08-16T09:00:00.0Z</tmNotice:notAfter>
  <tmNotice:label>example-one</tmNotice:label>
  <tmNotice:claim>
    <tmNotice:markName>Example One</tmNotice:markName>
    <tmNotice:holder entitlement="owner">
      <tmNotice:org>Example Inc.</tmNotice:org>
      <tmNotice:addr>
        <tmNotice:street>123 Example Dr.</tmNotice:street>
        <tmNotice:street>Suite 100</tmNotice:street>
        <tmNotice:city>Reston</tmNotice:city>
        <tmNotice:sp>VA</tmNotice:sp>
      </tmNotice:addr>
    </tmNotice:holder>
  </tmNotice:claim>
</tmNotice:notice>
```

```
<tmNotice:pc>20190</tmNotice:pc>
  <tmNotice:cc>US</tmNotice:cc>
</tmNotice:addr>
</tmNotice:holder>
<tmNotice:contact type="owner">
  <tmNotice:name>Joe Doe</tmNotice:name>
  <tmNotice:org>Example Inc.</tmNotice:org>
  <tmNotice:addr>
    <tmNotice:street>123 Example Dr.</tmNotice:street>
    <tmNotice:street>Suite 100</tmNotice:street>
    <tmNotice:city>Reston</tmNotice:city>
    <tmNotice:sp>VA</tmNotice:sp>
    <tmNotice:pc>20190</tmNotice:pc>
    <tmNotice:cc>US</tmNotice:cc>
  </tmNotice:addr>
  <tmNotice:voice x="4321">+1.7035555555</tmNotice:voice>
  <tmNotice:email>jdoe@example.com</tmNotice:email>
</tmNotice:contact>
<tmNotice:jurDesc jurCC="US">UNITED STATES OF AMERICA</tmNotice:jurDesc>
<tmNotice:classDesc classNum="35">
  Advertising; business management; business administration.
</tmNotice:classDesc>
<tmNotice:classDesc classNum="36">
  Insurance; financial affairs; monetary affairs; real estate.
</tmNotice:classDesc>
<tmNotice:goodsAndServices>
  Bardus populorum circumdabit se cum captiosus populum.
  Smert populorum circumdabit se cum captiosus populum qui eis differimus.
</tmNotice:goodsAndServices>
</tmNotice:claim>
<tmNotice:claim>
  <tmNotice:markName>Example-One</tmNotice:markName>
  <tmNotice:holder entitlement="owner">
    <tmNotice:org>Example S.A. de C.V.</tmNotice:org>
    <tmNotice:addr>
      <tmNotice:street>Calle conocida #343</tmNotice:street>
      <tmNotice:city>Conocida</tmNotice:city>
      <tmNotice:sp>SP</tmNotice:sp>
      <tmNotice:pc>82140</tmNotice:pc>
      <tmNotice:cc>BR</tmNotice:cc>
    </tmNotice:addr>
  </tmNotice:holder>
  <tmNotice:jurDesc jurCC="BR">BRAZIL</tmNotice:jurDesc>
  <tmNotice:goodsAndServices>
    Bardus populorum circumdabit se cum captiosus populum.
    Smert populorum circumdabit se cum captiosus populum qui eis differimus.
  </tmNotice:goodsAndServices>
</tmNotice:claim>
```

```

<tmNotice:claim>
  <tmNotice:markName>One</tmNotice:markName>
  <tmNotice:holder entitlement="owner">
    <tmNotice:org>One Corporation</tmNotice:org>
    <tmNotice:addr>
      <tmNotice:street>Otra calle</tmNotice:street>
      <tmNotice:city>Otra ciudad</tmNotice:city>
      <tmNotice:sp>OT</tmNotice:sp>
      <tmNotice:pc>383742</tmNotice:pc>
      <tmNotice:cc>CR</tmNotice:cc>
    </tmNotice:addr>
  </tmNotice:holder>
  <tmNotice:jurDesc jurCC="CR">COSTA RICA</tmNotice:jurDesc>
  <tmNotice:goodsAndServices>
    Bardus populorum circumdabit se cum captiosus populum.
    Smert populorum circumdabit se cum captiosus populum qui eis differimus.
  </tmNotice:goodsAndServices>
  <tmNotice:notExactMatch>
    <tmNotice:court>
      <tmNotice:refNum>234235</tmNotice:refNum>
      <tmNotice:cc>CR</tmNotice:cc>
      <tmNotice:courtName>Supreme Court of Justice of Costa Rica</tmNotice:courtName>
    </tmNotice:court>
  </tmNotice:notExactMatch>
</tmNotice:claim>
<tmNotice:claim>
  <tmNotice:markName>One Inc</tmNotice:markName>
  <tmNotice:holder entitlement="owner">
    <tmNotice:org>One SA de CV</tmNotice:org>
    <tmNotice:addr>
      <tmNotice:street>La calle</tmNotice:street>
      <tmNotice:city>La ciudad</tmNotice:city>
      <tmNotice:sp>CD</tmNotice:sp>
      <tmNotice:pc>34323</tmNotice:pc>
      <tmNotice:cc>AR</tmNotice:cc>
    </tmNotice:addr>
  </tmNotice:holder>
  <tmNotice:jurDesc jurCC="AR">ARGENTINA</tmNotice:jurDesc>
  <tmNotice:goodsAndServices>
    Bardus populorum circumdabit se cum captiosus populum.
    Smert populorum circumdabit se cum captiosus populum qui eis differimus.
  </tmNotice:goodsAndServices>
  <tmNotice:notExactMatch>
    <tmNotice:udrp>
      <tmNotice:caseNo>D2003-0499</tmNotice:caseNo>
      <tmNotice:udrpProvider>WIPO</tmNotice:udrpProvider>
    </tmNotice:udrp>
  </tmNotice:notExactMatch>

```

```
</tmNotice:claim>
</tmNotice:notice>
```

For the formal syntax of the TCN please refer to Section 7.1.

6.6. Sunrise List (SURL)

This section defines the format of the list containing every Domain Name Label (DNL) that matches a PRM eligible for Sunrise. The list is maintained by the TMDB and downloaded by Registries in regular intervals (see Section 5.4.2.1). The Registries use the Sunrise List during the Qualified Launch Program Period to check whether a requested DN matches a DNL of a PRM eligible for Sunrise.

The Sunrise List contains all the DNLs covered by a PRM eligible for Sunrise present in the TMDB at the datetime it is generated.

The Sunrise List is contained in a CSV formatted file that has the following structure:

- o first line: <version>,<Sunrise List creation datetime>

Where:

- + <version>, version of the file, this field MUST be 1.
- + <Sunrise List creation datetime>, date and time in UTC that the Sunrise List was created.

- o second line: a header line as specified in [RFC4180]

With the header names as follows:

DNL,insertion-datetime

- o One or more lines with: <DNL>,<DNL insertion datetime>

Where:

- + <DNL>, a Domain Name Label covered by a PRM eligible for Sunrise.
- + <DNL insertion datetime>, datetime in UTC that the DNL was first inserted into the Sunrise List. The possible two values of time for inserting a DNL to the Sunrise List are 00:00:00 and 12:00:00 UTC.

Example of a SURL

```
1,2012-08-16T00:00:00.0Z
DNL,insertion-datetime
example,2010-07-14T00:00:00.0Z
another-example,2012-08-16T00:00:00.0Z
anotherexample,2011-08-16T12:00:00.0Z
```

Figure 18

To provide authentication and integrity protection, the Sunrise List will be PGP signed by the TMDB (see also Section 5.1.1.4). The PGP signature of the Sunrise List can be found in the similar URI but with extension .sig as shown below.

The URL of the dy interface (Section 4.3.3) is:

- o < https://<tmdb-domain-name>/dnl/surl-latest.csv >
- o < https://<tmdb-domain-name>/dnl/surl-latest.sig >

7. Formal Syntax

7.1. Trademark Claims Notice (TCN)

The schema presented here is for a Trademark Claims Notice.

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.

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:tmNotice-1.0"
  xmlns:tmNotice="urn:ietf:params:xml:ns:tmNotice-1.0"
  xmlns:mark="urn:ietf:params:xml:ns:mark-1.0"
```

```
xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

<annotation>
  <documentation>
    Schema for representing a Trademark Claim Notice.
  </documentation>
</annotation>
<import namespace="urn:ietf:params:xml:ns:mark-1.0"/>
<element name="notice" type="tmNotice:noticeType"/>
<complexType name="holderType">
  <sequence>
    <element name="name" type="token" minOccurs="0"/>
    <element name="org" type="token" minOccurs="0"/>
    <element name="addr" type="tmNotice:addrType"/>
    <element name="voice" type="mark:el64Type" minOccurs="0"/>
    <element name="fax" type="mark:el64Type" minOccurs="0"/>
    <element name="email" type="mark:minTokenType" minOccurs="0"/>
  </sequence>
  <attribute name="entitlement" type="mark:entitlementType"/>
</complexType>
<complexType name="noticeType">
  <sequence>
    <element name="id" type="tmNotice:idType"/>
    <element name="notBefore" type="dateTime"/>
    <element name="notAfter" type="dateTime"/>
    <element name="label" type="mark:labelType"/>
    <element name="claim" type="tmNotice:claimType" minOccurs="0"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>
<complexType name="claimType">
  <sequence>
    <element name="markName" type="token"/>
    <element name="holder" type="tmNotice:holderType"
      maxOccurs="unbounded"/>
    <element name="contact" type="tmNotice:contactType" minOccurs="0"
      maxOccurs="unbounded"/>
    <element name="jurDesc" type="tmNotice:jurDescType"/>
    <element name="classDesc" type="tmNotice:classDescType"
      minOccurs="0" maxOccurs="unbounded"/>
    <element name="goodsAndServices" type="token"/>
    <element name="notExactMatch" type="tmNotice:noExactMatchType"
      minOccurs="0"/>
  </sequence>
</complexType>
<complexType name="jurDescType">
  <simpleContent>
    <extension base="token">
```



```
        <attribute name="jurCC" type="mark:ccType" use="required"/>
    </extension>
</simpleContent>
</complexType>
<complexType name="classDescType">
    <simpleContent>
        <extension base="token">
            <attribute name="classNum" type="integer" use="required"/>
        </extension>
    </simpleContent>
</complexType>
<complexType name="noExactMatchType">
    <choice maxOccurs="unbounded">
        <element name="udrp" type="tmNotice:udrpType"/>
        <element name="court" type="tmNotice:courtType"/>
    </choice>
</complexType>
<complexType name="udrpType">
    <sequence>
        <element name="caseNo" type="token"/>
        <element name="udrpProvider" type="token"/>
    </sequence>
</complexType>
<complexType name="courtType">
    <sequence>
        <element name="refNum" type="token"/>
        <element name="cc" type="mark:ccType"/>
        <element name="region" type="token" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="courtName" type="token"/>
    </sequence>
</complexType>
<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"/>
        <element name="pc" type="mark:pcType" minOccurs="0"/>
        <element name="cc" type="mark:ccType"/>
    </sequence>
</complexType>
<complexType name="contactType">
    <sequence>
        <element name="name" type="token"/>
        <element name="org" type="token" minOccurs="0"/>
        <element name="addr" type="tmNotice:addrType"/>
        <element name="voice" type="mark:e164Type"/>
        <element name="fax" type="mark:e164Type" minOccurs="0"/>
    </sequence>
</complexType>
```

```
        <element name="email" type="mark:minTokenType"/>
      </sequence>
      <attribute name="type" type="mark:contactTypeType"/>
    </complexType>
    <simpleType name="idType">
      <restriction base="token">
        <pattern value="[a-zA-F0-9]{8}\d{1,19}"/>
      </restriction>
    </simpleType>
  </schema>
END
```

8. Acknowledgements

This specification is a collaborative effort from several participants in the ICANN community. Bernie Hoeneisen participated as co-author until version 02 providing invaluable support for this document. This specification is based on a model spearheaded by: Chris Wright, Jeff Neuman, Jeff Eckhaus and Will Shorter. The author would also like to thank the thoughtful feedback provided by many in the tmch-tech mailing list, but particularly the extensive help provided by James Gould, James Mitchell and Francisco Arias. This document includes feedback received from the following individuals: Paul Hoffman.

9. IANA Considerations

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

Registration request for the Trademark Claims Notice:

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

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

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

10. Security Considerations

This specification uses HTTP Basic Authentication to provide a simple application-layer authentication service. HTTPS is used in all interfaces in order to protect against most common attacks. In addition, the client identifier is tied to a set of IP addresses that

are allowed to connect to the interfaces described in this document, providing an extra security measure.

The TMDB MUST provide credentials to the appropriate Registries and Registrars.

The TMDB MUST require the use of strong passwords by Registries and Registrars.

The TMDB, Registries and Registrars MUST use the best practices described in RFC 7525 or its successors.

11. References

11.1. Normative References

- [I-D.ietf-epext-tmch-smd]
Lozano, G., "Mark and Signed Mark Objects Mapping", draft-ietf-epext-tmch-smd-06 (work in progress), March 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.

11.2. Informative References

- [Claims50]
ICANN, "Implementation Notes: Trademark Claims Protection for Previously Abused Names", July 2013, <<https://newgtlds.icann.org/en/about/trademark-clearinghouse/previously-abused-16jul13-en.pdf>>.
- [ICANN-GTLD-AGB-20120604]
ICANN, "gTLD Applicant Guidebook Version 2012-06-04", June 2012, <<http://newgtlds.icann.org/en/applicants/agb/guidebook-full-04jun12-en.pdf>>.
- [ISO3166-2]
ISO, "International Standard for country codes and codes for their subdivisions", 2006, <http://www.iso.org/iso/home/standards/country_codes.htm>.

[MatchingRules]

ICANN, "Memorandum on Implementing Matching Rules", September 2012, <<https://newgtlds.icann.org/en/about/trademark-clearinghouse/matching-rules-24sep12-en.pdf>>.

[QLP-Addendum]

ICANN, "Qualified Launch Program Addendum", April 2014, <<https://newgtlds.icann.org/en/about/trademark-clearinghouse/rpm-requirements-qlp-addendum-10apr14-en.pdf>>.

- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <<http://www.rfc-editor.org/info/rfc2616>>.
- [RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, DOI 10.17487/RFC2617, June 1999, <<http://www.rfc-editor.org/info/rfc2617>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<http://www.rfc-editor.org/info/rfc2818>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<http://www.rfc-editor.org/info/rfc3339>>.
- [RFC4180] Shafranovich, Y., "Common Format and MIME Type for Comma-Separated Values (CSV) Files", RFC 4180, DOI 10.17487/RFC4180, October 2005, <<http://www.rfc-editor.org/info/rfc4180>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<http://www.rfc-editor.org/info/rfc4648>>.
- [RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <<http://www.rfc-editor.org/info/rfc4880>>.

- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<http://www.rfc-editor.org/info/rfc5280>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<http://www.rfc-editor.org/info/rfc5890>>.
- [RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <<http://www.rfc-editor.org/info/rfc7719>>.
- [RPM-Requirements]
ICANN, "Rights Protection Mechanism Requirements", September 2013, <<https://newgtlds.icann.org/en/about/trademark-clearinghouse/rpm-requirements-30sep13-en.pdf>>.
- [WIPO-NICE-CLASSES]
WIPO, "WIPO Nice Classification", 2015, <<http://www.wipo.int/classifications/nice/en>>.
- [WIPO.ST3]
WIPO, "Recommended standard on two-letter codes for the representation of states, other entities and intergovernmental organizations", March 2007, <http://www.iso.org/iso/home/standards/country_codes.htm>.

Author's Address

Gustavo Lozano
ICANN
12025 Waterfront Drive, Suite 300
Los Angeles 90292
US

Phone: +1.3103015800
Email: gustavo.lozano@icann.org

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2016

G. Lozano
ICANN
March 9, 2016

Mark and Signed Mark Objects Mapping
draft-ietf-eppext-tmch-smd-06

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.

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	2
1.1. Terminology	3
2. Object Description	4
2.1. Holder and Contacts objects	4
2.2. Mark	6
2.3. Signed Mark	9
2.4. Encoded Signed Mark	13
3. Formal Syntax	13
3.1. Signed Mark Schema	13
3.2. Mark Schema	15
4. Implementation Status	21
4.1. Verisign EPP SDK	21
4.2. Verisign Consolidated Top Level Domain (CTLD) SRS	22
4.3. Verisign .COM / .NET SRS	22
4.4. REngin v3.7	22
4.5. Uniregistry Corp. Shared Registry System (uSRS)	23
5. Acknowledgements	23
6. IANA Considerations	23
7. Security Considerations	24
8. References	25
8.1. Normative References	25
8.2. Informative References	26
Author's Address	26

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-epext-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 element 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.

- o 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.
- o 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>`, 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.
- * 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>`, 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 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:

- o 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.
- o 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.
- o A <smd:notBefore> element that contains the creation date and time of the SMD.
- o A <smd:notAfter> element that contains the expiration date and time of the SMD.
- o 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 version="1.0" encoding="UTF-8"?>
<smd:signedMark xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0"
id="smdl">
  <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:notBefore>2013-08-09T13:55:03.931Z</smd:notBefore>
<smd:notAfter>2017-07-23T22:00:00.000Z</smd:notAfter>
<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:holder>
    <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--validate</mark:label>
    <mark:label>test-etvalidate</mark:label>
    <mark:label>testetvalidate</mark:label>
    <mark:label>testvalidate</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:trademark>
</mark:mark>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
```



```
<SignedInfo>
  <CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
  <SignatureMethod
Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
  <Reference URI="#smd1">
    <Transforms>
      <Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
      </Transforms>
    <DigestMethod
Algorithm="http://www.w3.org/2001/04/xmenc#sha256" />
    <DigestValue>wgyW3nZPoEfpptlhRILKnOQnbdU6ArM7ShrAfHgDFg=</DigestValue>
  </Reference>
</SignedInfo>
  <SignatureValue>
jMu4PFyQGijBF0GWSEPFcJjmywCEqR2h4LD+ge6XQ+JnmKFFCuCZS/3SLKAX0L1w
QDF02e0Y69k2G7/LGE37X3vOflobFMlOgWja8+GMVraoto5xAd4/AF7eHukgAymD
o9toxo2h0yV4A4PmXzsU6S86XtCcUE+S/WM72nyn47zoUCzzPKHZBRYeWehVFQ+
jYRMIAMzM57HHQA+6eaXefRvtPETgU04aVIVSugc4OUAZZwbYcZrC6wOaQqqqAZi
30aPOBYbAvHMSmWSS+hFkbshomJfHxb97TD2grlYnRQIzqXk7WbHWy2SYdA+sI/Z
ipJsXNA6osTUw1CzA7jfwA==
  </SignatureValue>
  <KeyInfo>
    <X509Data>
      <X509Certificate>
MIIESTCCAzGgAwIBAgIBAjanBgkqhkiG9w0BAQsFADBIMQswCQYDVQQGEwJVUzEL
MAkGA1UECBMCQ0ExFDASBgNVBACTC0xvcyBBbmdlbGVzMRMwEQYDVQQKEwpJQ0FO
TiBUTUNIMRswGQYDVQQDEExJQ0FOTiBUTUNIIFRFU1QgQ0EwHhcNMjMwMjA4MDAw
MDAwWhcNMjMwMjA4MDAwMjMwMjA4MDAwWjBjBjBjBjBjBjBjBjBjBjBjBjBjBjBjBj
FDASBgNVBACTC0xvcyBBbmdlbGVzMRcwFQYDVQQKEw5WYXpZGF0b3IgcVElDSDEh
MB8GA1UEAxMYVmFsaWRhdG9yIFRnQ0ggVEVTVCBDRVJUMIIBIjanBgkqhkiG9w0B
AQEFAAOCAQ8AMIIBCgKCAQEAo/cwvXhbVYl0RDWWvOyeZpETVZVVCmCovUVNg/sw
WinuMgEWGvQFrz0xA04pEhXCFVv4evbUpekj5buqUlgmQyOsCKQlhoHTdPjvkC5u
pDqa5lFlkOTMaMkIQjs7aUKCmA4RG4tTTGK/Ejr1lix8/D0gHYVRldy1YPrMP+ou7
5b0VnIos+HifrAtrIv4qEqwLL4FTZAUpaCa2BmgXfy2CSRQbxD5OrlgcSa3vurh5
sPMCnXqaXmIXmQipS+DuEBqMM8tldaN7RYoJUEKRGVsNk5i9y2/7sJnlzyyUPf7v
L4GgDYqhJYWV6lDnXgx/Jd6CWxvsndf6scscQzUTEL+hywIDAQABO4H/MIH8MAwG
AlUdEwEB/wQCMAAwHQYDVR0OBBYEFpZEcIQcD/Bj2IFz/LErUo2ADJviMIGMBGNV
HSMEgYQwgYGAFO0/7kEh3FuEKS+Q/kYHaD/W6wihoWakZDBIMQswCQYDVQQGEwJV
UzELMAkGA1UECBMCQ0ExFDASBgNVBACTC0xvcyBBbmdlbGVzMRMwEQYDVQQKEwpJ
Q0FOTiBUTUNIMRswGQYDVQQDEExJQ0FOTiBUTUNIIFRFU1QgQ0GCAQEWdG9YDVR0P
AQH/BAQDAgeAMC4GA1UdHwQnMCUwI6AhoB+GHWH0dHA6Ly9jcmwuaWNhbm4ub3Jn
L3RtY2guY3J3MA0GCSqGSIb3DQEBcwUAA4IBAQB2qSy7ui+43cebKUKwWPrzz9y/
IkrMeJGKjo40n+9uekaw3DJ5EqiOf/qZ4pjBD++oR6BJCb6NQuQKwnoAz5lE4Ssu
y5+i93oTr3HfyVc4gNMIOhm1PS19l7DBKrbwbzAea/0jKWVz8vmV7TbfjxJD3AQo1R
bu5d5x6TijbdlFln05x0G0mrG7x5OUPuurihyiURpDpwH8KAHlwMcCpGXGFRtGKk
wydgyVYAty7otkl/z3bzkCVT34qPvF70sR6+QxUy8u0LzF5A/beYaZpxSYG31amL
```

```
AdXitTWfIpaIGea9lEGFM0L9+Bg7XzNn4nVLXokyEB3bgS4scG6QznX23FGk
  </X509Certificate>
  </X509Data>
  </KeyInfo>
</Signature>
</smd:signedMark>
```

NOTE: The example shown above includes white-spaces for indentation purposes. It is RECOMMENDED that SMDs do not include white-spaces between the XML elements, in order to mitigate risks of invalidating the digital signature when transferring of SMDs between applications takes place.

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.

```
<smd:encodedSignedMark
  xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPHNtZDpzaWduZWRNYXJ
rIHhtbG5zOnNtZD0idXJuOmllldGY6cGFyYW1zOnhtbDpuczpzaWduZWRNYXJrLTEuMCIGA
W... (base64 data elided for brevity) ...
PC9zbWQ6c2lnbmVkdWVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPHNtZDpzaWduZWRNYXJ
</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.

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: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"/>
</extension>
</complexContent>
</complexType>

<complexType name="issuerInfoType">
    <sequence>
        <element name="org" type="token"/>
        <element name="email" type="mark:minTokenType"/>
        <element name="url" type="token" minOccurs="0"/>
        <element name="voice" type="mark:el64Type" minOccurs="0"/>
    </sequence>
    <attribute name="issuerID" type="token" use="required"/>
</complexType>

<complexType name="encodedSignedMarkType">
    <simpleContent>
        <extension base="token">
            <attribute name="encoding" type="token" default="base64"/>
        </extension>
    </simpleContent>
</complexType>
</schema>
END

```

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"

```

```
xmlns="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">

<annotation>
  <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>
```

```
<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:el64Type" minOccurs="0"/>
<element name="fax" type="mark:el64Type" minOccurs="0"/>
<element name="email" type="mark:minTokenType" minOccurs="0"/>
</sequence>
<attribute name="entitlement" type="mark:entitlementType"/>
</complexType>

<complexType name="contactType">
  <sequence>
    <element name="name" type="token"/>
    <element name="org" type="token" minOccurs="0"/>
    <element name="addr" type="mark:addrType"/>
    <element name="voice" type="mark:el64Type"/>
    <element name="fax" type="mark:el64Type" minOccurs="0"/>
    <element name="email" type="mark:minTokenType"/>
  </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"
      minOccurs="0" 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" />
    <element name="pc" type="mark:pcType" minOccurs="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>
</schema>
```

END

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

URL: http://www.verisigninc.com/en_US/channel-resources/domain-registry-products/epp-sdks

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-epext-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-epext-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-epext-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-epext-tmch-smd.

Level of maturity: Operational Test Environment (OTE)

Coverage: Implements parsing and validation of all aspects of draft-ietf-epext-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,
<<http://newgtlds.icann.org/en/about/trademark-clearinghouse/rpm-requirements-30sep13-en.pdf>>.

[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,
<<https://www.itu.int/rec/T-REC-E.164-201011-I/en>>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997,
<<http://www.rfc-editor.org/info/rfc2119>>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004,
<<http://www.rfc-editor.org/info/rfc3688>>.

[RFC4051] Eastlake 3rd, D., "Additional XML Security Uniform Resource Identifiers (URIs)", RFC 4051, DOI 10.17487/RFC4051, April 2005,
<<http://www.rfc-editor.org/info/rfc4051>>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006,
<<http://www.rfc-editor.org/info/rfc4648>>.

- [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <<http://www.rfc-editor.org/info/rfc5322>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<http://www.rfc-editor.org/info/rfc5890>>.
- [WIPO-NICE-CLASSES] WIPO, "WIPO Nice Classification", 2015, <<http://www.wipo.int/classifications/nice/en>>.
- [WIPO.ST3] WIPO, "Recommended standard on two-letter codes for the representation of states, other entities and intergovernmental organizations", March 2007, <<http://www.wipo.int/standards/en/pdf/03-03-01.pdf>>.
- [XMLC14N] W3C Recommendation, "Exclusive XML Canonicalization Version 1.0", 2002, <<http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718>>.
- [XMLDSIG] W3C Recommendation, "XML Signature Syntax and Processing (Second Edition)", 2013, <<http://www.w3.org/TR/xmlsig-core1>>.

8.2. Informative References

- [I-D.ietf-eppext-tmch-func-spec] Lozano, G., "TMCH functional specifications", draft-ietf-eppext-tmch-func-spec-00 (work in progress), October 2015.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<http://www.rfc-editor.org/info/rfc5730>>.
- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <<http://www.rfc-editor.org/info/rfc6982>>.

Author's Address

Gustavo Lozano
ICANN
12025 Waterfront Drive, Suite 300
Los Angeles 90292
US

Phone: +1.3103015800
Email: gustavo.lozano@icann.org

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: April 15, 2016

N. Kong
J. Yao, Ed.
X. Li
CNNIC
J. Xie
CONAC
W. Tan
Cloud Registry
October 13, 2015

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

(<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. 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

14.1.	draft-kong-epp-bundle-mapping: Version 00	18
14.2.	draft-kong-epp-bundle-mapping: Version 01	18
14.3.	draft-kong-epp-bundle-mapping: Version 02	18
15.	References	18
15.1.	Normative References	18
15.2.	Informative References	19
	Authors' Addresses	20

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 bindled 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-xmlschema-1-20041028] and [W3C.REC-xmlschema-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:

- o Registered Domain Name (RDN), represents the valid domain name that users submitted for registration by the first time.
- o 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:

- o Registrar Ownership
- o Registration and Expiry Dates
- o Registrant, Admin, Billing, and Technical Contacts
- o Name Server Association
- o Domain Status
- o 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:

- o When performing a domain check, either BDN or RDN can be queried for the EPP command, and will return the same response.
- o 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.
- o 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.
- o 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.
- o 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.

- o 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.
- o 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
S:        a produced name
S:        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:

- o An <b-dn:rdn> element that contains the RDN, along with the attributes described below.
- o An OPTIONAL <b-dn:bdn> element that contains the BDN, along with the attributes described below.

The above elements contain the following attributes:

- o 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
S:          </domain:hostObj>
S:        </domain:ns>
S:      <domain:clID>ClientX</domain:clID>
```



```

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>
S:      <domain:pw>2fooBAR</domain:pw>
S:      </domain:authInfo>
S:      </domain:infData>
S:      </resData>
S:      <extension>
S:      <b-dn:infData
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: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:

- o An `<b-dn:rdn>` element that contains the RDN, along with the attributes described below.
- o An OPTIONAL `<b-dn:bdn>` element that contains the BDN, along with the attributes described below.

The above elements contain the following attribute:

- o 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:    </create>
C:    <extension>
C:      <b-dn:create
C:        xmlns:b-dn="urn:ietf:params:xml:ns:b-dn-1.0">
C:        <b-dn:rdn uLabel="U+5B9E" "U+4F8B".example>
C:          xn--fsq270a.example</b-dn:rdn>
C:        </b-dn:create>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
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:

```
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--fsq41a.example</b-dn:bdn>
S:        </b-dn:bundle>
S:      </b-dn:delData>
S:    </extension>
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.

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
```

```
</documentation>
</annotation>

<!--
Child elements found in EPP      commands.
-->
<element name="create" type="b-dn:createDataType"/>

<!--
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" />
```

```
<element name="bdn" type="b-dn:rdnType"
          minOccurs="0"   maxOccurs="unbounded" />

</sequence>
</complexType>

<complexType name="rdnType">
  <simpleContent>
    <extension base="eppcom:labelType">
      <attribute name="uLabel" type="eppcom:labelType"/>
    </extension>
  </simpleContent>
</complexType>

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

END

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:

- o 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:

- o URI: urn:ietf:params:xml:schema:b-dn-1.0
- o Registrant Contact: See the "Author's Address" section of this document.
- o 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.

- o CNNIC has implemented this extension in his EPP based Chinese domain name registration system.
- o 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

- o EPP extensiton for bundled domain name registrations.

14.2. draft-kong-epp-bundle-mapping: Version 01

- o Change the proposed category from EXP to STD.
- o Add the section of Implementation Status.
- o Refine the text, and update the examples.

14.3. draft-kong-epp-bundle-mapping: Version 02

- o Refine the texts.

15. References

15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<http://www.rfc-editor.org/info/rfc5730>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <<http://www.rfc-editor.org/info/rfc5731>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<http://www.rfc-editor.org/info/rfc5890>>.

- [RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <<http://www.rfc-editor.org/info/rfc5891>>.
- [RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.17487/RFC5892, August 2010, <<http://www.rfc-editor.org/info/rfc5892>>.
- [W3C.REC-xml-20040204]
Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium FirstEdition REC-xml-20040204", February 2004, <<http://www.w3.org/TR/2004/REC-xml-20040204>>.
- [W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028", October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>>.
- [W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028", October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>>.

15.2. Informative References

- [bundle.name]
ICANN, "Registry Services Technical Evaluation Panel (RSTEP) Report on Public Interest Registry's Request to Implement Technical Bundling in .NGO and .ONG", July 2014, <<https://www.icann.org/public-comments/rstep-technical-bundling-2014-07-29-en>>.
- [Final.Integrated.Issues.Report]
ICANN, "The IDN Variant Issues Project: A Study of Issues Related to the Management of IDN Variant TLDs", February 2012, <<http://www.icann.org/en/topics/idn/idn-vip-integrated-issues-final-clean-20feb12-en.pdf>>.

[RFC4290] Klensin, J., "Suggested Practices for Registration of Internationalized Domain Names (IDN)", RFC 4290, DOI 10.17487/RFC4290, December 2005, <<http://www.rfc-editor.org/info/rfc4290>>.

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: xieljg@conac.cn

Wil Tan
Cloud Registry
Suite 32 Seabridge House, 377 Kent St
Sydney, NSW 2000
Australia

Phone: +61 414 710899
Email: wil@cloudregistry.net

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: November 6, 2016

X. Jiagui
Teleinfo
J. Gould
VeriSign, Inc.
L. Hongyan
Teleinfo
May 5, 2016

Extensible Provisioning Protocol (EPP) China Name Verification Mapping
draft-xie-epext-nv-mapping-02

Abstract

This document describes an Extensible Provisioning Protocol (EPP) for the provisioning and management of Name Verification (NV) stored in a shared central repository in China. Specified in XML, the mapping defines EPP command syntax and semantics as applied to name verification.

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 6, 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.

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. Terminology	3
3. Definitions and Object Attributes	4
3.1. Definitions	4
3.2. Object Attributes	5
3.3. Name Verification Proofs	5
4. EPP Command Mapping	6
4.1. EPP Query Commands	6
4.1.1. EPP <check> Command	6
4.1.2. EPP <info> Command	8
4.1.3. EPP <transfer> Command	16
4.2. EPP Transform Commands	16
4.2.1. EPP <create> Command	16
4.2.2. EPP <delete> Command	22
4.2.3. EPP <renew> Command	22
4.2.4. EPP <transfer> Command	22
4.2.5. EPP <update> Command	22
4.3. Offline Review of Requested Actions	24
5. Formal Syntax	26
6. Internationalization Considerations	33
7. IANA Considerations	33
7.1. XML Namespace	33
7.2. EPP Extension Registry	34
8. Security considerations	34
9. Acknowledgements	35
10. Change History	35
10.1. draft-xie-eppext-nv-mapping-00: Version 00	35
10.2. Change from 00 to 01	35
10.3. Change from 01 to 02	35
11. Normative References	35

Authors' Addresses	36
------------------------------	----

1. Introduction

When creating a domain name which will be stored in a shared central repository, some registry administrative organizations require the verification of the domain name and the real name based on legal or policy requirements.

The domain name verification, means to verify the domain label is in compliance with laws, rules and regulations. The real name verification, means to verify that the registrant really exists and is authorized to register a domain name.

The verification of this document meets the requirements in China, but MAY be applicable outside of China.

In order to meet above requirements of the domain name registration, this document describes the Extensible Provisioning Protocol (EPP) Name Verification (NV) Mapping. 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-xmlschema-1-20041028] and [W3C.REC-xmlschema-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 mapping described in this document.

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].

"nv-1.0" in this document is used as an abbreviation for urn:ietf:params:xml:ns:nv-1.0. The XML namespace prefix "nv" 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.

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 and Object Attributes

3.1. Definitions

The following definitions are used in this document:

- o Domain Name Verification(DNV), represents the verification of the domain's label is in compliance with laws, rules and regulations.
- o Real Name Verification(RNV), represents the verification of the registrant(real name) is in compliance with laws, rules and regulations.
- o Name Verification(NV), represents DNV, RNV or both of them.
- o Verification Service Provider(VSP), collects the proof of materials for Name Verification(NV) and performs the verification.
- o Verification Code, which is described in [I-D.gould-eppext-verificationcode], is a formatted token, referred to as the Verification Code Token, that is digitally signed by a Verification Service Provider (VSP) using XML Signature in "W3C.CR-xmlsig-core2-20120124".
- o Signed Code, which is described in [I-D.gould-eppext-verificationcode], is the XML Signature format of the Verification Code.
- o Encoded Signed Code, which is described in [I-D.gould-eppext-verificationcode], is the "base64" encoded XML Signature format of the Verification Code.
- o Prohibited Name(PN), is a domain label that is prohibited from registration.
- o Restricted Name(RN), is a domain label that is restricted from registration. Additional information is needed during Domain Name Verification(DNV) to authorize the registration of a Restricted Name.

3.2. Object Attributes

An EPP NV 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.

- o Status Values. A NV object MUST always have one associated status value. The Status value can be set only by the server. The status value MAY be accompanied by a string of human-readable text that describes the rationale for the status applied to the object. The status of an object MAY change as a result of an action performed by a server operator. Status Value Descriptions:
 - * pendingCompliant. The object verification is not complete and is pending completion. Please refer to Section 4.3 for details on handling offline review of NV objects with the pendingComplaint status.
 - * compliant. The object is in compliance with the policy.
 - * nonCompliant. The object is not in compliance with the policy.
- o Dates and Times. Date and time attribute values MUST be represented in Universal Coordinated Time (UTC) using the Gregorian calendar. The extended date-time form using upper case "T" and "Z" characters defined in [W3C.REC-xmlschema-2-20041028] MUST be used to represent date-time values, as XML Schema does not support truncated date-time forms or lower case "T" and "Z" characters.
- o Authorization Information. Authorization information is associated with NV objects to facilitate query operations. Authorization information is assigned when a NV object is created, and it might be updated in the future. This specification describes password-based authorization information, though other mechanisms are possible.

3.3. Name Verification Proofs

When performing name verification, some Verification Service Providers(VSP) MAY need to collect the proof of materials to verify the real name of a registrant. The proof of materials is defined with the following enumerated values:

- o "poc" for Proof of Citizen(POC). The POC represents the citizen's identification card(ID) material.
- o "poe" for Proof of Enterprise(POE). The POE represents the Organization Code Certificate(OCC) or Business License(BL) material.
- o "poot" for Proof of Other Types(POOT). The POOT represents other certificate materials except the POC and POE.

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 NV via EPP.

4.1. EPP Query Commands

EPP provides three commands to retrieve NV information: <check> 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.

4.1.1. EPP <check> Command

The EPP <check> command is used to determine if the domain's label can be used to create a DNV object. It provides a hint that allows a client to anticipate the success or failure of creating a DNV object using the <create> command.

In addition to the standard EPP command elements, the <check> command MUST contain a <nv:check> element that identifies the nv namespace. The <nv:check> element contains the following child elements:

- o One or more <nv:name> elements that contain the domain labels to be queried.

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:      <nv:check xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
C:        <nv:name>example1</nv:name>
C:        <nv:name>example2</nv:name>
C:        <nv:name>example3</nv:name>
C:      </nv: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 <nv:chkData> element that identifies the NV namespace. The <nv:chkData> element contains one or more <nv:cd>elements that contain the following child elements:

- o A <nv:name> element that contains the queried domain label. This element MUST contain an "avail" attribute whose value indicates object availability (can it be created or not) at the moment the <check> command was completed. A value of "1" or "true" means that the object can be created. A value of "0" or "false" means that the object can not be created. This element SHOULD contain a "restricted" attribute whose value indicates this name is a RN or not, with a default value of "0". A value of "1" or "true" means that the object is a RN Name. A value of "0" or "false" means that the object is not restricted.
- o An OPTIONAL <nv:reason> element that MAY be provided when an object cannot be created. If present, this element contains server-specific text to help explain why the object cannot be created. 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:

```
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:      <nv:chkData
S:        xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:        <nv:cd>
S:          <nv:name avail="1">example1</nv:name>
S:        </nv:cd>
S:        <nv:cd>
S:          <nv:name avail="0">example2</nv:name>
S:          <nv:reason>In Prohibited Lists.</nv:reason>
S:        </nv:cd>
S:        <nv:cd>
S:          <nv:name avail="0" restricted="1">example3</nv:name>
S:        </nv:cd>
S:      </nv: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 NV object. The response to this command MAY vary depending on the identity of the querying client, and server policy towards unauthorized clients. If the querying client is the sponsoring client, all available information MUST be returned. If the querying client is not the sponsoring client but the client provides valid authorization information, all available information MUST be returned. If the querying client is not the sponsoring client and the client does not provide valid authorization information, server policy determines which OPTIONAL elements are returned.

In addition to the standard EPP command elements, the <info> command MUST contain a <nv:info> element that identifies the NV namespace. The <nv:info> element contains the following child elements:

- o A <nv:code> element that contains the Verification Code Token value. An "type" attribute MUST be used to identify the type of the query(Signed Code or Input Data). If the type is "signedCode", the successful response of the server MUST be the Signed Code of the verification code. If the type is "input", the successful response of the server MUST be the verification input data and the verification status.
- o An OPTIONAL <nv:authInfo> element that contains authorization information associated with the NV object. If this element is not provided or if the authorization information is invalid, server policy determines if the command is rejected or if response information will be returned to the client.

Example <info> command to query the signed code:

```
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:      <nv:info xmlns:nv="urn:ietf:params:xml:ns:nv-1.0"
C:        type="signedCode">
C:          <nv:code>abc-123</nv:code>
C:        </nv:info>
C:      </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <info> command to query the input data:

```
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:      <nv:info xmlns:nv="urn:ietf:params:xml:ns:nv-1.0"
C:        type="input">
C:          <nv:code>abc-123</nv:code>
C:        </nv:info>
C:      </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <info> command with authorization 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:    <info>
C:      <nv:info xmlns:nv="urn:ietf:params:xml:ns:nv-1.0"
C:        type="signedCode">
C:          <nv:code>abc-123</nv:code>
C:          <nv:authInfo>
C:            <nv:pw>2fooBAR</nv:pw>
C:          </nv:authInfo>
C:        </nv: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 <nv:infData> element that identifies the nv namespace. The <nv:infData> element has two forms based on the query type provided in the command: the Signed Code Form and the Input Form. The child element of the <nv:infData> element is defined for each form.

The Signed Code Form is returned when the command "type" attribute is set to "signedCode". The <nv:signedCode> element is used for the Signed Code Form that contains the following child elements:

- o A `<nv:code>` element that contains the Verification Code Token value of the signed code with the "type" attribute to indicate the type of NV object. The "type" attribute value of "domain" indicates a DNV object and "real-name" indicates a RNV object.
- o An OPTIONAL `<nv:status>` element that contains the current status using the status values defined in Section 3.2.
- o A `<nv:encodedSignedCode>` element include:
 - * A `<verificationCode:code>` element that is a "base64" encoded form of the digitally signed `<verificationCode:signedCode>` as defined in [I-D.gould-eppext-verificationcode].

Example <info> response of a Signed Code:

```
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:      <nv:infData xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:        <nv:signedCode>
S:          <nv:code type="domain">abc-123</nv:code>
S:          <nv:status s="compliant"/>
S:          <nv:authInfo>
S:            <nv:pw>2fooBAR</nv:pw>
S:          </nv:authInfo>
S:          <nv:encodedSignedCode>
S:ICAgICAgPHZlcmlmaWNhdGlvbkNvZGU6c2lnbmVkJ29kZQogICAgICAgIHhtbG5z
S:OnZlcmlmaWNhdGlvbkNvZGU9CiAgICAgICAgICAgICAidXJuOmllldGY6cGFyYW1zOnht
S:bDpuczp2ZXJpZmZl jYXRpb25Db2RlLTEuMCIKICAgICAgICAgIGlkPSJzaWduZWZWRD
S:b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMTIzPC92ZXJp
S:ZmZl jYXRpb25Db2RlOmNvZGU+CiAqPFNpZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3
```


S:dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CIAgIDxTaWduZWRJbmZvPgogICAg
S:PENhbm9uaWNhbGl6YXRpb25NZXRpb2QKIEFsZ29yaXRobT0iaHR0cDovL3d3dy53
S:My5vcmcvMjAwMS8xMC94bWwtZXhjLWwMxNG4jIi8+CIAgICA8U2lnbmF0dXJlTWV0
S:aG9kCiBBbGdvcmI0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveGlsZHNp
S:Zy1tb3JlI3JzYS1zaGEyNTYiLz4KICAgIDxSZWZlcmVuY2UgVVJJPSIjc2lnbmVk
S:Q29kZSI+CIAgICAgPFRyYW5zM9ybXhM+CIAgICAgIDxUcmFuc2Zvcml0KIEFsZ29y
S:aXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3B1
S:ZC1zaWduYXR1cmUiLz4KICAgICA8LlRyYW5zM9ybXhM+CIAgICAgPERpZ2Vzde1l
S:dGhvZAogQWxb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVu
S:YyNzaGEyNTYiLz4KIDxEaWdlc3RlYXNlZT53Z3lXM25aUG9FZnBwdGxoUklMS25P
S:UW5iZHRVnkFyTtdTaHJBZkhREZnPTwvRGlhZnZlcmF0dXJlVmFsdWU+CIAgICA8LlJlZmV
S:ZW5jZT4KICAgPC9TaWduZWRJbmZvPgogICAgIDxU2lnbmF0dXJlVmFsdWU+CIBqTXU0
S:UGZ5UUDpSkJGMEdXU0VQRkNKam15d0NFcVIyaDRMRctnZTZYUStKbmlLRkZDdUNa
S:Uy8zU0xLQXgwTDF3CiBRREZPMmUwWTY5azJHNY9MR0UzNlgzdG9mbG9iRk0xb0d3
S:ame4K0dNVnJhb3RvNXhBZDQvQUY3ZUhl1a2dBeW1ECiBvOXRveG9hMmgweVY0QTRQ
S:bVh6c1U2Uzg2WHRDY1VFK1MvV003Mm55bjQ3em9VQ3p6UEtIWkJSewVXZWhWrlEr
S:CiBqVWJNSUFNek01N0hIUUErNmVhWGVmUnZ0UEVUZlVPNGFWSVZTdWdjNE9VQVpa
S:d2JZY1pyQzZ3T2FRcXFXQVppCiAzMGFQT0JZYkF2SE1TbVdTUytoRmtic2hvbUpm
S:SHhiOTdURDJncmxZTnJRSXpxWGs3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z
S:VFV3MUN6QTdQZndBPT0KICAgPC9TaWduYXR1cmVWYXNlZT4KICAgPETleUluZm8+
S:CIAgICA8WduOURhdGE+CIAgICA8WduOUnlcnRpZmljYXRlPgogTUlJRvNUQ0NB
S:ekdnQXdkJkFnSUJBakFOQmdrcWhraUc5dzBCQVfzRkFEQmlNUNXN3Q1FZRFZRUUDF
S:d0pWVXpFTaogTUFrR0ExVUVDQk1DUTBFeEZEQVNCZ05WQkFjVEMweHZeUJCym1k
S:bGJHVnpNuk13RVFZRFZRUUtFd3BKUTBGTwogVGlCVVRVTklNUN3R1FZRFZRUURF
S:eEpKUTBGt1RpQlVUVU5JSUZSRlUxUWdrMEV3SGhjTk1UTXdNake0TURBdwogTURB
S:dldoY05NVGd3TWpBM01qTTFPVFU1V2pCc01Rc3dDUVlEVlFRR0V3SlZVekVMTUFr
S:R0ExVUVDQk1DUTBFeAogRkRBU0JnTlZCQWNUQzB4dmN5QkJibWRsYkdWek1SY3dG
S:UVlEVlFRS0V3NVdZV3hwWkdGMGIzSWdWRTFEU0RFAaogTUI4R0ExVUVBeE1ZVmlG
S:c2FXUmhkRz15SUZSTlEwZ2dWRVZUVkNCRFJWSlVNSU1CSWpBTkja3Foa2lHOXcw
S:QgogQVFFRkFBT0NBUThtBTUlJQkNnS0NBuUVBby9jd3ZYaGJWWwwwUkRXV3ZveWVa
S:cEVUVlplWVmNNQ292VVOZy9zdWogV2ludUlnRVdnVlFGcnoweEEwNHBFAfhDRlZ2
S:NGV2YlVwZwtKNWJ1cVUxZ21ReU9zQ0tRbGhPSFRkUGp2a0MldQogcERxYTUxRmxr
S:MFRNYUlrSVFqcZdhVUtDbUE0Ukc0dFRUR0svRWpSMWl4OC9EMGDlWVZSbGR5MVlQ
S:ck1QK291NwogNWJpVm5Jb3MrSGlmcKf0ck12NHFFcXdmTDRGVFPBVXBhQ2EyQmln
S:WGZ5MkntU1FieEQ1T3IxZ2NTYTN2dXJ0Nogc1BNQ054cWFYbUlYbVFPcFMRHVF
S:QnFNTt0bGRhTjdSWW9qVUULckdWc05rNWk5eTIvN3Nqbjf6eXlVUGY3dgogTDRH
S:Z0RZcWhKWVdWNjFEblhneC9KZDZDV3h2c25ERjZzY3NjUXpVVEVsK2h5d0lEQVFB
S:Qm80SC9NSUg4TUF3RwogQTFVZEV3RUIvd1FDTUFBd0hRWURWUjBPQkJZRUZQWkVj
S:SVFjRC9CaJJRnovTEVSdW8yQURKdmlNSUdNQmdOVgogSFNNRwdZUXdnWUdBRk8w
S:LzdrRWgzRnVFS1MrUS9rWUhhRC9XNndpaG9XYWtaREJpTVFzd0NRWURWUWFHRXdk
S:VgogVXpFTE1Ba0dBMVVFQ0JNQ1EwRXhGREFTQmdOVkjbY1RDMHh2Y3lCQmJtZGxi
S:RlZ6TVJND0VRWURWUWFLRXdwSgogUTBGt1RpQlVUVU5JTVJzd0dRWURWUWFERXhk
S:SlEwRk9UaUJVVVFVOSU1GUkZVMVFnUTBHQ0FRRXdeZ1lEVlIwUAogQVFILOJBUURB
S:Z2VBTUM0R0ExVWRId1FuTUNvd0k2QWwhVjQitHSFdoMGRIQTZMeTlqY213dWFXThhi
S:bTRlYjNkbgogTDNSdFkyZ3VZM0pzTUEwR0NTcUdTSWIzRFFFQkN3VUFBNElCQVFC
S:MnFTeTdlaSs0M2NlYktVS3dXUHJ6ejl5LwogSWtyTWVKR0tqbzQwbis5dWVrYXcz
S:REolRXFPt2YvcVo0cGpCRCSrb1I2QkpdYjZOUXVRS3dub0F6NWxFNFnzdQogeTUR
S:aTkzblQzSGZ5VmM0Z05NSW9IbTFQUze5bDdEQktyYndiekFlYS8waktXVnpydm1W

```

S:N1RCZmp4RDNBW8xUgogYlU1ZEJyNklqYmRMRmxuTzV4MEcwbXJHN3g1T1VQdXVy
S:aWh5aVVSceZEchdIOEtBSDF3TWNDcFhHWEZSdEdLawogd3lkZ3lWWUF0eTdvdGts
S:L3ozYlprQ1ZUMzRnUHZGNzBzUjYrUXhVeTh1MEx6RjVBL2JlWWFacHhTWUczMWft
S:TAogQWRYaXRUV0ZpcGFJR2VhOWxFR0ZNMEw5K0JnN1h6Tm40blZMWG9reUVCm2Jn
S:UzRzY0c2UXpuWDIzRkdrCiAgIDwvWduOUnlcnRpZmljYXRlPgogICA8L1glMD1E
S:YXRhPgogICA8L0tleUluZm8+CiAgPC9TaWduYXRlcmU+CgkJPc92ZXJpZmljYXRp
S:b25Db2RlOnNpZ25lZENvZGU+Cg==
S:      </nv:encodedSignedCode>
S:      </nv:signedCode>
S:      </nv:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>

```

The Input Code Form is returned when the command "type" attribute is set to "input". The <nv:input> element is used for the Input Form and contains a choice of two different child elements dependent on the type of NV object that matches the <nv:code> in the command. The <nv:dnv> child element is used for a DNV object and the <nv:rnv> child element is used for a RNV object.

The <nv:dnv> element is used for a DNV object and contains the following child elements:

- o A <nv:name> element that contains the label of the domain.
- o An OPTIONAL <nv:rnvCode> element containing the Verification Code Token value of a RNV object used for verification of a Restricted Name.

Example <info> response of a DNV:

```
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:      <nv:infData xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:        <nv:input>
S:          <nv:dnv>
S:            <nv:name>example</nv:name>
S:          </nv:dnv>
S:          <nv:authInfo>
S:            <nv:pw>2fooBAR</nv:pw>
S:          </nv:authInfo>
S:        </nv:input>
S:      </nv:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The <nv:rnv> element is used for a RNV object. The "role" attribute MUST be used to identify the role of the RNV object with the possible values of "person" or "org".

The <nv:rnv> element contains the following child elements:

- o A <nv:name> element that contains the full name of the contact.
- o A <nv:num> element that contains the citizen or the organization ID of the contact.
- o A <nv:proofType> element that contains the proof material type of the contact based on the enumerated values defined in Name Verification Proofs (Section 3.3).
- o Zero or more <nv:document> elements that contains the following child elements:
 - * A <nv:fileType> element contains the type of the file.

- * A <nv:fileContent> element contains the "base64" encoded content of the file.

Example <info> response of a RNV person:

```
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:      <nv:infData xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:        <nv:input>
S:          <nv:rnv role="person">
S:            <nv:name>John Xie</nv:name>
S:            <nv:num>1234567890</nv:num>
S:            <nv:proofType>poc</nv:proofType>
S:            <nv:document>
S:              <nv:fileType>jpg</nv:fileType>
S:              <nv:fileContent>EABQRAQAAAAAAAAAAAAAAAAAAD
S:            </nv:fileContent>
S:          </nv:document>
S:        </nv:rnv>
S:        <nv:authInfo>
S:          <nv:pw>2fooBAR</nv:pw>
S:        </nv:authInfo>
S:      </nv:input>
S:    </nv:infData>
S:  </resData>
S:  <trID>
S:    <clTRID>ABC-12345</clTRID>
S:    <svTRID>54322-XYZ</svTRID>
S:  </trID>
S: </response>
S:</epp>
```

Example <info> response of a RNV organization:

```
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:      <nv:infData xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
```

```
S:      <nv:input>
S:      <nv:rnv role="org">
S:      <nv:name>John Xie</nv:name>
S:      <nv:num>1234567890</nv:num>
S:      <nv:proofType>poe</nv:proofType>
S:      <nv:document>
S:      <nv:fileType>jpg</nv:fileType>
S:      <nv:fileContent>EABQRAQAAAAAAAAAAAAAAAAAAD
S:      </nv:fileContent>
S:      </nv:document>
S:      </nv:rnv>
S:      <nv:authInfo>
S:      <nv:pw>2fooBAR</nv:pw>
S:      </nv:authInfo>
S:      </nv:input>
S:      </nv:infData>
S:      </resData>
S:      <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:      </trID>
S:      </response>
S: </epp>
```

A server with a different information-return policy MAY provide less information in a response for an unauthorized client.

An EPP error response MUST be returned if an <info> command cannot be processed for any reason.

4.1.3. EPP <transfer> Command

Transfer semantics do not apply to Name Verification (NV) objects, so there is no mapping defined for the EPP <transfer> command.

4.2. EPP Transform Commands

EPP provides five commands to transform NV 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.

4.2.1. EPP <create> Command

The EPP <create> command provides a transform operation that allows a client to create an NV object. In addition to the standard EPP command elements, the <create> command MUST contain a <nv:create>

element that identifies the NV namespace. The <nv:create> elements contains a choice of two different child elements dependent on the type of NV object to create. The <nv:dnv> child element is used to create a DNV object and the <nv:rnv> child element is used to create a RNV object. AN <nv:authInfo> element contains authorization information to be associated with the NV object.

- o The <nv:dnv> element is used for a DNV object and contains the following child elements:
 - * A <nv:name> element that contains the label of the domain.
 - * An OPTIONAL <nv:rnvCode> element containing the Verification Code Token value of a RNV object used for verification of a Restricted Name.
- o The <nv:rnv> element is used for a RNV object. The "role" attribute MUST be used to identify the role of the RNV object with the possible values of "person" or "org". The <nv:rnv> element contains the following child elements:
 - * A <nv:name> element that contains the full name of the contact.
 - * A <nv:num> element that contains the citizen or the organization ID of the contact.
 - * A <nv:proofType> element that contains the proof material type of the contact based on the enumerated values defined in Name Verification Proofs (Section 3.3).
 - * Zero or more <nv:document> elements that contains the following child elements:
 - + A <nv:fileType> element contains the type of the file.
 - + A <nv:fileContent> element contains the "base64" encoded content of the file.

Example <create> command for a DNV 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:    <create>
C:      <nv:create xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
C:        <nv:dnv>
C:          <nv:name>example</nv:name>
C:        </nv:dnv>
```

```

C:      <nv:authInfo>
C:      <nv:pw>2fooBAR</nv:pw>
C:      </nv:authInfo>
C:      </nv:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>

```

Example <create> command for a RNV person 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:    <create>
C:      <nv:create xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
C:        <nv:rnv role="person">
C:          <nv:name>John Xie</nv:name>
C:          <nv:num>1234567890</nv:num>
C:          <nv:proofType>poe</nv:proofType>
C:          <nv:document>
C:            <nv:fileType>jpg</nv:fileType>
C:            <nv:fileContent>EABQRAQAAAAAAAAAAAAAAAAAAAAAD
C:              </nv:fileContent>
C:          </nv:document>
C:        </nv:rnv>
C:        <nv:authInfo>
C:          <nv:pw>2fooBAR</nv:pw>
C:        </nv:authInfo>
C:      </nv:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>

```

Example <create> command for an RNV organization:

```

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:      <nv:create xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
C:        <nv:rnv role="org">
C:          <nv:name>John Xie</nv:name>
C:          <nv:num>1234567890</nv:num>
C:          <nv:proofType>poe</nv:proofType>
C:          <nv:document>
C:            <nv:fileType>jpg</nv:fileType>

```

```

C:      <nv:fileContent>EABQRAQAAAAAAAAAAAAAAAAAAAAAD
C:      </nv:fileContent>
C:      </nv:document>
C:      </nv:rnv>
C:      <nv:authInfo>
C:      <nv:pw>2fooBAR</nv:pw>
C:      </nv:authInfo>
C:      </nv:create>
C:      </create>
C:      <clTRID>ABC-12345</clTRID>
C:      </command>
C: </epp>

```

When a <create> command has been processed successfully, the EPP <resData> element MUST contain a child <nv:creData> element that identifies the nv namespace. <nv:creData> element contains the either a <nv:success> element on success or a <nv:failed> element on failure.

- o The <nv:success> element contains the following child elements:
 - * A <nv:code> element that contains the id of the verification code with the required "type" attribute that defines the type of the verification code.
 - * A <nv:status> element that contains the current status using the status values defined in Section 3.2.
 - * A <nv:crDate> element that contains the date and time of nv object creation.
 - * A <nv:encodedSignedCode> element include:
 - + A <verificationCode:code>element that is a "base64" encoded form of the digitally signed <verificationCode:signedCode> as defined in [I-D.gould-eppext-verificationcode].
- o The <nv:failed> element contains the following child elements:
 - * A <nv:status> element that contains the current status using the status values defined in Section 3.2.
 - * A <nv:msg> element containing a human-readable description of the reason of the failure. The language of the response is identified via an OPTIONAL "lang" attribute. If not specified, the default attribute value MUST be "en" (English).

Example <create> response of success:


```
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:     <nv:creData xmlns:nv="urn:iETF:params:xml:ns:nv-1.0">  
S:       <nv:succes>  
S:         <nv:code type="domain">abc-123</nv:code>  
S:         <nv:status s="compliant"/>  
S:         <nv:crDate>2015-08-17T22:00:00.0Z</nv:crDate>  
S:         <nv:encodedSignedCode>  
S:ICAgICAgPHZlcmhmaWNhdGlvbkNvZGU6c2lnbmVkJQ29kZQogICAgICAgIHhtbG5z  
S:OnZlcmhmaWNhdGlvbkNvZGU9CiAgICAgICAgICAidXJuOmlldGY6cGFyYWwzOnht  
S:bDpuczp2ZXJpZmlljYXRpb25Db2RlLTUUMCIKICAgICAgICAgIGlkPSJzaWduZWRD  
S:b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMTIzPC92ZXJp  
S:ZmlljYXRpb25Db2RlOmNvZGU+CiAgPFNPZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3  
S:dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CIAgIDxTaWduZWRRJmZvPgogICAg  
S:PENhbm9uaWNhbgI6YXRpb25NZXRRob2QKIEFsZ29yaXRobT0iaHR0cDovL3d3dy53  
S:My5vcmcvMjAwMS8xMC94bWwtZXhjLWMwNG4jiI8+CIAgICA8U2lnbmF0dXJlTWV0  
S:Zyag9kCiBBbGdvcml0aG0Imh0dHA6Ly93d3cudzdMub3JnLzIwMDExMDQxZW50  
S:S:ay1tb3JlIjZYS1zaGEyNTYiLz4KICAgIDxSZWZlcmluY2UgVVJJPSIjc2lnbmVkJQ29kZSI+CIAgICAgPFryYW5zM9ybXM+CIAgICAgIDxUcmFuc2Zvcml0KIEFsZ29y  
S:S:aXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3Bl  
S:S:ZC1zaWduYXRlcmUiLz4KICAgICA8L1RyYW5zM9ybXM+CIAgICAgPERpZ2VzeE1l  
S:S:dGhvZAogQWxnbn3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVu  
S:S:YyNzaGEyNTYiLz4KIDxEaWdlc3RlYXNlZT53Z3lXM25aUG9fZnBwdGxoUklMS25P  
S:S:UW5iZHRVNkFyTTdTahJBZkhREZnPTwvRGlnZXN0VmFsdWU+CIAgICA8L1JlZmVy  
S:S:ZW5jZT4KICAgPC9TaWduZWRRJmZvPgogICA8U2lnbmF0dXJlVmFsdWU+CIBqTXU0  
S:S:UGZ5UUdpSkJGMEdXU0VQRkNKam15d0NFcVIyaDRMRCTnZTZYUStKbmlLRkZDdUNa  
S:S:Uy8zU0xLQXgwTDF3CiBRREZPMmUwWTY5azJHNy9MR0UzNlgzdk9mbG9iRk0xb0d3  
S:S:ame4K0dNVnJhb3RvNXhBZDQvQUY3ZUhla2dBew1ECiBvOXRveG9hMmgweVY0QTRQ  
S:S:bVh6c1U2uzg2WHRDYL1VFk1MvV003Mm55bjQ3em9VQ3p6UEtIWkjSeWVXZWWhWRLEr  
S:S:CibqWVJNSUFnek01N0hIUUERnmVhVGVMUnZ0UEUVUZ1VPNGFWSVZTDwdjNE9VQVpa  
S:S:d2JZY1pyQzZ3T2FRcXFxQVppCiAzMGFQT0JZYkF2SE1TbvDtUytoRmtic2hvbUpm  
S:S:SHhiOTdURDJncmxZTnJRSPxWGS3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z  
S:S:VFV3MUN6QTdqZndBPT0KICAgPC9TaWduYXRlcmVWYXNlZT4KICAgPETleUluZm8+  
S:S:CIAgICA8WDUwOURhdGE+CIAgICA8WDUwOUNlcnRpZmlljYXRlpGogTUljRVNUQ0NB  
S:S:ekdnQXdjQkFnSUJBakFOQmdrcWhraUc5dzBCQVFrZkFEQmlNUXN3Q1FZRFRZRUUDF  
S:S:d0pWVXPFTaogTUFR0EXVUVDQk1DUTBFEEZEQVNCZ05WQkFjVEFMweHZjeUJCymIk  
S:S:bGJHVNpNUK13RVFRFRZRUUtfd3BKUTBGTwogVGlCVVRVTklNUUN3R1FZRFRZRUURF  
S:S:eEPKUTBGTLRpQ1VUVU5JSUZSRlUxUWdrMEV3SGhjTk1UTXDnake0TURBdwogTURB  
S:S:dldoY05NVGd3TWpBM01qtTFPVFU1V2pCc01Rc3dDUVL1EVLFRR0V3SlZVekVMTUFR  
S:S:R0EXVUVDQk1DUTBFEEagRkrBU0JnTlZCQQWUZF0B4dmN5QkjiBWRsYkdWek1SY3dg  
S:S:UVLEvlFRS0V3NVdZV3hwWkdGMGIzSWdWRNTEUQzFAogTUJI4R0EXVUUBee1ZlMOG  
S:S:c2FXUhmhkRz15SUZSTLeWZ2dWRVZUVVNCRFJWSlVSuFSAogTUjCSWPBTkja3Foa21HOXCw  
S:S:QogQOVFFRkFBT0NBUTHBTULJOkNNs0NBUUVBby9jd3ZYAGJWWWWWUkRXV3ZveWVw
```

```
S:CEVUVlPwMnNQ292VVZ0Zy9zdwogV2ludUlnRVdnVlFGcnoweEEwNHBFAfhDRlZ2
S:NGV2YlVwZWtKNWJlcvUxZ21ReU9zQ0trBgHPSFRkUGp2a0MldQogcERxYTUxRmxr
S:MFRNYUlrSVFqcZdhVUtDbUE0Ukc0dFRUR0svRWpSMWl40C9EMGdIWVZSbGR5MVlQ
S:cklQK29lNwogNWWJPVm5Jb3MrSGlmckf0ckl2NHFFcXdMTDRGVfPbVXBhQ2EyQm1n
S:WGZ5MkNTUlfieEQ1T3IxZ2NTYTn2dXJoNQogclBNQ054cWfYbUlYbVfPcFMrRHVF
S:QnFNTTh0bGRhTjdsSWW9qVUVLckdWc05rNwK5eTivN3Nqbjf6eXlVUGY3dgogTDRH
S:ZORZcWhKWvdWnjFEblhneC9KZDZDV3h2c25ERjZzY3NjUXpVVEVVsK2h5d0lEQVFB
S:Qm80SC9NSUg4TUF3RwogQTFVZEV3RUIvd1FDTUFBd0hRWURWUjBPQkJZRUZQWkvj
S:SVFjRC9CaJJJRnrvTEVSDW8yQURKdmlNSUdNQmdOVgogSFNNRwdZUXdnWUDBRk8w
S:LzdrRWGzRnVfSlMrUS9rWUhhRC9XNndpaG9XYWtaREJpTVfVzd0NRWURWUVFHRXdk
S:VgogVXpFTElBa0dBmVVFQ0JNQlEwRXhGREFTQmdOVkjbY1RDMHh2Y3lCQmJtZGxi
S:RlZ6TVJNd0VRWURWUVFLRXdwSgogUTBGT1RpQlVUVU5JTVJZzd0dRWURWUVFERXhk
S:SlEwRk9UaUJVVfVOSUlGukZVMVFnUTBHQ0FRRXdeZ1lEVlIwUAogQVFIL0JBUURB
S:Z2VBTUM0R0ExVWRidlFuTUNvd0k2QWhvQitHSFdoMGRIQTZMeTlqY213dWFXtmhi
S:bTR1YjNkbgogTDNSdFkyZ3VZM0pzTUEwR0NTcUdTswIzRFFFQkN3VUFBNElCQVFC
S:MnFTeTd1aSs0M2NlYktVS3dXUHJ6ejl5LwogSWtyTWVKR0tqbzQwbis5dWVrYXcz
S:REo1RXFpT2YvcVo0cGpCRCSrb1I2QkpDYjZOUXVRS3dub0F6NWxFNFnzdQogeTUR
S:aTkzb1QzSGZ5VmM0Z05NSW9IbTFQUZe5bDdeQktyYndiekFlYS8waktXVnpydm1W
S:NlRCZmp4RDNBuW8xUgogYlU1ZEJyNklqYmRMRmxuTzV4MEcwbXJHN3g1TlVQdXVy
S:aWh5aVVSceZEchdIOEtBSDF3TWNDcFhHWEZSdeDLaWogd3lkZ3lWWUF0eTdvdGts
S:L3ozYlprQlZUMzRnUHZGNZBzUjYrUXhVeThlMEX6RjVBL2JlWWFacHhTWUcZMWfT
S:TAoZQWRyAXRUvZpcGFJR2VhOWxFR0ZNMEw5K0JnN1h6Tm40blZMWG9reUVCm2Jn
S:UzRzY0c2UXpuWDIzRkdrCiAgIDwvWDUwOUNlcnRpZmljYXRlPgogICA8Llg1MDle
S:YXRhPgogICA8L0tleUluZm8+CiAgPC9TaWduYXRlcmlU+CgkJPc92ZXJpZmljYXRp
S:b25Db2RlOnNpZ25lZENvZGU+Cg==
S:      </nv:encodedSignedCode>
S:      </nv:success>
S:      </nv:creData>
S:    </resData>
S:  <trID>
S:    <clTRID>ABC-12345</clTRID>
S:    <svTRID>54321-XYZ</svTRID>
S:  </trID>
S: </response>
S:</epp>
```

Example <create> response of failed:

```
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:      <nv:creData xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:        <nv:failed>
S:          <nv:status s="nonCompliant"/>
```

```
S:      <nv:msg lang="en">
S:      The name of the object is not correct.
S:      </nv:msg>
S:      </nv:failed>
S:      </nv:creData>
S:      </resData>
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 <create> command cannot be processed for any reason.

4.2.2. EPP <delete> Command

Delete semantics do not apply to Name Verification (NV) objects, so there is no mapping defined for the EPP <delete> command.

4.2.3. EPP <renew> Command

Renew semantics do not apply to Name Verification (NV) objects, so there is no mapping defined for the EPP <renew> command.

4.2.4. EPP <transfer> Command

Transfer semantics do not apply to Name Verification (NV) 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 NV object. In addition to the standard EPP command elements, the <update> command MUST contain a <nv:update> element that identifies the NV namespace. The <nv:update> element contains the following child elements:

- o A <nv:code> element that contains the code of the a NV object to be updated.
- o A <nv:chg> element that contains object attribute values to be changed.

A <nv:chg> element contains the following child elements:

- o A <nv:authInfo> element that contains authorization information associated with the NV object. This mapping includes a password-based authentication mechanism, but the schema allows new mechanisms to be defined in new schemas.

Example <update> 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:    <update>
C:      <nv:update xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
C:        <nv:code>abc-123</nv:code>
C:        <nv:chg>
C:          <nv:authInfo>
C:            <nv:pw>2BARfoo</nv:pw>
C:          </nv:authInfo>
C:        </nv:chg>
C:      </nv:update>
C:    </update>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</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.

4.3. Offline Review of Requested Actions

Commands are processed by a server in the order they are received from a client. Though an immediate response confirming receipt and processing of the command is produced by the server, a server operator MAY perform an offline review of requested transform commands before completing the requested action. In such situations, the response from the server MUST clearly note that the transform command has been received and processed but that the requested action is pending. The status of the corresponding object MUST clearly reflect processing of the pending action. The server MUST notify the client when offline processing of the action has been completed.

Examples describing a <create> command that requires offline review are included here. Note the result code and message returned in response to the <create> command.

```
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:      <nv:creData
S:        xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:        <nv:pending>
S:          <nv:code type="domain">abc-123</nv:code>
S:          <nv:status s="pendingCompliant"/>
S:          <nv:crDate>2015-09-03T22:00:00.0Z</nv:crDate>
S:        </nv:pending>
S:      </nv:creData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The status of the NV object after returning this response MUST include "pendingCompliant". The server operator reviews the request offline, and informs the client of the outcome of the review either by queuing a service message for retrieval via the <poll> command or by using an out-of-band mechanism to inform the client of the outcome of the review.

The service message MUST contain text that describes the notification in the child <msg> element of the response <msgQ> element. In addition, the EPP <resData> element MUST contain a child <nv:panData> element that identifies the NV namespace. The <nv:panData> element contains the following child elements:

A <nv:code> element that contains the id of the verification code with the required "type" attribute that defines the type of the verification code.

A <nv:paStatus> element that contains the current status descriptors associated with the NV.

A <nv:msg> element containing a human-readable description of the result. The language of the response is identified via an OPTIONAL "lang" attribute. If not specified, the default attribute value MUST be "en" (English).

A <nv:paDate> element that contains the date and time describing when review of the requested action was completed.

Example "review completed" service message:

```
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>2015-09-04T22:01:00.0Z</qDate>
S:      <msg>Pending action completed successfully.</msg>
S:    </msgQ>
S:    <resData>
S:      <nv:panData
S:        xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:        <nv:code type="domain">abc-123</nv:code>
S:        <nv:paStatus s="compliant"/>
S:        <nv:msg>The object has passed verification,
S:          signed code was generated.</nv:msg>
S:        <nv:paDate>2015-09-04T22:00:00.0Z</nv:paDate>
S:      </nv:panData>
S:    </resData>
S:    <trID>
S:      <clTRID>BCD-23456</clTRID>
S:      <svTRID>65432-WXY</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5. Formal Syntax

An EPP object NV 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:nv-1.0"
  xmlns:nv="urn:ietf:params:xml:ns:nv-1.0"
  xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
  xmlns:verificationCode=
    "urn:ietf:params:xml:ns:verificationCode-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
```

```
    elementFormDefault="qualified">

<annotation>
  <documentation>
    Extensible Provisioning Protocol v1.0
    Name Verification provisioning schema.
  </documentation>
</annotation>

<!--
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:verificationCode-1.0"/>

<!--
Child elements found in EPP commands.
-->
<element name="check" type="nv:mNameType"/>
<element name="create" type="nv:createType"/>
<element name="info" type="nv:infoType"/>
<element name="update" type="nv:updateType"/>

<!--
Child element of <check> command.
-->
<complexType name="mNameType">
  <sequence>
    <element name="name" type="eppcom:labelType"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>

<!--
Child elements of the <create> command.
-->
<complexType name="createType">
  <sequence>
    <choice>
      <element name="dnv" type="nv:dnvType"/>
      <element name="rnv" type="nv:rnvType"/>
    </choice>
    <element name="authInfo" type="nv:authInfoChgType"/>
  </sequence>
</complexType>
```



```
<complexType name="dnvType">
  <sequence>
    <element name="name" type="eppcom:labelType"/>
    <element name="rnvCode" type="token"
      minOccurs="0"/>
  </sequence>
</complexType>

<complexType name="rnvType">
  <sequence>
    <element name="name" type="nv:nameType"/>
    <element name="num" type="nv:nameType"/>
    <element name="proofType" type="nv:proofEumType"/>
    <element name="document" type="nv:documentType"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="role" type="nv:roleType"
    default="person"/>
</complexType>

<simpleType name="proofEumType">
  <restriction base="token">
    <enumeration value="poc"/>
    <enumeration value="poe"/>
    <enumeration value="poot"/>
  </restriction>
</simpleType>

<simpleType name="roleType">
  <restriction base="token">
    <enumeration value="person"/>
    <enumeration value="org"/>
  </restriction>
</simpleType>

<simpleType name="nameType">
  <restriction base="token">
    <minLength value="1"/>
    <maxLength value="255"/>
  </restriction>
</simpleType>

<complexType name="documentType">
  <sequence>
    <element name="fileType" type="nv:fileType"/>
    <element name="fileContent" type="base64Binary"/>
  </sequence>
</complexType>
```

```
<simpleType name="fileType">
  <restriction base="token">
    <enumeration value="pdf"/>
    <enumeration value="jpg"/>
  </restriction>
</simpleType>

<!--
Child elements of the <info> command.
-->
<complexType name="infoType">
  <sequence>
    <element name="code" type="token"/>
    <element name="authInfo" type="nv:authInfoChgType"
      minOccurs="0"/>
  </sequence>
  <attribute name="type" type="nv:infoFormType"
    default="signedCode"/>
</complexType>

<simpleType name="infoFormType">
  <restriction base="token">
    <enumeration value="input"/>
    <enumeration value="signedCode"/>
  </restriction>
</simpleType>

<!--
Child elements of the <update> command.
-->
<complexType name="updateType">
  <sequence>
    <element name="code" type="token"/>
    <element name="chg" type="nv:chgType"/>
  </sequence>
</complexType>

<!--
Data elements that can be changed.
-->
<complexType name="chgType">
  <sequence>
    <element name="authInfo" type="nv:authInfoChgType"/>
  </sequence>
</complexType>
```

```
<!--
Data elements of authInfoChgType.
-->
<complexType name="authInfoChgType">
  <choice>
    <element name="pw" type="eppcom:pwAuthInfoType"/>
    <element name="ext" type="eppcom:extAuthInfoType"/>
  </choice>
</complexType>

<!--
Child response elements.
-->
<element name="chkData" type="nv:chkDataType"/>
<element name="creData" type="nv:creDataType"/>
<element name="panData" type="nv:panDataType"/>
<element name="infData" type="nv:infDataType"/>

<!--
<check> response elements.
-->
<complexType name="chkDataType">
  <sequence>
    <element name="cd" type="nv:checkType"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>

<complexType name="checkType">
  <sequence>
    <element name="name" type="nv:checkNameType"/>
    <element name="reason" type="eppcom:reasonType"
      minOccurs="0"/>
  </sequence>
</complexType>

<complexType name="checkNameType">
  <simpleContent>
    <extension base="eppcom:labelType">
      <attribute name="avail" type="boolean"
        use="required"/>
      <attribute name="restricted" type="boolean"
        default="0"/>
    </extension>
  </simpleContent>
</complexType>

<!--
```

```
<create> response elements.
-->
<complexType name="creDataType">
  <choice>
    <element name="success" type="nv:successtype"/>
    <element name="failed" type="nv:failedType"/>
    <element name="pending" type="nv:pendingType"/>
  </choice>
</complexType>

<complexType name="successtype">
  <sequence>
    <element name="code" type="nv:verificationCodeType"/>
    <element name="status" type="nv:statusType"/>
    <element name="crDate" type="dateTime"/>
    <element name="encodedSignedCode"
      type="verificationCode:encodedSignedCodeType"/>
  </sequence>
</complexType>

<complexType name="failedType">
  <sequence>
    <element name="status" type="nv:statusType"/>
    <element name="msg" type="nv:msgType"/>
  </sequence>
</complexType>

<complexType name="pendingType">
  <sequence>
    <element name="code" type="nv:verificationCodeType"/>
    <element name="status" type="nv:statusType"/>
    <element name="crDate" type="dateTime"/>
  </sequence>
</complexType>

<!--
Pending action notification response elements.
-->
<complexType name="panDataType">
  <sequence>
    <element name="code" type="nv:verificationCodeType"/>
    <element name="paStatus" type="nv:statusType"/>
    <element name="msg" type="nv:msgType"/>
    <element name="paDate" type="dateTime"/>
  </sequence>
</complexType>

<!--
```

```
<info> response elements.
-->
<complexType name="infDataType">
  <sequence>
    <choice>
      <element name="signedCode" type="nv:signedCodeType"/>
      <element name="input" type="nv:createType"/>
    </choice>
  </sequence>
</complexType>

<complexType name="signedCodeType">
  <sequence>
    <element name="code" type="nv:verificationCodeType"/>
    <element name="status" type="nv:statusType"
      minOccurs="0"/>
    <element name="authInfo" type="nv:authInfoChgType"/>
    <element name="encodedSignedCode"
      type="verificationCode:encodedSignedCodeType"/>
  </sequence>
</complexType>

<complexType name="verificationCodeType">
  <simpleContent>
    <extension base="token">
      <attribute name="type" type="token"
        use="required"/>
    </extension>
  </simpleContent>
</complexType>

<!--
Status is a combination of attributes and an optional
human-readable message that may be expressed in languages other
than English.
-->
<complexType name="statusType">
  <simpleContent>
    <extension base="normalizedString">
      <attribute name="s" type="nv:statusValueType"
        use="required"/>
      <attribute name="lang" type="language"
        default="en"/>
    </extension>
  </simpleContent>
</complexType>

<complexType name="msgType">
```

```
<simpleContent>
  <extension base="normalizedString">
    <attribute name="lang" type="language"
      default="en"/>
  </extension>
</simpleContent>
</complexType>

<simpleType name="statusValueType">
  <restriction base="token">
    <enumeration value="compliant"/>
    <enumeration value="nonCompliant"/>
    <enumeration value="pendingCompliant"/>
  </restriction>
</simpleType>

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

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, 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 assign the following two URI.

Registration request for the NV namespace:

- o URI: urn:ietf:params:xml:ns:nv-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 NV XML schema:

- o URI: urn:ietf:params:xml:schema:nv-1.0
- o Registrant Contact: See the "Author's Address" section of this document.
- o 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: Extensible Provisioning Protocol (EPP) China Name Verification Mapping

Document status: Informational

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

8. Security considerations

Verification Code Tokens are digitally signed using XML Signature technology. The security considerations described in Section 12 of the W3C XML Signature Syntax and Processing Candidate Recommendation [W3C.CR-xmlsig-core2-20120124] apply to this specification as well. The object mapping 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.

9. Acknowledgements

The authors especially thank the author of [RFC5730].

Useful comments and contributions were made by TBD.

10. Change History

RFC Editor: Please remove this section.

10.1. draft-xie-eppext-nv-mapping-00: Version 00

- o First draft.

10.2. Change from 00 to 01

1. Made the <nv:code> element of the panDataType and pendingType require the "type" attribute in the XML schema.
2. Fixed the XML schema to include the <nv:rncCode> OPTIONAL element.
3. Added the support for the OPTIONAL "lang" attribute for the <nv:msg> element of the <nv:failed> and <nv:panData> elements.

10.3. Change from 01 to 02

- o Ping update.

11. Normative References

- [I-D.gould-eppext-verificationcode]
Gould, J., "Verification Code Extension for the Extensible Provisioning Protocol (EPP)", draft-gould-eppext-verificationcode-03 (work in progress), March 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<http://www.rfc-editor.org/info/rfc5730>>.

- [RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <<http://www.rfc-editor.org/info/rfc7451>>.
- [W3C.CR-xmlsig-core2-20120124]
Bartel, M., Boyer, J., Fox, B., LaMacchia, B., and E. Simon, "XML Signature Syntax and Processing Version 2.0", W3C Candidate Recommendation 24 January 2012", January 2012, <<http://www.w3.org/TR/2012/CR-xmlsig-core2-20120124/>>.
- [W3C.REC-xml-20040204]
Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium FirstEdition REC-xml-20040204", February 2004, <<http://www.w3.org/TR/2004/REC-xml-20040204>>.
- [W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028", October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>>.
- [W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028", October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>>.

Authors' Addresses

Xie Jiagui
Teleinfo
1#-21,gaolizhang Street,Haidian District
Beijing, Beijing 100095
China

Phone: +86 10 5884 6931
Email: xiejiagui@teleinfo.cn

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

Email: jgould@verisign.com
URI: <http://www.verisign.com>

Liu Hongyan
Teleinfo
1#-21,gaolizhang Street,Haidian District
Beijing, Beijing 100095
China

Phone: +86 10 5884 6931
Email: liuhongyan@teleinfo.cn

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2016

L. Zhou
N. Kong
J. Wei
X. Lee
CNNIC
J. Gould
VeriSign, Inc.
March 9, 2016

Reseller Extension for the Extensible Provisioning Protocol (EPP)
draft-zhou-epext-reseller-03

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 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 September 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.

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. Acknowledgement	14
10. References	14
10.1. Normative References	14
10.2. Informative References	15
Appendix A. Change Log	16
Authors' Addresses	16

1. Introduction

Domain resellers are the individuals or companies act 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-epext-reseller-mapping]. All reseller objects are identified by a server-unique identifier

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-epext-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:

```
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>example.com</domain:name>
C:        <domain:authInfo>
C:          <domain:pw>fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:info>
C:    </info>
C:    <clTRID>ngcl-mIFICBNP</clTRID>
C:  </command>
C:</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 <resellerext:infData> element that identifies the extension namespace if the domain object has data associated with this extension and based on its service policy. The <resellerext:infData> element contains the following child elements:

- o A <resellerext:id> element that contains the identifier of the reseller of a sponsoring registrar.

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 lang="en-US">Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>jdl234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="billing">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:ns>
S:          <domain:hostObj>ns1.example.com</domain:hostObj>
S:        </domain:ns>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2015-02-06T04:01:21.0Z</domain:crDate>
S:        <domain:exDate>2018-02-06T04:01:21.0Z</domain:exDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <rgp:infData xmlns:rgp="urn:ietf:params:xml:ns:rgp-1.0">
S:        <rgp:rgpStatus s="addPeriod"/>
S:      </rgp:infData>
S:      <resellerext:infData xmlns:resellerext="urn:ietf:params:xml:ns:resellere
xt-1.0">
S:        <resellerext:id>myreseller</resellerext:id>
S:      </resellerext:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ngcl-IvJjzMZc</clTRID>
S:      <svTRID>test142AWQONJZ</svTRID>
S:    </trID>
S:  </response>
S:</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:

- o A <resellerext:id> element that contains the identifier of the reseller of a sponsoring registrar.

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 xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:period unit="y">3</domain:period>
C:        <domain:ns>
C:          <domain:hostObj>ns1.example.com</domain:hostObj>
C:        </domain:ns>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:contact type="billing">sh8013</domain:contact>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw roid="dddd-dddd">fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <resellerext:create xmlns:resellerext="urn:ietf:params:xml:ns:resellerext-1.0">
C:        <resellerext:id>myreseller</resellerext:id>
C:      </resellerext:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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:

- o An OPTIONAL <resellerext:add> element that contains attribute values to be added to the object.
- o An OPTIONAL <resellerext:rem> element that contains attribute values to be removed from the object.
- o An OPTIONAL <resellerext:chg> element that contains attribute values to be changed.

At least one and only one <resellerext:add>, <resellerext:rem> or <resellerext:chg> element MUST be provided. The <resellerext:add>, <resellerext:rem> and <resellerext:chg> elements contain the following child element:

- o A <resellerext:id> element that contains the identifier of the reseller of a sponsoring registrar.

Example <update> command, adding 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:add>
C:          <resellerext:id>myreseller</resellerext:id>
C:        </resellerext:add>
C:      </resellerext:update>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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:

```
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:chg>
C:          <resellerext:id>myreseller</resellerext:id>
C:        </resellerext:chg>
C:      </resellerext:update>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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:resellerext="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"/>

<!-- Child elements of the <resellerext: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 <resellerext: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>
    <!-- agent identifier that sells the domain, e.g. registrar, reseller -->
    <element name="id" type="eppcom:clIDType" minOccurs="0"/>
  </sequence>
</complexType>

<!-- Child response element -->

<element name="infData" type="resellerext:infDataType"/>
```

```
<!-- <resellerext:infData> response elements -->

<complexType name="infDataType">
  <sequence>
    <!-- agent identifier that sells the domain, e.g. registrar, reseller -->
    <element name="id" type="eppcom:clIDType" minOccurs="0"/>
  </sequence>
</complexType>

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

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 assign the following URI.

Registration request for the reseller namespace:

- o URI: urn:ietf:params:xml:ns:reseller-1.0
- o Registrant Contact: See the "Author's Address" section of this document.
- o 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. Acknowledgement

The authors would like to thank Rik Ribbers, Marc Groeneweg and Patrick Mevzek for their careful review and valuable comments.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.

- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<http://www.rfc-editor.org/info/rfc5730>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <<http://www.rfc-editor.org/info/rfc5731>>.
- [RFC5732] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Host Mapping", STD 69, RFC 5732, DOI 10.17487/RFC5732, August 2009, <<http://www.rfc-editor.org/info/rfc5732>>.
- [RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.17487/RFC5733, August 2009, <<http://www.rfc-editor.org/info/rfc5733>>.
- [RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <<http://www.rfc-editor.org/info/rfc7451>>.
- [W3C.REC-xml-20040204]
Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium First Edition REC-xml-20040204, February 2004, <<http://www.w3.org/TR/2004/REC-xml-20040204>>.
- [W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028, October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>>.
- [W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>>.

10.2. Informative References

- [ID.draft-zhou-eppept-reseller-mapping]
Zhou, L., Kong, N., Qi, C., Lee, X., and J. Gould, "Extensible Provisioning Protocol (EPP) Reseller Mapping", Oct 2015, <<http://tools.ietf.org/html/draft-zhou-eppept-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.

-02:

- * Changed author information.
- * Updated xml typos <reseller:infData> to <resellerext:infData> in <info> response.

-03:

- * Changed author information.
- * Updated section 3.1.
- * Removed reseller name element in <info> response.
- * Added acknowledgement.
- * Revised the typo "resellerr" to "resellerext".

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

Junkai Wei
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 3494
Email: weijunkai@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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2016

L. Zhou
N. Kong
G. Zhou
X. Lee
CNNIC
J. Gould
VeriSign, Inc.
March 9, 2016

Extensible Provisioning Protocol (EPP) Reseller Mapping
draft-zhou-eppext-reseller-mapping-03

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 September 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.

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	5
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	6
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	26
7. IANA Considerations	26
7.1. XML Namespace	26
7.2. EPP Extension Registry	27
8. Security Considerations	27
9. Acknowledgement	27
10. Normative References	27
Appendix A. Change Log	28

Authors' Addresses	29
--------------------	----

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. All reseller objects are identified by a server-unique identifier.

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:

- o ok: the normal status value for the reseller object.
- o readonly: transform commands submitted with the reseller identifier in the reseller extension would not be allowed.
- o 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.

4.1.1.1. 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:

- o One or more <reseller:id> elements that contain the server-unique identifier of the reseller objects to be queried.

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:      <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:

- o 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.

- o 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:

```
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:

- o 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:

- o A <reseller:id> element that contains the server-unique identifier of the reseller object, as defined in Section 3.1.
- o A <reseller:roid> element that contains the Repository Object Identifier assigned to the reseller object when the object was created.
- o A <reseller:state> element that contains the operational state of the reseller, as defined in Section 3.3.
- o 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 URL to the website of the reseller.
- 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.
- o 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.
- o 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.
- o 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:postalInfo>
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: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:

- o A <reseller:id> element that contains the desired server-unique identifier for the reseller to be created, as defined in Section 3.1.
- o A <reseller:state> element that contains the operational status of the reseller, as defined in Section 3.3.
- o 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 URL to the website of the reseller.
- o Zero or more OPTIONAL <reseller:contact> elements that contain identifiers for the human or organizational social information objects associated with the reseller object.

- o 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:    <clTRID>ABC-12345</clTRID>
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:

- o A `<reseller:id>` element that contains the server-unique identifier for the created reseller, as defined in Section 3.1.
- o A `<reseller:crDate>` element that contains the date and time of reseller-object creation.

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="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:    <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 `<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:

- o 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
C:        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:

- o A <reseller:id> element that contains the server-unique identifier of the reseller object to be updated, as defined in Section 3.1.
- o An OPTIONAL <reseller:add> element that contains attribute values to be added to the object.
- o An OPTIONAL <reseller:rem> element that contains attribute values to be removed from the object.
- o 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:

- o 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:

- o 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 URL to the website of the reseller.

- o 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:

```
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:      <reseller:update
C:        xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0">
C:        <reseller:id>res1523</reseller:id>
C:        <reseller:add>
C:          <reseller:contact type="tech">sh8013</reseller:contact>
C:        </reseller:add>
C:        <reseller:chg>
C:          <reseller:state>readonly</reseller:state>
C:          <reseller:postalInfo type="int">
C:            <reseller:addr>
C:              <reseller:street>124 Example Dr.</reseller:street>
C:              <reseller:street>Suite 200</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>+1.7034444444</reseller:voice>
C:          <reseller:fax/>
C:          <reseller:disclose flag="1">
C:            <reseller:voice/>
C:            <reseller:email/>
C:          </reseller:disclose>
C:        </reseller:chg>
C:      </reseller:update>
C:    </update>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</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"/>
  <import namespace="urn:ietf:params:xml:ns:domain-1.0"/>

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

```



```
        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>

<complexType name="postalInfoType">
  <sequence>
    <element name="name" type="contact:postalLineType"/>
    <element name="addr" type="contact:addrType"/>
  </sequence>
  <attribute name="type" type="contact:postalInfoEnumType"
    use="required"/>
</complexType>

<complexType name="discloseType">
  <sequence>
    <element name="name" 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>
```

```
<!--
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"
      minOccurs="0"/>
    <element name="email" type="eppcom:minTokenType"/>
    <element name="url" type="anyURI"
      minOccurs="0"/>
    <element name="contact" type="domain: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"
      minOccurs="0"/>
    <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="domain: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"/>
  </sequence>
</complexType>
```

```

        <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="addr" type="contact:addrType"
          minOccurs="0"/>
      </sequence>
      <attribute name="type" type="contact:postalInfoEnumType"
        use="required"/>
    </complexType>

    <!--
Child response elements.
-->
    <element name="chkData" type="contact:chkDataType"/>
    <element name="creData" type="contact:creDataType"/>
    <element name="infData" type="reseller:infDataType"/>

    <!--
<info> response elements.
-->
    <complexType name="infDataType">
      <sequence>
        <element name="id" type="eppcom:clIDType"/>
        <element name="roid" type="eppcom:roidType"/>
        <element name="state" type="reseller:stateType"/>
        <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"
          minOccurs="0"/>
        <element name="email" type="eppcom:minTokenType"/>
        <element name="url" type="anyURI"
          minOccurs="0"/>
        <element name="contact" type="domain:contactType"
          minOccurs="0" maxOccurs="3"/>
        <element name="clID" type="eppcom:clIDType"/>
        <element name="crID" type="eppcom:clIDType"/>

```

```
<element name="crDate" type="dateTime"/>
<element name="upID" type="eppcom:clIDType"
  minOccurs="0"/>
<element name="upDate" type="dateTime"
  minOccurs="0"/>
<element name="disclose" type="reseller:discloseType"
  minOccurs="0"/>
</sequence>
</complexType>

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

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 assign the following URI.

Registration request for the reseller namespace:

- o URI: urn:iETF:params:xml:ns:reseller-1.0
- o Registrant Contact: See the "Author's Address" section of this document.
- o 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. Acknowledgement

The authors would like to thank Rik Ribbers, Marc Groeneweg and Patrick Mevzek for their careful review and valuable comments.

10. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<http://www.rfc-editor.org/info/rfc5730>>.
- [RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.17487/RFC5733, August 2009, <<http://www.rfc-editor.org/info/rfc5733>>.
- [RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <<http://www.rfc-editor.org/info/rfc7451>>.
- [W3C.REC-xml-20040204]
Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium First Edition REC-xml-20040204, February 2004, <<http://www.w3.org/TR/2004/REC-xml-20040204>>.
- [W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028, October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>>.
- [W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>>.

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.

-02:

- * Changed author information.
- * Updated url definition.
- * Updated XML schema.

-03:

- * Changed author information.
- * Updated section 3.1.
- * Refactored the XSD file. Added <chgPostalInfoType> element.
- * Added acknowledgement.

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

Guiqing Zhou
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 2692
Email: zhouguiqing@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