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

Table of Contents

1. Introduction

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

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

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

1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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

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

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

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

2. Migrating to Newer Versions of This Extension

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

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

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

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

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

3. Extension Elements

3.1. Client Commands

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

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

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

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

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

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-eppext-launchphase]) or when an object is deleted during the relevant Grace Period (see below).

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

3.4.2. Grace Periods

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

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

If the "grace-period" attribute is omitted, then clients SHOULD NOT make any assumption about the grace period of the fee.

3.4.3. Correlation between Refundability and Grace Periods

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

3.4.4. Applicability

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

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

3.5. Account Balance

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

Whether or not the <fee:balance> is included in responses is a matter of server policy. However, if a server chooses to offer support for this element, it MUST be included in responses to all "transform" commands (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:

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

Example <check> command:

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

When the server receives a <check> command that includes the extension elements described above, its response MUST (subject to the exception described below) contain an <extension> element, which MUST

contain a child <fee:chkData> element. The <fee:chkData> element MUST contain a <fee:cd> element for each object referenced in the <check> element in the command.

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

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

- A <fee:object> element, which contains a copy of the child element of the <check> element of the command, to which the fee information relates.
- A <fee:command> element, which contains the same command that appeared in the corresponding <fee:object> element. This element MAY have the OPTIONAL "phase" and "subphase" elements, which MUST match the same attributes in the corresponding <fee:object> element.
- A <fee:currency> element, which contains the same currency code that appeared in the <fee:currency> element of the command. If no <fee:currency> element appeared in the command, then the client's default billing currency should be used.
- An OPTIONAL <fee:period> element, which contains the same unit that appeared in the <fee:currency> element of the command. If the value of the preceding <fee:command> element is "restore", this element MUST NOT be included. Otherwise it MUST be included. If no <fee:period> appeared in command (and the command is not "restore") then this element MUST have a value of 1 year.
- Zero or more <fee:fee> elements.
- Zero or more <fee:credit> elements.
- An OPTIONAL <fee:class> element.
- An OPTIONAL <fee:reason> element.

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

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

elements. If the "avail" attribute is true, then the <fee:cd> element MUST NOT contain a <fee:reason> element.

Example <check> response:

```
S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <resData>
S:       <domain:chkData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:cd>
S:           <domain:name avail="1">example.com</domain:name>
S:         </domain:cd>
S:         <domain:cd>
S:           <domain:name avail="1">example.net</domain:name>
S:         </domain:cd>
S:         <domain:cd>
S:           <domain:name avail="1">example.xyz</domain:name>
S:         </domain:cd>
S:       </domain:chkData>
S:     </resData>
S:     <extension>
S:       <fee:chkData
S:           xmlns:fee="urn:ietf:params:xml:ns:fee-0.11"
S:           xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <fee:cd avail="1">
S:           <fee:object>
S:             <domain:name>example.com</domain:name>
S:           </fee:object>
S:           <fee:command>create</fee:command>
S:           <fee:currency>USD</fee:currency>
S:           <fee:period unit="y">1</fee:period>
S:           <fee:fee
S:             description="Registration Fee"
S:             refundable="1"
S:             grace-period="P5D">5.00</fee:fee>
S:         </fee:cd>
S:         <fee:cd avail="1">
S:           <fee:object>
S:             <domain:name>example.com</domain:name>
S:           </fee:object>
S:           <fee:command>create</fee:command>
S:           <fee:currency>USD</fee:currency>
S:           <fee:period unit="y">1</fee:period>
```

```
S:           <fee:fee
S:             description="Registration Fee"
S:             refundable="1"
S:             grace-period="P5D">5.00</fee:fee>
S:         </fee:cd>
S:         <fee: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 valudate the value of the element. If the value is incorrect, the "avail" attribute of the corresponding <fee:cd> element MUST be false.

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

Server operators must provide clear documentation to client operators which of the above models it supports.

5.1.2. EPP Transfer Query Command

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

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

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

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

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

Example <transfer> query response:

```
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:

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

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

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

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

Example <create> command:

```
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>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:       <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:

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

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

Example <delete> response:

```
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:

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

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

- A <fee:currency> element;

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

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

Example <renew> command:

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

Example <renew> response:

```
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:

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

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

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

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

Example <transfer> command:

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

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

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

element, which contains the following child elements:

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

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

Example <update> command:

```
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.

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

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

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

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

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

- James Gould of Verisign
- Luis Munoz of ISC

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

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: December 11, 2016

R. Carney
GoDaddy Inc.
J. Galvin
Afilias USA

June 9, 2016

Non-standard Domain Fees File Format Specification
draft-carney-regext-domain-fees-00

Abstract

This document defines the file format for the storage of non-standard domain name fees and related details for a top level 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 December 11, 2016.

Table of Contents

1. Introduction

This document defines the file format for the storage of non-standard domain name fees and related details for a top level domain name registry, designed to facilitate interoperability and reusability among domain name registries and registrars.

The increased number of domain name registries and registrars, the greater number of registration fees for domain names in a single Top Level Domain (TLD), and to make accommodations for possible real-time communications issues between registries and registrars has driven the need for this standard sharable file.

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. General File Format Elements

File format will be a comma separated values (CSV) file and MUST use the US-ASCII character set. See informational RFC 4180 and RFC 7111 for details on the CSV file format.

Filename has two forms depending on content. For a file containing only a list of domain names in one Top Level Domain (TLD) the filename MUST be <TLD>-nonstandardnames-<YYYY-MM-DDThhmmss>.csv. For a file containing a list of domain names in multiple Top Level Domains (TLDs) the filename MUST be nonstandardnames-<YYYY-MM-DDThhmmss>.csv. For an IDN TLD <TLD> MUST be ace-encoded (i.e. in the "xn--" format). Additionally, <YYYY-MM-DDThhmmss> MUST be the date and time (in UTC) the file was created.

The first row MUST be the column headings: TLD, Domain Name, Status, Description, Currency, Domain Create Fee (Yearly), Domain Renew Fee (Yearly), Domain Transfer Fee (Yearly), Domain Restore Fee (per Restore), Effective Date. The remaining rows will contain the data (all unavailable names) from the registry in the following structure: <TLD>, <Domain Name>, <Status>, <Description>, <Currency>, <Domain Create Fee (Yearly)>, <Domain Renew Fee (Yearly)>, <Domain Transfer Fee (Yearly)>, <Domain Restore Fee (per Restore)>, <Effective Date>. Where <TLD> is the TLD of the domain name, <Domain Name> is the string of characters that represent the domain name, <Status> is one of five values (REGISTRY REGISTERED, REGISTERED, AVAILABLE, REGISTRY RESERVED, POLICY RESERVED), <Description> is the descriptive name for the premium tier to which the name is allocated, <Currency> is the ISO 4217 Alpha Currency Code for the currency required by the EPP commands used with the SRS, <Domain Create Fee (Yearly)> is the 1-year Initial Registration fee to the Registrar incurred on a 1-year domain create EPP command and does not reflect any marketing or rebate reductions, <Domain Renew Fee (Yearly)> is the 1-year renewal fee to the Registrar incurred on a 1-year domain renewal EPP command and does not reflect any marketing or rebate reductions, <Domain Transfer Fee (Yearly)> is the fee to the Registrar that is incurred when the domain is transferred between Registrars, <Domain Restore Fee (per Restore)> is the fee to the Registrar to restore a domain that is in the Redemption Period, <Effective Date> is the date and time the fee listed will go into effect in UTC time using the combined date/time format from ISO 8601 (YYYY-MM-DDThh:mm:ss.sZ). If the <TLD> and/or <Domain Name> is an IDN it MUST be ace-encoded (i.e. in the "xn--" format).

3. Examples

3.1. Single TLD File Example

Example of a file that contains domain names from a single TLD with non-standard fees.

FileName: example-nonstandardnames-2016-05-01T010000.csv

```
TLD,Domain Name,Status,Description,Currency,Domain Create Fee
(Yearly),Domain Renew Fee (Yearly),Domain Transfer Fee
(Yearly),Domain Restore Fee (per Restore),Effective Date
example,e.example,REGISTRY
RESERVED,A,USD,200,200,200,40,2016-02-05T09:31:40.2Z
example,ex.example,POLICY RESERVED,A,USD,200,200,200,40,
example,example.example,AVAILABLE,B,USD,500.75,500.75,500.75,40,
example,xn--4gqvdy3r.example,AVAILABLE,A,USD,200,200,200,40,2016-1
1-03T00:00:00.0Z
```

3.2. Multiple TLDs File Example

Example of a file that contains domain names from multiple TLDs with non-standard fees.

FileName: nonstandardnames-2016-05-01T010000.csv

```
TLD,Domain Name,Status,Description,Currency,Domain Create Fee
(Yearly),Domain Renew Fee (Yearly),Domain Transfer Fee
(Yearly),Domain Restore Fee (per Restore),Effective Date
example,e.example,REGISTRY
RESERVED,A,USD,200,200,200,40,2016-02-05T09:31:40.2Z
test,ex.test,POLICY RESERVED,A,USD,200,200,200,40,
another, xn--
4gqvdy3r.another,AVAILABLE,B,USD,500.75,500.75,500.75,40,
```

3.3. Fee Change File Example

If the fee for a domain name is changing or moving to reserved or unreserved in the future, there SHOULD be two entries for the name: an entry for the current fee and status of the name (This entry will be removed once the change takes place.) and an entry for the future fee of the name with the effective date in YYYY-MM-DDThh:mm:ss.sZ format.

Example of a file that contains a future fee change.

FileName: nonstandardnames-2016-05-01T010000.csv

```
TLD,Domain Name,Status,Description,Currency,Domain Create Fee
(Yearly),Domain Renew Fee (Yearly),Domain Transfer Fee
(Yearly),Domain Restore Fee (per Restore),Effective Date
example,e.example,REGISTRY
RESERVED,A,USD,200,200,200,40,2016-02-05T09:31:40.2Z
example,e.example,REGISTRY RESERVED,B,USD,100,100,100,40,
```

4. Security Considerations

The file format described in this document does not provide any security services.

5. Acknowledgements

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

- Christine Turner of GoDaddy Inc.
- Jody Kolker of GoDaddy Inc.

- Richard Merdinger of GoDaddy Inc.
- Mike Runcieman of Afilias Canada
- Wayne Beaver of Afilias USA

6. Change History

7. 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>.

[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>.

[RFC7111] Hausenblas, M., Wilde, E., and J. Tennison, "URI Fragment Identifiers for the text/csv Media Type", RFC 7111, DOI 10.17487/RFC7111, January 2014, <http://www.rfc-editor.org/info/rfc7111>.

Authors' Addresses

Roger Carney
GoDaddy Inc.
14455 N. Hayden Rd. #219
Scottsdale, AZ 85260
US

Email: rcarney@godaddy.com
URI: http://www.godaddy.com

James Galvin
Afilias USA
300 Welsh Road, Building 3, Suite 105
Horsham, PA 19044
US

Email: jgalvin@afilias.info
URI: http://www.afilias.info

Registration Protocols Extensions
Internet-Draft
Intended status: Informational
Expires: July 22, 2018

A. Blinn
R. Carney
GoDaddy Inc.

January 18, 2018

Domain Connect API - Communications between DNS Provider and Services
draft-carney-regext-domainconnect-03

Abstract

This document provides information related to the Domain Connect API that was built to support communications between DNS Providers and Service Providers (hosting, social, email, hardware, etc.).

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 https://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 22, 2018.

Table of Contents

1. Introduction

Configuring a service at a Service Provider to work with a domain has historically been a complex task that is difficult for users.

Typically, a customer would try to configure their service by entering their domain name with the Service Provider. The Service Provider then used a number of techniques with mixed reliability to discover the DNS Provider. This might include DNS queries for nameservers, queries to whois, and mapping tables to determine the registrar or the company providing DNS.

Once the Service Provider discovered the DNS Provider, they typically gave the customer instructions for proper configuration of DNS. This

might include help text, screen shots, or even links to the appropriate tools.

Discovery of the DNS Provider in this manner is unreliable, and providing instructions to users would present a number of technologies (DNS record types, TTLs, Hostnames, etc.) and processes they didn't understand. These instructions authored by the Service Provider often quickly become out of date, further confusing the issue for users.

The goal of this specificatoin is to create a system where Service Providers can easily enable their applications/services to work with the domain names of their customers. This includes both discovery of the DNS Provider and subsequent modification of DNS.

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 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.

3. Definitions

The following definitions are used in this document:

- Service Providers - refers to entities that provide products and services attached to domain names. Examples include web hosting providers (such as Wix or SquareSpace), email Service Providers (such as Microsoft or Google) and potentially even hardware manufacturers with DNS-enabled devices including home routers or automation controls (such as Linksys, Nest, and Philips).
- DNS Providers - refers to entities that provide DNS services such as registrars (e.g. GoDaddy, 1and1) or standalone DNS services (e.g. CloudFlare).
- Customer/User - refers to the end-user of these services.
- Templates/Service Templates - refers to a file that describes a set of changes to DNS and domain functionality to enable a specific service.
- Root Domain refers to a registered domain (e.g. example.com or example.co.uk) or a delegated zone in DNS.
- Sub Domain refers to a sub-domain of a root domain (e.g. sub.example.com or sub.example.co.uk).

4. The API

The system will be implemented using simple web based interactions and standard authentication protocols. The creation and modification of DNS settings will be done through the application of templates instead of direct manipulation of individual DNS records.

Templates are core to this proposal as they describe a service owned by a Service Provider, and contain all of the information necessary to to enable and operate/maintain a service.

The individual records may be identified by a group identifier. This allows for the application of templates in different stages. For example, an email provider might first set a TXT record to verify the domain, and later set an MX record to configure email delivery. While done separately, both changes are fundamentally part of the same service.

It is important that templates be constrained to an individual service, as later removal of a template would remove all associated records.

Templates can also contain variable portions, as often values of data in the template change based on the implementation and/or user of the Service Provider (e.g. the IP address of a service, a customer id, etc).

Configuration and onboarding of templates between the DNS Provider and the Service Provider is seen as a manual process. The template is defined by the Service Provider and given to the DNS Provider. Future versions of this specification may allow for an independent repository of templates. For now, the templates are all published at http://domainconnect.org.

By basing the protocol on templates instead of DNS Records, several advantages are achieved. The DNS Provider has very explicit knowledge and control on the settings being changed to enable a service. The system is also more secure as templates are tightly controlled and contained.

To attach a domain name to a service provided by a Service Provider, the customer would first enter their domain name.

Instead of relying on examination of the nameservers and mapping these to DNS Providers, DNS Provider discovery would be handled through simple records in DNS and an API. The Service Provider can query for a specific record in the zone to determine a REST endpoint to initiate the protocol. A Domain Connect compliant DNS Provider

would return information about that domain and how to configure it using Domain Connect.

For the application of the changes to DNS, there are two use cases. The first is a synchronous web flow, and the second is an asynchronous flow using OAuth and an API.

It should be noted that a DNS Provider may choose to only implement one of the flows. As a matter of practice many Service Providers are based on the synchronous flow, with only a handful of them based on the asynchronous OAuth flow. So, many DNS providers may opt to only implement the synchronous flow.

It should also be noted that individual services may work with the synchronous flow only, the asynchronous flow only, or with both.

4.1. The Synchronous Flow

This flow is tailored for the Service Provider that requires a one-time synchronous change to DNS.

The user would first enter their domain name at the Service Provider website.

After the Service Provider determines the DNS Provider, the Service Provider might display a link to the user indicating that they can "Connect their Domain" to the service.

After clicking the link, the user is directed to a browser window on the DNS Provider’s site. This is typically in another tab or in a new browser window, but can also be in place navigation with a return url. This link would pass the domain name being modified, the service provider and template being enabled, and any additional parameters needed to configure the service.

Once at the DNS Provider site, the user would be asked to authenticate if necessary.

After authenticating at the DNS Provider, the DNS Provider would verify the domain name is owned by the user. The DNS Provider would also verify other parameters passed in are valid and would prompt the user to give consent for making the change to DNS. The DNS Provider could also warn the user of services that would be disabled by applying this change to DNS.

Assuming the user grants this consent, the DNS changes would be applied. Upon successful application of the DNS changes, the popup

window or tab would be closed. If in place the user would be redirected back to the service provider.

4.2. The Asynchorous Flow

The asynchronous OAuth flow is tailored for the Service Provider that wishes to make changes to DNS asynchronously with respect to the user interaction, or wishes to make multiple or additional changes to DNS over time.

The OAuth based authentication and authorization flow begins similarly to the web based synchronous flow. The Service Provider determines the DNS Provider and links to the consent dialog at the DNS Provider where the user signs in, the ownership of the domain is verified, and the consent is granted.

However, instead of applying the DNS changes on user consent, OAuth access is granted to the Service Provider. An OAuth access code is generated and handed back to the Service Provider. The Service Provider then requests an access (bearer) token.

The permission granted in the OAuth token is a right for the Service Provider to apply a template to the specific domain owned by a specific user.

The Service Provider would later call an API that applies this template to the domain, including any necessary parameters along with the access token(s). As in all OAuth flows, access can be revoked by the user at any time. This would be done on the DNS Provider’s user experience.

If the OAuth flow is used, once a Service Provider has an OAuth token the Domain Connect API becomes available for use. The Domain Connect API is a simple REST service.

This REST service allows the application or removal of the changes in the template on a domain name. The domain name, user, and template must be authorized through the OAuth token and corresponding access token.

Additional parameters are expected to be passed as name/value pairs on the query string of each API call.

4.3. DNS Provider Initiated Flows

A DNS Provider may with to expose interesting services that the user could attach to their domain. An example would be suggesting to a

user that they might want to connect their domain to a partner Service Provider.

If the template for the service is static, it is possible to simply apply the template.

However, often the template has some dynamic elements. For this scenario, the DNS Provider need simply call a URL at the Service Provider. The Service Provider can then sign the user in, collect any necessary information, and call the normal web-based flows described above.

4.4. DNS Provider Discovery

In order to facilitate discovery of the DNS Provider from a domain name, a domain will contain a record in DNS.

This record will be a simple TXT record containing a URL used as a prefix for calling a discovery API. This record will be named domainconnect.

An example of this record might contain:

domainconnect.godaddy.com

As a practical matter of implementation, the DNS Provider need not contain a copy of this data in each and every zone. Instead, the DNS Provider needs simply to respond to the DNS query for the domainconnect TXT record with the appropriate data.

How this is implemented is up to the DNS Provider.

For example, the DNS Provider may not store the data inside a TXT record for the domain, opting instead to put a CNAME in the zone and have the TXT record in the target of the CNAME.

Once the URL prefix is discovered, it can be used by the Service Provider to determine the additional settings for using Domain Connect on this domain at the DNS Provider. This is done by calling a REST API.

```
GET
https://{domainconnect}/v2/{domain}/settings
```

This will return a JSON structure containing the settings to use for Domain Connect on the domain name (passed in on the path) at the DNS Provider. This JSON structure will contain the following fields:

- o providerName: The name of the DNS Provider suitable for display on the Service Provider UX.
- o urlSyncUX: The URL Prefix for linking to the UX elements of Domain Connect for the synchronous flow at the DNS Provider.
- o urlAsyncUX: The URL Prefix for linking to the UX elements of Domain Connect for the asynchronous flow at the DNS Provider
- o urlAPI: This is the URL Prefix for the REST API.
- o width: This is the width of the popup window for granting consent when navigated in a popup. Default value is 750px.
- o height: This is the height of the popup window for granting consent when navigated in a popup. Default value is 750px.

As an example, the JSON returned by this call might contain.

```
{
"providerName": "GoDaddy",
"urlSyncUX": "https://domainconnect.godaddy.com",
"urlAsyncUX": "https://domainconnect.godaddy.com",
"urlAPI" : "https://api.domainconnect.godaddy.com",
"width" : 750,
"height" : 750
}
```

If the DNS Provider is not implementing the synchronous flow, the urlSyncUX is not required. Similarly, if the DNS Provider is not implementing the asynchronous flow the urlAsyncUX is not required.

4.5. Domain Connect Details

Domain Connect contains endpoints in the form of URLs.

The first set of endpoints are for the UX that the Service Provider links to. These are for the UX which includes the web-based flow where the user clicks on the link, and the OAuth flow where the user clicks on the link for consent.

The second set of endpoints are for the API, largely for the asynchronous OAuth flow via REST.

All endpoints begin with a root URL for the DNS Provider such as https://connect.dnsprovider.com/

They may also include any prefix at the discretion of the DNS Provider, for example, https://connect.dnsprovider.com/api/

The root URLs for the UX endpoints and the API endpoints are returned in the JSON payload during DNS Provider discovery.

4.5.1. Synchronous Flow

4.5.1.1. Query Supported Template

```
GET
{urlAPI}/v2/domainTemplates/
providers/{providerId}/services/{serviceId}
```

This URL can be used by the Service Provider to determine if the DNS Provider supports a specific template through the synchronous flow.

Returning a status of 200 without a body indicates the template is supported. Returning a status of 404 indicates the template is not supported.

4.5.1.2. Apply Template

```
GET
{urlAPI}/v2/domainTemplates/
providers/{providerId}/services/{serviceId}/apply?[properties]
```

This is the URL used to apply a template to a domain. It is called from the Service Provider to start the Domain Connect Protocol.

This URL should be called in a new browser tab or in a popup browser window. The DNS Provider would sign the user in, verify domain ownership, and ask for confirmation of application of the template.

It is also likely that the DNS Provider would warn the user of existing settings that would change and/or services that would be disrupted as part of applying this template. The fidelity of this warning is left to the DNS Provider.

Upon completion of the application of the template the DNS Provider would close this tab or window.

Parameters/properties passed to this URL include:

- domain: This parameter contains the domain name being configured.
- name/value pairs: Any variable fields consumed by this template. The name portion of this API call corresponds to the variable(s) specified in the template and the value corresponds to the value that should be used when applying the template.
- groupId: This OPTIONAL parameter specifies the group of changes from the template to apply. If no group is specified, all changes are applied. Multiple groups can be specified in comma delimited format.

- o sig: An optional signature of the query string. See Security Considerations section.

An example query string is below:

```
GET
https://webconnect.dnsprovider.com/v2/domainTemplates/providers/coolprovider.com/services/hosting/apply?www=192.168.42.42&m=192.168.42.43&domain=example.com
```

This call indicates that the Service Provider wishes to connect the domain example.com to the service using the template identified by the composite key of the provider (coolprovider.com) and the service owned by them (hosting). In this example, there are two variables in this template, "www" and "m" which both require values (in this case each requires an IP address). These variables are passed as name/value pairs.

4.5.1.3. Security Considerations

By applying a template with parameters, there is a security consideration that must be taken into account

Consider an email template where the IP address of the MX record is a passed in variable. A bad actor could generate a URL with a bad IP. If an end user is convinced to click on this URL (say in a phishing email), they would land on the DNS Provider site to confirm the change. To the user, this would appear to be a valid request to configure the domain. Yet the IP would be hijacking the service.

Not all templates have this problem. But when they do, there are two options.

One option would be to not enable the synchronous flow and use asynchronous OAuth. But as will be seen below, OAuth has both a higher implementation burden and requires onboarding between each Service and DNS Provider.

As another option, digitally signing the query string will be enabled. The signature will be appended as an additional query string parameter, properly URL encoded and of the form:

```
sig=NLOQQm6ikGC2FlFvFZqIFNCZqlaC4B%2FQDwS6iCwIElMWhXMgRnRE17zhLtdLFie
WkyqKa4I%2FOoFaAgd%2FAl%2ByzDd3sM2X1JVF5ELjTlj84jZ4KOEIdnbgkEeO%2FTkY
RrPkwcmcHMwc4RuX%2Fqio8vKYxJaKLKeVGpUNSKo7zkq3XIRgyxoLSRKxmlSTHFAz4Lv
YXPWo6SHDoVcRvElWj18Um13sSXuX4KhtOLym2yImHpboEi4m2Ziigc%2BNHZE0VvHUR7
wZgDaB01z8hFm5ATF%2B8swjandMRf2Lr4Syv4qTxMNT61r62EWFkt5t9nhxMgss6z4pf
DVFZ3vYwSJDGuRpEQ%3D%3D
```

The Service Provider can generate this signature using a private key. The DNS Provider can then verify the signature using the public key.

key=_dcpubkeyv1

This example indicates that the public key can be found by doing a DNS query for a TXT record called _dcpubkeyv1.

Since the public key may be greater than 255 characters, multiple TXT records may exist for the DNS TXT query. For a public key of:

```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1dCqv7JEzUOfbhWKB9mTRsv3O
9Vzy1Tz3UQlIDGpnVrTPBJDQTXUhxUMREEOBKo+rOjHZqfYnSmlkgu1dnBEO8bsELQL8G
jS4zsjdA53gRk2SDxuzcB4fK+NCDfnRHut5nG0S3U4cq4DuGrMDFVBwxH1duTsqDNgIOO
fNTsFcWSVXoSSTqCCMGbj8Vt51umDhWQAj06lf50qP2/
jMNs2G+KTlk3dBHx3wtqYLvdcop1Tk5xBD64BPJ9uwm8KlDNHe+8O+cC9j04Ji8B2K0/
PzAj90xnb8XJy/
EM124hpT9lMgpHKBUvdeurJYweC6oP41gsTf5LrpjnyIy9j5FHPCQIDAQAB
```

There would be several TXT records. The records would be of the form:

```
p=1,a=RS256,d=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1dCqv7JE
zUOfbhWKB9mTRsv3O9Vzy1Tz3UQlIDGpnVrTPBJDQTXUhxUMREEOBKo+rOjHZqfYnS
mlkgu1dn
p=2,a=RS256,d=BEO8bsELQL8GjS4zsjdA53gRk2SDxuzcB4fK+NCDfnRHut5nG0S3
U4cq4DuGrMDFVBwxH1duTsqDNgIOOfNTsFcWSVXoSSTqCCMGbj8Vt51umDhWQAj06l
f5
p=3,a=RS256,d=NCDfnRHut5nG0S3U4cq4DuGrMDFVBwxH1duTsqDNgIOOfNTsFcWS
VXoSSTqCCMGbj8Vt51umDhWQAj06lf50qP2/
jMNs2G+KTlk3dBHx3wtqYLvdcop1Tk5xBD64BPJ9
p=4,a=RS256,d=uwm8KlDNHe+8O+cC9j04Ji8B2K0/PzAj90xnb8XJy/
EM124hpT9lMgpHKBUvdeurJYweC6oP41gsTf5LrpjnyIy9j5FHPCQIDAQAB
```

Here the public key is broken into four records in DNS, and the data also indicates that the signing algorithm is an RSA Signature with SHA-256.

It should be noted that the above data was generated for a query string:

a=1&b=2&ip=10.10.10.10&domain=foobar.com

Support for signing the query string and verification is optional. Not all services require this level of security, and not all DNS Providers will support this signing for the synchronous flow.

There are circumstances where the Service Provider may wish to verify that the template was successfully applied. Without domain connect, this typically involved the Service Provider querying DNS to see if the settings had been applied.

This same technique works with Domain Connect, and if necessary can be triggered either manually on the Service Provider site or automatically upon page/window activation in the browser.

Automatic notification via callback URLs were considered in earlier drafts, and subsequently dropped due to their lack of reliability and difficulty in getting a consistent implementation across DNS Providers.

4.5.2. Asynchronous Flow: OAuth

Using the OAuth flow is a more advanced use case, needed by Service Providers that have more complex configurations that may require multiple steps and/or are asynchronous from the user's interaction.

Details of an OAuth implementation are beyond the scope of this specification. Instead, an overview of how OAuth fits with Domain Connect is given here.

Service providers wishing to use the OAuth flow must register as an OAuth client with the DNS Provider. This is envisioned as a manual process.

To register, the Service Provider would provide (in addition to their template) one or more callback URLs that specify where the customer will be redirected after the provider authorization. In return, the DNS Provider will give the Service Provider a client id and secret which will be used when requesting tokens as part of the OAuth process flow.

4.5.2.1. Getting an Authorization Code

```
GET
{urlAsyncUX}/v2/domainTemplates/
providers/{providerId}/services/{serviceId}
```

To initiate the OAuth flow the Service Provider would link to the DNS Provider to gain consent.

This endpoint is similar to the synchronous flow described above, and will handle authenticating the user, verification of domain ownership, and asking for the user's permission to allow the Service Provider to make the specified changes to the domain. Similarly, the

DNS Provider will often want to warn the user that (eventual) application of this template might change existing records and/or disrupt existing services attached to the domain.

While the variables for the applied template would be provided later, the values of some variables are often necessary in the consent flow to determine conflicts. As such, any variables impacting conflicting records needs to be provided in the consent flow. Today this includes variables in hosts, and variables in the data portion for certain TXT records. As conflict resolution evolves, this list may grow.

Upon successful authorization, verification, and consent, the DNS Provider will direct the end user’s browser to the redirect URI provided in the request, appending the authorization code as a query parameter of "code".

Upon error, the DNS Provider will direct the end user’s browser to the redirect URI provided in the request, appending the error code as a query parameter "error".

The following describes the values to be included in the query string parameters for the request for the OAuth consent flow.

- domain: This parameter contains the domain name being configured.
- client_id: This is the client id that was provided by the DNS Provider, to the Service Provider during registration.
- redirect_uri: The location to direct the client’s browser to upon successful authorization, or upon error.
- response_type: OPTIONAL. If included should be the string ’code’ to indicate an authorization code is being requested.
- scope: This is the name of the template that is being requested.
- state: OPTIONAL but recommended. This is a random, unique string passed along to prevent CSRF. It will be returned as a parameter when redirecting to the redirect_url described above.
- name/value pairs: Required for fields that impact the conflict detection. This includes variables used in hosts and data in TXT records.

4.5.2.2. Requesting an Access Token

```
POST {urlAPI}/v2/OAuth/access_token
```

Once authorization has been granted the Service Provider must use the Authorization Code provided to request an Access Token. The OAuth specification recommends that the Authorization Token be a short lived token, and a reasonable recommended setting is ten minutes. As

such this exchange needs to be completed before that time has expired or the process will need to be repeated.

This token exchange is done via a server to server API call from the Service Provider to the DNS Provider.

The Access Token granted will also have a longer lifespan, but also can expire. To get a new access token, the Refresh Token is used.

The following describes the POST parameters to be included in the request.

- code: The authorization code that was provided in the previous step when the customer accepted the authorization request, or the refresh_token for a subsequent access token.
- redirect_uri: OPTIONAL. If included, needs to be the same redirect uri provided in the previous step, simple for verification.
- grant_type: The type of code in the request. Usually the string 'authorization_code' or 'refresh_token'.
- client_id: This is the client id that was provided by the DNS Provider, to the Service Provider during registration.
- client_secret: The secret provided to the Service Provider during registration.

Upon successful token exchange, the DNS Provider will return a response with 4 properties in the body of the response.

- access_token: The access token to be used when making API requests.
- token_type: Always the string "bearer".
- expires_in: The number of seconds until the access_token expires.
- refresh_token: The token that can be used to request new access tokens when this one has expired.

4.5.2.3. Making Requests with Access Tokens

Once the Service Provider has the access token, they can call the DNS Provider's API to make change to DNS on behalf of the user.

All calls to this API pass the access token in the Authorization Header of the request to the call to the API. More details can be found in the OAuth specifications, but as an example:

```
GET /resource/1 HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

4.5.2.4. Apply Template to Domain

```
POST
{urlAPI}/v2/domainTemplates/
providers/{providerId}/services/{serviceId}/apply?[properties]
```

The primary function of the API is to apply a template to a customer domain.

While the providerId and serviceId are also implied in the authorization, these are on the path for consistency with the synchronous flows. If not matching what is in the authorization, an error would be returned.

When applying a template to a domain, it is possible that a conflict may exist with previous settings. While it is recommended that conflicts be detected when the user grants consent, because OAuth is asynchronous it is possible that a new conflict was introduced by the user.

While it is up to the DNS Provider to determine what constitutes a conflict (see section on Conflicts below), when one is detected an error will be returned by default. This error will enumerate the conflicting records in a format described below.

Because the user isn’t present at the time of this error, it is up the Service Provider to determine how to handle this error. Some providers may decide to notify the user. Others may decide to apply their template anyway using the "force" parameter. This parameter will bypass error checks for conflicts, and after the call the service will be in its desired state.

Calls to apply a template via OAuth require the following parameters:

- domain: This contains the domain name being configured. It must match the domain in the authorization token.
- name/value pairs: Any variable fields consumed by this template. The name portion of this API call corresponds to the variable(s) specified in the record and the value corresponds to the value that should be used when applying the template as per the implementation notes.
- groupId: This OPTIONAL parameter specifies the group of changes in the template to apply. If omitted, all changes are applied. This can also be a comma separated list of groupIds.
- force: This OPTIONAL parameter specifies that the template should be applied independently of any conflicts that may exist on the domain. This can be a value of 0 or 1.

An example call is below. In this example, it is contemplated that there are two variables in this template, "www" and "m" which both require values (in this case each requires an IP address). These variables are passed as name/value pairs.

```
POST
https://connect.dnsprovider.com/v2/domainTemplates/providers/coolp
rovider.com/services/hosting/
apply?www=192.168.42.42&m=192.168.42.43&force=1
```

The API must validate the access token for the Service Provider and that the domain belongs to the customer and is represented by the token being presented. With these checks passing, the template may be applied to the domain after verifying that doing so would not cause an error condition, either because of problems with required variables or the current state of the domain itself (for example, already having a conflicting template applied).

Results of this call can include information indicating success, or an error. Errors will be 400 status codes, with the following codes defined.

- Success (20*): A response of an http status code of 204 indicates that call was successful and the template applied. Note that any 200 level code should be considered a success.
- Unauthorized (401): A response of a 401 indicates that caller is not authorized to make this call. This can be because the token was revoked, or other access issues.
- Error (404,422): This indicates something wrong with the request itself, such as bad parameters.
- Failed (409): This indicates that the call was good, and the caller authorized, but the change could not be applied due to a conflicting template or a domain state that prevents updates. Errors due to conflicts will only be returned when force is not equal to 1.

When a 409 is returned, the body of the response will contain details of the error. This will be JSON containing the error code, a message suitable for developers, and an array of tuples containing the conflicting records type, host, and data element.

4.5.2.5. Revert Template

```
POST
{urlAPI}/v2/domainTemplates/
providers/{providerId}/services/{serviceId}/revert?domain={domain}
```

This API allows the removal of a template from a customer domain using an OAuth request.

The provider and service name in the authorizatoin must match the values in the URL. So must the domain name on the query string.

This call must validate that the template requested exists and has been applied to the domain by the Service Provider or a warning must be returned that the call would have no effect. This call must validate that there is a valid authorization token for the domain passed in or an error condition must be reported.

An example query string might look like:

```
POST
https://connect.dnsprovider.com/v2/domainTemplates/providers/coolp
rovider.com/services/hosting/revert?domain=example.com
```

Response codes are identical to above.

4.5.2.6. Revoke Access

Like all OAuth flows, the user can revoke the access at any time using UX at the DNS Provider site. So the Service Provider needs to be aware that their access to the API may be denied.

4.6. Domain Connect Objects and Templates

Templates are not versioned. Instead, if a breaking change is made to a template it is recommended that a new template be created. While on the surface versioning looks appealing, the reality is that the settings in a template rarely change. This is because a successful service will have many customers with settings in their DNS, some applied by templates using this protocol, and some manually applied. As such changes to the template need to be done in a manner that accounts for existing customers.

A template is defined as a standard JSON data structure containing the following data:

- providerId: The unique identifier of the Service Provider that created this template. This is used in the URLs to identify the Service Provider. To ensure non-coordinated uniqueness, this would be the domain name of the Service Provider.
- providerName: The name of the Service Provider. This will be displayed to the user.
- templateId: The name or identifier of the template. This is used in URLs to identify the template.

- templateName: The friendly name of this service. This will be displayed to the user.
- logoUrl: A graphical logo for use in any web-based flow. This is a URL to a graphical logo sufficient for retrieval.
- description: A textual description of what this template attempts to do. This is meant to assist integrators, and therefore should not be displayed to the user.
- syncBlock: Indicates that the synchronous protocol should not be enabled for this template.
- synchPubKeyDomain: When present, indicates that calls to apply a template synchronously will be digitally signed. This element contains the domain name for querying a TXT record from DNS.
- launchUrl: OPTIONAL. A URL suitable for a DNS Provider to call to initiate the execution of this template. This allows the flow to begin with the DNS Provider as described above.
- records: A list of records for the template.

Each template record is an entry that contains a type and several optional parameters based on the value.

For all entries of a record template other than "type" and "groupId", the value can contain variables denoted by %<variable name>%. These are the values substituted at runtime when writing into DNS.

It should be noted that as a best practice, the variable should be equal to the portion of the values in the template that change as little as possible.

For example, say a Service Provider requires a CNAME of one of three values for their users: s01.example.com, s02.example.com, and s03.example.com.

The value in the template could simply contain %servercluster%, and the fully qualified string passed in. Alternatively, the value in the template could contain s%var%.example.com. By placing more fixed data into the template, the data is more constrained.

Each record will contain the following elements:

- type: Describes the type of record in DNS, or the operation impacting DNS. Valid values include: A, AAAA, CNAME, MX, TXT, SRV, NS, APEXCNAME, REDIR301, or REDIR302.
- groupId: This OPTIONAL parameter identifies the group the record belongs to when applying changes.
- host: The host for A, AAAA, CNAME, TXT, and MX values. This is the hostname in DNS.
- pointsTo: The pointsTo location for A, AAAA, CNAME and MX records.

- ttl: This is the time-to-live for the record in DNS. Valid for A, AAAA, CNAME, TXT, MX, and SRV records.
- data: This is the data for a TXT record in DNS.
- priority: This is the priority for an MX or SRV record in DNS.
- weight: This is the weight for the SRV record.
- port: This is the port for the SRV record.
- protocol: This is the protocol for the SRV record.
- service: This is the protocol for the SRV record.

4.7. Operational and Implementation Considerations

The DNS Provider is responsible for handling of the conflicts with records already existing in the DNS Zone. This includes detection of conflicts, removing conflicts when a new template is applied, and merging records when appropriate.

For example, if the application of a template through the web based flow would interfere with previously set DNS records (either through another template or manual settings), it is envisioned that the user would be asked to confirm the clearing of the previously set template. If it would interfere with DNS records accessible through a previously issued OAuth flow, the provider could revoke the previously issued token.

Similarly, when granting an OAuth token that interferes with a previously issued OAuth token, access to the old token could automatically be revoked.

Manual changes to DNS through the DNS Provider could have appropriate warnings in place to prevent unwanted changes; with overrides being possible and removing conflicting templates.

The behavior of these interactions is left to the sophistication of the DNS Provider. However, a general recommendation is to ensure that a newly configured service works correctly.

A proposing handling of records is as follows (if not otherwise specified, conflicts occur if the records have the same name):

> Replace records of the same type for A, AAAA, MX, CNAME, APEXCNAME, SRV. If the template specifies an A or AAAA, the respective AAAA or A record should be removed to avoid IPv4 and IPv6 pointing to different services
> Append to the existing records of the same type for TXT
> Replace any record for CNAME
> Remove any CNAME record existing at the same or parent level to any records added by the template

Additional record types and/or extensions to records in the template can be implemented on a per DNS Provider basis. However, care should be taken when defining extensions so as to not conflict with other protocols and standards. Certain record names are reserved for use in DNS for protocols like DNSSEC (DNSKEY, RRSIG) at the registry level.

Defining these optional extensions in an open manner as part of this specification is highly recommended. The following are the initial optional extensions a DNS Provider/Service Provider may support.

Some Service Providers desire the behavior of a CNAME record, but in the apex record. This would allow for an A Record at the root of the domain but dynamically determined at runtime.

The recommended record type for DNS Providers that wish to support this an APEXCNAME record. Additional fields included with this record would include pointsTo and TTL.

Defining a standard for such functionality in DNS is beyond the scope of this specification. But for DNS Providers that support this functionality, using the same record type name across DNS Providers allows template reuse.

Some Service Providers desire a redirection service associated with the A Record. A typical example is a service that requires a redirect of the domain (e.g. example.com) to the www variant (www.example.com). The www would often contain a CNAME.

Since implementation of a redirection service is typically simple, it is recommended that service providers implement redirection on their own. But for DNS Providers that have a redirection service, supporting simple templates with this functionality may be desired.

While technically not a "record" in DNS, when supporting this optional functionality, it is recommended that this be implemented using two new record types.

REDIR301 and REDIR302 would implement 301 and 302 redirects respectively. Associated with this record would be a single field called the "target", containing the target domain of the redirect.

Several service providers have asked for functionality supporting an update to the nameserver records at the registrar associated with the domain.

This functionality is again deemed as optional and up to the DNS Provider to determine if they desire to support this functionality.

When implementing this, two records will be provided. NS1 and NS2, each containing a pointsTo argument.

Requests have also been made to allow for updates to the DS record for DNSSEC. This record is required at the registry to enable DNSSEC, but can only be written by the registrar.

Note that the registrar may or may not be the DNS Provider, but in this case the implementation of updates of the DS record into the registry would be handled exclusively by the registrar.

For DNS Providers that support this record, the record type should be DS. Values will be keyTag, algorithm, digestType, and digest.

Variables in templates that are hard-coded host names are the responsibility of the DNS Provider to protect. That is, DNS Providers are responsible for ensuring that host names do not interfere with known values (such as m. or www. or mail.) or internal names that provide critical functionality that is outside the scope of this specification.

This template format is intended for internal use by a DNS Provider and there are no codified API endpoints for creation or modification of these objects. API endpoints do not use this object directly. Instead, API endpoints reference a template by ID and then provide key/value pairs that match any variable values in these record objects.

However, by defining a standard template format it is believed it will make it easier for Service Providers to share their provisioning across DNS Providers. Further revisions of this specification may include a repository for publishing and consuming these templates. For now, templates are maintained at http://domainconnect.org.

Implementers are responsible for data integrity and should use the record type field to validate that variable input meets the criteria for each different data type.

Some considerations are necessary for configuring a domain (example.com) vs. a sub-domain (sub.example.com) for a Service.

The DNS Provider will only implement the _domainconnect record at the domain level. This means that during discovery, the Service Provider would need to call the root domain for this information.

The DNS Provider should support configuring services on domains vs. sub-domains.

If the template is identical for the root and for the sub-domain, the Service Provider simply needs to call domain connect with the fully qualified domain name. Here passing in sub.example.com vs. example.com to the domain connect flow is all that is necessary.

If there are differences, two templates would be created and the Service Provider would invoke the appropriate version.

It is also highly recommended that this approach be taken, vs. variables for host names passed into the template.

Example Records: Single static host record

Consider a template for setting a single host record. The records section of the template would have a single record of type "A" and could have a value of:

```
[{
''type'': ''A'',
''host'': ''www'',
''pointsTo'': ''192.168.1.1'',
''ttl'': 600
}]
```

This would have no variable substitution and the application of this template to a domain would simply set the host name "www" to the IP address "192.168.1.1"

Example Records: Single variable host record for A

In the case of a template for setting a single host record from a variable, the template would have a single record of type "A" and could have a value of:

```
[{
''type'': ''A'',
''host'': ''@'',
''pointsTo'': ''192.168.1.%srv%'',
''ttl'': 600
}]
```

A query string with a key/value pair of

srv=8

would cause the application of this template to a domain to set the host name for the apex A record to the IP address "192.168.1.8" with a TTL of 600.

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:validate-1.0

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

6. Acknowledgements

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

- Chris Ambler of GoDaddy Inc.
- Jody Kolker of GoDaddy Inc.

7. Change History

7.1. Change from 02 to 03

Added width/height JSON values returned by DNS Provider Discovery. Corrected text of GET method for getting the authorization token. Added clarifying text to Group ID description parameter of the apply template POST method. Quite a few minor edits and clarifications that were found during implementation, especially in the Implementation Considerations section.

7.2. Change from 01 to 02

Added new GET method for Service Providers to determine if the DNS Provider supports a specific template. Some other minor edits for clarification.

7.3. Change from 00 to 01

Minor edits and clarifications found during implementation.

8. 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, <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>.

Authors' Addresses

Arnold Blinn
GoDaddy Inc.
14455 N. Hayden Rd. #219
Scottsdale, AZ 85260
US

Email: arnoldb@godaddy.com
URI: http://www.godaddy.com

Roger Carney
GoDaddy Inc.
14455 N. Hayden Rd. #219
Scottsdale, AZ 85260
US

Email: rcarney@godaddy.com
URI: http://www.godaddy.com

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: December 11, 2016

R. Carney
GoDaddy Inc.
J. Galvin
Afilias USA

June 9, 2016

Unavailable Domain Name File Format Specification
draft-carney-regext-unavailable-domains-00

Abstract

This document defines the file format for the storage of unavailable domain names and related details for a top level 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 December 11, 2016.

Table of Contents

1. Introduction

This document specifies a file format for the storage of unavailable domain names and related details for a top level domain (TLD) name registry, designed to facilitate interoperability and reusability among domain name registries and registrars.

The increased number of domain name registries and registrars, the greater number of reasons for domain names not being available, and to make accommodations for possible real-time communications issues between registries and registrars has driven the need for this standard sharable file.

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. General File Format Elements

File format will be a comma separated values (CSV) file and MUST use the US-ASCII character set. See informational RFC 4180 and RFC 7111 for details on the CSV file format.

Filename has two forms depending on content. For a file containing only a list of unavailable domain names in one Top Level Domain (TLD) the filename MUST be <TLD>-unavailablenames-<YYYY-MM-DDThhmmss>.csv. For a file containing a list of unavailable domain names in multiple Top Level Domains (TLDs) the filename MUST be unavailablenames-<YYYY-MM-DDThhmmss>.csv. For an IDN TLD <TLD> MUST be ace-encoded (i.e. in the "xn--" format). <YYYY-MM-DDThhmmss> MUST be the date and time (in UTC) the file was created indicating that all the names in the file were unavailable on that date at the time the file was created.

The first row MUST be the column headings: TLD, Domain Name, Status. The remaining rows will contain the data (all unavailable names) from the registry in the following structure: <TLD>, <Domain Name>, <Status>. Where <TLD> is the TLD of the domain name, <Domain Name> is the string of characters that represent the unavailable name, and <Status> is one of four values (REGISTERED, REGISTRY RESERVED, POLICY RESERVED, IDN VARIANT RESERVED). If the <TLD> and/or <Domain Name> is an IDN it MUST be ace-encoded (i.e. in the "xn--" format).

All domain name labels that are not available for registration in the SRS, for any reason, to the registrar channel MUST be included in this file.

3. Examples

3.1. Single TLD File Example

Example of a file that contains unavailable domain names from a single TLD.

FileName: example-unavailablenames-2016-05-01T010000.csv

```
TLD,Domain Name,Status
example,e.example,REGISTRY RESERVED
example,ex.example,POLICY RESERVED
example,example.example,REGISTERED
example,xn--4gqvdy3r.example,REGISTERED
```

3.2. Multiple TLDs File Example

Example of a file that contains unavailable domain names from multiple TLDs.

FileName: unavailablenames-2016-05-01T010000.csv

```
TLD,Domain Name,Status
example,e.example,REGISTRY RESERVED
test,ex.test,POLICY RESERVED
another,xn--4gqvdy3r.another,REGISTERED
```

4. Security Considerations

The file format described in this document does not provide any security services.

5. Acknowledgements

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

- Christine Turner of GoDaddy Inc.
- Jody Kolker of GoDaddy Inc.
- Richard Merdinger of GoDaddy Inc.
- Mike Runcieman of Afilias Canada
- Wayne Beaver of Afilias USA

6. Change History

7. 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>.

[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>.

[RFC7111] Hausenblas, M., Wilde, E., and J. Tennison, "URI Fragment Identifiers for the text/csv Media Type", RFC 7111, DOI 10.17487/RFC7111, January 2014, <http://www.rfc-editor.org/info/rfc7111>.

Authors' Addresses

Roger Carney
GoDaddy Inc.
14455 N. Hayden Rd. #219
Scottsdale, AZ 85260
US

Email: rcarney@godaddy.com
URI: http://www.godaddy.com

James Galvin
Afilias USA
300 Welsh Road, Building 3, Suite 105
Horsham, PA 19044
US

Email: jgalvin@afilias.info
URI: http://www.afilias.info

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2017

R. Carney
J. Snitker
GoDaddy Inc.

October 18, 2016

Validate Extension for the Extensible Provisioning Protocol (EPP)
draft-carney-regext-validate-01

Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension mapping for the validation of contact and eligibility data.

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 21, 2017.

Copyright Notice

Table of Contents

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 by which EPP clients and servers can reliably validate contact and eligibility data.

With the increased number of restrictions on contacts and required data points (license, ids, etc.) to register a domain name, a way to validate the data points prior to issuing a transform command is becoming more important.

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.

2. Object Attributes

This extension adds additional elements to EPP object mappings like the EPP domain name mapping [RFC5733]. Only those new elements are described here.

Key Value provides a flexible mechanism to share data between the client and the server. The <validate:kv> element defines the data, with two required simple attributes, key and value, and an optional contactType attribute for specificity in the response, more details below.

- An example <validate:kv key="VATID" value="0123456789"/>.
- An example <validate:kv contactType="Admin" key="contact:cc" value="Invalid country code for admin contact, must be MX."/>.

3. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in [RFC5730]. The command mappings described here are specifically for the Validate Extension

3.1. EPP Query Commands

EPP provides four 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, <poll> to discover and retrieve service messages queued by the server, and <transfer> to retrieve object transfer status information.

3.1.1. EPP <check> Command

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

The command frame MAY contain an <extension> element which MAY contain one child <validate:check> element with the following child element(s):

- one or more <validate:contact> element(s) for each contact that is to be validated that contains the contact type of the contact to be validated.

The <validate:contact> element MUST contain the following child elements:

- one <validate:cd> element.
- zero or more <validate:kv> elements.

The <validate:cd> element MUST contain the following child elements:

- one <validate:id> element.
- an OPTIONAL <validate:postalInfo> element.
- an OPTIONAL <validate:voice> element.
- an OPTIONAL <validate:fax> element.
- an OPTIONAL <validate:email> element.
- an OPTIONAL <validate:authInfo> element.
- an OPTIONAL <validate:disclose> element.

The following is an example of the <check> command using the <validate> extension.

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
C:  xmlns:validate="urn:ietf:params:xml:ns:validate-0.1"
C:  xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
C:  <command>
C:    <check>
C:      <validate:check>
C:        <validate:contact contactType="registrant" tld="COM">
C:          <validate:cd>
C:            <validate:id>sh8013</validate:id>
C:            <validate:postalInfo type="int">
C:              <contact:name>John Doe</contact:name>
C:              <contact:org>Example Inc.</contact:org>
C:              <contact:addr>
C:                <contact:street>123 Example Dr.</contact:street>
C:                <contact:street>Suite 100</contact:street>
C:                <contact:city>Dulles</contact:city>
C:                <contact:sp>VA</contact:sp>
C:                <contact:pc>20166-6503</contact:pc>
C:                <contact:cc>US</contact:cc>
C:              </contact:addr>
C:          </validate:postalInfo>
C:          <validate:voice>+1.7035555555</validate:voice>
C:          <validate:fax>+1.7035555556</validate:fax>
C:          <validate:email>jdoe@example.com</validate:email>
```

```
C:          <validate:authInfo>
C:            <contact:pw>2fooBAR</contact:pw>
C:          </validate:authInfo>
C:          <validate:disclose flag="0">
C:            <contact:voice/>
C:            <contact:email/>
C:          </validate:disclose>
C:        </validate:cd>
C:        <validate:kv key="VAT" value="1234567890"/>
C:      </validate:contact>
C:      <validate:contact contactType="tech" tld="COM">
C:        <validate:cd>
C:          <validate:id>sh8013</validate:id>
C:        </validate:cd>
C:      </validate:contact>
C:      <validate:contact contactType="admin" tld="COM">
C:        <validate:cd>
C:          <validate:id>sh8014</validate:id>
C:          <validate:postalInfo type="int">
C:            <contact:name>John Doe</contact:name>
C:            <contact:org>Example Inc.</contact:org>
C:            <contact:addr>
C:              <contact:street>123 Example Dr.</contact:street>
C:              <contact:street>Suite 100</contact:street>
C:              <contact:city>Dulles</contact:city>
C:              <contact:sp>VA</contact:sp>
C:              <contact:pc>20166-6503</contact:pc>
C:              <contact:cc>US</contact:cc>
C:            </contact:addr>
C:          </validate:postalInfo>
C:          <validate:voice>+1.7035555555</validate:voice>
C:          <validate:fax>+1.7035555556</validate:fax>
C:          <validate:email>jdoe@example.com</validate:email>
C:          <validate:authInfo>
C:            <contact:pw>2fooBAR</contact:pw>
C:          </validate:authInfo>
C:          <validate:disclose flag="0">
C:            <contact:voice/>
C:            <contact:email/>
C:          </validate:disclose>
C:        </validate:cd>
C:      </validate:contact>
C:      <validate:contact contactType="billing" tld="COM">
C:        <validate:cd>
C:          <validate:id>sh8014</validate:id>
C:        </validate:cd>
C:      </validate:contact>
C:    </validate:check>
```

```
C:    </check>
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 contain an <extension> element, which MUST contain a child <validate:chkData> element. The <validate:chkData> element MUST contain a <validate:cd> element for each <validate:check> element contained in the <check> command. The <validate:cd> element MUST contain the following child elements:

- one <validate:id> element.
- one <validate:response> element.
- zero or more <validate:kv> elements.

The following is an example of the <check> response using the <validate> 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:      <validate:chkData
S:        xmlns:validate="urn:ietf:params:xml:ns:validate-0.1">
S:        <validate:cd>
S:          <validate:id>sh8013</validate:id>
S:          <validate:response>1000</validate:response>
S:        </validate:cd>
S:        <validate:cd>
S:          <validate:id>sh8014</validate:id>
S:          <validate:response>2306</validate:response>
S:          <validate:kv key="contact:city" value="City not valid
S:            for state."/>
S:          <validate:kv contactType="Admin" key="contact:cc"
S:            value="Invalid country code for admin, must be mx."/>
S:          <validate:kv contactType="Billing" key="VAT" value="VAT
S:            required for Billing contact."/>
S:        </validate:cd>
S:      </validate:chkData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-ZYX</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.1.2. EPP <info> Command

This extension does not add any elements to the EPP <info> command or <info> response.

3.1.3. EPP <poll> Command

This extension does not add any elements to the EPP <poll> command or <poll> response.

3.1.4. EPP <transfer> Command

This extension does not add any elements to the EPP <transfer> command or <transfer> response.

3.2. EPP Query Commands

EPP provides five commands to transform objects: <create> to create an instance of an object with a server, <delete> to remove an instance of an object from a server, <renew> to extend the validity period of an object, <transfer> to manage changes in client sponsorship of an object, and <update> to change information.

3.2.1. EPP <create> Command

This extension does not add any elements to the EPP <create> command or <create> response.

3.2.2. EPP <delete> Command

This extension does not add any elements to the EPP <delete> command or <delete> response.

3.2.3. EPP <renew> Command

This extension does not add any elements to the EPP <renew> command or <renew> response.

3.2.4. EPP <transfer> Command

This extension does not add any elements to the EPP <transfer> command or <transfer> response.

3.2.5. EPP <update> Command

This extension does not add any elements to the EPP <update> command or <update> response.

4. Formal Syntax

One schema is presented here that is the EPP Validate 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. Validate Extension Schema

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema
   targetNamespace="urn:ietf:params:xml:ns:validate-0.1"
   xmlns:validate="urn:ietf:params:xml:ns:validate-0.1"
   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">

   <annotation>
     <documentation>
       Extensible Provisioning Protocol v1.0
       Validate Object Extension
     </documentation>
   </annotation>

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

<!--
Child elements of the <check> command.
-->
   <element name="check" type="validate:checkType"/>

   <complexType name="checkType">
     <sequence>
       <element name="contact"
         type="validate:validateContactType"
         maxOccurs="unbounded" />
     </sequence>
   </complexType>

   <complexType name="validateContactType">
     <sequence>
       <element name="cd"
         type="validate:checkDataType"/>
       <element name="kv"
         type="validate:kvType" minOccurs="0"
         maxOccurs="unbounded" />
```

```
      </sequence>
      <attribute name="contactType" type="eppcom:labelType"
        use="required"/>
      <attribute name="tld"
        type="eppcom:labelType" use="required"/>
    </complexType>

    <complexType name="checkDataType">
      <sequence>
        <element name="id"
          type="eppcom:clIDType" />
        <element name="postalInfo"
          type="contact:postalInfoType"
          minOccurs="0" maxOccurs="2" />
        <element name="voice"
          type="contact:e164Type" minOccurs="0" />
        <element name="fax"
          type="contact:e164Type" minOccurs="0" />
        <element name="email"
          type="eppcom:minTokenType" minOccurs="0"/>
        <element name="authInfo"
          type="contact:authInfoType"
          minOccurs="0"/>
        <element name="disclose"
          type="contact:discloseType"
          minOccurs="0" />
      </sequence>
    </complexType>

    <complexType name="kvType">
      <attribute name="contactType"
        type="eppcom:labelType" use="optional" />
      <attribute name="key"
        type="validate:keyType" use="required" />
     <attribute name="value"
        type="validate:valueType" use="required" />
    </complexType>

    <simpleType name="keyType">
      <restriction base="token">
        <minLength value="1" />
      </restriction>
    </simpleType>

    <simpleType name="valueType">
      <restriction base="token">
        <minLength value="0" />
      </restriction>
```

```
      </simpleType>

   <!--
   Child elements of the <check> response.
   -->
      <element name="chkData" type="validate:chkDataType" />

      <complexType name="chkDataType">
        <sequence>
          <element name="cd"
            type="validate:resCreateDataType" maxOccurs="unbounded" />
        </sequence>
      </complexType>

      <complexType name="resCreateDataType">
        <sequence>
          <element name="id"
            type="eppcom:clIDType" />
          <element name="response"
            type="epp:resultCodeType" />
          <element name="kv"
            type="validate:kvType"
            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]. The following URI assignment is requested of IANA:

URI: ietf:params:xml:ns:validate-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: "Validate 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 thank the following persons for their feedback and suggestions:

- Kevin Allendorf of GoDaddy Inc.
- Jody Kolker of GoDaddy Inc.
- James Gould of Verisign Inc

8. Change History

8.1. Change from 00 to 01

After review and broad feedback, extensive changes have been made transforming the original document from a standalone extension command to an extension using the <check> command and response framework.

9. 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>.

Authors' Addresses

Roger Carney
GoDaddy Inc.
14455 N. Hayden Rd. #219
Scottsdale, AZ 85260
US

Email: rcarney@godaddy.com
URI: http://www.godaddy.com

Joseph Snitker
GoDaddy Inc.
14455 N. Hayden Rd. #219
Scottsdale, AZ 85260
US

Email: jsnitker@godaddy.com
URI: http://www.godaddy.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

Table of Contents

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.

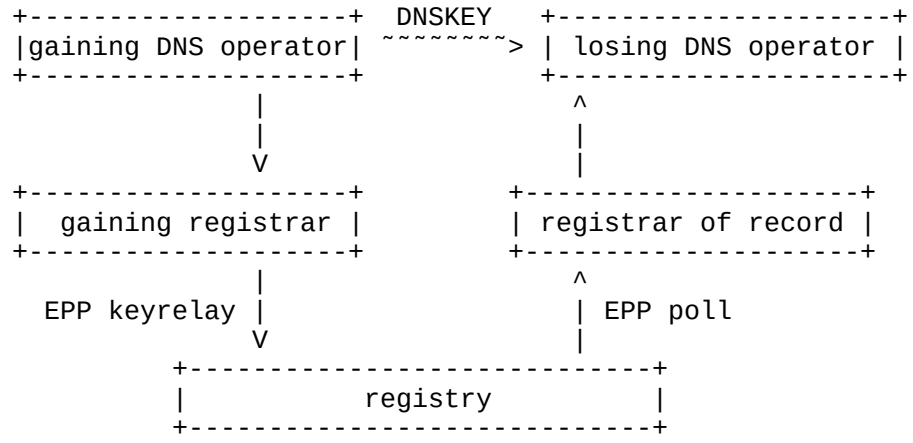

Figure 1: Transfer of DNSSEC key material.

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:

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

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

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

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

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

3. EPP Command Mapping

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

3.1. EPP Query Commands

EPP provides three commands to retrieve object information: <check> to determine if an object is known to the server, <info> to retrieve

detailed information associated with an object, and <transfer> to retrieve object transfer status information.

3.1.1. EPP <check> Command

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

3.1.2. EPP <info> Command

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

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

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

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>cmlraXN0aGViZXN0</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>cmlraXN0aGViZXN0</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 succesfully 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 authoritive 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-eppext-keyrelay-00

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

2. IDnits formatting ajustments

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

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

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

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

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

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

2. Updated the wording

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

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

2. Restored Acknowledgement section, thanks to Marco Davids

3. Incorperated feedback from Patrick Mevzek

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

1. Incorperated 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: 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 statue 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.

Table of Contents

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 statue or treaty.

This specification is intended to be used in the gTLD space, but nothing precludes the use of this format by other entities.

The detailed policy regarding the public key infrastructure (PKI), authorized validators, and other requirements must be defined based on the local policy of the entities using this specification. In the case of gTLDs, the detailed policy regarding the use of this specification is defined in the Rights Protection Mechanism Requirements document (see [ICANN-TMCH]), and the PKI is defined in [I-D.ietf-eppext-tmch-func-spec]. Implementations will need to implement such a PKI (or an equivalent) in order for the signatures defined in this document to have any useful semantics.

The objects specified in this document can be referenced by application protocols like the Extensible Provisioning Protocol (EPP), defined in [RFC5730].

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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

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

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

2. Object Description

This section defines the Mark and Signed Mark objects. Empty complex element types and abstract elements are defined to support additional Mark and Signed Mark definitions using XML schema substitution groups. Support for replacement through the XML schema substitution groups is included in the description of the objects.

This section defines some elements as OPTIONAL. If an elements is not defined as OPTIONAL, then it MUST be included in the object.

The following elements are defined as telephone numbers: <mark:voice>, <mark:fax> and <smd:voice>. The representation of telephone numbers in this specification is derived from structures defined in [ITU.E164.2005]. Telephone numbers described in this mapping are character strings that MUST begin with a plus sign ("+", ASCII value 0x002B), followed by a country code defined in [ITU.E164.2005], followed by a dot (".", ASCII value 0x002E), followed by a sequence of digits representing the telephone number. An optional "x" attribute is provided to note telephone extension information.

The following elements are defined as email addresses: <mark:email> and <smd:email>. Email address syntax is defined in [RFC5322].

2.1. Holder and Contacts objects

Marks are linked to Holder objects and optionally linked to Contact objects. This section defines the <mark:holder> and <mark:contact> objects.

- The child elements of <mark:holder> include:
  - A <mark:name> element that contains the name of the individual holder of the mark. At least one of <mark:name> and <mark:org>

MUST be specified, and <mark:name> is OPTIONAL if <mark:org> is specified.

* A <mark:org> element that contains the name of the organization holder of the mark. At least one of <mark:name> and <mark:org> MUST be specified, and <mark:org> is OPTIONAL if <mark:name> is specified.

* A <mark:addr> element that contains the address information of the holder of a mark. A <mark:addr> contains the following child elements:

  + One, two or three OPTIONAL <mark:street> elements that contains the holder's street address.

  + A <mark:city> element that contains the holder's city.

  + An OPTIONAL <mark:sp> element that contains the holder's state or province.

  + An OPTIONAL <mark:pc> element that contains the holder's postal code.

  + A <mark:cc> element that contains the holder's country code. This a two-character code from [ISO3166-2].

* An OPTIONAL <mark:voice> element that contains the holder's voice telephone number.

* An OPTIONAL <mark:fax> element that contains the holder's facsimile telephone number.

* An OPTIONAL <mark:email> element that contains the email address of the holder.

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:

- The <smd:id> that uniquely identifies an SMD in relation to a repository of SMDs potentially maintained by more than one issuer. The <smd:id> value is a concatenation of the local identifier, followed by a hyphen ("-", ASCII value 0x002D), followed by the issuer identifier.

- A <smd:issuerInfo> element that contains the information of the issuer of the mark registration. A "issuerID" attribute is used to specify the issuer identifier. The child elements include:

  * A <smd:org> element that contains the organization name of the issuer.

  * A <smd:email> element that contains the issuer customer support email address.

  * An OPTIONAL <smd:url> element that contains the HTTP or HTTPS URL of the issuer’s site.

  * An OPTIONAL <smd:voice> element that contains the issuer’s voice telephone number.

- A <smd:notBefore> element that contains the creation date and time of the SMD.

- A <smd:notAfter> element that contains the expiration date and time of the SMD.

- A <mark:mark> element that contains the mark information as defined in the Mark (Section 2.2) section.

The following is an example of an SMD:

```
<?xml version="1.0" encoding="UTF-8"?>
<smd:signedMark xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0"
id="smd1">
  <smd:id>0000001751376056503931-65535</smd:id>
  <smd:issuerInfo issuerID="65535">
    <smd:org>ICANN TMCH TESTING TMV</smd:org>
    <smd:email>notavailable@example.com</smd:email>
    <smd:url>https://www.example.com</smd:url>
    <smd:voice>+32.000000</smd:voice>
  </smd:issuerInfo>
```

```
   <smd: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>000520137346897313734689 73-65535</mark:id>
       <mark:markName>Test &amp;amp; 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/xmlenc#sha256"/>
 <DigestValue>wgyW3nZPoEfpptlhRILKnOQnbdtU6ArM7ShrAfHgDFg=</DigestValue>
    </Reference>
   </SignedInfo>
   <SignatureValue>
 jMu4PfyQGiJBF0GWSEPFCJjmywCEqR2h4LD+ge6XQ+JnmKFFCuCZS/3SLKAx0L1w
 QDFO2e0Y69k2G7/LGE37X3vOflobFM1oGwja8+GMVraoto5xAd4/AF7eHukgAymD
 o9toxoa2h0yV4A4PmXzsU6S86XtCcUE+S/WM72nyn47zoUCzzPKHZBRyeWehVFQ+
 jYRMIAMzM57HHQA+6eaXefRvtPETgUO4aVIVSugc4OUAZZwbYcZrC6wOaQqqqAZi
 30aPOBYbAvHMSmWSS+hFkbshomJfHxb97TD2grlYNrQIzqXk7WbHWy2SYdA+sI/Z
 ipJsXNa6osTUw1CzA7jfwA==
   </SignatureValue>
   <KeyInfo>
    <X509Data>
    <X509Certificate>
 MIIESTCCAzGgAwIBAgIBAjANBgkqhkiG9w0BAQsFADBiMQswCQYDVQQGEwJVUzEL
 MAkGA1UECBMCQ0ExFDASBgNVBAcTC0xvcyBBbmdlbGVzMRMwEQYDVQQKEwpJQ0FO
 TiBUTUNIMRswGQYDVQQDExJJQ0FOTiBUTUNIIFRFU1QgQ0EwHhcNMTMwMjA4MDAw
 MDAwWhcNMTgwMjA3MjM1OTU5WjBsMQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ0Ex
 FDASBgNVBAcTC0xvcyBBbmdlbGVzMRcwFQYDVQQKEw5WYWxpZGF0b3IgVE1DSDEh
 MB8GA1UEAxMYVmFsaWRhdG9yIFRNQ0ggVEVTVCBDRVJUMIIBIjANBgkqhkiG9w0B
 AQEFAAOCAQ8AMIIBCgKCAQEAo/cwvXhbVYl0RDWWvoyeZpETVZVVcMCovUVNg/sw
 WinuMgEWgVQFrz0xA04pEhXCFVv4evbUpekJ5buqU1gmQyOsCKQlhOHTdPjvkC5u
 pDqa51Flk0TMaMkIQjs7aUKCmA4RG4tTTGK/EjR1ix8/D0gHYVRldy1YPrMP+ou7
 5bOVnIos+HifrAtrIv4qEqwLL4FTZAUpaCa2BmgXfy2CSRQbxD5Or1gcSa3vurh5
 sPMCNxqaXmIXmQipS+DuEBqMM8tldaN7RYojUEKrGVsNk5i9y2/7sjn1zyyUPf7v
 L4GgDYqhJYWV61DnXgx/Jd6CWxvsnDF6scscQzUTEl+hywIDAQABo4H/MIH8MAwG
 A1UdEwEB/wQCMAAwHQYDVR0OBBYEFPZEcIQcD/Bj2IFz/LERuo2ADJviMIGMBgNV
 HSMEgYQwgYGAFO0/7kEh3FuEKS+Q/kYHaD/W6wihoWakZDBiMQswCQYDVQQGEwJV
 UzELMAkGA1UECBMCQ0ExFDASBgNVBAcTC0xvcyBBbmdlbGVzMRMwEQYDVQQKEwpJ
 Q0FOTiBUTUNIMRswGQYDVQQDExJJQ0FOTiBUTUNIIFRFU1QgQ0GCAQEwDgYDVR0P
 AQH/BAQDAgeAMC4GA1UdHwQnMCUwI6AhoB+GHWh0dHA6Ly9jcmwuaWNhbm4ub3Jn
 L3RtY2guY3JsMA0GCSqGSIb3DQEBCwUAA4IBAQB2qSy7ui+43cebKUKwWPrzz9y/
 IkrMeJGKjo40n+9uekaw3DJ5EqiOf/qZ4pjBD++oR6BJCb6NQuQKwnoAz5lE4Ssu
 y5+i93oT3HfyVc4gNMIoHm1PS19l7DBKrbwbzAea/0jKWVzrvmV7TBfjxD3AQo1R
 bU5dBr6IjbdLFlnO5x0G0mrG7x5OUPuurihyiURpFDpwH8KAH1wMcCpXGXFRtGKk
 wydgyVYAty7otkl/z3bZkCVT34gPvF70sR6+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
rIHhtbG5zOnNtZD0idXJuOmlldGY6cGFyYW1zOnhtbDpuczpzaWduZWRNYXJrLTEuMCIgaW
... (base64 data elided for brevity) ...
PC9zbWQ6c2lnbmVkTWFyaz4=
</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

```
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"/>
```

```
            <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:e164Type" 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


```
 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:e164Type" minOccurs="0"/>
      <element name="fax" type="mark:e164Type" 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:e164Type"/>
      <element name="fax" type="mark:e164Type" 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"
        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"/>
```

```
      <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>

  <!--
  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-eppext-tmch-smd for a variety of Top Level Domains (TLD's).

Level of maturity: Production

Coverage: Implements parsing and validation of all aspects of draft-ietf-eppext-tmch-smd including the Signed Mark, the Encoded Signed Mark, and the contained Mark. Implements the encoding of the Mark in supporting the response of draft-ietf-eppext-launchphase.

Licensing: Proprietary

Contact: jgould@verisign.com

4.3. Verisign .COM / .NET SRS

Organization: Verisign Inc.

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

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

Level of maturity: Operational Test Environment (OTE)

Coverage: Implements parsing and validation of all aspects of draft-ietf-eppext-tmch-smd including the Signed Mark, the Encoded Signed Mark, and the contained Mark.

Licensing: Proprietary

Contact: jgould@verisign.com

4.4. REngin v3.7

Organisation: Domain Name Services (Pty) Ltd

Name: REngin v3.7

Description: Server side implementation only

Level of maturity: Production

Coverage: All aspects of draft-ietf-eppext-tmch-smd have been implemented

Licensing: Proprietary Licensing with Maintenance Contracts

Contact: info@dnservices.co.za

URL: http://domain-name.services

4.5. Uniregistry Corp. Shared Registry System (uSRS)

Organization: Uniregistry Corp.

Name: Uniregistry Corp. Shared Registry System (uSRS)

Description: Uniregistry's Shared Registry System implements the server-side of draft-ietf-eppext-tmch-smd for its TLD registry.

Level of maturity: Production

Coverage: Implements parsing and validation of all aspects of draft-ietf-eppext-tmch-smd including the Signed Mark, the Encoded Signed Mark, and the contained Mark. Implements the encoding of the Mark in supporting the response of draft-ietf-eppext-launchphase.

Licensing: Proprietary

Contact: fobispo@uniregistry.link

5. Acknowledgements

Special thanks to Chris Wright for creating the first prototype of a SMD; James Gould, Wil Tan and Gavin Brown for creating the mark and SMD definitions in their EPP draft launch extension on which this draft is based. Portions of the security section were shamefully copied from RFC5105. The author would like to acknowledge the following individuals for their contributions to this document: Scott Hollenbeck and Jan Jansen.

6. IANA Considerations

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

Registration request for the signed mark namespace:

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

Registrant Contact: IESG

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

Registration request for the signed mark schema:

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

Registrant Contact: IESG

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

Registration request for the mark namespace:

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

Registrant Contact: IESG

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

Registration request for the mark schema:

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

Registrant Contact: IESG

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

7. Security Considerations

The security of a Signed Mark object depends on the security of the underlying XML DSIG algorithms. As such, all the security considerations from [XMLDSIG] apply here as well.

The digital signature algorithm used in Signed Mark objects SHOULD be RSA-SHA256 [RFC4051]. The size of the RSA key SHOULD be at least 2048 bits. A valid reason for choosing something else would be if RSA-SHA256 would be deemed to not provide sufficient security.

In the case of the ICANN Trademark Clearinghouse (TMCH), Signed Mark objects use the algorithms for digesting and signing recommended in this document.

Signed Marks are used primarily for sunrise domain name registrations in gTLDs, but other third parties might be using them. A party using Signed Marks should verify that the digital signature is valid based on local policy. In the case of gTLDs, the RPM Requirements document [ICANN-TMCH] defines such policy, and the PKI is defined in [I-D.ietf-eppext-tmch-func-spec]. Implementations will need to implement such a PKI (or an equivalent) in order for the signatures defined in this document to have any useful semantics.

8. References

8.1. Normative References

[ICANN-TMCH]
ICANN, "ICANN Trademark Clearinghouse, Rights Protection Mechanism Requirements", 2013, <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/xmldsig-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: Informational
Expires: April 7, 2020

N. Kong
Consultant
J. Yao
L. Zhou
CNNIC
W. Tan
Cloud Registry
J. Xie

October 5, 2019

Extensible Provisioning Protocol (EPP) Domain Name Mapping Extension for Strict Bundling Registration
draft-ietf-regext-bundling-registration-11

Abstract

This document describes an extension of Extensible Provisioning Protocol (EPP) domain name mapping for the provisioning and management of strict 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. This is a non-standard proprietary extension.

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 https://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 7, 2020.

Table of Contents

1. Introduction

Bundled domain names are those which share the same TLD but whose second level labels are variants, or those which have identical second level labels for which certain parameters are shared in different TLDs. For an example, Public Interest Registry has requested to implement bundling of second level domains for .NGO and .ONG. So we have two kinds of bundled domain names. The first one is in the form of "V-label.TLD" in which the second level label (V-label) is a variant sharing the same TLD; Second one is in the form of "LABEL.V-tld" in which the second level label(LABEL) remains the same but ending with a different TLD (V-tld).

Bundled domain names normally share some attributes. Policy-wise bundling can be implemented in three ways. The first one is strict bundling, which requires all bundled names to share many same attributes. When creating, updating, or transferring of any of the bundled domain names, all bundled domain names will be created, updated or transferred atomically. The second one is partial bundling, which requires the bundled domain names to be registered by the same registrant. The third one is relaxed bundling, which has no specific requirements on the domain registration. This document mainly addresses the strict bundling names registration.

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, most registries offering Chinese Domain Name (CDN) adopt a registration policy whereby a registrant can apply for 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 share a common set of attributes.

The basic Extensible Provisioning Protocol (EPP) domain name mapping [RFC5731] provides the facility for single domain name registration. It does not specify how to register the strict bundled names which share many of the attributes.

In order to meet the above requirements of strict bundled name registration, this document describes an extension of the EPP domain name mapping [RFC5731] for the provisioning and management of bundled names. This document describes a non-standard proprietary extension.

This extension is specially useful for registries of practising Chinese domain name registration. This document is specified using 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 mapping described in this document.

This document uses many IDN concepts, so a thorough understanding of the IDNs for Application (IDNA, described in [RFC5890], [RFC5891], and [RFC5892]) and the variant approach discussed in [RFC4290] is assumed.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

uLabel in this document is used to express the U-label of an internationalized domain name as a series of characters where non-ASCII characters will be represented in the format of "&#xXXXX;" where XXXX is a UNICODE point by using the XML escaping mechanism. U-Label is defined in [RFC5890].

The XML namespace prefix "b-dn" is used for the namespace "urn:ietf:params:xml:ns:epp:b-dn", but implementations MUST NOT rely 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

The following definitions are used in this document:

- Registered Domain Name (RDN), represents the valid domain name that users submitted for the initial registration.

- 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 places them in the registration package together with the RDN.

The bundled names share many properties, such as expiration date and sponsoring registrar, by sharing the same domain object. So when users update any property of a domain object within a bundle package, that property of all other domain objects in the bundle package will be updated at the same time.

5. Requirement for Bundling Registration of Names

The bundled names whether they are in the form of "V-label.TLD" or in the form of "LABEL.V-tld" should share some parameter or attributes associated with domain names. Typically, bundled names will share the following parameters or attributes:

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

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 of the bundle names 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 registered, 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 are returned 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.

For example: <b-dn:rdn uLabel="实例.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="實例.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:
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
S:            </domain:name>
S:            <domain:reason>This associated domain name is
S:              a produced name based on bundle name policy.
S:            </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, unless some registration policy has some special processing, 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 registration policy. The <b-dn:infData> element contains the <b-dn:bundle> which has the following child elements:

- An <b-dn:rdn> element that contains the RDN, along with the attribute described below.

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

The above elements contain the following attribute:

- 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
S:        </domain:crDate>
S:        <domain:exDate>2012-04-03T22:00:00.0Z
S:        </domain:exDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <b-dn:infData
S:        xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example">
```

```
S:            xn--fsq270a.example
S:          </b-dn:rdn>
S:          <b-dn:bdn uLabel="&#x5BE6;&#x4F8B;.example">
S:            xn--fsqz41a.example
S:          </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 theses commands have been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. Unless some registration policy has some special processing, this EPP <extension> element SHOULD contain the <b-dn:bundle> which has the following child elements:

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

The above elements contain the following attribute:

- An optional "uLabel" attribute represents the U-label of the element.

7.2.1. EPP <create> Command

This extension defines additional elements to extend the EPP <create> command described in the EPP domain name mapping [RFC5731] for bundled names registration.

In addition to the EPP command elements described in the EPP domain mapping [RFC5731], the <create> command SHALL contain an <extension> element. Unless some registration policy has some special processing, the <extension> element SHOULD contain a child <b-dn:create> element that identifies the bundle namespace, and a child <b-dn:rdn> element that identifies the U-Label form of the registered domain name with the uLabel attribute.

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:epp:b-dn">
C:        <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example">
C:          xn--fsq270a.example
C:        </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, unless some registration policy has some special processing, 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 registration policy. The <b-dn:creData> element contains the <b-dn:bundle> element.

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>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:epp:b-dn">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example">
S:            xn--fsq270a.example
S:          </b-dn:rdn>
S:          <b-dn:bdn uLabel="&#x5BE6;&#x4F8B;.example" >
S:            xn--fsqz41a.example
S:          </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>
```

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, unless some registration policy has some special processing, 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 registration 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:epp:b-dn">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example">
S:            xn--fsq270a.example
S:          </b-dn:rdn>
S:          <b-dn:bdn uLabel="&#x5BE6;&#x4F8B;.example">
S:            xn--fsqz41a.example
S:          </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 <extension> element SHALL be contained in the response if the domain object has data associated with bundled names. Unless some registration policy has some special processing, this EPP <extension> element SHOULD contain the <b-dn:renData> which contains <b-dn:bundle> element.

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>xn--fsq270a.example</domain:name>
S:        <domain:exDate>2012-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:renData>
S:    </resData>
S:    <extension>
S:      <b-dn:renData
S:        xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example">
S:            xn--fsq270a.example
S:          </b-dn:rdn>
S:          <b-dn:bdn uLabel="&#x5BE6;&#x4F8B;.example" >
S:            xn--fsqz41a.example
S:          </b-dn:bdn>
S:        </b-dn:bundle>
S:      </b-dn:renData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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]. However, additional elements are defined for the <transfer> response in the EPP object mapping. When the command has been processed successfully, the EPP <extension> element SHALL be contained in the response if the domain object has data associated with bundled names. Unless some registration policy has some special processing, this EPP <extension> element SHOULD contain the <b-dn:trnData> which contains <b-dn:bundle> element.

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>xn--fsq270a.example</domain:name>
S:        <domain:trStatus>pending</domain:trStatus>
S:        <domain:reID>ClientX</domain:reID>
S:        <domain:reDate>2011-04-03T22:00:00.0Z</domain:reDate>
S:        <domain:acID>ClientY</domain:acID>
S:        <domain:acDate>2011-04-08T22:00:00.0Z</domain:acDate>
S:        <domain:exDate>2012-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:trnData>
S:    </resData>
S:    <extension>
S:      <b-dn:trnData
S:        xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example">
S:            xn--fsq270a.example
S:          </b-dn:rdn>
S:          <b-dn:bdn uLabel="&#x5BE6;&#x4F8B;.example">
S:            xn--fsqz41a.example
S:          </b-dn:bdn>
S:        </b-dn:bundle>
S:      </b-dn:trnData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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]. However, additional elements are defined for the <update> response in the EPP object mapping. When the command has been processed successfully, the EPP <extension> element SHALL be contained in the response if the domain object has data associated with bundled names. Unless some

registration policy has some special processing, this EPP <extension> element SHOULD contain the <b-dn:upData> which contains <b-dn:bundle> 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:    <extension>
S:      <b-dn:upData
S:        xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example" >
S:            xn--fsq270a.example
S:          </b-dn:rdn>
S:          <b-dn:bdn uLabel="&#x5BE6;&#x4F8B;.example">
S:            xn--fsqz41a.example
S:          </b-dn:bdn>
S:        </b-dn:bundle>
S:      </b-dn:upData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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:epp:b-dn"
  xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn"
  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"/>

  <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"/>
    </sequence>
  </complexType>

  <!--
    Child response elements in <b-dn:infData>, <b-dn:delData>,
    <b-dn:creData>, <b-dn:renData>, <b-dn:trnData> and <b-dn:upData>.
  -->
  <element name="infData" type="b-dn:bundleDataType"/>
  <element name="delData" type="b-dn:bundleDataType"/>
  <element name="creData" type="b-dn:bundleDataType"/>
  <element name="renData" type="b-dn:bundleDataType"/>
  <element name="trnData" type="b-dn:bundleDataType"/>
  <element name="upData" type="b-dn:bundleDataType"/>

  <complexType name="bundleDataType">
    <sequence>
      <element name="bundle" type="b-dn:bundleType" />
    </sequence>
  </complexType>
```

```
<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 assignment the following two URIs.

Registration request for the IDN namespace:

- URI: urn:ietf:params:xml:ns:epp:b-dn

- Registrant Contact: See the "Author's Address" section of this document.
- XML: None. Namespace URI does not represent an XML specification.

Registration request for the IDN XML schema:

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

The EPP extension described in this document should be registered by IANA in the "Extensions for the Extensible Provisioning Protocol (EPP)" registry described in [RFC7451]. The details of the registration are as follows:

- Name of Extension: "Domain Name Mapping Extension for Strict Bundling Registration"
- Document status: Informational
- Reference: This document
- Registrant Name and Email Address: IESG, iesg@ietf.org
- Top-Level Domains (TLDs): Any
- IPR Disclosure: https://datatracker.ietf.org/ipr/
- Status: Active
- Notes: None

11. Security Considerations

Some registries and registrars have more than 15 years of the bundled registration of domain names (especially Chinese domain names). They have not found any significant security issues. One principle that the registry and registrar should let the registrants know is that bundled registered domain names will be created, transferred, updated, and deleted together as a group. The registrants for bundled domain names should remember this principle when doing some operations to these domain names. [RFC5730] also introduces some security consideration.

This document does not take a position regarding whether or not the bundled domain names share a DS/DNSKEY key. The DNS administrator can choose whether DS/DNSKEY information can be shared or not. If a DS/DNSKEY key is shared then the bundled domain names share fate if there is a key compromise.

12. Implementation Status

Note to RFC Editor: Please remove this section before publication.

- The Chinese Domain Name Consortium(CDNC) including CNNIC, TWNIC, HKIRC, MONIC, SGNIC and more have followed the principles defined in this document for many years.

- CNNIC and TELEINFO have implemented this extension in their EPP based Chinese domain name registration system.

- Public Interest Registry, has requested to implement technical bundling of second level domains for .NGO and .ONG. This means that by registering and purchasing a domain in the .ngo TLD, for an example, the NGO registrant is also registering and purchasing the corresponding name in the .ong TLD (and vice-versa for registrations in .ong).

- Patrick Mevzek has released a new version of Net::DRI, an EPP client (Perl library, free software) implementing this extension.

13. Acknowledgements

The authors especially thank the authors of [RFC5730] and [RFC5731] and the following ones of CNNIC: Weiping Yang, Chao Qi.

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-ietf-regext-bundle-registration: Version 00

- accepted as WG document.

14.2. draft-ietf-regext-bundle-registration: Version 01

- make this document to focus on the restrict bundled domain name registration.

14.3. draft-ietf-regext-bundle-registration: Version 02

- Update the section of implementation status.

14.4. draft-ietf-regext-bundle-registration: Version 03

- This document is changed to informational category.
- Refine the text.

14.5. draft-ietf-regext-bundle-registration: Version 04

- Update the implementation section.
- Refine the text.

14.6. draft-ietf-regext-bundle-registration: Version 05

- Scope the XML namespaces to include 'epp'.

14.7. draft-ietf-regext-bundle-registration: Version 06

- add some examples for the transfer, update and renew command
- add some text to security consideration

14.8. draft-ietf-regext-bundle-registration: Version 07

- Update IANA consideration section based on Scott's comments
- Update security consideration based on Chair and Patrick Mevzek's comments

14.9. draft-ietf-regext-bundle-registration: Version 08

- Refine some texts.

14.10. draft-ietf-regext-bundle-registration: Version 09

- Refine the texts.

14.11. draft-ietf-regext-bundle-registration: Version 10

- Update the texts based on IETF LC.

14.12. draft-ietf-regext-bundle-registration: Version 11

- Update the texts based on AD's comment.

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, <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>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://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, <https://www.rfc-editor.org/info/rfc5892>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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

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

Authors' Addresses

Ning Kong
Consultant

Email: ietfing@gmail.com

Jiankang Yao
CNNIC
4 South 4th Street,Zhongguancun,Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 3007
Email: yaojk@cnnic.cn

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

Phone: +86 10 5881 2677
Email: zhoulinlin@cnnic.cn

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

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

Jiagui Xie

Email: jiagui1984@163.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 1, 2017

J. Gould
VeriSign, Inc.

October 28, 2016

Extensible Provisioning Protocol (EPP) and Registration Data Access Protocol (RDAP) Status Mapping
draft-ietf-regext-epp-rdap-status-mapping-04

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 May 1, 2017.

Table of Contents

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 Extensible Provisioning Protocol (EPP) Domain Name Mapping [RFC5731], section 2.3 of the Extensible Provisioning Protocol (EPP) Host Mapping [RFC5732], section 2.2 of the Extensible Provisioning Protocol (EPP) Contact Mapping [RFC5733], and section 3.1 of Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP) [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. EPP provides status codes that allow distinguishing the case that an action is prohibited because of server policy from the case that an action is prohibited because of a client request. The ability to make this distinction needs to be preserved in RDAP.

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; This grace period is provided after the initial registration of the object. If the object is deleted by the client during this period, the server provides a credit to the client for the cost of the registration.

autoRenewPeriod = auto renew period; This grace period is provided after an object registration period expires and is extended (renewed) automatically by the server. If the object is deleted by the client during this period, the server provides a credit to the client for the cost of the auto renewal.

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

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

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

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

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

pendingRestore = pending restore; An object is in the process of being restored after being in the redemption period state.

redemptionPeriod = redemption period; 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; This grace period is provided after an object registration period is explicitly extended (renewed) by the client. If the object is deleted by the client during this period, the server provides a credit to the client for the cost of the renewal.

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

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

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

serverUpdateProhibited = server update prohibited; The server set the status so that requests to update the object (other than to remove this status) MUST be rejected.

serverHold = server hold; The server set the status so that DNS delegation information MUST NOT be published for the object.
transferPeriod = transfer period; This grace period is provided after the successful transfer of object registration sponsorship from one client to another client. If the object is deleted by the client during this period, the server provides a credit to the client 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: This grace period is provided after the initial registration of the object. If the object is deleted by the client

during this period, the server provides a credit to the client for the cost of the registration. This maps to the Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP) [RFC3915] 'addPeriod' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: auto renew period

Type: status

Description: This grace period is provided after an object registration period expires and is extended (renewed) automatically by the server. If the object is deleted by the client during this period, the server provides a credit to the client for the cost of the auto renewal. This maps to the Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP) [RFC3915] 'autoRenewPeriod' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: client delete prohibited

Type: status

Description: The client requested that requests to delete the object MUST be rejected. This maps to the Extensible Provisioning Protocol (EPP) Domain Name Mapping [RFC5731], Extensible Provisioning Protocol (EPP) Host Mapping [RFC5732], and Extensible Provisioning Protocol (EPP) Contact Mapping [RFC5733] 'clientDeleteProhibited' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: client hold

Type: status

Description: The client requested that the DNS delegation information MUST NOT be published for the object. This maps to the Extensible Provisioning Protocol (EPP) Domain Name Mapping [RFC5731] 'clientHold' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: client renew prohibited

Type: status

Description: The client requested that requests to renew the object MUST be rejected. This maps to the Extensible Provisioning Protocol (EPP) Domain Name Mapping [RFC5731] 'clientRenewProhibited' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: client transfer prohibited

Type: status

Description: The client requested that requests to transfer the object MUST be rejected. This maps to the Extensible Provisioning Protocol (EPP) Domain Name Mapping [RFC5731] and Extensible Provisioning Protocol (EPP) Contact Mapping [RFC5733] 'clientTransferProhibited' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: client update prohibited

Type: status

Description: The client requested that requests to update the object (other than to remove this status) MUST be rejected. This maps to the Extensible Provisioning Protocol (EPP) Domain Name Mapping [RFC5731], Extensible Provisioning Protocol (EPP) Host Mapping [RFC5732], and Extensible Provisioning Protocol (EPP) Contact Mapping [RFC5733] 'clientUpdateProhibited' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: pending restore

Type: status

Description: An object is in the process of being restored after being in the redemption period state. This maps to the Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP) [RFC3915] 'pendingRestore' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: redemption period

Type: status

Description: 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. This maps to the Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP) [RFC3915] 'redemptionPeriod' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: renew period

Type: status

Description: This grace period is provided after an object registration period is explicitly extended (renewed) by the client. If the object is deleted by the client during this period, the server provides a credit to the client for the cost of the renewal. This maps to the Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP) [RFC3915] 'renewPeriod' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: server delete prohibited

Type: status

Description: The server set the status so that requests to delete the object MUST be rejected. This maps to the Extensible Provisioning Protocol (EPP) Domain Name Mapping [RFC5731], Extensible Provisioning Protocol (EPP) Host Mapping [RFC5732], and Extensible Provisioning Protocol (EPP) Contact Mapping [RFC5733] 'serverDeleteProhibited' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: server renew prohibited

Type: status

Description: The server set the status so that requests to renew the object MUST be rejected. This maps to the Extensible Provisioning Protocol (EPP) Domain Name Mapping [RFC5731] 'serverRenewProhibited' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: server transfer prohibited

Type: status

Description: The server set the status so that requests to transfer the object MUST be rejected. This maps to the Extensible Provisioning Protocol (EPP) Domain Name Mapping [RFC5731] and Extensible Provisioning Protocol (EPP) Contact Mapping [RFC5733] 'serverTransferProhibited' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: server update prohibited

Type: status

Description: The server set the status so that requests to update the object (other than to remove this status) MUST be rejected. This maps to the Extensible Provisioning Protocol (EPP) Domain Name Mapping [RFC5731], Extensible Provisioning Protocol (EPP) Host Mapping [RFC5732], and Extensible Provisioning Protocol (EPP) Contact Mapping [RFC5733] 'serverUpdateProhibited' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: server hold

Type: status

Description: The server set the status so that DNS delegation information MUST NOT be published for the object. This maps to the Extensible Provisioning Protocol (EPP) Domain Name Mapping [RFC5731] 'serverHold' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: transfer period

Type: status

Description: This grace period is provided after the successful transfer of object registration sponsorship from one client to another client. If the object is deleted by the client during this period, the server provides a credit to the client for the cost of the transfer. This maps to the Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP) [RFC3915] 'transferPeriod' status.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

4. Security Considerations

The status values described in this document can be subject to server-side information disclosure policies that restrict display of the values to authorized clients. Implementers may wish to review [RFC7481] for a description of the RDAP security services that can be used to implement information disclosure policies.

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.

[RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", RFC 7481, DOI 10.17487/RFC7481, March 2015, <http://www.rfc-editor.org/info/rfc7481>.

[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. Acknowledgements

Suggestions that have been incorporated into this document were provided by Andrew Newton, Scott Hollenbeck, Jim Galvin, Gustavo Lozano, and Robert Sparks.

Appendix B. Change History

B.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.

B.2. Change from 01 to 02

1. Ping update.

B.3. Change from 02 to 03

1. Ping update.

B.4. Change from 03 to REGEXT 00

1. Changed to regext working group draft by changing draft-gould-epp-rdap-status-mapping to draft-ietf-regext-epp-rdap-status-mapping.

B.5. Change from REGEXT 00 to REGEXT 01

1. Updated based on regext mailing feedback from Scott Hollenbeck that included updating the registrant for the registration of the new statuses to IESG and iesg@ietf.org, and revising the security section. Changed to standards track based on suggestion by Jim Galvin and support from Gustavo Lozano on the regext mailing list.

B.6. Change from REGEXT 01 to REGEXT 02

1. Updated the text associated with distinguishing client and server prohibited statuses in RDAP based on feedback by Robert Sparks on the regext mailing list.
2. Removed the "For DNR that indicates" text from the description of the statuses based on feedback by Robert Sparks on the regext mailing list.
3. Made a few editorial changes to the status descriptions including referring to "redemption period" instead of "redemptionPeriod" and referring to "object" instead of "domain name".
4. Changed all references of "registrar" to "client" and "registry" to "server" in the status descriptions to be consistent.

B.7. Change from REGEXT 02 to REGEXT 03

1. Updated descriptions of the add period, auto renew period, renew period, and transfer period statuses to better reflect what the status is in RFC 3915, based on feedback by Robert Sparks on the regext mailing list.

B.8. Change from REGEXT 03 to REGEXT 04

1. Updated the descriptions of the JSON Values Registry entries to include a reference back to the appropriate EPP RFC status, based on feedback by Sabrina Tanamal from IANA.

Author's Address

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

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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: June 15, 2018

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

December 12, 2017

Launch Phase Mapping for the Extensible Provisioning Protocol (EPP) draft-ietf-regext-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 June 15, 2018.

Copyright Notice

Table of Contents

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 as they begin operation 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. Multiple launch phases and multiple models are supported to enable the launch of a domain name registry. What is supported and what is validated is up to server policy. Communication of the server policy is typically performed using an out-of-band mechanism that is not specified in this document.

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, RFC 7848 [RFC7848] 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. The use of "..." is used as shorthand for elements defined outside this document.

A Launch Registration is a domain name registration during a launch phase when the server uses a "first-come, first-served" model. Only

a single registration for a domain name can exist in the server at a time.

A Launch Application represents the intent to register a domain name during a launch phase when the server accepts multiple applications for a domain name and the server later selects one of the applications to allocate as a registration. Many Launch Applications for a domain name can exist in the server at a time.

The XML namespace prefix "launch" is used for the namespace "urn:ietf:params:xml:ns:launch-1.0", 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.

The XML namespace prefix "smd" is used for the [RFC7848] namespace "urn:ietf:params:xml:ns:signedMark-1.0", 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.

The XML namespace prefix "mark" is used for the [RFC7848] namespace "urn:ietf:params:xml:ns:mark-1.0", 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 <domain:create> command 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.

2.2. Validator Identifier

The Validator Identifier is the identifier unique to the server, 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 unique set of Validator Identifier values supported by the server is up to server policy. 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 [RFC7848]. Both the Validator Identifier and the Issuer Identifier used MUST be unique in the server. If the ICANN TMCH is not used or multiple Trademark Validators are used, the server MUST define the list of supported validator identifiers and MUST make this information available to clients using a mutually acceptable, out-of-band mechanism.

The Validator Identifier may define a non-Trademark Validator that supports a form of claims, where claims and a Validator Identifier can be used for purposes beyond trademarks.

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, 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 phase, as defined in the Section 2.3.1, in which a Claims Notice must be displayed to a prospective registrant of a domain name that matches trademarks.

open: A 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 Section 2.3.1, 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>.

2.3.1. Trademark Claims Phase

The Trademark Claims Phase is when a Claims Notice must be displayed to a prospective registrant of a domain name that matches trademarks. See [I-D.ietf-regext-tmch-func-spec] for additional details of trademark claims handling. The source of the trademarks is a Trademark Validator and the source of the Claims Notice information is a Claim Notice Information Service (CNIS), which may be directly linked to a Trademark Validator. The client interfaces with the server to determine if a trademark exists for a domain name, interfaces with a CNIS to get the Claims Notice information, and interfaces with the server to pass the Claims Notice acceptance information in a create command. This document supports the Trademark Claims Phase in two ways including:

Claims Check Form: Is defined in Section 3.1.1 and is used to determine whether or not there are any matching trademarks for a domain name. If there is at least one matching trademark that exists for the domain name, a claims key is returned. The mapping of domain names and the claims keys is based on an out-of-band interface between the server and the Trademark Validator. The CNIS associated with the claims key Validator Identifier (Section 2.2) MUST accept the claims key as the basis for retrieving the claims information.

Claims Create Form: Is defined in Section 3.3.2 and is used to pass the Claims Notice acceptance information in a create command. The notice identifier (<launch:noticeID>) format, validation rules, and server processing is up to the interface between the server and the Trademark Validator. The CNIS associated with the Validator Identifier (Section 2.2) MUST generate a notice identifier compliant with the <launch:noticeID> element.

The following shows the Trademark Claims Phase registration flow:

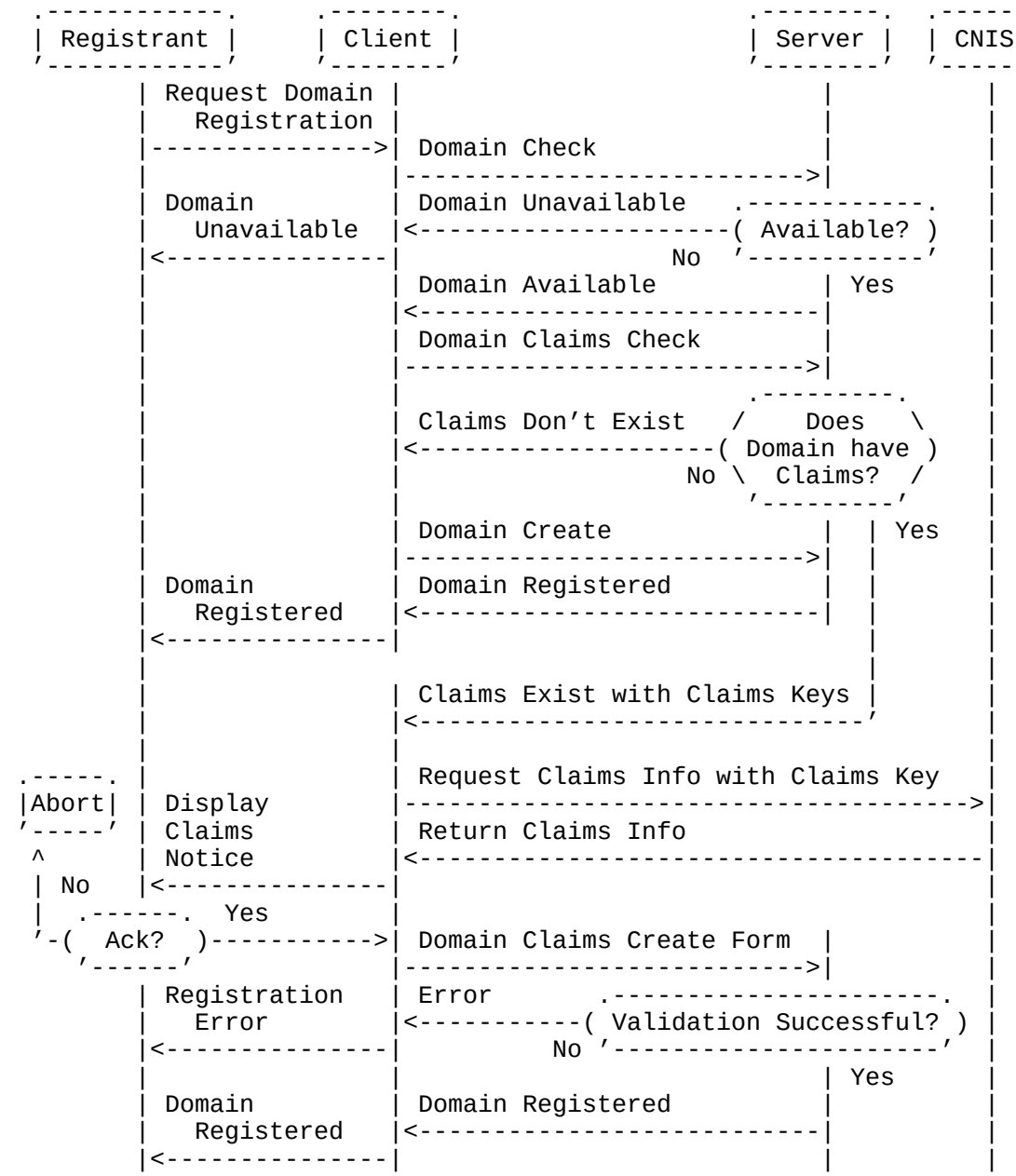

Figure 1

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 ("allocated" and "rejected").

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. This is a possible end state of an application or registration object.
rejected: The application or registration object was not provisioned. This 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, as defined in [RFC5646], 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 use one of the non-"custom" status values.

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

The transitions between the states is a matter of server policy. This diagram defines one possible set of permitted transitions.

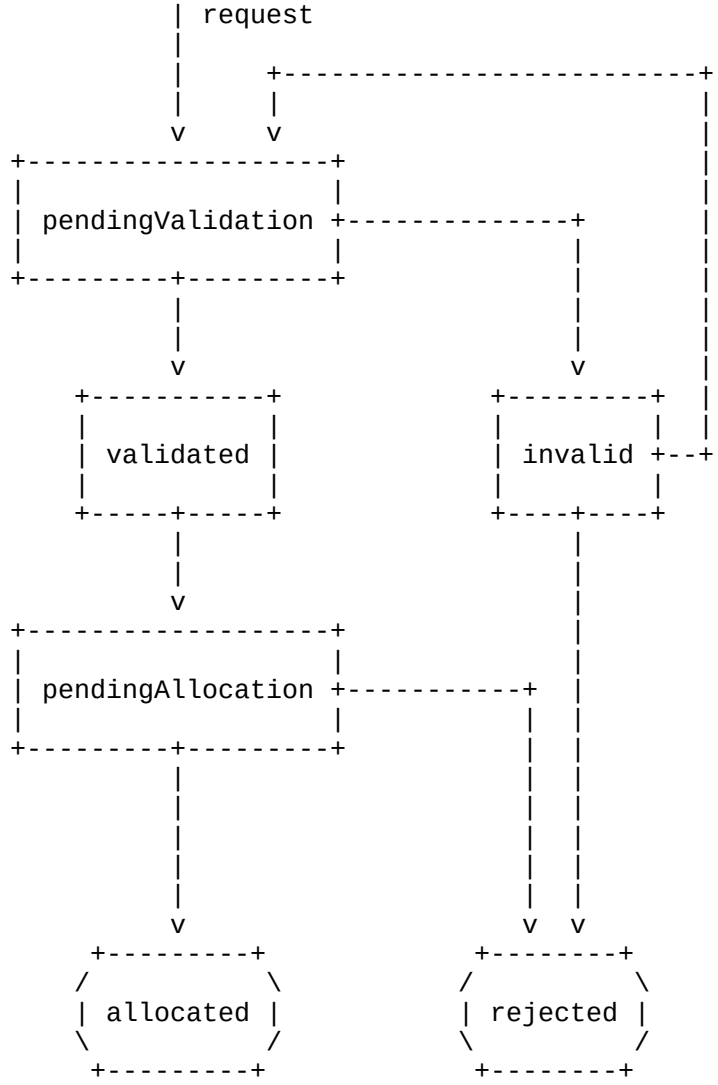

Figure 2

2.5. Poll Messaging

A Launch Application MUST be handled as an EPP domain name object as specified in RFC 5731 [RFC5731], with the "pendingCreate" status and with the launch status values defined in Section 2.4. A Launch Registration MUST be handled as an EPP domain name object as specified in RFC 5731 [RFC5731], with the "pendingCreate" status and 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 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 [RFC7848]. 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 [RFC7848].

2.6.3. Digital Signature

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

2.6.3.1. <smd:signedMark> element

The <smd:signedMark> element contains the digitally signed mark information. The <smd:signedMark> element is defined in [RFC7848]. 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 [RFC7848].

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 [RFC7848]. 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 [RFC7848].

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 according to Section 2.3.

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 ICANN 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/rJ1NrDO92vDsAzf7EQzgjX4R0000000001
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/rJ1NrDO92vDsAzf7EQzgjX4R0000000001
S:        </launch:claimKey>
S:        <launch:claimKey validatorID="custom-tmch">
S:        20140423200/1/2/3/rJ1Nr2vDsAzasdff7EasdfgjX4R000000002
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. The server SHOULD validate the value and return an EPP error result code of 2306 if it is invalid.

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 Trademark 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 ICANN 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/rJ1NrDO92vDsAzf7EQzgjX4R0000000001
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/rJ1NrDO92vDsAzf7EQzgjX4R0000000001
S:        </launch:claimKey>
S:        <launch:claimKey validatorID="custom-tmch">
S:        20140423200/1/2/3/rJ1Nr2vDsAzasdff7EasdfgjX4R000000002
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) can be 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. The server SHOULD validate the value and return an EPP error result code of 2306 if it is invalid.
<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 only if the "includeMark" attribute is "true" in the command.

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, and return an EPP error result code of 2306 if the type is incorrect. The <launch:create> element contains the following child elements:

<launch:phase>: The identifier for the launch phase. The server SHOULD validate the value according to Section 2.3.
<launch:codeMark> or <smd:signedMark> or <smd:encodedSignedMark>:

<launch:codeMark>: Zero or more <launch:codeMark> elements. The <launch:codeMark> child elements are defined in the <launch:codeMark> element (Section 2.6.1) section.
<smd:signedMark>: Zero or more <smd:signedMark> elements. The <smd:signedMark> child elements are defined in the <smd:signedMark> element (Section 2.6.3.1) section.
<smd:encodedSignedMark>: Zero or more <smd:encodedSignedMark> elements. The <smd:encodedSignedMark> child elements are defined in the <smd:encodedSignedMark> element (Section 2.6.3.2) section.

The following is an example <create> domain command using the <launch:create> extension, following the "code" validation model, with multiple sunrise codes:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domain.example</domain:name>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:        <launch:phase>sunrise</launch:phase>
C:        <launch:codeMark>
C:          <launch:code validatorID="sample1">
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: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>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:        <launch:phase>sunrise</launch:phase>
C:        <launch:codeMark>
C:          <launch:code validatorID="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>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:       type="application">
C:        <launch:phase>sunrise</launch:phase>
C:        <smd:signedMark id="signedMark"
C:         xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
C:         ...
C:        </smd:signedMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

The following is an example <create> domain command using the <launch:create> extension, following the "signed mark" validation model, with the base64 encoded signed mark information:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
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:        <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, and return an EPP error result code of 2306 if the type is incorrect. 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 according to Section 2.3.
<launch:notice>: One or more <launch:notice> elements that contain the following child elements:

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

The following is an example <create> domain command using the <launch:create> extension with the <launch:notice> information for the "tmch" and the "custom-tmch" validators, for the "claims" launch phase:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domain.example</domain:name>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain: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, and return an EPP error result code of 2306 if the type is incorrect.

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 according to Section 2.3.

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: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 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 add a <launch:creData> element along to 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:applicationID>
S:      </launch:creData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.4. EPP <update> Command

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

An EPP <update> command with the extension sent to a server that does not support launch applications will fail. 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. The server SHOULD validate the value and return an EPP error result code of 2306 if it is invalid.
<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. The server SHOULD validate the value and return an EPP error result code of 2306 if it is invalid.
<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

```
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 "custom", 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 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>

    <!-- 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: IESG
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: IESG
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 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-ietf-regext-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-regext-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-regext-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-regext-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-regext-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.

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 Harald Alvestrand, Ben Campbell, Spencer Dawkins, Jothan Frakes, Keith Gaughan, Seth Goldman, Scott Hollenbeck, Michael Holloway, Jan Jansen, Rubens Kuhl, Mirja Kuhlewind, Warren Kumari, Ben Levac, Gustavo Lozano, Klaus Malorny, Alexander Mayrhofer, Alexey Melnikov, Patrick Mevzek, James Mitchell, Francisco Obispo, Mike O'Connell, Eric Rescoria, Bernhard Reutner-Fischer, Sabrina Tanamal, Trung Tran, Ulrich Wisser and Sharon Wodjenski.

Some of the description of the Trademark Claims Phase was based on the work done by Gustavo Lozano in the ICANN TMCH functional specifications.

9. References

9.1. 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, <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>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[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>.

[RFC7848] Lozano, G., "Mark and Signed Mark Objects Mapping", RFC 7848, DOI 10.17487/RFC7848, June 2016, <https://www.rfc-editor.org/info/rfc7848>.

[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>.

9.2. Informative References

[I-D.ietf-regext-tmch-func-spec] Lozano, G., "ICANN TMCH functional specifications", draft-ietf-regext-tmch-func-spec-03 (work in progress), July 2017.

[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. 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 "extensed" to "extended" in "Availability Check Form" section.
    8. Broke the mix of create forms in the "EPP <create> Command" section to a fourth "Mixed Create Form" with its own sub-section.
    9. Removed "displayed or" from "displayed or accepted" in the <launch:acceptedDate> description.

10. Replaced "given domain name is supported" with "given domain name are supported" in the "Create Response" section.
11. Changed the reference of 2303 (object does not exist) in the "Security Considerations" section to 2201 (authorization error).
12. Added arrow from "invalid" status to "pendingValidation" status and "pendingAllocation" status to "rejected" status in the State Transition Diagram.

4. Added the "C:" and "S:" example prefixes and related text in the "Conventions Used in This Document" section.

A.11. Change from 10 to 11

1. Moved the claims check response <launch:chkData> element under the <extension> element instead of the <resData> element based on the request from Francisco Obispo.

A.12. Change from 11 to 12

1. Added support for multiple validator identifiers for claims notices and marks based on a request and text provided by Mike O'Connell.
2. Removed domain:exDate element from example in section 3.3.5 based on a request from Seth Goldman on the provreg list.
3. Added clarifying text for clients not passing the launch extension on update and delete commands to servers that do not support launch applications based on a request from Sharon Wodjenski on the provreg list.

A.13. Change from 12 to EPPEXT 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 EPPEXT 00 to EPPEXT 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 EPPEXT 01 to EPPEXT 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 EPPEXT 02 to EPPEXT 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 EPPEXT 03 to EPPEXT 04

1. Amended XML Namespace section of IANA Considerations, added EPP Extension Registry section.

A.18. Change EPPEXT 04 to EPPEXT 05

1. Added a missing comma to the descripton 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 EPPEXT 05 to EPPEXT 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 EPPEXT 06 to EPPEXT 07

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

A.21. Change from EPPEXT 07 to REGEXT 00

1. Changed to regext working group draft by changing draft-ietf-eppext-launchphase to draft-ietf-regext-launchphase and by changing references of draft-ietf-eppext-tmch-func-spec to draft-ietf-regext-tmch-func-spec.

A.22. Change from REGEXT 00 to REGEXT 01

1. Fixed reference of Claims Check Command to Trademark Check Command in the Trademark Check Form section.
2. Replaced reference of draft-ietf-eppext-tmch-smd to RFC 7848.

A.23. Change from REGEXT 01 to REGEXT 02

1. Removed the reference to ietf-regext-tmch-func-spec and briefly described the trademark claims phase that is relavent to draft-ietf-regext-launchphase.

A.24. Change from REGEXT 02 to REGEXT 03

1. Ping update.

A.25. Change from REGEXT 03 to REGEXT 04

1. Updates based on feedback from Scott Hollenbeck that include:

    1. Nit on reference to RFC 7848 in section 1.
    2. Added reference to <domain:create> for the request to create a Launch Application in section 2.1.
    3. Removed the second paragraph of section 2.1 describing the option of creating an application identifier for a Launch Registration.
    4. Provided clarification in section 2.2 on the responsibility of the server to ensure that the supported validator identifiers are unique.
    5. Updated the text in section 2.5 referencing the domain name object in RFC 5731.
    6. Updated the copyright to 2017 in section 4.1.

A.26. Change from REGEXT 04 to REGEXT 05

1. Updates based on feedback from Ulrich Wisser that include:

    1. Updated reference to obsoleted RFC 6982 with RFC 7942.
    2. Moved RFC 7451 reference from normative to informative.

A.27. Change from REGEXT 05 to REGEXT 06

1. Updates based on feedback from Adam Roach that include:

    1. Added an informative reference to draft-ietf-regext-tmch-func-spec in section 2.3.1 "Trademark Claims Phase".
    2. Added formal definition of a Launch Registration and Launch Application to section 1.1.
    3. Updated the description of the Validator Identifier to indicate that the uniqueness is based on server policy.
    4. Updated "Does Domain have Claims?" "No" and "Yes" branch labels in Figure 1.
    5. Updated the description of the <launch:phase> element in the commands to explicitly specify the return of a 2306 EPP error result when invalid or referring to section 2.3 for validation.
    6. Fixed indentation of the <launch:applicationID> and <launch:status> elements in the section 2.5 examples.
    7. Updated the description of the <mark:mark> element in the info response.
    8. Added returning an EPP error result code of 2306 if the "type" attribute is incorrect in section 3.3.1, 3.3.2, and 3.3.3.
    9. Made small change in the description of the Create Response in section 3.3.5.
    10. Updated the Registrant Contact in section 7 to the IESG.

A.28. Change from REGEXT 06 to REGEXT 07

1. Updates based on feedback from Mirja Kuhlewind that include:

    1. In the Security Considerations section, change must to MUST in "Updates to, and deletion of an application object MUST be restricted to clients authorized to perform the said operation on the object".
2. Updates based on feedback from Warren Kumari that include:

    1. Removed the comma from "The Validator Identifier is the identifier, that is unique..." not needed due to change from Harald Alvestrand’s feedback.
3. Updates based on feedback from Alexey Melnikov that include:

    1. Added a Normative Reference to RFC 5646 for the "language" attribute.
    2. Replace identifer with identifier".
    3. Remove "for" in "Enumeration of for launch phase values"
4. Updates based on feedback from Harald Alvestrand that include:

1. Removed the references to the unused "launch-1.0", "signedMark-1.0", and "mark-1.0" abbreviations and revised the XML namespace prefix definitions for "launch", "smd", and "mark".
2. Replace "that is unique to the server" to "unique to the server" in the Validator Identifier section.
3. Replaced ", including the "allocated" and "rejected" statuses" with "("allocated" and "rejected")" in the Status Values section.
4. Replaced "Is a possible end state" with "This is a possible end state" in the definition of the "allocated" and "rejected" statuses in the Status Values section.
5. Add the preamble "The transitions between the states is a matter of server policy. This diagram defines one possible set of permitted transitions." to the State Transition diagram.
6. Split the first sentence of the Poll Messaging section into two sentences, one for the Launch Application and one for the Launch Registration.
7. Remove "either" and replace "or" with an "and" in the first sentence of the Digital Signature section for clarity and to be more consistent with the description of the "signed mark" validation model.

5. Updates based on feedback from Ben Campbell that include:

1. Replacement of "that" with "which" in the first sentence of the Validator Identifier section not needed due change from Harald Alvestrand’s feedback.
2. Avoid using RFC 2119 in the Launch Phases definitions, which resulted in change MAY to may in the definition of the "open" phase and MUST to must in the definition of the "claims" phase.
3. Change "SHOULD" to "should" in the sentence "For example, the <launch:phase> element SHOULD be <launch:phase name="landrush">claims</launch:phase>".
4. Change "MUST" to "must" in the sentence "The Trademark Claims Phase is when a Claims Notice MUST be displayed to a prospective registrant of a domain name that matches trademarks".
5. Change "MAY" to "may" in the sentence "Claim Notice Information Service (CNIS), which MAY be directly linked to a Trademark Validator.", where MAY can be lowercase may".
6. Remove "that" from the sentence "The <launch:codeMark> element that is used by the "code", "mark", and "code with mark" validation models, has the following child elements".
7. Added the consistent use of colons ":" at the end of the hangText labels to address adding whitespace between handing indent list entries.

8. Revised the first sentence, of the second paragraph, of the "EPP <update> Command" section, to read "An EPP <update> command with the extension sent to a server that does not support launch applications will fail.".
9. Revised the "The server SHOULD NOT use the "custom" status value" to "The server SHOULD use one of the non-"custom" status values" in the Status Values section.
10. Revised "Both the Validator Identifier and the Issuer Identifier used MUST be unique" to "Both the Validator Identifier and the Issuer Identifier used MUST be unique in the server" in the Validator Identifier section.
11. Revised "The Validator Identifier MAY define a non-Trademark Validator that supports a form of claims" to "The Validator Identifier may define a non-Trademark Validator that supports a form of claims, where claims and a Validator Identifier can be used for purposes beyond trademarks" in the Validator Identifier section.

6. Updates based on feedback from Eric Rescoria that include:

1. Replaced the duplicate Claims Check Form and Claims Create Form in the list of the two ways the document supports the Trademark Claims Phase, to refer to the section by number instead of by name.
2. Added "The use of "..." is used as shorthand for elements defined outside this document" added to the "In examples,..." paragraph of the Conventions Used in This Document section.
3. Added "When using digital signatures the server MUST validate the digital signature" to the Digital Signature section.
4. Removed "post-launch" to the description of the "open" phase in the Launch Phases section.
5. Add the sentences "Multiple launch phases and multiple models are supported to enable the launch of a domain name registry. What is supported and what is validated is up to server policy. Communication of the server policy is typically performed using an out-of-band mechanism that is not specified in this document." to the second paragraph of the Introduction section.

7. Updates based on feedback from Spencer Dawkins that include:

1. Replace "during their initial launch" with "as they begin operation" in the Introduction section.

8. Updates based on feedback from Sabrina Tanamal that include:

1. Pretty print the XML schema to address inconsistent indenting.

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: June 10, 2017

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

December 7, 2016

Extensible Provisioning Protocol (EPP) Reseller Mapping
draft-ietf-regext-reseller-01

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 June 10, 2017.

Copyright Notice

Table of Contents

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:

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

3.4. Parent Identifier

There can be more than one layer of resellers. The parent identifier, as defined with the <reseller:parentId> element, represents the parent reseller identifier in a child reseller. The parent identifier is not defined for the top level reseller, namely the registrar of the registry. An N-tier reseller has a parent reseller and at least one child reseller. A reseller customer has a parent reseller and no child resellers.

Loops SHOULD be prohibited. If reseller A has B as parent identifier, reseller B must not have reseller A as parent identifier.

3.5. URL

The URL represents the reseller web home page, as defined with the <reseller:url> element.

3.6. Disclosure of Data Elements and Attributes

This document supports the same disclosure features described in Section 2.9 of with the use of the <reseller:disclose> element. [RFC5733].

The <reseller:disclose> element MUST contain at least one of the following child elements:

<reseller:name type="int"/>

<reseller:name type="loc"/>

<reseller:addr type="int"/>

<reseller:addr type="loc"/>

<reseller:voice/>

<reseller:fax/>

<reseller:email/>

<reseller:url/>

<reseller:contact/>

4. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730]. The command mappings described here are specifically for use in provisioning and managing reseller information via EPP.

4.1. EPP Query Commands

EPP provides two commands to retrieve domain information: <check> to determine if a reseller object can be provisioned within a repository, and <info> to retrieve detailed information associated with a reseller object. This document does not define a mapping for the EPP <transfer> command.

4.1.1. EPP <check> Command

The EPP <check> command is used to determine if an object can be provisioned within a repository. It provides a hint that allows a client to anticipate the success or failure of provisioning an object using the <create> command, as object-provisioning requirements are ultimately a matter of server policy.

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

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

Example <check> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <check>
C:      <reseller:check
C:       xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0">
C:        <reseller:id>res1523</reseller:id>
C:        <reseller:id>re1523</reseller:id>
C:        <reseller:id>1523res</reseller:id>
C:      </reseller:check>
C:    </check>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When a <check> command has been processed successfully, the EPP <resData> element MUST contain a child <reseller:chkData> element that identifies the reseller namespace. The <reseller:chkData> element contains one or more <reseller:cd> elements that contain the following child elements:

- A <reseller:id> element that identifies the queried object. This element MUST contain an "avail" attribute whose value indicates object availability (can it be provisioned or not) at the moment the <check> command was completed. A value of "1" or "true" means that the object can be provisioned. A value of "0" or "false" means that the object cannot be provisioned.

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:

- A <reseller:id> element that contains the server-unique identifier of the reseller object to be queried.

Example <info> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <reseller:info
C:       xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0">
C:        <reseller:id>res1523</reseller:id>
C:      </reseller:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

- A <reseller:id> element that contains the server-unique identifier of the reseller object, as defined in Section 3.1.

- A <reseller:roid> element that contains the Repository Object IDentifier assigned to the reseller object when the object was created.

- A <reseller:state> element that contains the operational state of the reseller, as defined in Section 3.3.

- An OPTIONAL <reseller:parentId> element that contains the identifier of the parent object, as defined in Section 3.4.

- One or two <reseller:postalInfo> elements that contain postal-address information. Two elements are provided so that address information can be provided in both internationalized and localized forms; a "type" attribute is used to identify the two forms. If an internationalized form (type="int") is provided, element content MUST be represented in a subset of UTF-8 that can be represented in the 7-bit US-ASCII character set. If a localized form (type="loc") is provided, element content MAY be

represented in unrestricted UTF-8. The <reseller:postalInfo> element contains the following child elements:

* A <reseller:name> element that contains the name of the reseller, which SHOULD be the name of the organization.

* A <reseller:addr> element that contains address information associated with the reseller. A <reseller:addr> element contains the following child elements:

  + One, two, or three OPTIONAL <reseller:street> elements that contain the reseller's street address.

  + A <reseller:city> element that contains the reseller's city.

  + An OPTIONAL <reseller:sp> element that contains the reseller's state or province.

  + An OPTIONAL <reseller:pc> element that contains the reseller's postal code.

  + A <reseller:cc> element that contains the reseller's country code.

o An OPTIONAL <reseller:voice> element that contains the reseller's voice telephone number.

o An OPTIONAL <reseller:fax> element that contains the reseller's facsimile telephone number.

o A <reseller:email> element that contains the reseller's email address.

o A <reseller:url> element that contains the 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.

- A <reseller:crID> element that contains the identifier of the client that created the reseller object.
- A <reseller:crDate> element that contains the date and time of reseller-object creation.
- A <reseller:upID> element that contains the identifier of the client that last updated the reseller object. This element MUST NOT be present if the reseller has never been modified.
- A <reseller:upDate> element that contains the date and time of the most recent reseller-object modification. This element MUST NOT be present if the reseller object has never been modified.
- An OPTIONAL <reseller:disclose> element that identifies elements that require exceptional server-operator handling to allow or restrict disclosure to third parties. See Section 3.6 for a description of the child elements contained within the <reseller:disclose> element.

Example <info> response for the sponsoring client:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <reseller:infData
S:       xmlns:reseller="urn:ietf:params:xml:ns:reseller-1.0">
S:        <reseller:id>res1523</reseller:id>
S:        <reseller:roid>res1523-REP</reseller:roid>
S:        <reseller:state>ok</reseller:state>
S:        <reseller:parentId>1523res</reseller:parentId>
S:        <reseller:postalInfo type="int">
S:          <reseller:name>Example Reseller Inc.</reseller:name>
S:          <reseller:addr>
S:            <reseller:street>123 Example Dr.</reseller:street>
S:            <reseller:street>Suite 100</reseller:street>
S:            <reseller:city>Dulles</reseller:city>
S:            <reseller:sp>VA</reseller:sp>
S:            <reseller:pc>20166-6503</reseller:pc>
S:            <reseller:cc>US</reseller:cc>
S:          </reseller:addr>
S:        </reseller: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:

- A <reseller:id> element that contains the desired server-unique identifier for the reseller to be created, as defined in Section 3.1.
- A <reseller:state> element that contains the operational status of the reseller, as defined in Section 3.3.
- An OPTIONAL <reseller:parentId> element that contains the identifier of the parent object, as defined in Section 3.4.

- 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.
- An OPTIONAL <reseller:voice> element that contains the reseller’s voice telephone number.
- An OPTIONAL <reseller:fax> element that contains the reseller’s facsimile telephone number.
- A <reseller:email> element that contains the reseller’s email address.
- A <reseller:url> element that contains the URL to the website of the reseller.
- 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:

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

Example <create> response:

```
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:

- 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:

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

At least one <reseller:add>, <reseller:rem> or <reseller:rem> element MUST be provided if the command is not being extended. All of these elements MAY be omitted if an <update> extension is present. The <reseller:add> and <reseller:rem> elements contain the following child element:

- Zero or more <reseller:contact> elements that contain the identifiers for contact objects to be associated with or removed from the reseller object. Contact object identifiers MUST be known to the server before the contact object can be associated with the reseller object.

A <reseller:chg> element contains the following OPTIONAL child elements. At least one child element MUST be present:

- A <reseller:state> element that contains the operational status of the reseller.

- A <reseller:parentId> element that contains the identifier of the parent object.

- 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.

- An <reseller:voice> element that contains the reseller’s voice telephone number.

- An <reseller:fax> element that contains the reseller’s facsimile telephone number.

- A <reseller:email> element that contains the reseller’s email address.

- 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
```

```
      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"/>
```

```
          <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 assignment the following URI.

Registration request for the reseller namespace:

- URI: urn:ietf:params:xml:ns:reseller-1.0
- Registrant Contact: See the "Author’s Address" section of this document.
- XML: See the "Formal Syntax" section of this document.

7.2. EPP Extension Registry

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

Name of Extension: Domain Reseller Object Extension

Document status: Standards Track

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

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

TLDs: any

IPR Disclosure: none

Status: active

Notes: none

8. Security Considerations

Authorization information described in [RFC5733] is not supported in this document. If the querying client is not the sponsoring registrar of the reseller, not all the object information is accessible. The disclose element defined in [RFC5733] is used to allow or restrict disclosure of object elements to third parties. Other mechanism, such as defining a registry customized authorization information list according to their local policies and regulations, is also possible.

9. 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 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>.

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.

WG document-00: WG document submitted

WG document-01: Keep document alive for further discussion. Reseller object or entity object with multiple roles?

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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: June 10, 2017

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

December 7, 2016

Reseller Extension for the Extensible Provisioning Protocol (EPP)
draft-ietf-regext-reseller-ext-01

Abstract

This mapping, an extension to EPP object mappings like the EPP domain name mapping [RFC5731], to support assigning a reseller to any existing object (domain, host, contact) as well as any future objects. Specified in Extensible Markup Language (XML), this extended mapping is applied to provide additional features required for the provisioning of 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 June 10, 2017.

Copyright Notice

Table of Contents

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-ietf-regext-reseller] 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-ietf-regext-reseller]. 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-ietf-regext-reseller].

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:

- 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>jd1234</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:

- 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="ddddd-dddd">fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <resellerext:create xmlns:resellerext="urn:ietf:params:xml:ns:resellerex
t-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:

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

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

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

Example <update> command, adding a reseller:

```
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:resellerex
t-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:resellerex
t-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:resellerex
t-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 assignment the following URI.

Registration request for the reseller namespace:

- URI: urn:ietf:params:xml:ns:reseller-1.0
- Registrant Contact: See the "Author’s Address" section of this document.
- XML: See the "Formal Syntax" section of this document.

7.2. EPP Extension Registry

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

Name of Extension: Domain Reseller Extension

Document status: Standards Track

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

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

TLDs: any

IPR Disclosure: none

Status: active

Notes: none

8. Security Considerations

The object mapping extension described in this document does not provide any other security services or introduce any additional considerations beyond those described by [RFC5730], [RFC5731], [RFC5732] and [RFC5733] or those caused by the protocol layers used by EPP.

9. 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 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>.

10.2. Informative References

[ID.draft-ietf-regext-reseller] Zhou, L., Kong, N., Zhou, G., Lee, X., and J. Gould, "Extensible Provisioning Protocol (EPP) Reseller Mapping", Jun 2016, <http://tools.ietf.org/html/draft-ietf-regext-reseller>.

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".

WG document-00: WG document submitted

WG document-01: Keep document alive for further discussion. The requirement of reseller information is clear for both registrar and registry. What we should reach a consensus is whether the extension should support only a name or ID and name.

Authors' Addresses

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

Phone: +86 10 5881 2677
Email: zhoulinlin@cnnic.cn

Ning Kong
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 3147
Email: nkong@cnnic.cn

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: Informational
Expires: May 23, 2021

G. Lozano
ICANN

Nov 19, 2020

ICANN TMCH functional specifications
draft-ietf-regext-tmch-func-spec-09

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 https://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 23, 2021.

Copyright Notice

Table of Contents

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:

- 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.

- 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", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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.

Extensible Markup Language (XML) 1.0 as described in [W3C.REC-xml-20081126] and XML Schema notation as described in [W3C.REC-xmlschema-1-20041028] and [W3C.REC-xmlschema-2-20041028] are used in this specification.

3. Glossary

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

- 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.

- CA: Certificate Authority, see [RFC5280].

- CNIS, Claims Notice Information Service: This service provides Trademark Claims Notices (TCN) to Registrars.

- 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.

- CRL: Certificate Revocation List, see [RFC5280].

- CSV: Comma-Separated Values, see [RFC4180]

- Date and time, datetime: Date and time are specified following the standard "Date and Time on the Internet specification", see [RFC3339].

- DN, Domain Name, domain name: see definition of Domain name in [RFC8499].

- 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.

- DNL, Domain Name Label, the DNL is an A-label or NR-LDH label (see [RFC5890]).

- DNL List: A list of DNLs that are covered by a PRM.

- DNS: Domain Name System, see [RFC8499].

- 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.

- EPP: The Extensible Provisioning Protocol, see definition of EPP in [RFC8499].
- FQDN: Fully-Qualified Domain Name, see definition of FQDN in [RFC8499].
- HTTP: Hypertext Transfer Protocol, see [RFC7230] and [RFC7231].
- HTTPS: HTTP over TLS (Transport Layer Security), [RFC2818].
- IDN: Internationalized Domain Name, see definition of IDN in [RFC8499].
- 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.
- 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).
- Matching Rules: Some trademarks entitled to inclusion in the TMDB include characters that are impermissible in the domain name system (DNS) as a DNL. The TMV changes (using the ICANN TMCH Matching Rules [MatchingRules]) certain DNS-impermissible characters in a trademark into DNS-permissible equivalent characters
- NORDN, Notification of Registered Domain Names: The process by which Registries upload their recent LORDN to the TMDB.
- PGP: Pretty Good Privacy, see [RFC4880]
- PKI: Public Key Infrastructure, see [RFC5280].
- PRM, Pre-registered mark: Mark that has been pre-registered with the ICANN TMCH.
- 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.
- Registrant: see definition of Registrant in [RFC8499].

- Registrar, Domain Name Registrar: see definition of Registrar in [RFC8499].

- Registry, Domain Name Registry, Registry Operator: see definition of Registry in [RFC8499]. A Registry Operator is the contracting party with ICANN for the TLD.

- 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 [RFC7848]. An SMD generated by an ICANN-approved trademark validator (TMV) contains both the signed token and the TMV's PKIX certificate.

- 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.

- 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.

- SRS: Shared Registration System, see also [ICANN-GTLD-AGB-20120604].

- SURL, Sunrise List: The list of DNLs that are covered by a PRM and eligible for Sunrise.

- 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.

- TLD: Top-Level Domain Name, see definition of TLD in [RFC8499].

- 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.

- 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.

- 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.

- TMH, Trademark Holder: The person or organization owning rights on a mark.

- 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.

- 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.

- Trademark Claims, Claims: Provides information to enhance the understanding of the Trademark rights being claimed by the TMH.

- 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.

- 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>.

- 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.

- 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

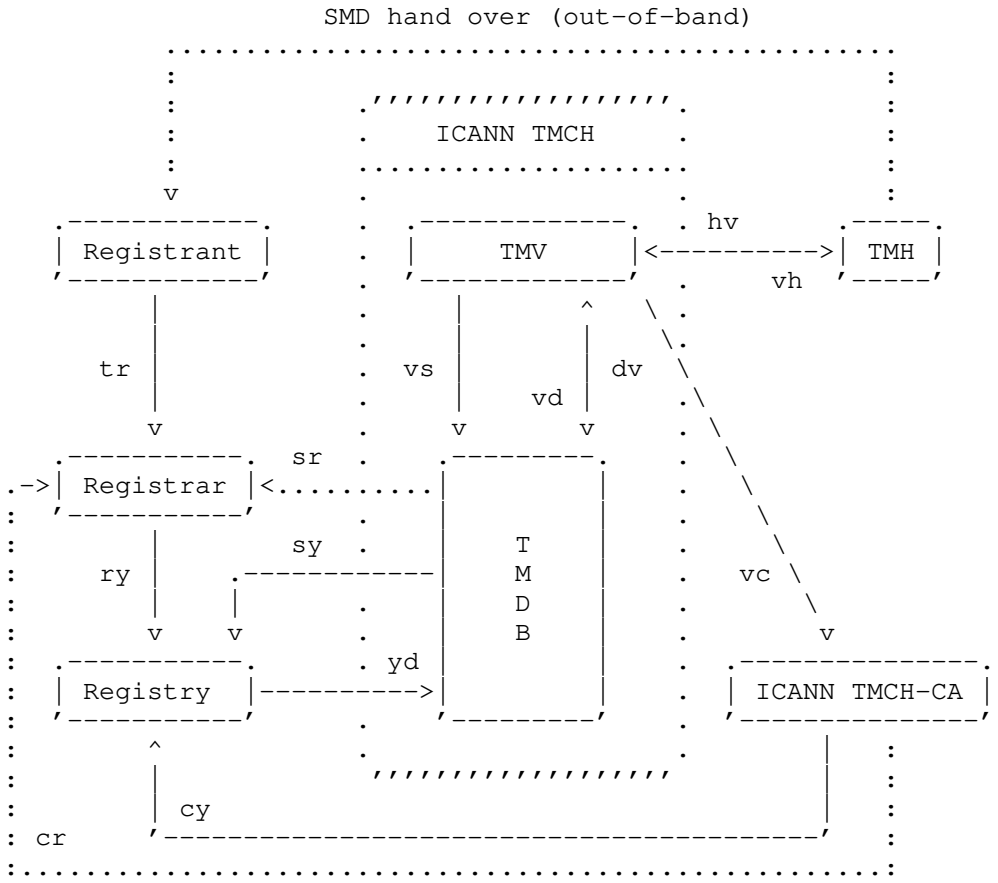

Figure 1

4.2. Trademark Claims Period

Architecture of the Trademark Claims Period

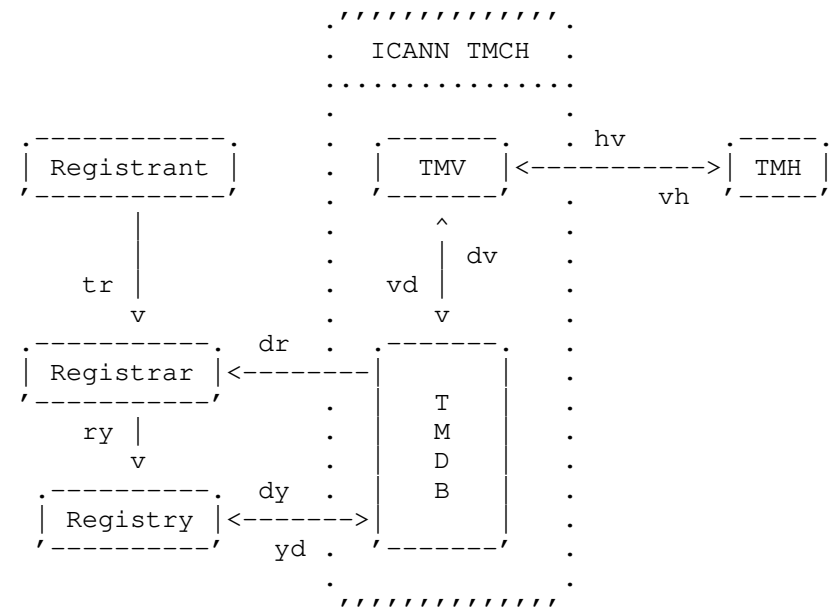

Figure 2

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 DNs 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:

- During the Sunrise Period to fetch the SMD Revocation List from the TMBD via the sy interface (Section 4.3.11).
- During the Trademark Claims Period to fetch the DNL List from the TMDB via the dy interface (Section 4.3.3).
- 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:

- Fetch the SMD Revocation List via the sy interface (Section 4.3.11).
- Fetch the DNL List from the TMDB via the dy interface (Section 4.3.3).
- 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 (see [RFC7235]). 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:

- 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:

- 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).
- 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:

- During the Sunrise Period to (optionally) fetch the SMD Revocation List from the TMDB via the sr interface (Section 4.3.12).
- 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:

- Fetch the SMD Revocation List via the sr interface (Section 4.3.12).
- 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:

- 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:

- 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

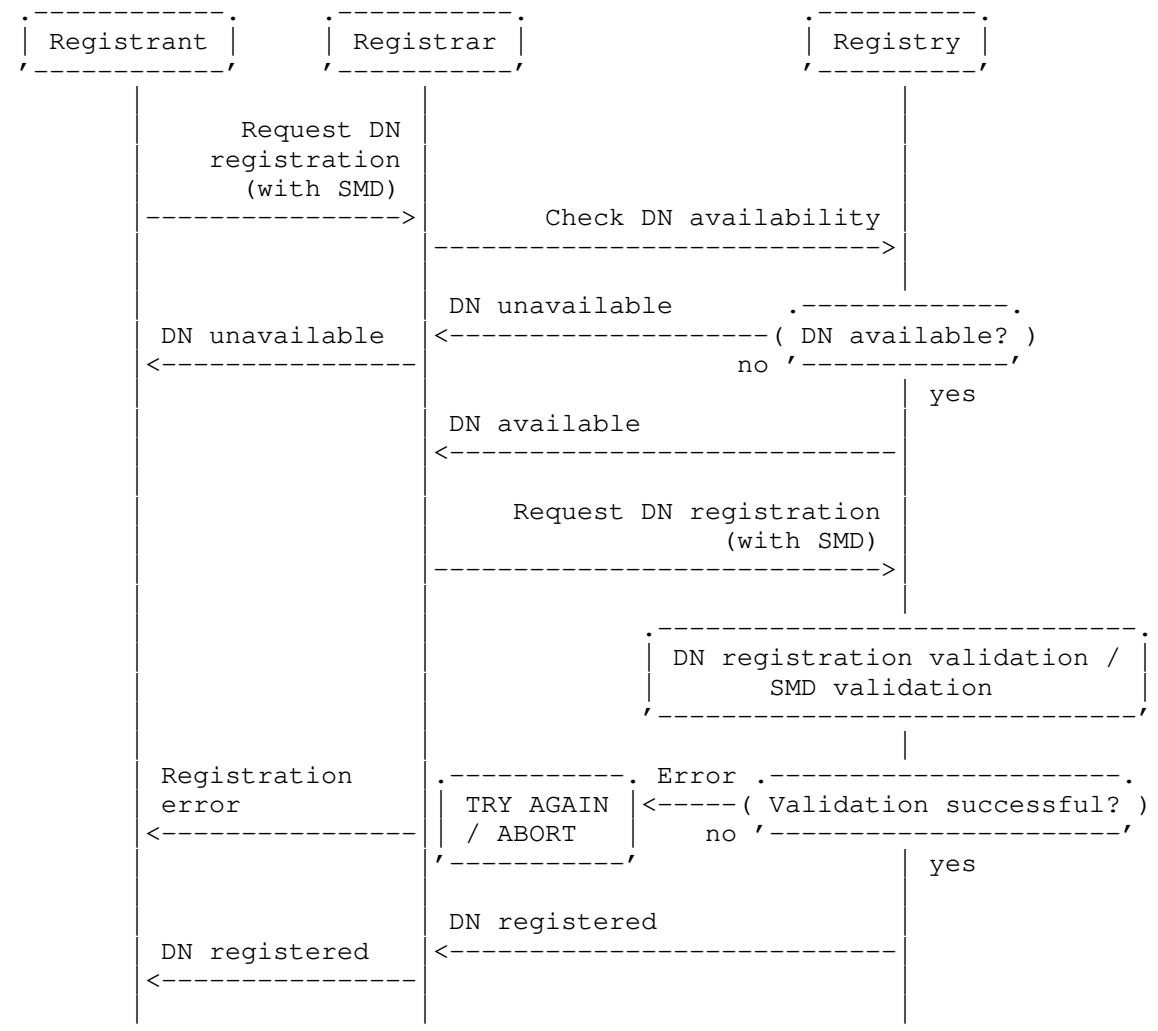

Figure 3

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 of the DN being effectively allocated matches one of the labels (<mark:label>) elements in the SMD. For example, if the DN "xn--mgbachtv.xn--mgbh0fb" is being effectively allocated, the leftmost DNL would be "xn--mgbachtv".

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

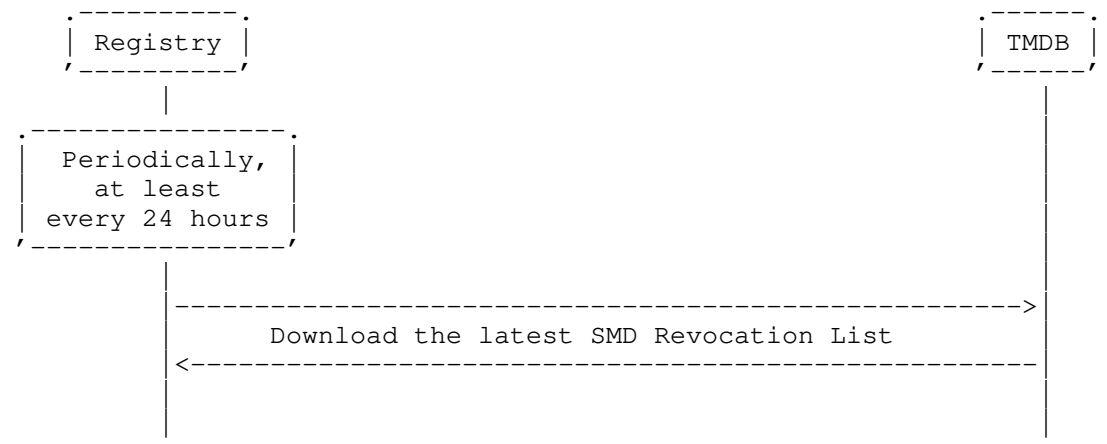

Figure 4

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 DNs effectively allocated during the Sunrise or Trademark Claims Period (same applies to DNs 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

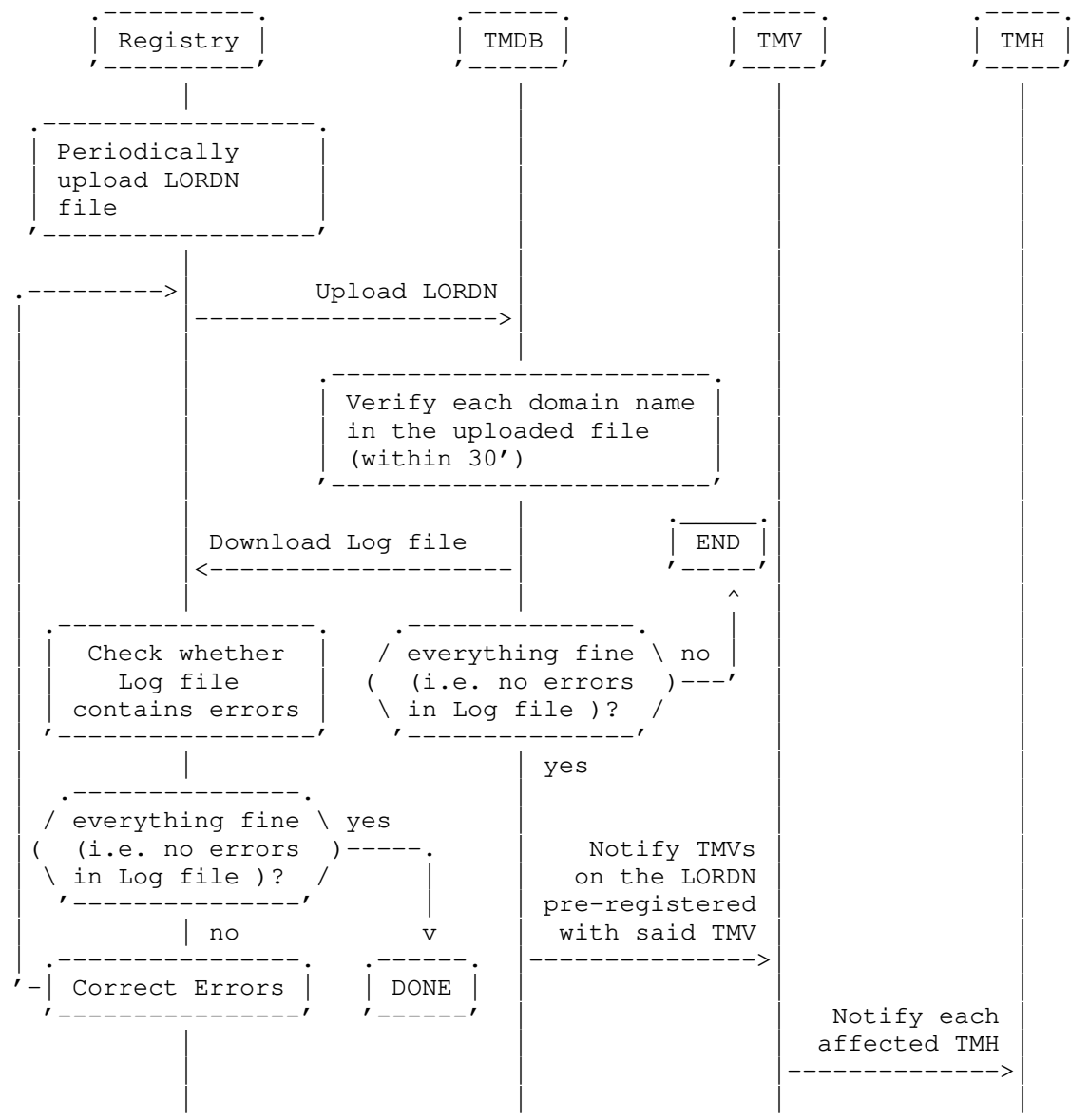

Figure 6

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

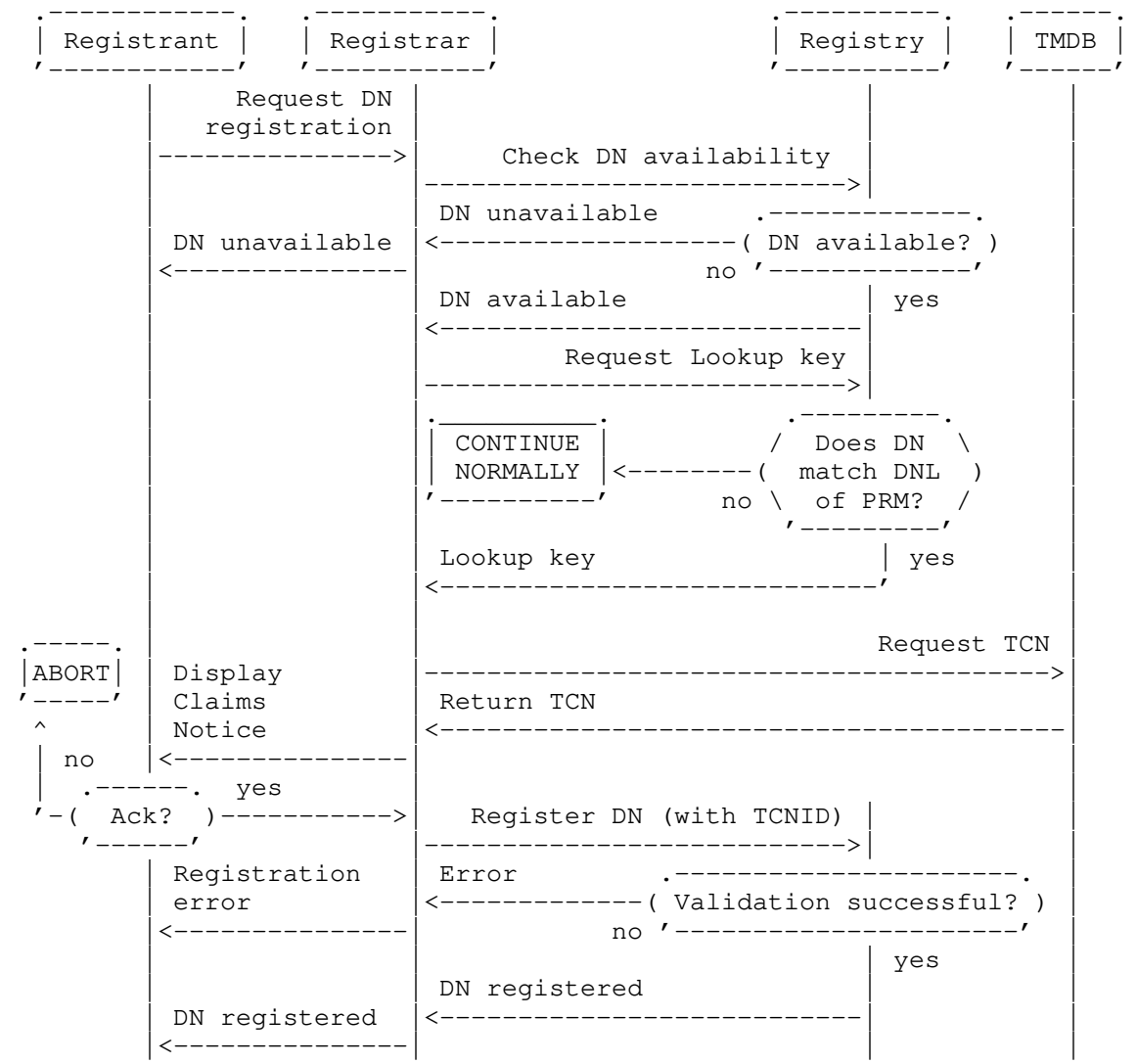

Figure 7

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:

- 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.

- Registries MUST provide the Lookup Key only when queried about a specific DN.

- 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.

- 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.

- 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 within the window of time defined by ICANN policy. In the gTLD round of 2012, Registrars verified that the acceptance datetime was less than or equal to 48 hours in the past, as there were no defined ICANN policies at that time. Implementers should be aware that ICANN policy may define this value in the future.

4. Using the leftmost DNL of the DN being effectively allocated, 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. For example, if the DN "xn--mgbachtv.xn--mgbh0fb" is being effectively allocated, the leftmost DNL would be "xn--mgbachtv".

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. TMBD 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

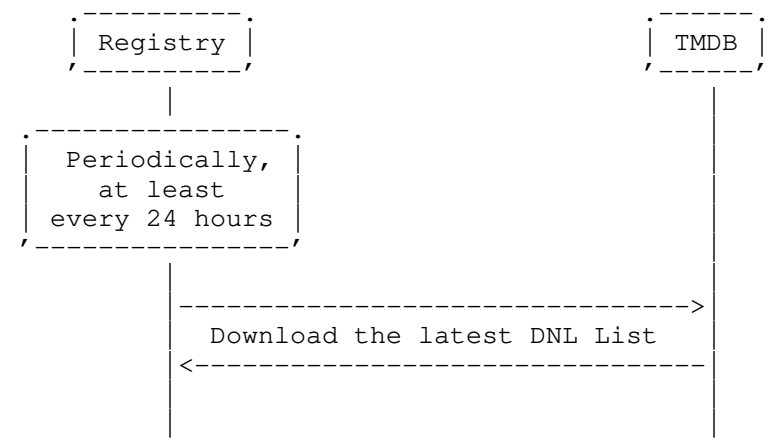

Figure 8

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 of the DN being effectively allocated matches the label (<tmNotice:label>) element in the TCN. For example, if the DN "xn--mgbachtv.xn--mgbh0fb" is being effectively allocated, the leftmost DNL would be "xn--mgbachtv".

    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.

Currently TCNs are generated twice a day by the TMDB. The expiration date (<tmNotice:notAfter>) of each TCN MUST be set to a value defined by ICANN policy. In the gTLD round of 2012, the TMDB set the expiration value to 48 hours in to the future as there were no defined ICANN policies at that time. Implementers should be aware that ICANN policy may define this value in the future.

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. TMBD 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:

- If the leftmost DNL 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. For example, if the DN "xn--mgbachtv.xn--mgbh0fb" is being effectively allocated, the leftmost DNL would be "xn--mgbachtv".

- 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.

- 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.

- 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:

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

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

- SMD Revocation List (see Section 5.2.3.1)

o CNIS (see Section 5.3.5.1)

5.4.2. TMBD 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

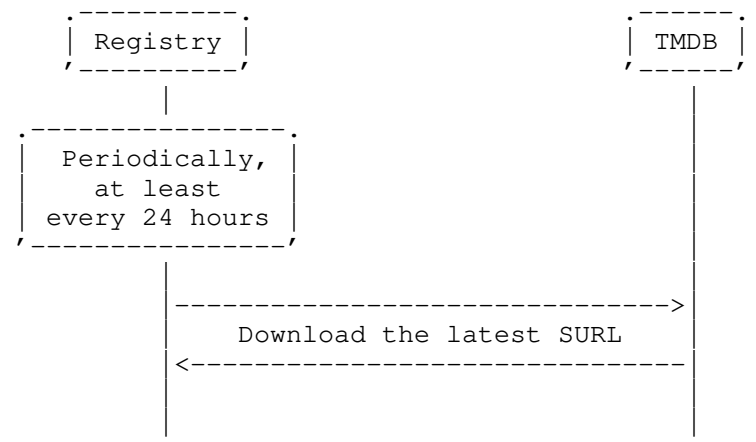

Figure 9

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:

- 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.

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

  With the header names as follows:

  DNL,lookup-key,insertion-datetime

- 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 hex character. This is the first, second and third hex character 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/rJ1NrDO92vDsAzf7EQzgjX4R0000000001,\
   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:

- 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.

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

  With the header names as follows:

  smd-id,insertion-datetime

- 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:

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

- < 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:

- Sunrise LORDN file:

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

- 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:

- Sunrise LORDN file (during QLP Period):

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

- 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:

- 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
  ```

- 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.

- The interface provides a HTTP/401 status code if the credentials provided does not authorize the Registry Operator to upload a LORDN file.

- 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.

- 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:

- 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 form 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

- 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 form 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:

- Sunrise LORDN Log file:

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

- 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:

- 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.

- 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.

- The interface provides a HTTP/400 status code if the request is incorrect.

- The interface provides a HTTP/401 status code if the credentials provided does not authorize the Registry Operator to download the LORDN Log file.

- The interface provides a HTTP/404 status code if the LORDN Transaction Identifier is incorrect.

- 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 0000000000000000001 and TLD "example" the URI would be:

< https://<tmdb-domain-name>/LORDN/example/sunrise/0000000000000000001/result >

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

- 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 identified 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".

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

  With the header names as follows:

roid,result-code

- 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:

- ok: Success, DN Line accepted by the TMDB.

- warn: a warning is issued, DN Line accepted by the TMDB.

- 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 the 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 QLP LORDN file out 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 U-label or NR-LDH labels (see [RFC5890])>
4. notBefore: <begin validity>
5. notAfter: <end validity>
6. -----BEGIN ENCODED SMD-----
7. <encoded SMD (see [RFC7848])>
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:

- 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 0000000000000000001 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 of the DN being effectively allocated, 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, if the DN "xn--mgbachtv.xn--mgbh0fb" is being effectively allocated, the leftmost DNL would be "xn--mgbachtv".

Example of computation of the TCN Checksum:

CRC32(example-one128194920092233720368547758 07)=370d0b7c

- A <tmNotice:notBefore> element that contains the start of the validity date and time of the TCN.
- A <tmNotice:notAfter> element that contains the expiration date and time of the TCN.
- A <tmNotice:label> element that contains the DNL covered by a PRM.
- 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: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">USA</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.
    </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.
    </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.
     </tmNotice:goodsAndServices>
     <tmNotice:notExactMatch>
       <tmNotice:court>
         <tmNotice:refNum>234235</tmNotice:refNum>
         <tmNotice:cc>CR</tmNotice:cc>
         <tmNotice:courtName>Supreme Court of Spain</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.
     </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:

- 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.

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

  With the header names as follows:

  DNL,insertion-datetime

- 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:

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

```
<CODE BEGINS>
<?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:e164Type" minOccurs="0"/>
       <element name="fax" type="mark:e164Type" 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"/>
      <element name="email" type="mark:minTokenType"/>
    </sequence>
    <attribute name="type" type="mark:contactTypeType"/>
  </complexType>
  <simpleType name="idType">
    <restriction base="token">
      <pattern value="[a-fA-F0-9]{8}\d{1,19}"/>
    </restriction>
```

```
   </simpleType>
 </schema>
 <CODE ENDS>
```

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 feedbak 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. Change History

[[RFC Editor: Please remove this section.]]

9.1. Version 04

1. Ping update.

9.2. Version 05

1. Ping update.

9.3. Version 06

1. Updated the terminology text to reflect the text in RFC8174.

2. Updated the reference of RFC7719 to RFC8499.

3. Updated the matching rules document reference to link to the latest version.

9.4. Version 07

1. Changes based on the feedback provided here: https://mailarchive.ietf.org/arch/msg/regext/xcZPOAajlUJzgPgZBuqlIWRcFZg/

2. Changes based on the feedback provided here: https://mailarchive.ietf.org/arch/msg/regext/MdOhSomd6_djLcthfw5mxWZkbWY

9.5. Version 08

1. Fixed issues detected by idnits tool.

9.6. Version 09

1. Ping update.

10. 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 Trademark Claims Notice namespace:

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

Registrant Contact: IETF <regext@ietf.org>

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

Registration request for the Trademark Claims Notice XML schema:

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

Registrant Contact: IETF <regext@ietf.org>

XML: See Section 7.1 of this document.

11. 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.

12. References

12.1. Normative 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>.

[MatchingRules]
ICANN, "Memorandum on Implementing Matching Rules", July 2016, <https://newgtlds.icann.org/en/about/trademark-clearinghouse/matching-rules-14jul16-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>.

[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>.

[RFC7848] Lozano, G., "Mark and Signed Mark Objects Mapping", RFC 7848, DOI 10.17487/RFC7848, June 2016, <https://www.rfc-editor.org/info/rfc7848>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RPM-Requirements]
ICANN, "Rights Protection Mechanism Requirements", September 2013, <https://newgtlds.icann.org/en/about/trademark-clearinghouse/rpm-requirements-30sep13-en.pdf>.

[W3C.REC-xml-20081126]
Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition) REC-xml-20081126", November 2008, <https://www.w3.org/TR/2008/REC-xml-20081126/>.

[W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition REC-xmlschema-1-20041028", October 2004, <https://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 REC-xmlschema-2-20041028", October 2004, <https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>.

12.2. Informative References

[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>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc4180>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc5890>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <https://www.rfc-editor.org/info/rfc7235>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

[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: June 20, 2019

G. Lozano
ICANN

Dec 17, 2018

Nameserver objects sharing the same name, support for the Registration Data Access Protocol (RDAP)
draft-lozano-rdap-nameservers-sharing-name-03

Abstract

This document describes a Registration Data Access Protocol (RDAP) extension that may be used to retrieve the registration information of a particular nameserver object sharing the name with other nameserver objects.

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 https://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 June 20, 2019.

Copyright Notice

Table of Contents

1. Introduction

The RDAP protocol described in RFCs 7480-7484 supports nameserver object lookup based on the name of the nameserver (see section 3.1.4 of [RFC7482]), therefore it may not be possible to retrieve the registration information of a particular nameserver object sharing the name with other nameserver objects.

This document describes a Registration Data Access Protocol (RDAP) extension that may be used to retrieve the registration information of a particular nameserver object sharing the name with other nameserver objects.

This specification is intended to be used by Domain Name Registries (DNRs) that support the coexistence of multiple external hosts (see [RFC5732]) sharing the same name in the repository.

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].

3. RDAP Conformance object

A server that conforms to this specification MUST include the string literal "rdap_nameservers_sharing_name" in the "rdapConformance" member of the topmost JSON object of all responses provided by the server.

An example of a rdapConformance data structure including this extension:

```
"rdapConformance" :
[
 "rdap_level_0",
 "rdap_nameservers_sharing_name"
]
```

4. Signaling the existence of nameservers sharing the same name

A server that conforms to this specification MUST include a "links" member in a nameserver object, if a different nameserver object in the central repository shares the same name.

The "links" array MUST include a "rel" of "collection", "type" of "application/rdap+json", and a "href" pointing to a nameserver search method by nameserver name.

The following is an example of a nameserver object that includes a links member in order to signal the existence of other nameserver objects sharing the same name.

```
{
 "objectClassName" : "nameserver",
 "handle" : "ROID123",
 "ldhName" : "ns1.foo.test",
 "status" : [ "active" ],
 "ipAddresses" :
 {
  "v4": [ "192.0.2.1", "192.0.2.2" ],
  "v6": [ "2001:db8::123" ]
 },
 "links" :
 [
  {
   "value" : "https://example.com/rdap/nameserver/ns1.foo.test",
   "rel" : "collection",
   "href" : "https://example.com/rdap/nameservers?name=ns1.foo.test",
   "type" : "application/rdap+json"
  }
 ],
 "events" :
 [
  {
   "eventAction" : "registration",
   "eventDate" : "1990-12-31T23:59:59Z"
  },
  {
   "eventAction" : "last changed",
   "eventDate" : "1991-12-31T23:59:59Z",
   "eventActor" : "joe@example.com"
  }
 ]
}
```

5. Nameserver search by nameserver name

An RDAP service that conforms to this specification MUST support nameserver search by nameserver name as described in section 3.2.2. of [RFC7482].

The following is an elided example of a response to a /nameservers?name search for a nameserver sharing the name with another nameserver object in the central repository.

```
{
 "rdapConformance" :
 [
  "rdap_level_0",
  "rdap_nameservers_sharing_name"
 ],
 ...
 "nameserverSearchResults" :
 [
  {
   "objectClassName" : "nameserver",
   "handle" : "ROID123",
   "ldhName" : "ns1.foo.test",
   "entities" :
   [
    {
     "objectClassName" : "entity",
     "handle" : "Rr1",
     "roles" : [ "registrar" ],
     ...
    },
    ...
   ],
   ...
  },
  {
   "objectClassName" : "nameserver",
   "handle" : "ROID321",
   "ldhName" : "ns1.foo.test",
   "entities" :
   [
    {
     "objectClassName" : "entity",
     "handle" : "Rr2",
     "roles" : [ "registrar" ],
     ...
    },
    ...
   ],
   ...
  }
 ]
}
```

6. Nameserver-by-handle path segment specification

A server that conforms to this specification MUST support lookup queries of nameserver objects by the handle of the nameserver using the custom path "nameserver_handle". The custom path "nameserver_handle" adhere to the extensibility mechanism described in Section 5 of [RFC7482].

The appropriated structure for a response to a "nameserver_handle" lookup query is the same as the structure used for a response to a nameserver lookup query defined in section 3.1.4 of [RFC7482].

Syntax: nameserver_handle/<handle>

The <handle> parameter represents a nameserver identifier whose syntax is specific to the registration provider.

The following URL would be used to find information for the nameserver associated with handle ROID123:

https://example.com/rdap/nameserver_handle/ROID123

7. Acknowledgements

TBD.

8. Change History

[[RFC Editor: Please remove this section.]]

8.1. Version 02

1. Ping update.

8.2. Version 03

1. Ping update.

9. IANA Considerations

The following values have been registered in the IANA RDAP Extensions registry:

Extension identifier: rdap_nameservers_sharing_name
Registry operator: N/A
Specification: draft-lozano-rdap-nameservers-sharing-name
Contact: See Author's Address section in the specification

Intended Usage: This document describes a Registration Data Access Protocol (RDAP) extension that may be used to retrieve the registration information of a particular nameserver object sharing the name with other nameserver objects.

10. Security Considerations

The RDAP extension described in this document do not provide any security services beyond those described by RDAP (see RFCs 7480-7484), and protocol layers used by RDAP. The security considerations described in these other specifications apply to this specification as well.

11. References

11.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7482] Newton, A. and S. Hollenbeck, "Registration Data Access Protocol (RDAP) Query Format", RFC 7482, DOI 10.17487/RFC7482, March 2015, <https://www.rfc-editor.org/info/rfc7482>.

11.2. Informative References

[RFC5732] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Host Mapping", STD 69, RFC 5732, DOI 10.17487/RFC5732, August 2009, <https://www.rfc-editor.org/info/rfc5732>.

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: January 8, 2017

G. Lozano
ICANN

July 07, 2016

Transformation of Contact Information Extension Mapping for the Extensible Provisioning Protocol (EPP) draft-lozano-regext-epp-transf-contact-inf-00

Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension mapping for the provisioning and management of transformation (i.e. translation and transliteration) of contact information for objects stored in a shared central repository. Specified in XML, this mapping extends the EPP contact mapping.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 8, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) described in [RFC5730]. This mapping, an extension of the contact mapping described in [RFC5733], is specified using the Extensible Markup Language (XML) 1.0 [W3C.REC-xml] and XML Schema notation ([W3C.REC-xmlschema-1] [W3C.REC-xmlschema-2]).

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.

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

Generic TLDs (gTLDs) Domain Name Registries and Registrars operating according to an agreement with the Internet Corporation for Assigned Names and Numbers (ICANN) are required to follow ICANN Consensus Policies.

On 28/September/2015, the ICANN Board adopted the GNSO Council Policy Recommendations concerning the translation and transliteration of

contact information as presented in the "Final Report on the Translation and Transliteration of Contact Information Policy Development Process" (T&T Report, see [TTOCI]).

The T&T Report recommends:

- Whilst noting that a Whois replacement system should be capable of receiving input in the form of non-ASCII script contact information, the Working Group recommends its data fields be stored and displayed in a way that allows for easy identification of what the different data entries represent and what language(s)/script(s) have been used by the registered name holder

This specification is intended to be used by gTLD Domain Name Registries and Registrars in order to support this recommendation.

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].

In examples, "C:" represents lines sent by a protocol client, and "S:" represents lines returned by a protocol server. "////" is used to note element values that have been shortened to better fit page boundaries. Indentation and white space in examples is provided only to illustrate element relationships and is not a mandatory feature of this protocol.

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.

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

This extension adds additional elements to the EPP contact mapping [RFC5733]. Only those new elements are described here.

Postal-address information in [RFC5733] can be provided in both internationalized and localized forms; a "type" attribute is used to identify the two forms. The use of this extension requires that:

- If two <contact:postalInfo> elements are defined for a contact object, one <contact:postalInfo> is the authoritative information from which other transformations (i.e. translations or transliterations) are derived, and the other <contact:postalInfo> is a transformation.

- One and only one <contact:postalInfo> element MUST be defined as authoritative.

2.1. Common Elements

The following common elements are used in this extension:

- <ird:country>, contains the name of the country in the <addr> element. A "lang" attribute contains the language identifier. Language identifiers MUST be structured as documented in [RFC5646].

- <ird:transliterationStd>, contains the transliteration standard used for the transformation. This element MUST only be present if the transformation is a transliteration.

2.2. Postal Information Objects

The following postal information objects are used in this extension:

2.2.1. Contact Postal Information

The <ird:contactPostalInfo> is used to specify the Internationalized Registration Data (IRD) information of the <contact:postalInfo>. A required "type" attribute is used to specify the relation to a <contact:postalInfo> element. Possible values for the "type" attribute are: localized ("loc") or internationalized ("int"). A required "infSource" attribute is used to specify the source (i.e. "registry", "registrar", "reseller", or "registrant") of the <contact:postalInfo> information. A required "authOrTransMechanism" attribute is used to define which of the two types ("loc" or "int") of <contact:postalInfo> is the authoritative information from which the transformations are derived. Alternatively, if the <contact:postalInfo> is not authoritative, then it has been transformed (i.e. "translation" or "transliteration").

The <ird:contactPostalInfo> object contains the following child elements:

- An <ird:nameLang> element contains the language identifier of a <contact:name> element.

- An <ird:addrLang> element contains the language identifier of a <contact:addr> element.

- An OPTIONAL <ird:orgLang> element contains the language identifier of a <contact:org> element.

- An <ird:country> element, see definition in (see Section 2.1).

- An OPTIONAL <ird:transliterationStd> element, see definition in (see Section 2.1).

2.2.2. Additional Postal Information

The <ird:additionalPostalInfo> is used to specify additional transformations of the authoritative <contact:postalInfo> element. A required "infSource" attribute is used to specify the source (i.e. "registry", "registrar", "reseller", or "registrant") of the additional transformation. An required "conversionMechanism" attribute is used to define the transformation mechanism (i.e. "transliteration" or "translation") of the authoritative <contact:postalInfo> information.

The <ird:additionalPostalInfo> object contains the following child elements:

- An <ird:id> element that contains the server-unique identifier of the <ird:additionalPostalInfo>. When adding a new <ird:additionalPostalInfo>, the <ird:id> element contains the desired server-unique identifier.

- A <ird:name> element that contains the transformation of <contact:name>. A "lang" attribute contains the language identifier.

- An OPTIONAL <ird:org> element that contains the transformation of <contact:org>. A "lang" attribute contains the language identifier.

- An <ird:addr> element that contains the transformation of <contact:addr>. A "lang" attribute contains the language identifier. A <ird:addr> contains the following child elements:

  * One, two, or three OPTIONAL <ird:street> elements that contain the transformation of <contact:street>.

  * A <ird:city> element that contains the transformation of <contact:city>.

    * An OPTIONAL <ird:sp> element that contains the transformation of <contact:sp>

- An <ird:country> element, see definition in (see Section 2.1).

- An OPTIONAL <ird:transliterationStd> element, see definition in (see Section 2.1).

- An OPTIONAL <ird:disclose> is used by the client to identify elements that require exceptional server-operator handling to allow or restrict disclosure to third parties. The <ird:disclose> element MUST contain a "flag" attribute. The "flag" attribute contains an XML Schema boolean value. A value of "true" or "1" (one) notes a client preference to allow disclosure of the specified elements as an exception to the stated data-collection policy. A value of "false" or "0" (zero) notes a client preference to not allow disclosure of the specified elements as an exception to the stated data-collection policy. The <ird:disclose> element MUST contain at least one of the following child elements:

      <ird:name />

      <ird:org />

      <ird:addr />

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 provisioning and managing of transformation of contact information via EPP.

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 EPP contact mapping [RFC5733].

3.1.2. EPP <info> command

This extension does not add any elements to the EPP <info> command described in the EPP contact mapping [RFC5733]. However, additional elements are defined for the <info> response.

When an <info> command has been processed successfully the EPP <extension> element MUST contain a child <ird:infData> element. The <ird:infData> element contains the following child elements:

One or two <ird:contactPostalInfo> elements.

Zero or more OPTIONAL <ird:additionalPostalInfo> elements.

Example <info> 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:      <contact:infData
S:        xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
S:        <contact:id>sh8013</contact:id>
S:        <contact:roid>SH8013-REP</contact:roid>
S:        <contact:status s="linked"/>
S:        <contact:status s="clientDeleteProhibited"/>
S:        <contact:postalInfo type="loc">
S:          <contact:name>&#1048;&#1074;&#1072;&#1085; &#1055;&#1077;&#1090;&#10
88;&#1086;&#1074;&#1080;&#1095; &#1057;&#1080;&#1076;&#1086;&#1088;&#1086;&#1074
;</contact:name>
S:          <contact:addr>
S:            <contact:street>8343 &#1044;&#1088;&#1072;&#1075;&#1072;&#1090;&#1
091;&#1096;</contact:street>
S:            <contact:city>&#1041;&#1086;&#1073;&#1088;&#1091;&#1081;&#1089;&#1
082;</contact:city>
S:            <contact:pc>20166-6503</contact:pc>
S:            <contact:cc>RU</contact:cc>
S:          </contact:addr>
S:        </contact:postalInfo>
S:        <contact:postalInfo type="int">
S:          <contact:name>Ivan Petrovich Sidorov</contact:name>
S:          <contact:addr>
S:            <contact:street>8343 Dragatush</contact:street>
S:            <contact:city>Babruysk</contact:city>
S:            <contact:pc>20166-6503</contact:pc>
S:            <contact:cc>RU</contact:cc>
S:          </contact:addr>
S:        </contact:postalInfo>
S:        <contact:voice x="1234">+1.7035555555</contact:voice>
```

```
S:        <contact:fax>+1.7035555556</contact:fax>
S:        <contact:email>ivan@example.com</contact:email>
S:        <contact:clID>ClientY</contact:clID>
S:        <contact:crID>ClientX</contact:crID>
S:        <contact:crDate>1999-04-03T22:00:00.0Z</contact:crDate>
S:        <contact:upID>ClientX</contact:upID>
S:        <contact:upDate>1999-12-03T09:00:00.0Z</contact:upDate>
S:        <contact:trDate>2000-04-08T09:00:00.0Z</contact:trDate>
S:        <contact:authInfo>
S:          <contact:pw>2fooBAR</contact:pw>
S:        </contact:authInfo>
S:        <contact:disclose flag="0">
S:          <contact:voice/>
S:          <contact:email/>
S:        </contact:disclose>
S:      </contact:infData>
S:    </resData>
S:    <extension>
S:      <ird:infData
S:        xmlns:ird="urn:ietf:params:xml:ns:ird-1.0"
S:        xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
S:        <ird:contactPostalInfo infSource="registrant"
S:          type="loc" authOrTransMechanism="authoritative">
S:          <ird:nameLang>ru</ird:nameLang>
S:          <ird:addrLang>ru</ird:addrLang>
S:          <ird:country lang="ru">&#1056;&#1086;&#1089;&#1089;&#1080;&#1103;</i
rd:country>
S:        </ird:contactPostalInfo>
S:        <ird:contactPostalInfo infSource="registrar"
S:          type="int" authOrTransMechanism="translation">
S:          <ird:nameLang>en</ird:nameLang>
S:          <ird:addrLang>en</ird:addrLang>
S:          <ird:country lang="en">Russia</ird:country>
S:        </ird:contactPostalInfo>
S:        <ird:additionalPostalInfo infSource="registrar"
S:          transMechanism="transliteration">
S:          <ird:id>sh8013-3</ird:id>
S:          <ird:name lang="ru-Latn">Ivan Petrovi&#269; Sidorov</ird:name>
S:          <ird:addr lang="ru-Latn">
S:            <ird:street>8343 Dragatus</ird:street>
S:            <ird:city>Bobrujsk</ird:city>
S:            <ird:pc>20166-6503</ird:pc>
S:            <ird:cc>RU</ird:cc>
S:          </ird:addr>
S:          <ird:country lang="ru-Latn">Rossia</ird:country>
S:          <ird:transliterationStd>iso9</ird:transliterationStd>
S:          <ird:disclose flag="0">
S:            <ird:name/>
S:          </ird:disclose>
```

```
S:        </ird:additionalPostalInfo>
S:        <ird:additionalPostalInfo infSource="registrar"
S:          transMechanism="translation">
S:          <ird:id>sh8013-4</ird:id>
S:          <ird:name lang="es">Ivan Petrovich Sidorov</ird:name>
S:          <ird:addr lang="es">
S:            <ird:street>8343 Dragatush</ird:street>
S:            <ird:city>Babruysk</ird:city>
S:            <ird:pc>20166-6503</ird:pc>
S:            <ird:cc>RU</ird:cc>
S:          </ird:addr>
S:          <ird:country lang="es">Rusia</ird:country>
S:          <ird:disclose flag="0">
S:            <ird:name/>
S:          </ird:disclose>
S:        </ird:additionalPostalInfo>
S:      </ird: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> command or <check> response described in the EPP contact mapping [RFC5733].

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 for the EPP <create> command described in the EPP contact mapping [RFC5733]. When a <create> command that includes this extension has been processed successfully, the response MUST include this EPP <extension> with a child <ird:infData> element.

The EPP <create> command provides a transform operation that allows a client to create a contact object. In addition to the EPP command elements described in the EPP contact mapping [RFC5733], the command MUST contain an <extension> element, and the <extension> element MUST contain a child <ird:infData> element that identifies the extension namespace if the client wants to associate data defined in this extension to the contact object. The <ird:infData> element contains the following child elements:

One or two <ird:contactPostalInfo> elements.

Zero or more OPTIONAL <ird:additionalPostalInfo> elements.

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:      <contact:create
C:        xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
C:        <contact:id>sh8013</contact:id>
C:        <contact:postalInfo type="loc">
C:          <contact:name>&#1048;&#1074;&#1072;&#1085; &#1055;&#1077;&#1090;&#10
88;&#1086;&#1074;&#1080;&#1095; &#1057;&#1080;&#1076;&#1086;&#1088;&#1086;&#1074
;</contact:name>
C:          <contact:addr>
C:            <contact:street>8343 &#1044;&#1088;&#1072;&#1075;&#1072;&#1090;&#1
091;&#1096;</contact:street>
C:            <contact:city>&#1041;&#1086;&#1073;&#1088;&#1091;&#1081;&#1089;&#1
082;</contact:city>
C:            <contact:pc>20166-6503</contact:pc>
C:            <contact:cc>RU</contact:cc>
C:          </contact:addr>
C:        </contact:postalInfo>
C:        <contact:postalInfo type="int">
C:          <contact:name>Ivan Petrovich Sidorov</contact:name>
C:          <contact:addr>
C:            <contact:street>8343 Dragatush</contact:street>
C:            <contact:city>Babruysk</contact:city>
C:            <contact:pc>20166-6503</contact:pc>
C:            <contact:cc>RU</contact:cc>
C:          </contact:addr>
C:        </contact:postalInfo>
C:        <contact:voice x="1234">+1.7035555555</contact:voice>
C:        <contact:fax>+1.7035555556</contact:fax>
C:        <contact:email>ivan@example.com</contact:email>
C:        <contact:authInfo>
C:          <contact:pw>2fooBAR</contact:pw>
C:        </contact:authInfo>
C:        <contact:disclose flag="0">
C:          <contact:voice/>
```

```
C:          <contact:email/>
C:        </contact:disclose>
C:      </contact:create>
C:    </create>
C:    <extension>
C:      <ird:infData
C:        xmlns:ird="urn:ietf:params:xml:ns:ird-1.0" >
C:        <ird:contactPostalInfo infSource="registrant"
C:          type="loc" authOrTransMechanism="authoritative">
C:          <ird:nameLang>ru</ird:nameLang>
C:          <ird:addrLang>ru</ird:addrLang>
C:          <ird:country lang="ru">&#1056;&#1086;&#1089;&#1089;&#1080;&#1103;</i
rd:country>
C:        </ird:contactPostalInfo>
C:        <ird:contactPostalInfo infSource="registrar"
C:          type="int" authOrTransMechanism="translation">
C:          <ird:nameLang>en</ird:nameLang>
C:          <ird:addrLang>en</ird:addrLang>
C:          <ird:country lang="en">Russia</ird:country>
C:        </ird:contactPostalInfo>
C:        <ird:additionalPostalInfo infSource="registrar"
C:          transMechanism="transliteration">
C:          <ird:id>sh8013-3</ird:id>
C:          <ird:name lang="ru-Latn">Ivan Petrovi&#269; Sidorov</ird:name>
C:          <ird:addr lang="ru-Latn">
C:            <ird:street>8343 Dragatus</ird:street>
C:            <ird:city>Bobrujsk</ird:city>
C:            <ird:pc>20166-6503</ird:pc>
C:            <ird:cc>RU</ird:cc>
C:          </ird:addr>
C:          <ird:country lang="ru-Latn">Rossia</ird:country>
C:          <ird:transliterationStd>iso9</ird:transliterationStd>
C:          <ird:disclose flag="0">
C:            <ird:name/>
C:          </ird:disclose>
C:        </ird:additionalPostalInfo>
C:        <ird:additionalPostalInfo infSource="registrar"
C:          transMechanism="translation">
C:          <ird:id>sh8013-4</ird:id>
C:          <ird:name lang="es">Ivan Petrovich Sidorov</ird:name>
C:          <ird:addr lang="es">
C:            <ird:street>8343 Dragatush</ird:street>
C:            <ird:city>Babruysk</ird:city>
C:            <ird:pc>20166-6503</ird:pc>
C:            <ird:cc>RU</ird:cc>
C:          </ird:addr>
C:          <ird:country lang="es">Rusia</ird:country>
C:          <ird:disclose flag="0">
C:            <ird:name/>
```

```
C:          </ird:disclose>
C:        </ird:additionalPostalInfo>
C:      </ird:infData>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

3.2.2. EPP <delete> command

This extension does not add any elements to the EPP <delete> command or <delete> response described in the EPP contact mapping [RFC5733].

3.2.3. EPP <renew> command

This extension does not add any elements to the EPP <renew> command or <renew> response described in the EPP contact mapping [RFC5733].

3.2.4. EPP <transfer> command

This extension does not add any elements to the EPP <transfer> command or <transfer> response described in the EPP contact mapping [RFC5733].

3.2.5. EPP <update> command

This extension defines additional elements for the EPP <update> command described in the EPP contact mapping [RFC5733]. When an <update> command that includes this extension has been processed successfully, the response MUST include this EPP <extension> with a child <ird:infData> element.

The EPP <update> command provides a transform operation that allows a client to modify the attributes of a contact object. In addition to the EPP command elements described in the EPP contact mapping, the command MUST contain an <extension> element, and the <extension> element MUST contain a child <ird:update> element that identifies the extension namespace if the client wants to update the contact object with data defined in this extension. The <ird:update> element contains a <ird:add> element to add ird information or, a <ird:rem> element to remove ird information.

The order of the <ird:rem> and <ird:add> elements is significant, where the server MUST first remove the existing elements prior to adding the new elements.

The <ird:update> element contains the following child elements:

An OPTIONAL <ird:add> element that is used to add IRD information to a contact object. The <ird:add> element MUST contain the following child elements:

One or two <ird:contactPostalInfo> elements.

Zero or more OPTIONAL <ird:additionalPostalInfo> elements.

An OPTIONAL <ird:rem> element that is used to remove IRD information of a contact object. The <ird:rem> element MUST contain at least one of the following child elements:

Zero or two OPTIONAL <ird:contactPostalInfoRem> elements the contains the type ("loc" or "int) of <ird:contactPostalInfo> to be removed.

Zero or more OPTIONAL <ird:id> elements.

Example of <update>:

```
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:      <contact:update
C:        xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
C:        <contact:id>sh8013</contact:id>
C:        <contact:add>
C:          <contact:status s="clientDeleteProhibited"/>
C:        </contact:add>
C:      </contact:update>
C:    </update>
C:    <extension>
C:      <ird:update
C:        xmlns:ird="urn:ietf:params:xml:ns:ird-1.0" >
C:        <ird:rem>
C:          <ird:contactPostalInfoRem>int</ird:contactPostalInfoRem>
C:          <ird:id>sh8013-1</ird:id>
C:          <ird:id>sh8013-2</ird:id>
C:        </ird:rem>
C:        <ird:add>
C:          <ird:contactPostalInfo infSource="registrant"
C:            type="loc" authOrTransMechanism="authoritative">
C:            <ird:nameLang>ru</ird:nameLang>
C:            <ird:addrLang>ru</ird:addrLang>
C:            <ird:country lang="ru">&#1056;&#1086;&#1089;&#1089;&#1080;&#1103;<
/ird:country>
C:          </ird:contactPostalInfo>
C:          <ird:contactPostalInfo infSource="registrar"
```

```
C:            type="int" authOrTransMechanism="translation">
C:            <ird:nameLang>en</ird:nameLang>
C:            <ird:addrLang>en</ird:addrLang>
C:            <ird:country lang="en">Russia</ird:country>
C:          </ird:contactPostalInfo>
C:          <ird:additionalPostalInfo infSource="registrar"
C:            transMechanism="transliteration">
C:            <ird:id>sh8013-3</ird:id>
C:            <ird:name lang="ru-Latn">Ivan Petrovi&#269; Sidorov</ird:name>
C:            <ird:addr lang="ru-Latn">
C:              <ird:street>8343 Dragatus</ird:street>
C:              <ird:city>Bobrujsk</ird:city>
C:              <ird:pc>20166-6503</ird:pc>
C:              <ird:cc>RU</ird:cc>
C:            </ird:addr>
C:            <ird:country lang="ru-Latn">Rossia</ird:country>
C:            <ird:transliterationStd>iso9</ird:transliterationStd>
C:            <ird:disclose flag="0">
C:              <ird:name/>
C:            </ird:disclose>
C:          </ird:additionalPostalInfo>
C:          <ird:additionalPostalInfo infSource="registrar"
C:            transMechanism="translation">
C:            <ird:id>sh8013-4</ird:id>
C:            <ird:name lang="es">Ivan Petrovich Sidorov</ird:name>
C:            <ird:addr lang="es">
C:              <ird:street>8343 Dragatush</ird:street>
C:              <ird:city>Babruysk</ird:city>
C:              <ird:pc>20166-6503</ird:pc>
C:              <ird:cc>RU</ird:cc>
C:            </ird:addr>
C:            <ird:country lang="es">Rusia</ird:country>
C:            <ird:disclose flag="0">
C:              <ird:name/>
C:            </ird:disclose>
C:          </ird:additionalPostalInfo>
C:        </ird:add>
C:      </ird:update>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

4. 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.

4.1. ird Schema


```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:ird-1.0"
  xmlns:ird="urn:ietf:params:xml:ns:ird-1.0"
  xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
  xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
  xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

  <import namespace="urn:ietf:params:xml:ns:contact-1.0"
    schemaLocation="contact-1.0.xsd"/>
  <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 contact extension
      schema for provisioning and management the transformation
      (i.e. transliteration or translation) of an entity
      object.
    </documentation>
  </annotation>

  <element name="infData" type="ird:infDataType"/>
  <element name="update" type="ird:updateDataType"/>

  <complexType name="updateDataType">
    <sequence>
      <element name="rem" type="ird:remType"
        minOccurs="0"/>
      <element name="add" type="ird:infDataType"
```

```
          minOccurs="0"/>
      </sequence>
    </complexType>

    <complexType name="remType">
      <sequence>
        <element name="contactPostalInfoRem"
          type="contact:postalInfoEnumType"
          minOccurs="0" maxOccurs="2"/>
        <element name="id" type="eppcom:clIDType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </complexType>


    <complexType name="infDataType">
      <sequence>
        <element name="contactPostalInfo"
          type="ird:contactPostalInfoDataType"
          minOccurs="1" maxOccurs="2"/>
        <element name="additionalPostalInfo"
          type="ird:additionalPostalInfoDataType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </complexType>

    <complexType name="contactPostalInfoDataType">
      <sequence>
        <element name="nameLang" type="language" />
        <element name="orgLang" type="language" minOccurs="0"/>
        <element name="addrLang" type="language" />
        <element name="country" type="ird:countryType" />
        <element name="transliterationStd" type="token"
          minOccurs="0"/>
      </sequence>
      <attribute name="type" type="contact:postalInfoEnumType"
        use="required"/>
      <attribute name="infSource" type="ird:infSourceEnumType"
        use="required"/>
      <attribute name="authOrTransMechanism"
        type="ird:authOrTransMechanismEnumType" />
    </complexType>

    <complexType name="additionalPostalInfoDataType">
      <sequence>
        <element name="id" type="eppcom:clIDType"/>
        <element name="name" type="ird:name" />
        <element name="org" type="ird:org"
```

```
          minOccurs="0"/>
        <element name="addr" type="ird:addrType"/>
        <element name="country" type="ird:countryType" />
        <element name="transliterationStd" type="token"
          minOccurs="0"/>
        <element name="disclose" type="ird:discloseType"
          minOccurs="0"/>
      </sequence>
      <attribute name="infSource" type="ird:infSourceEnumType"
        use="required"/>
      <attribute name="transMechanism"
        type="ird:conversionMechanismEnumType" use="required"/>
    </complexType>

    <complexType name="name">
      <simpleContent>
        <extension base="contact:postalLineType">
          <attribute name="lang" type="language" use="required"/>
        </extension>
      </simpleContent>
    </complexType>

    <complexType name="org">
      <simpleContent>
        <extension base="contact:optPostalLineType">
          <attribute name="lang" type="language" use="required"/>
        </extension>
      </simpleContent>
    </complexType>

    <complexType name="addrType">
      <sequence>
        <element name="street" type="contact:optPostalLineType"
          minOccurs="0" maxOccurs="3"/>
        <element name="city" type="contact:postalLineType"/>
        <element name="sp" type="contact:optPostalLineType"
          minOccurs="0"/>
        <element name="pc" type="contact:pcType"
          minOccurs="0"/>
        <element name="cc" type="contact:ccType"/>
      </sequence>
      <attribute name="lang" type="language" use="required" />
    </complexType>

    <complexType name="countryType">
      <simpleContent>
        <extension base="token">
          <attribute name="lang" type="language" use="required" />
```

```
        </extension>
      </simpleContent>
    </complexType>

    <simpleType name="infSourceEnumType">
      <restriction base="token">
        <enumeration value="registry"/>
        <enumeration value="registrar"/>
        <enumeration value="reseller"/>
        <enumeration value="registrant"/>
      </restriction>
    </simpleType>

    <simpleType name="conversionMechanismEnumType">
      <restriction base="token">
        <enumeration value="translation"/>
        <enumeration value="transliteration"/>
      </restriction>
    </simpleType>

    <simpleType name="authOrTransMechanismEnumType">
      <restriction base="token">
        <enumeration value="authoritative"/>
        <enumeration value="translation"/>
        <enumeration value="transliteration"/>
      </restriction>
    </simpleType>

    <complexType name="discloseType">
      <sequence>
        <element name="name" minOccurs="0"/>
        <element name="org" minOccurs="0"/>
        <element name="addr" minOccurs="0"/>
        <element name="voice" minOccurs="0"/>
        <element name="fax" minOccurs="0"/>
        <element name="email" minOccurs="0"/>
      </sequence>
      <attribute name="flag" type="boolean" use="required"/>
    </complexType>

  </schema>
  END
```

5. Acknowledgements

TBD.

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 ird namespace:

URI: urn:ietf:params:xml:ns:ird-1.0"

Registrant Contact: IESG

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

Registration request for the ird schema:

URI: urn:ietf:params:xml:schema:ird-1.0"

Registrant Contact: IESG

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

7. Internationalization Considerations

The internationalization considerations of EPP described in [RFC5730] apply to this specification as well.

8. Security Considerations

The mapping extensions described in this document do not provide any security services beyond those described by EPP [RFC5730], the EPP contact mapping [RFC5733], and protocol layers used by EPP. The security considerations described in these other specifications apply to this specification as well.

9. References

9.1. 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>.

[RFC3735] Hollenbeck, S., "Guidelines for Extending the Extensible Provisioning Protocol (EPP)", RFC 3735, DOI 10.17487/RFC3735, March 2004, <http://www.rfc-editor.org/info/rfc3735>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <http://www.rfc-editor.org/info/rfc5646>.

[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>.

[W3C.REC-xml]
W3C, "Extensible Markup Language (XML) 1.0 (Second Edition)", Oct 2000, <https://www.w3.org/TR/REC-xml/>.

[W3C.REC-xmlschema-1]
W3C, "XML Schema Part 1: Structures Second Edition", Oct 2004, <http://www.w3.org/TR/xUmlschema-1/>.

[W3C.REC-xmlschema-2]
W3C, "XML Schema Part 2: Structures Second Edition", Oct 2004, <https://www.w3.org/TR/xmlschema-2/>.

9.2. Informative References

[TTOCI] ICANN, "Final Report on the Translation and Transliteration of Contact Information Policy Development Process", June 2015, <http://gnso.icann.org/en/issues/gtlds/translation-transliteration-contact-final-12jun15-en.pdf>.

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: December 26, 2016

G. Lozano
ICANN

June 24, 2016

RDAP Transformation of Contact Information
draft-lozano-regext-rdap-transf-contact-inf-00

Abstract

This document adds support for RDAP Transformation (i.e. translation and transliteration) of Contact Information.

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 26, 2016.

Copyright Notice

Table of Contents

1. Introduction

Generic TLDs (gTLDs) Domain Name Registries and Registrars operating according to an agreement with the Internet Corporation for Assigned Names and Numbers (ICANN), are required to follow ICANN Consensus Policies.

On 2015/09/28, the ICANN Board adopted the GNSO Council Policy Recommendations concerning the translation and transliteration of contact information as presented in the "Final Report on the Translation and Transliteration of Contact Information Policy Development Process" (T&T Report, see [TTOCI]).

The T&T Report recommends:

1. Contact information that has been transformed (i.e. transliteration or translated) must be marked as such and their source(s) indicated.

This specification is intended to be used by gTLD Domain Name Registries and Registrars in order to support this recommendation.

2. RDAP Conformance object

A server that conforms to this specification MUST include the string literal "rdap_transformation_of_contact_information" in the "rdapConformance" member of the topmost JSON object of all responses provided by the server.

An example of a rdapConformance data structure including this extension:

```
"rdapConformance" :
[
 "rdap_level_0",
 "rdap_transformation_of_contact_information"
]
```

3. Transformation object class

A transformation object class represents the transformation (i.e. transliteration or translation) of an entity object.

The transformation object class uses jCard [RFC7095] to represent contact information, such as postal addresses, email addresses, phone numbers and names of organizations and individuals. Many of the types of information that can be represented with jCard have no use in RDAP, such as birthdays, anniversaries, and gender.

The transformation object class contains the following members:

- objectClassName -- the string "transformation".
- vcardArray -- a jCard with the entity's contact information. The language property MUST be included in the jCard object in case of a translation in order to indicate the language that the contact information was translated to.
- sourceOfTransformation -- a string containing the source of the transformation. The possible values of sourceOfTransformation are: registry, registrar or reseller.
- typeOfTransformation -- a string containing the type of the transformation. The possible values of typeOfTransformation are: translation or transliteration.
- An OPTIONAL transliterationStandard -- a string containing the standard (e.g. ISO 9, ISO 7098, etc.) used for the transliteration. This element MUST be present if typeOfTransformation is transliteration.

4. Transformations object class

The transformations object is an array of transformation objects (defined in Section 3) included in an entity object.

If a transformations object is included, the parent entity object MUST show the contact information as originally inputted by the user.

The following is an example of a JSON domain object, that includes a transformations object, representing a forward DNS delegation point that might be served by a DNR.

```
{
  "objectClassName":"domain",
  "handle":"XXXX",
  "ldhName":"xn--fo-5ja.example",
  "unicodeName":"foo.example",
  "variants":[
    {
      "relation":[
        "registered",
        "conjoined"
      ],
      "variantNames":[
        {
          "ldhName":"xn--fo-cka.example",
          "unicodeName":"foo.example"
        },
        {
          "ldhName":"xn--fo-fka.example",
          "unicodeName":"foo.example"
        }
      ]
    },
    {
      "relation":[
        "unregistered",
        "registration restricted"
      ],
      "idnTable":".EXAMPLE Swedish",
      "variantNames":[
        {
          "ldhName":"xn--fo-8ja.example",
          "unicodeName":"foo.example"
        }
      ]
    }
  ],
  "status":[
    "locked",
    "transfer prohibited"
  ],
  "publicIds":[
    {
      "type":"ENS_Auth ID",
      "identifier":"1234567890"
    }
  ],
  "nameservers":[
    {
```

```
        "objectClassName":"nameserver",
        "handle":"XXXX",
        "ldhName":"ns1.example.com",
        "status":[
          "active"
        ],
        "ipAddresses":{
          "v6":[
            "2001:db8::123",
            "2001:db8::124"
          ],
          "v4":[
            "192.0.2.1",
            "192.0.2.2"
          ]
        },
        "remarks":[
          {
            "description":[
              "She sells sea shells down by the sea shore.",
              "Originally written by Terry Sullivan."
            ]
          }
        ],
        "links":[
          {
            "value":"http://example.net/nameserver/XXXX",
            "rel":"self",
            "href":"http://example.net/nameserver/XXXX",
            "type":"application/rdap+json"
          }
        ],
        "events":[
          {
            "eventAction":"registration",
            "eventDate":"1990-12-31T23:59:59Z"
          },
          {
            "eventAction":"last changed",
            "eventDate":"1991-12-31T23:59:59Z"
          }
        ]
      },
      {
        "objectClassName":"nameserver",
        "handle":"XXXX",
        "ldhName":"ns2.example.com",
        "status":[
```

```
          "active"
        ],
        "ipAddresses":{
          "v6":[
            "2001:db8::125",
            "2001:db8::126"
          ],
          "v4":[
            "192.0.2.3",
            "192.0.2.4"
          ]
        },
        "remarks":[
          {
            "description":[
              "She sells sea shells down by the sea shore.",
              "Originally written by Terry Sullivan."
            ]
          }
        ],
        "links":[
          {
            "value":"http://example.net/nameserver/XXXX",
            "rel":"self",
            "href":"http://example.net/nameserver/XXXX",
            "type":"application/rdap+json"
          }
        ],
        "events":[
          {
            "eventAction":"registration",
            "eventDate":"1990-12-31T23:59:59Z"
          },
          {
            "eventAction":"last changed",
            "eventDate":"1991-12-31T23:59:59Z"
          }
        ]
      }
    ],
    "secureDNS":{
      "zoneSigned":true,
      "delegationSigned":true,
      "maxSigLife":604800,
      "keyData":[
        {
          "flags":257,
          "protocol":3,
```

```
          "algorithm":1,
          "publicKey":"AQPJ////4Q==",
          "events":[
            {
              "eventAction":"last changed",
              "eventDate":"2012-07-23T05:15:47Z"
            }
          ]
        }
      ]
    },
    "remarks":[
      {
        "description":[
          "She sells sea shells down by the sea shore.",
          "Originally written by Terry Sullivan."
        ]
      }
    ],
    "links":[
      {
        "value":"http://example.net/domain/XXXX",
        "rel":"self",
        "href":"http://example.net/domain/XXXX",
        "type":"application/rdap+json"
      }
    ],
    "port43":"whois.example.net",
    "events":[
      {
        "eventAction":"registration",
        "eventDate":"1990-12-31T23:59:59Z"
      },
      {
        "eventAction":"last changed",
        "eventDate":"1991-12-31T23:59:59Z",
        "eventActor":"joe@example.com"
      },
      {
        "eventAction":"transfer",
        "eventDate":"1991-12-31T23:59:59Z",
        "eventActor":"joe@example.com"
      },
      {
        "eventAction":"expiration",
        "eventDate":"2016-12-31T23:59:59Z",
        "eventActor":"joe@example.com"
      }
```

```
    ],
    "entities":[
      {
        "objectClassName":"entity",
        "handle":"XXXX",
        "vcardArray":[
          "vcard",
          [
            ["version", {}, "text", "4.0"],
            ["fn",{"language":"zh"},"text","&#29579;&#20116;"],
            ["adr",
              {"language":"zh"},
              "text",
              [
                "",
                "1827&#21495;",
                "1047 &#29275;&#23665;&#29645;&#25252;&#33181;&#26449;",
                "&#36830;&#20113;&#28207;&#24066;",
                "&#27743;&#33487;&#30465;",
                "&#20013;&#22269;"
              ]
            ],
            ["tel", {},
              "uri", "tel:+86146-7515-1740"
            ],
            ["email", {},
              "text", "&#29579;&#20116;@&#20363;.&#20363;"
            ]
          ]
        ],
        "transformations":[
          {
            "objectClassName":"transformation",
            "vcardArray":[
              "vcard",
              [
                ["version", {}, "text", "4.0"],
                ["fn", {"language":"en"}, "text", "Wang Wu"],
                ["adr",
                  {"language":"en"},
                  "text",
                  [
                    "",
                    "No. 1827",
                    "1047 Niu Shan Zhen Hu Xi Cun",
                    "Lianyungang City",
                    "Jiangsu Province",
                    "China"
```

```
                ]
              ],
              ["tel", {},
                "uri", "tel:+86146-7515-1740"
              ],
              ["email", {},
                "text", "&#29579;&#20116;@&#20363;.&#20363;"
              ]
            ]
          ],
          "sourceOfTransformation":"registry",
          "typeOfTransformation":"translation"
        },
        {
          "objectClassName":"transformation",
          "vcardArray":[
            "vcard",
            [
              ["version", {}, "text", "4.0"],
              ["fn",{"language":"sp"},"text","Wang Wu"],
              ["adr",
                {"language":"sp"},
                "text",
                [
                  "",
                  "No. 1827",
                  "1047 Niu Shan Zhen Hu Xi Cun",
                  "Ciudad Lianyungang",
                  "Provincia Jiangsu",
                  "China"
                ]
              ],
              ["tel", {},
                "uri", "tel:+86146-7515-1740"
              ],
              ["email", {},
                "text", "&#29579;&#20116;@&#20363;.&#20363;"
              ]
            ]
          ],
          "sourceOfTransformation":"registry",
          "typeOfTransformation":"translation"
        }
      ],
      "status":[
        "validated",
        "locked"
      ],
```

```
        "roles":[
          "registrant"
        ],
        "remarks":[
          {
            "description":[
              "She sells sea shells down by the sea shore.",
              "Originally written by Terry Sullivan."
            ]
          }
        ],
        "links":[
          {
            "value":"http://example.net/entity/xxxx",
            "rel":"self",
            "href":"http://example.net/entity/xxxx",
            "type":"application/rdap+json"
          }
        ],
        "events":[
          {
            "eventAction":"registration",
            "eventDate":"1990-12-31T23:59:59Z"
          },
          {
            "eventAction":"last changed",
            "eventDate":"1991-12-31T23:59:59Z"
          }
        ]
      }
    ]
  }
```

5. Acknowledgements

TBD.

6. IANA Considerations

TBD.

7. Security Considerations

TBD.

8. References

8.1. Normative References

[RFC7095] Kewisch, P., "jCard: The JSON Format for vCard", RFC 7095, DOI 10.17487/RFC7095, January 2014, <http://www.rfc-editor.org/info/rfc7095>.

8.2. Informative References

[TTOCI] ICANN, "Final Report on the Translation and Transliteration of Contact Information Policy Development Process", June 2015, <http://gnso.icann.org/en/issues/gtlds/translation-transliteration-contact-final-12jun15-en.pdf>.

Author's Address

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

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