

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 7, 2019

J. Gould
VeriSign, Inc.
K. Feher
Neustar
October 04, 2018

Allocation Token Extension for the Extensible Provisioning Protocol
(EPP)
draft-ietf-regext-allocation-token-12

Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension for including an Allocation Token in "query" and "transform" commands. The Allocation Token is used as a credential that authorizes a client to request the allocation of a specific object from the server, using one of the EPP transform commands including create and transfer.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Conventions Used in This Document	3
2. Object Attributes	4
2.1. Allocation Token	4
3. EPP Command Mapping	5
3.1. EPP Query Commands	5
3.1.1. EPP <check> Command	5
3.1.2. EPP <info> Command	9
3.1.3. EPP <transfer> Query Command	11
3.2. EPP Transform Commands	12
3.2.1. EPP <create> Command	12
3.2.2. EPP <delete> Command	13
3.2.3. EPP <renew> Command	13
3.2.4. EPP <transfer> Command	13
3.2.5. EPP <update> Command	14
4. Formal Syntax	15
4.1. Allocation Token Extension Schema	15
5. IANA Considerations	16
5.1. XML Namespace	16
5.2. EPP Extension Registry	16
6. Implementation Status	16
6.1. Verisign EPP SDK	17
6.2. Neustar EPP SDK	17
6.3. Neustar gTLD SRS	18
6.4. Net::DRI	18
7. Security Considerations	19
8. Acknowledgements	19
9. References	19
9.1. Normative References	19
9.2. Informative References	20
Appendix A. Change History	20
A.1. Change from 00 to 01	20
A.2. Change from 01 to 02	20
A.3. Change from 02 to 03	21
A.4. Change from 03 to 04	21
A.5. Change from 04 to REGEXT 00	21
A.6. Change from REGEXT 00 to REGEXT 01	21
A.7. Change from REGEXT 01 to REGEXT 02	21
A.8. Change from REGEXT 02 to REGEXT 03	21
A.9. Change from REGEXT 03 to REGEXT 04	21
A.10. Change from REGEXT 04 to REGEXT 05	21
A.11. Change from REGEXT 05 to REGEXT 06	22

A.12. Change from REGEXT 06 to REGEXT 07	23
A.13. Change from REGEXT 07 to REGEXT 08	23
A.14. Change from REGEXT 08 to REGEXT 09	24
A.15. Change from REGEXT 09 to REGEXT 10	24
A.16. Change from REGEXT 10 to REGEXT 11	25
A.17. Change from REGEXT 11 to REGEXT 12	26
Authors' Addresses	26

1. Introduction

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This mapping, an extension to EPP object mappings like the EPP domain name mapping [RFC5731], supports passing an Allocation Token as a credential that authorizes a client to request the allocation of a specific object from the server, using one of the EPP transform commands including create and transfer.

Allocation is when a server assigns the sponsoring client of an object based on the use of an Allocation Token credential. Examples include allocating a registration based on a pre-eligibility Allocation Token, allocating a premium domain name registration based on an auction Allocation Token, allocating a registration based on a founders Allocation Token, and allocating an existing domain name held by the server or by a different sponsoring client based on an Allocation Token passed with a transfer command.

Clients pass an Allocation Token to the server for validation, and the server determines if the supplied Allocation Token is one supported by the server. It is up to server policy which EPP transform commands and which objects require the Allocation Token. The Allocation Token MAY be returned to an authorized client for passing out-of-band to a client that uses it with an EPP transform command.

1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "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.

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

The XML namespace prefix "allocationToken" is used for the namespace "urn:ietf:params:xml:ns:allocationToken-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 "abc123" token value is used as a placeholder value in the examples. The server MUST support token values that follow the Security Considerations (Section 7) section.

The domain object attribute values, including the "2fooBAR" <domain:pw> value, in the examples are provided for illustration purposes only. Refer to [RFC5731] for details on the domain object attributes.

2. Object Attributes

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

2.1. Allocation Token

The Allocation Token is a simple XML "token" type. The exact format of the Allocation Token is up to server policy. The server MAY have the Allocation Token for each object to match against the Allocation Token passed by the client to authorize the allocation of the object. The <allocationToken:allocationToken> element is used for all of the supported EPP commands as well as the info response. If the supplied Allocation Token passed to the server does not apply to the object, the server MUST return an EPP error result code of 2201.

Authorization information, like what is defined in the EPP domain name mapping [RFC5731], is associated with objects to facilitate transfer operations. The authorization information is assigned when an object is created. The Allocation Token and the authorization information are both credentials, but used for different purposes and used in different ways. The Allocation Token is used to facilitate the allocation of an object instead of transferring the sponsorship of the object. The Allocation Token is not managed by the client, but is validated by the server to authorize assigning the initial sponsoring client of the object.

An example <allocationToken:allocationToken> element with value of "abc123":

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

3. EPP Command Mapping

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

3.1. EPP Query Commands

EPP provides three commands to retrieve object information: <check> to determine if an object can be provisioned, <info> to retrieve information associated with an object, and <transfer> to retrieve object transfer status information.

3.1.1. EPP <check> Command

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

This extension allows clients to check the availability of an object with an Allocation Token, as described in Section 2.1. Clients can check if an object can be created using the Allocation Token. The Allocation Token is applied to all object names included in the EPP <check> command.

Example <check> command for the allocation.example domain name using the <allocationToken:allocationToken> extension with the allocation token of 'abc123':

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

If the query was successful, the server replies with a <check> response providing the availability status of the queried object based on the following Allocation Token cases, where the object is otherwise available:

1. If an object requires an Allocation Token and the Allocation Token does apply to the object, then the server MUST return the availability status as available (e.g., "avail" attribute is "1" or "true").
2. If an object requires an Allocation Token and the Allocation Token does not apply to the object, then the server SHOULD return the availability status as unavailable (e.g., "avail" attribute is "0" or "false").
3. If an object does not require an Allocation Token, the server MAY return the availability status as available (e.g., "avail" attribute is "1" or "true").

Example <check> domain response for a <check> command using the <allocationToken:allocationToken> extension:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S: <response>
S:   <result code="1000">
S:     <msg lang="en-US">Command completed successfully</msg>
S:   </result>
S:   <resData>
S:     <domain:chkData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:       <domain:cd>
S:         <domain:name avail="1">allocation.example</domain:name>
S:       </domain:cd>
S:     </domain:chkData>
S:   </resData>
S:   <trID>
S:     <clTRID>ABC-DEF-12345</clTRID>
S:     <svTRID>54321-XYZ</svTRID>
S:   </trID>
S: </response>
S:</epp>
```

Example <check> command with the <allocationToken:allocationToken> extension for the allocation.example and allocation2.example domain names. Availability of allocation.example and allocation2.example domain names are based on the Allocation Token 'abc123':

```
C:<?xml version="1.0" encoding="UTF-8"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <check>
C:      <domain:check
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>allocation.example</domain:name>
C:          <domain:name>allocation2.example</domain:name>
C:        </domain:check>
C:      </check>
C:    <extension>
C:      <allocationToken:allocationToken
C:        xmlns:allocationToken=
C:          "urn:ietf:params:xml:ns:allocationToken-1.0">
C:        abc123
C:      </allocationToken:allocationToken>
C:    </extension>
C:  <clTRID>ABC-DEF-12345</clTRID>
C: </command>
C:</epp>
```


Example <check> domain response for multiple domain names in the <check> command using the <allocationToken:allocationToken> extension, where the Allocation Token 'abc123' matches allocation.example but does not match allocation2.example:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S: <response>
S:   <result code="1000">
S:     <msg lang="en-US">Command completed successfully</msg>
S:   </result>
S:   <resData>
S:     <domain:chkData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:       <domain:cd>
S:         <domain:name avail="1">allocation.example</domain:name>
S:       </domain:cd>
S:       <domain:cd>
S:         <domain:name avail="0">allocation2.example</domain:name>
S:         <domain:reason>Allocation Token mismatch</domain:reason>
S:       </domain:cd>
S:     </domain:chkData>
S:   </resData>
S:   <trID>
S:     <clTRID>ABC-DEF-12345</clTRID>
S:     <svTRID>54321-XYZ</svTRID>
S:   </trID>
S: </response>
S:</epp>
```

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

3.1.2. EPP <info> Command

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

The EPP <info> command allows a client to request information associated with an existing object. Authorized clients MAY retrieve the Allocation Token (Section 2.1) along with the other object information by supplying the <allocationToken:info> element in the command. The <allocationToken:info> element is an empty element that serves as a marker to the server to return the <allocationToken:allocationToken> element in the info response. If the client is not authorized to receive the Allocation Token, the server MUST return an EPP error result code of 2201. If the client

is authorized to receive the Allocation Token, but there is no Allocation Token associated with the object, the server MUST return an EPP error result code of 2303. The authorization is subject to server policy.

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

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

If the query was successful, the server replies with an <allocationToken:allocationToken> element along with the regular EPP <resData>. The <allocationToken:allocationToken> element is described in Section 2.1.

Example <info> domain response using the
<allocationToken:allocationToken> extension:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>allocation.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:      <allocationToken:allocationToken
S:        xmlns:allocationToken=
S:          "urn:ietf:params:xml:ns:allocationToken-1.0">
S:        abc123
S:      </allocationToken:allocationToken>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.1.3. EPP <transfer> Query Command

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

3.2. EPP Transform Commands

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

3.2.1. EPP <create> Command

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

The EPP <create> command provides a transform operation that allows a client to create an instance of an object. In addition to the EPP command elements described in an object mapping like [RFC5731], the command MUST contain a child <allocationToken:allocationToken> element for the client to be authorized to create and allocate the object. If the Allocation Token does not apply to the object, the server MUST return an EPP error result code of 2201.

Example <create> command to create a domain object with an Allocation Token:

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

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

3.2.2. EPP <delete> Command

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

3.2.3. EPP <renew> Command

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

3.2.4. EPP <transfer> Command

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

The EPP <transfer> request command provides a transform operation that allows a client to request the transfer of an object. In addition to the EPP command elements described in an object mapping like [RFC5731], the command MUST contain a child <allocationToken:allocationToken> element for the client to be authorized to transfer and allocate the object. The authorization associated with the Allocation Token is in addition to and does not replace the authorization mechanism defined for the object's <transfer> request command. If the Allocation Token is invalid or not required for the object, the server MUST return an EPP error result code of 2201.

Example <transfer> request command to allocate the domain object with the Allocation Token:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <transfer op="request">
C:      <domain:transfer
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example1.tld</domain:name>
C:        <domain:period unit="y">1</domain:period>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:transfer>
C:    </transfer>
C:    <extension>
C:      <allocationToken:allocationToken
C:        xmlns:allocationToken=
C:          "urn:ietf:params:xml:ns:allocationToken-1.0">
C:        abc123
C:      </allocationToken:allocationToken>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

3.2.5. EPP <update> Command

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

4. Formal Syntax

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

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

4.1. Allocation Token Extension Schema

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:allocationToken="urn:ietf:params:xml:ns:allocationToken-1.0"
  targetNamespace="urn:ietf:params:xml:ns:allocationToken-1.0"
  elementFormDefault="qualified">
  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
      Allocation Token Extension
    </documentation>
  </annotation>

  <!-- Element used in info command to get allocation token. -->
  <element name="info">
    <complexType>
      <complexContent>
        <restriction base="anyType" />
      </complexContent>
    </complexType>
  </element>

  <!-- Allocation Token used in transform
  commands and info response -->
  <element name="allocationToken"
    type="allocationToken:allocationTokenType" />
  <simpleType name="allocationTokenType">
    <restriction base="token">
      <minLength value="1" />
    </restriction>
  </simpleType>

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

5. IANA Considerations

5.1. XML Namespace

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

Registration request for the allocationToken namespace:

URI: urn:ietf:params:xml:ns:allocationToken-1.0
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.

Registration request for the allocationToken XML schema:

URI: urn:ietf:params:xml:schema:allocationToken-1.0
Registrant Contact: IESG
XML: See the "Formal Syntax" section of this document.

5.2. EPP Extension Registry

The following registration of the EPP Extension Registry, described in [RFC7451], is requested:

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

Document status: Standards Track

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

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

6. 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-allocation-token.

Level of maturity: Production

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

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

6.2. Neustar EPP SDK

Organisation: Neustar Inc.

Name: Neustar EPP SDK

Description: The Neustar EPP SDK includes a full client implementation of draft-ietf-regext-allocation-token.

Level of maturity: Production

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: quoc-anh.np@team.neustar

URL: <http://registrytoolkit.neustar>

6.3. Neustar gTLD SRS

Organisation: Neustar Inc.

Name: Neustar generic Top Level Domain (gTLD) Shared Registry System (SRS).

Description: The Neustar gTLD SRS implements the server side of draft-ietf-regext-allocation-token for several Top Level Domains.

Level of maturity: Production

Coverage: All server side aspects of the protocol are implemented.

Licensing: Proprietary

Contact: quoc-anh.np@team.neustar

6.4. Net::DRI

Organization: Dot and Co

Name: Net::DRI

Description: Net::DRI implements the client-side of draft-ietf-regext-allocation-token.

Level of maturity: Production

Coverage: All client-side aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: netdri@dotandco.com

7. Security Considerations

The mapping described in this document does 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.

The mapping acts as a conduit for the passing of Allocation Tokens between a client and a server. The definition of the Allocation Token SHOULD be defined outside of this mapping. The following are security considerations in the definition and use of an Allocation Token:

1. An Allocation Token should be considered secret information by the client and SHOULD be protected at rest and MUST be protected in transit.
2. An Allocation Token should be single use, meaning it should be unique per object and per allocation operation.
3. An Allocation Token should have a limited life with some form of expiry in the Allocation Token if generated by a trusted 3rd third party, or with a server-side expiry if generated by the server.
4. An Allocation Token should use a strong random value if it is based on an unsigned code.
5. An Allocation Token should leverage digital signatures to confirm its authenticity if generated by a trusted 3rd party.
6. An Allocation Token that is signed XML should be encoded (e.g., base64 [RFC4648]) to mitigate server validation issues.

8. Acknowledgements

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

Special suggestions that have been incorporated into this document were provided by Ben Campbell, Scott Hollenbeck, Benjamin Kaduk, Mirja Kuehlewind, Rubens Kuhl, Alexander Mayrhofer, Patrick Mevzek, Eric Rescoria, and Adam Roach.

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>>.
- [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>>.
- [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

- [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>>.
- [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>>.

Appendix A. Change History

A.1. Change from 00 to 01

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

A.2. Change from 01 to 02

1. Ping update.

A.3. Change from 02 to 03

1. Ping update.

A.4. Change from 03 to 04

1. Updated the authors for the draft.

A.5. Change from 04 to REGEXT 00

1. Changed to regext working group draft by changing draft-gould-allocation-token to draft-ietf-regext-allocation-token.

A.6. Change from REGEXT 00 to REGEXT 01

1. Ping update.

A.7. Change from REGEXT 01 to REGEXT 02

1. Added the Implementation Status section.

A.8. Change from REGEXT 02 to REGEXT 03

1. Changed Neustar author to Kal Feher.

A.9. Change from REGEXT 03 to REGEXT 04

1. Added Neustar implementation to the Implementation Status section.

A.10. Change from REGEXT 04 to REGEXT 05

1. Updates based on feedback from Patrick Mevzek, that include:
 1. Remove "or code" from the Abstract section.
 2. Add a missing "to" in "an allocation token TO one of the EPP..." in the Introduction section.
 3. Reword the "The allocation token is known to the server..." sentence in the Introduction section.
 4. Modify the "The allocation token MAY be returned to an authorized client for passing out-of-band to a client that uses it with an EPP transform command" to clarify who the two separate clients are.
 5. Removed an unneeded ":" from the EPP <transfer> Command and EPP <update> Command sections.

A.11. Change from REGEXT 05 to REGEXT 06

1. Fix description of Neustar gTLD SRS based on feedback from Rubens Kuhl.
2. Updates based on feedback from Alexander Mayrhofer, that include:
 1. Making all references to Allocation Token to use the upper case form.
 2. Revise the language of the abstract to include "for including an Allocation Token in query and transform commands. The Allocation Token is used as a credential that authorizes a client to request the allocation of a specific object from the server, using one of the EPP transform commands..."
 3. Replace the title "EPP <transfer> Command" with "EPP <transfer> Query Command" for section 3.1.3.
 4. Revise the second sentence of the Introduction to "The mapping, ..., supports passing an Allocation Token..."
 5. Change "support" to "require" in the Introduction sentence "It is up to server policy which EPP transform commands and which objects support the Allocation Token."
 6. Add the definition of Allocation to the Introduction.
 7. Removed "transform" from "all of the supported EPP transform commands" in the "Allocation Token" section, since the Allocation Token can be used with the "check" command as well.
 8. Remove the word "same" from "The same <allocationToken:allocationToken> element is used for all..." in the "Allocation Token" section.
 9. Change the description of the use of the 2201 error in the "Allocation Token" section, the "EPP <create> Command" section, the "EPP <transfer> Command" section, and the "EPP <update> Command" section.
 10. Revise "<check> to determine if an object is known to the server..." to "<check> to determine if an object can be provisioned..." and remove "detailed" in the description of the <info> in the "EPP Query Commands" section.
 11. Add missing description of the expected <check> response behavior.
 12. Replaced the example reason "Invalid domain-token pair" with "Allocation Token mismatch".
 13. Replace "information on" with "information associated with" in the "EPP <info> Command" section.
 14. Removed the "that identifies the extension namespace", the ", defined in...", the Allocation Token links from the error response sentences, and the "object referencing the <allocationToken:info> element" in the "EPP <info> Command" section.

15. Added "The authorization is subject to server policy." to the "EPP <info> Command" section.
 16. Replace "or <transfer> response" with "or <transfer> query response" in the "EPP <transfer> Query Command" section.
 17. Replace "create an object" with "create an instance of an object" in the "EPP <create> Command" section.
 18. Revised the sentence to include "the command MUST contain a child <allocationToken:allocationToken> element for the client to be authorized to create and allocate the object" in the "EPP <create> Command" section.
 19. Removed the reference to section 2.1 and the namespace identification text in the "EPP <transfer> Command" section.
 20. Added "The authorization associated with the Allocation Token is in addition to and does not replace the authorization mechanism defined for the object's <transfer> request command." to the "EPP <transfer> Command" section.
 21. Modified the first sentence of the "EPP Extension Registry" section to read "The following registration of the EPP Extension Registry, described in RFC7451, is requested"
 22. Removed support with using the Allocation Token with an empty extension of update (e.g., release command), based on the confusion and lack of known applicability.
3. Updates based on feedback from Scott Hollenbeck, that include:
 1. Revised XML schema to included a minimum length of 1 for the allocationTokenType.
 2. Revised the "IANA Considerations" section to include the registration of the XML schema.
 3. Revised the "Security Considerations" section to include considerations for the definition of the Allocation Tokens.
- A.12. Change from REGEXT 06 to REGEXT 07
1. Updates based on feedback from Patrick Mevzek:
 1. Updated obsoleted RFC 7942 to RFC 7942.
 2. Moved RFC 7451 to an informational reference.
- A.13. Change from REGEXT 07 to REGEXT 08
1. Changed Kal Feher's contact e-mail address.
 2. Changed Neustar's Implementation Status contact e-mail address.
 3. Added the Net::DRI sub-section to the Implementation Status section.

A.14. Change from REGEXT 08 to REGEXT 09

1. Updates based on the AD review by Adam Roach, that include:
 1. In "Abstract", set "query" and "transform" off in some way (e.g., using quotation marks)
 2. In "Conventions Used in This Document", please update to use the boilerplate from RFC 8174.
 3. Remove "allocationToken-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:allocationToken-1.0".
 4. In "Allocation Token", change "The server MUST have the Allocation Token" to "The server MAY have the Allocation Token".
 5. In "EPP <check> Command", change "This extension allow clients" to "This extension allows clients".
 6. Use domains reserved by RFC 2026 for the examples. The example domain "example.tld" was changed to "allocation.example" and the example domain "example2.tld" was changed to "allocation2.example".
 7. In "EPP <info> Command", change "...the server MUST return an EPP error result code of 2303 object referencing the <allocationToken:info> element." to "...the server MUST return an EPP error result code of 2303."
 8. In "EPP <transfer> Query Command", remove "the" before "RFC5730".
 9. In "EPP <transfer> Command", change "If the Allocation Token does not apply to the object..." to "If the Allocation Token is invalid or not required for the object...".
 10. In "XML Namespace", remove the sentence "The following URI assignment is requested of IANA:"
 11. In "Security Considerations", change "An Allocation Token should is" to "An Allocation Token that is". Also informatively cite RFC 4648 for the base64 reference.
2. Change "ietf:params:xml:ns:allocationToken-1.0" to "ietf:params:xml:schema:allocationToken-1.0" for the XML schema IANA registration.

A.15. Change from REGEXT 09 to REGEXT 10

1. Changed "auhorization" to "authorization" in the "EPP <info> Command" section.
2. Added 'If an object does not require an Allocation Token, the server MAY return the availability status as available (e.g., "avail" attribute is "1" or "true").' to the check response cases, based on feedback by Mirja Kuehlewind.
3. Changed the definition of the <info> element in the XML schema to only allow an empty element, based on IANA's expert review.

4. Added normative language to the storage and transport of the Allocation Token, in the "Security Considerations" section, based on feedback from Eric Rescoria.
 5. Changed "The definition of the Allocation Token is defined outside of this mapping" to "The definition of the Allocation Token SHOULD be defined outside of this mapping", in the "Security Considerations" section, based on feedback from Eric Rescoria.
 6. Added the missing "urn:" prefix with the IANA URI registrations.
 7. The URL for the BCP 14 was removed based on feedback from Alissa Cooper.
 8. Updates based on review by Benjamin Kaduk, that include:
 1. Added the second paragraph to the "Allocation Token" section to describe the difference (motivation) of using the Allocation Token versus the EPP RFC authorization mechanism.
 2. Added a paragraph to the "Conventions Used in This Document" section for the use of the "abc123" token value and the use of domain object "2fooBAR" password value in the examples.
 3. Changed the "A client MUST pass an Allocation Token known to the server to be authorized to use one of the supported EPP transform commands." sentence in the "Introduction" section to "Clients pass an Allocation Token to the server for validation, and the server determines if the supplied Allocation Token is one supported by the server."
 4. Changed the "Indentation and white space in the examples are provided only to illustrate element relationships and are not REQUIRED in the protocol." sentence in the "Conventions Used in This Document" section to "Indentation and white space in the examples are provided only to illustrate element relationships and are not REQUIRED in the protocol."
 5. Changed the "Authorized clients MAY retrieve..." sentence in the "EPP <info> Command" section.
 6. Changed the "If the query was successful..." sentence in the "EPP <info> Command" section.
 7. Added "supplied" to the "If the supplied Allocation Token passed..." sentence in the "Allocation Token" section.
 8. Removed an extra newline in the <annotation> element in the "Allocation Token Extension Schema" section.
- A.16. Change from REGEXT 10 to REGEXT 11
1. Removed the old duplicate "Authorized clients MAY retrieve..." sentence from section 3.1.2 "EPP <info> Command".

A.17. Change from REGEXT 11 to REGEXT 12

1. Revised the example <check> domain response to first include the positive case for allocation.example, and to second include the negative case for allocation2.example, based on feedback from Ben Campbell. The caption was revised for the example to include the text ", where the Allocation Token 'abc123' matches allocation.example but does not match allocation2.example".

Authors' Addresses

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

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

Kal Feher
Neustar
lvl 8/10 Queens Road
Melbourne, VIC 3004
AU

Email: ietf@feherfamily.org
URI: <http://www.neustar.biz>

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.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	3
2. Terminology	4
3. Definitions	5
4. Overview	5
5. Requirement for Bundling Registration of Names	5
6. Object Attributes	6
6.1. RDN	6
6.2. BDN	7
7. EPP Command Mapping	7
7.1. EPP Query Commands	7
7.1.1. EPP <check> Command	7
7.1.2. EPP <info> Command	8
7.1.3. EPP <transfer> Query Command	10
7.2. EPP Transform Commands	10
7.2.1. EPP <create> Command	11
7.2.2. EPP <delete> Command	13
7.2.3. EPP <renew> Command	14
7.2.4. EPP <transfer> Command	15
7.2.5. EPP <update> Command	16
8. Formal Syntax	17
9. Internationalization Considerations	19
10. IANA Considerations	19
11. Security Considerations	20
12. Implementation Status	21
13. Acknowledgements	21
14. Change History	21
14.1. draft-ietf-regext-bundle-registration: Version 00	21
14.2. draft-ietf-regext-bundle-registration: Version 01	21
14.3. draft-ietf-regext-bundle-registration: Version 02	22
14.4. draft-ietf-regext-bundle-registration: Version 03	22
14.5. draft-ietf-regext-bundle-registration: Version 04	22
14.6. draft-ietf-regext-bundle-registration: Version 05	22
14.7. draft-ietf-regext-bundle-registration: Version 06	22
14.8. draft-ietf-regext-bundle-registration: Version 07	22
14.9. draft-ietf-regext-bundle-registration: Version 08	22
14.10. draft-ietf-regext-bundle-registration: Version 09	22
14.11. draft-ietf-regext-bundle-registration: Version 10	22
14.12. draft-ietf-regext-bundle-registration: Version 11	23

15. References	23
15.1. Normative References	23
15.2. Informative References	24
Authors' Addresses	24

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:

- o Registered Domain Name (RDN), represents the valid domain name that users submitted for the initial registration.
- o Bundled Domain Name (BDN), represents the bundled domain name produced according to the bundled domain name registration policy.

4. Overview

Domain registries have traditionally adopted a registration model whereby metadata relating to a domain name, such as its expiration date and sponsoring registrar, are stored as properties of the domain object. The domain object is then considered an atomic unit of registration, on which operations such as update, renewal and deletion may be performed.

Bundled names brought about the need for multiple domain names to be registered and managed as a single package. In this model, the registry typically accepts a domain registration request (i.e. EPP domain <create> command) containing the domain name to be registered. This domain name is referred to as the RDN in this document. As part of the processing of the registration request, the registry generates a set of bundled names that are related to the RDN, either programmatically or with the guidance of registration policies, and 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:

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

- o Applicable grace periods (Add Grace Period, Renewal Grace Period, Auto-Renewal Grace Period, Transfer Grace Period, and Redemption Grace Period)

Because the domain names are bundled and share the same parameters or attributes, the EPP command should do some processing for these requirements:

- o When performing a domain check, either BDN or RDN can be queried for the EPP command, and will return the same response.
- o When performing a domain info, either BDN or RDN can be queried, the same response will include both BDN and RDN information with the same attributes.
- o When performing a domain Create, either 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--fsqz4la.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:

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

The above elements contain the following attribute:

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

Example <info> response for an authorized client:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>xn--fsq270a.example</domain:name>
S:        <domain:roid>58812678-domain</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>123</domain:registrant>
S:        <domain:contact type="admin">123</domain:contact>
S:        <domain:contact type="tech">123</domain:contact>
S:        <domain:ns>
S:          <domain:hostObj>ns1.example.cn
S:        </domain:hostObj>
S:      </domain:ns>
S:      <domain:clID>ClientX</domain:clID>
S:      <domain:crID>ClientY</domain:crID>
S:      <domain:crDate>2011-04-03T22:00:00.0Z
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 these 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:

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

The above elements contain the following attribute:

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

7.2.1.1. 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 assign the following two URIs.

Registration request for the IDN namespace:

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

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

Registration request for the IDN XML schema:

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

- o Name of Extension: "Domain Name Mapping Extension for Strict Bundling Registration"
- o Document status: Informational
- o Reference: This document
- o Registrant Name and Email Address: IESG, iesg@ietf.org
- o Top-Level Domains (TLDs): Any
- o IPR Disclosure: <https://datatracker.ietf.org/ipr/>
- o Status: Active
- o 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.

- o 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.
- o CNNIC and TELEINFO have implemented this extension in their EPP based Chinese domain name registration system.
- o Public Interest Registry, has requested to implement technical bundling of second level domains for .NGO and .ONG. This means that by registering and purchasing a domain in the .ngo TLD, for an example, the NGO registrant is also registering and purchasing the corresponding name in the .ong TLD (and vice-versa for registrations in .ong).
- o 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

- o accepted as WG document.

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

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

- 14.3. draft-ietf-regext-bundle-registration: Version 02
 - o Update the section of implementation status.
- 14.4. draft-ietf-regext-bundle-registration: Version 03
 - o This document is changed to informational category.
 - o Refine the text.
- 14.5. draft-ietf-regext-bundle-registration: Version 04
 - o Update the implementation section.
 - o Refine the text.
- 14.6. draft-ietf-regext-bundle-registration: Version 05
 - o Scope the XML namespaces to include 'epp'.
- 14.7. draft-ietf-regext-bundle-registration: Version 06
 - o add some examples for the transfer, update and renew command
 - o add some text to security consideration
- 14.8. draft-ietf-regext-bundle-registration: Version 07
 - o Update IANA consideration section based on Scott's comments
 - o Update security consideration based on Chair and Patrick Mevzek's comments
- 14.9. draft-ietf-regext-bundle-registration: Version 08
 - o Refine some texts.
- 14.10. draft-ietf-regext-bundle-registration: Version 09
 - o Refine the texts.
- 14.11. draft-ietf-regext-bundle-registration: Version 10
 - o Update the texts based on IETF LC.

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

- o 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 First Edition REC-xml-20040204", February 2004, <<http://www.w3.org/TR/2004/REC-xml-20040204>>.

[W3C.REC-xmlschema-1-20041028]

Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028", October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>>.

[W3C.REC-xmlschema-2-20041028]

Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028", October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>>.

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: July 8, 2019

J. Gould
VeriSign, Inc.
K. Feher
Neustar
January 4, 2019

Change Poll Extension for the Extensible Provisioning Protocol (EPP)
draft-ietf-regext-change-poll-12

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 8, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Conventions Used in This Document	3
2. Object Attributes	4
2.1. Operation	4
2.2. State	5
2.3. Who	5
2.4. Dates and Times	6
3. EPP Command Mapping	6
3.1. EPP Query Commands	6
3.1.1. EPP <check> Command	6
3.1.2. EPP <info> Command	6
3.1.3. EPP <transfer> Command	16
3.2. EPP Transform Commands	16
3.2.1. EPP <create> Command	16
3.2.2. EPP <delete> Command	16
3.2.3. EPP <renew> Command	16
3.2.4. EPP <transfer> Command	16
3.2.5. EPP <update> Command	16
4. Formal Syntax	16
4.1. Change Poll Extension Schema	17
5. IANA Considerations	19
5.1. XML Namespace	19
5.2. EPP Extension Registry	20
6. Implementation Status	20
6.1. Verisign EPP SDK	21
6.2. Verisign Consolidated Top Level Domain (CTLD) SRS	21
6.3. Verisign .COM / .NET SRS	22
6.4. Neustar EPP SDK	22
7. Security Considerations	22
8. Acknowledgements	22
9. References	23
9.1. Normative References	23
9.2. Informative References	24
Appendix A. Change History	24
A.1. Change from 00 to 01	24
A.2. Change from 01 to 02	24

A.3.	Change from 02 to 03	24
A.4.	Change from 03 to 04	24
A.5.	Change from 04 to 05	24
A.6.	Change from 05 to REGEXT 00	24
A.7.	Change from REGEXT 00 to REGEXT 01	24
A.8.	Change from REGEXT 01 to REGEXT 02	25
A.9.	Change from REGEXT 02 to REGEXT 03	25
A.10.	Change from REGEXT 03 to REGEXT 04	25
A.11.	Change from REGEXT 04 to REGEXT 05	25
A.12.	Change from REGEXT 05 to REGEXT 06	25
A.13.	Change from REGEXT 06 to REGEXT 07	25
A.14.	Change from REGEXT 07 to REGEXT 08	26
A.15.	Change from REGEXT 08 to REGEXT 09	26
A.16.	Change from REGEXT 09 to REGEXT 10	26
A.17.	Change from REGEXT 10 to REGEXT 11	27
A.18.	Change from REGEXT 11 to REGEXT 12	27
Authors'	Addresses	27

1. Introduction

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

1.1. Conventions Used in This Document

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.

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 XML namespace prefix "changePoll" is used for the namespace "urn:ietf:params:xml:ns:changePoll-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 EPP object mappings like the EPP domain name mapping [RFC5731]. Only those new elements are described here.

2.1. Operation

An operation consists of any transform operation that impacts objects that the client sponsors and should be notified of. The <changePoll:operation> element defines the operation. The OPTIONAL "op" attribute is an identifier, represented in the 7-bit US-ASCII character set defined in [RFC0020], that is used to define a sub-operation or the name of a "custom" operation. The enumerated list of <changePoll:operation> values is:

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

"custom": Custom operation that MUST set the "op" attribute with the custom operation name. The custom operations supported is up to server policy.

2.2. State

The state attribute reflects the state of the object "before" or "after" the operation. The state is defined using the OPTIONAL "state" attribute of the <changePoll:changeData> element, with the possible values "before" or "after" and with a default value of "after". The server MAY support both the "before" state and the "after" state of the operation, by using one poll message for the "before" state and one poll message for the "after" state. The "before" state poll message MUST be inserted into the message queue prior to the "after" state poll message.

For operations in Section 2.1 that don't have an "after" state, the server MUST use the "before" state poll message. For example, for the "delete" operation with the "op" attribute set to "purge", or the "autoPurge" operation, the server includes the state of the object prior to being purged in the "before" state poll message.

For operations in Section 2.1 that don't have a "before" state, the server MUST use the "after" state poll message. For example, for the "create" operation, the server includes the state of the object after creation in the "after" state poll message.

2.3. Who

The <changePoll:who> element defines who executed the operation for audit purposes. It is a freeform value that is strictly meant for audit purposes and not meant to drive client-side logic. The scheme used for the possible set of <changePoll:who> element values is up to server policy. The server MAY identify the <changePoll:who> element value based on:

"Identifier": Unique user identifier of the user that executed the operation. An example is "ClientX".

"Name": Name of the user that executed the operation. An example is "John Doe".

"Role": Role of the user that executed operation. An example is "CSR" for a Customer Support Representative or "Batch" for a server batch.

2.4. Dates and Times

Date and time attribute values MUST be represented in Universal Coordinated Time (UTC) using the Gregorian calendar. The extended date-time form using upper case "T" and "Z" characters defined in [W3C.REC-xmlschema-2-20041028] MUST be used to represent date-time values, as XML Schema does not support truncated date-time forms or lower case "T" and "Z" characters.

3. EPP Command Mapping

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

3.1. EPP Query Commands

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

3.1.1. EPP <check> Command

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

3.1.2. EPP <info> Command

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

This extension adds operation detail of EPP object mapping operations Section 2.1 to an EPP poll response, as described in [RFC5730]. The extension is an extension of the EPP object mapping info response. Any transform operation to an object defined in an EPP object mapping by a client other than the sponsoring client MAY result in extending the <info> response of the object for inserting an EPP poll message with the operation detail. The sponsoring client will then receive the state of the object with operation detail like what, who, when, and why the object was changed. The <changePoll:changeData> element contains the operation detail along with an indication of whether the object reflects the state before or after the operation as defined in Section 2.2. The <changePoll:changeData> element includes the operation detail with the following child elements:

<changePoll:operation>: Transform operation executed on the object as defined in Section 2.1.

<changePoll:date>: Date and time when the operation was executed.

<changePoll:svTRID>: Server transaction identifier of the operation.
<changePoll:who>: Who executed the operation as defined in
Section 2.3.

<changePoll:caseId>: OPTIONAL case identifier associated with the operation. The required "type" attribute defines the type of case. The OPTIONAL "name" attribute is an identifier, represented in the 7-bit US-ASCII character set defined in [RFC0020], that is used to define the name of the "custom" case type. The enumerated list of case types is:

udrp: a Uniform Domain-Name Dispute-Resolution Policy (UDRP) case.

urs: a Uniform Rapid Suspension (URS) case.

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

<changePoll:reason>: OPTIONAL reason for executing the operation. If present, this element contains the server-specific text to help explain the reason the operation was executed. This text MUST be represented in the response language previously negotiated with the client; an OPTIONAL "lang" attribute MAY be present to identify the language if the negotiated value is something other than the default value of "en" (English).

Example poll <info> response with the <changePoll:changeData> extension for a URS lock transaction on the domain.example domain name, with the "before" state. The "before" state is reflected in the <resData> block:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg lang="en-US">
S:        Command completed successfully; ack to dequeue</msg>
S:      </result>
S:    <msgQ id="201" count="1">
S:      <qDate>2013-10-22T14:25:57.0Z</qDate>
S:      <msg>Registry initiated update of domain.</msg>
S:    </msgQ>
S:  <resData>
S:    <domain:infData
S:      xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:      <domain:name>domain.example</domain:name>
S:      <domain:roid>EXAMPLE1-REP</domain:roid>
S:      <domain:status s="ok"/>
S:      <domain:registrant>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:exDate>2014-04-03T22:00:00.0Z</domain:exDate>
S:    </domain:infData>
S:  </resData>
S:  <extension>
S:    <changePoll:changeData
S:      xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0"
S:      state="before">
S:      <changePoll:operation>update</changePoll:operation>
S:      <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
S:      <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:      <changePoll:who>URS Admin</changePoll:who>
S:      <changePoll:caseId type="urs">urs123</changePoll:caseId>
S:      <changePoll:reason>URS Lock</changePoll:reason>
S:    </changePoll:changeData>
S:  </extension>
S:  <trID>
S:    <clTRID>ABC-12345</clTRID>
S:    <svTRID>54321-XYZ</svTRID>
S:  </trID>
S: </response>
S:</epp>
```

Example poll <info> response with the <changePoll:changeData> extension for a URS lock transaction on the domain.example domain name, with the "after" state. The "after" state is reflected in the

<resData> block:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg lang="en-US">
S:        Command completed successfully; ack to dequeue</msg>
S:      </result>
S:    <msgQ id="202" count="1">
S:      <qDate>2013-10-22T14:25:57.0Z</qDate>
S:      <msg>Registry initiated update of domain.</msg>
S:    </msgQ>
S:  <resData>
S:    <domain:infData
S:      xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:      <domain:name>domain.example</domain:name>
S:      <domain:roid>EXAMPLE1-REP</domain:roid>
S:      <domain:status s="serverUpdateProhibited"/>
S:      <domain:status s="serverDeleteProhibited"/>
S:      <domain:status s="serverTransferProhibited"/>
S:      <domain:registrant>jdl234</domain:registrant>
S:      <domain:contact type="admin">sh8013</domain:contact>
S:      <domain:contact type="tech">sh8013</domain:contact>
S:      <domain:clID>ClientX</domain:clID>
S:      <domain:crID>ClientY</domain:crID>
S:      <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
S:      <domain:upID>ClientZ</domain:upID>
S:      <domain:upDate>2013-10-22T14:25:57.0Z</domain:upDate>
S:      <domain:exDate>2014-04-03T22:00:00.0Z</domain:exDate>
S:    </domain:infData>
S:  </resData>
S:  <extension>
S:    <changePoll:changeData
S:      xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0"
S:      state="after">
S:      <changePoll:operation>update</changePoll:operation>
S:      <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
S:      <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:      <changePoll:who>URS Admin</changePoll:who>
S:      <changePoll:caseId type="urs">urs123</changePoll:caseId>
S:      <changePoll:reason>URS Lock</changePoll:reason>
S:    </changePoll:changeData>
S:  </extension>
S:  <trID>
S:    <clTRID>ABC-12345</clTRID>
S:    <svTRID>54321-XYZ</svTRID>
S:  </trID>
S: </response>
S:</epp>
```

Example poll <info> response with the <changePoll:changeData> extension for a custom "sync" operation on the domain.example domain name, with the default "after" state. The "after" state is reflected in the <resData> block:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:    </result>
S:    <msgQ id="201" count="1">
S:      <qDate>2013-10-22T14:25:57.0Z</qDate>
S:    <msg>Registry initiated Sync of Domain Expiration Date</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:upID>ClientZ</domain:upID>
S:        <domain:upDate>2013-10-22T14:25:57.0Z</domain:upDate>
S:        <domain:exDate>2014-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <changePoll:changeData
S:        xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0">
S:        <changePoll:operation op="sync">custom
S:      </changePoll:operation>
S:      <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
S:      <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:      <changePoll:who>CSR</changePoll:who>
S:      <changePoll:reason lang="en">Customer sync request
S:    </changePoll:reason>
S:    </changePoll:changeData>
S:  </extension>
S:  <trID>
S:    <clTRID>ABC-12345</clTRID>
S:    <svTRID>54321-XYZ</svTRID>
S:  </trID>
S: </response>
S:</epp>
```

Example poll <info> response with the <changePoll:changeData> extension for a "delete" operation on the domain.example domain name that is immediately purged, with the "before" state. The "before" state is reflected in the <resData> block:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:    </result>
S:    <msgQ id="200" count="1">
S:      <qDate>2013-10-22T14:25:57.0Z</qDate>
S:      <msg>Registry initiated delete of
S:        domain resulting in immediate purge.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:clID>ClientX</domain:clID>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <changePoll:changeData
S:        xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0"
S:        state="before">
S:        <changePoll:operation op="purge">delete
S:        </changePoll:operation>
S:        <changePoll:date>2013-10-22T14:25:57.0Z
S:        </changePoll:date>
S:        <changePoll:svTRID>12345-XYZ
S:        </changePoll:svTRID>
S:        <changePoll:who>ClientZ
S:        </changePoll:who>
S:        <changePoll:reason>Court order
S:        </changePoll:reason>
S:      </changePoll:changeData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```


Example poll <info> response with the <changePoll:changeData> extension for an "autoPurge" operation on the domain.example domain name that previously had the "pendingDelete" status, with the "before" state. The "before" state is reflected in the <resData> block:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue
S:    </msg>
S:    </result>
S:    <msgQ id="200" count="1">
S:      <qDate>2013-10-22T14:25:57.0Z</qDate>
S:      <msg>Registry purged domain with pendingDelete status.
S:    </msg>
S:    </msgQ>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:clID>ClientX</domain:clID>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <changePoll:changeData
S:        xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0"
S:        state="before">
S:        <changePoll:operation>autoPurge
S:      </changePoll:operation>
S:      <changePoll:date>2013-10-22T14:25:57.0Z
S:    </changePoll:date>
S:    <changePoll:svTRID>12345-XYZ
S:  </changePoll:svTRID>
S:    <changePoll:who>Batch
S:  </changePoll:who>
S:    <changePoll:reason>Past pendingDelete 5 day period
S:  </changePoll:reason>
S:    </changePoll:changeData>
S:  </extension>
S:  <trID>
S:    <clTRID>ABC-12345</clTRID>
S:    <svTRID>54321-XYZ</svTRID>
S:  </trID>
S: </response>
S:</epp>
```

Example poll <info> response with the <changePoll:changeData> extension for an "update" operation on the ns1.domain.example host, with the default "after" state. The "after" state is reflected in the <resData> block:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:    </result>
S:    <msgQ id="201" count="1">
S:      <qDate>2013-10-22T14:25:57.0Z</qDate>
S:      <msg>Registry initiated update of host.</msg>
S:    </msgQ>
S:    <resData>
S:      <host:infData
S:        xmlns:host="urn:ietf:params:xml:ns:host-1.0">
S:        <host:name>ns1.domain.example</host:name>
S:        <host:roid>NS1_EXAMPLE1-REP</host:roid>
S:        <host:status s="linked"/>
S:        <host:status s="serverUpdateProhibited"/>
S:        <host:status s="serverDeleteProhibited"/>
S:        <host:addr ip="v4">192.0.2.2</host:addr>
S:        <host:addr ip="v6">2001:db8:0:0:1:0:0:1</host:addr>
S:        <host:clID>ClientX</host:clID>
S:        <host:crID>ClientY</host:crID>
S:        <host:crDate>2012-04-03T22:00:00.0Z</host:crDate>
S:        <host:upID>ClientY</host:upID>
S:        <host:upDate>2013-10-22T14:25:57.0Z</host:upDate>
S:      </host:infData>
S:    </resData>
S:    <extension>
S:      <changePoll:changeData
S:        xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0">
S:        <changePoll:operation>update</changePoll:operation>
S:        <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
S:        <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:        <changePoll:who>ClientZ</changePoll:who>
S:        <changePoll:reason>Host Lock</changePoll:reason>
S:      </changePoll:changeData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.1.3. EPP <transfer> Command

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

3.2. EPP Transform Commands

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

3.2.1. EPP <create> Command

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

3.2.2. EPP <delete> Command

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

3.2.3. EPP <renew> Command

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

3.2.4. EPP <transfer> Command

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

3.2.5. EPP <update> Command

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

4. Formal Syntax

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

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

4.1. Change Poll Extension Schema

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

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

    <annotation>
      <documentation>
        Extensible Provisioning Protocol v1.0
        Change Poll Mapping Schema.
      </documentation>
    </annotation>

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

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

```
<!--
  Enumerated list of operations, with extensibility via "custom".
-->
<simpleType name="operationEnum">
  <restriction base="token">
    <enumeration value="create"/>
    <enumeration value="delete"/>
    <enumeration value="renew"/>
    <enumeration value="transfer"/>
    <enumeration value="update"/>
    <enumeration value="restore"/>
    <enumeration value="autoRenew"/>
    <enumeration value="autoDelete"/>
    <enumeration value="autoPurge"/>
    <enumeration value="custom"/>
  </restriction>
</simpleType>

<!--
  Enumerated of state of the object in the poll message.
-->
<simpleType name="stateType">
  <restriction base="token">
    <enumeration value="before"/>
    <enumeration value="after"/>
  </restriction>
</simpleType>

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

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

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

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

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

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

5. IANA Considerations

5.1. XML Namespace

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

Registration request for the changePoll namespace:

URI: urn:ietf:params:xml:ns:changePoll-1.0
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.

Registration request for the changePoll XML schema:

URI: urn:ietf:params:xml:ns:changePoll-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: "Change Poll Extension for the Extensible Provisioning Protocol (EPP)"

Document status: Standards Track

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

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

6. 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-change-poll.

Level of maturity: Production

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

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

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-change-poll for a variety of Top Level Domains (TLD's).

Level of maturity: Production

Coverage: The "after" state poll message for an "update" transform operation of a domain name due to server policy.

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 and .NET implements the server-side of draft-ietf-regext-change-poll.

Level of maturity: Production

Coverage: The "after" state poll message for an "update" transform operation of a domain name due to server policy.

Licensing: Proprietary

Contact: jgould@verisign.com

6.4. Neustar EPP SDK

Organisation: Neustar Inc.

Name: Neustar EPP SDK

Description: The Neustar EPP SDK includes a full client implementation of draft-ietf-regext-change-poll.

Level of maturity: Production

Coverage: All client side aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: quoc-anh.np@team.neustar

7. Security Considerations

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

8. Acknowledgements

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

Special suggestions that have been incorporated into this document were provided by Scott Hollenbeck, Michael Holloway, and Patrick Mevzek.

9. References

9.1. Normative References

- [RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <<https://www.rfc-editor.org/info/rfc20>>.
- [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>>.
- [RFC3915] Hollenbeck, S., "Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP)", RFC 3915, DOI 10.17487/RFC3915, September 2004, <<https://www.rfc-editor.org/info/rfc3915>>.
- [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>>.
- [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>>.
- [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>>.

9.2. Informative References

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

Appendix A. Change History

A.1. Change from 00 to 01

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

A.2. Change from 01 to 02

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

A.3. Change from 02 to 03

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

A.4. Change from 03 to 04

1. Updated the authors for the draft.

A.5. Change from 04 to 05

1. Ping update.

A.6. Change from 05 to REGEXT 00

1. Changed to regext working group draft by changing draft-gould-change-poll to draft-ietf-regext-change-poll.

A.7. Change from REGEXT 00 to REGEXT 01

1. Ping update.

A.8. Change from REGEXT 01 to REGEXT 02

1. Added the Implementation Status section.

A.9. Change from REGEXT 02 to REGEXT 03

1. Changed Neustar author to Kal Feher.

A.10. Change from REGEXT 03 to REGEXT 04

1. Added Neustar implementation to the Implementation Status section.

A.11. Change from REGEXT 04 to REGEXT 05

1. Updates based on feedback from Patrick Mevzek, that include:
 1. Added a missing comma to "Using this extension, clients" in the Introduction section.
 2. Modified the description of the "transfer", "restore", and "custom" operations to include "MUST set the "op" attribute" language.
 3. Rephrased the first sentence of the Who section.
 4. Added references to the <changePoll:who> element in the Who section.
 5. Revise the sentence that describes how the extension extends the info response in the EPP <info> Command section.
 6. Refer to EPP Object Mapping as EPP object mapping throughout the document.
 7. Add a Dates and Times section to the Object Attributes section.

A.12. Change from REGEXT 05 to REGEXT 06

1. Added the "State" sub-section to the "Object Attributes" section to describe the expected behavior for the "before" and "after" states, based on feedback from Patrick Mevzek.
2. Added a colon suffix to each hangText entry to provide better separation.

A.13. Change from REGEXT 06 to REGEXT 07

1. Updates based on feedback from Scott Hollenbeck, that include:
 1. Changed MAY to may in the Abstract.
 2. Revised the "IANA Considerations" section to include the registration of the XML schema.

3. Revised the description of the <changePoll:caseId> "name" attribute and the "changePoll:operation" "op" attribute as containing 7-bit US-ASCII identifiers for the case type or the operation type, respectively.

A.14. Change from REGEXT 07 to REGEXT 08

1. Updated obsoleted RFC 6982 to RFC 7942.
2. Moved RFC 7451 to an informational reference based on a check done by the Idnits Tool.
3. Changed Kal Feher's contact e-mail address.
4. Changed Neustar's Implementation Status contact e-mail address.

A.15. Change from REGEXT 08 to REGEXT 09

1. Fixed Section 1.1 (Conventions) to contain the updated language (e.g. "NOT RECOMMENDED", RFC 8174, BCP 14), based on feedback from the Document Shepherd.

A.16. Change from REGEXT 09 to REGEXT 10

1. Updates based on the AD review by Adam Roach, that include:
 1. Fix the "purge" and "autoPurge" examples to use the normative "before" state instead of the default "after" state.
 2. Added the sentences "The extension only extends the EPP <poll> response in [RFC5730] and does not extend the EPP <poll> command. Please refer to [RFC5730] for information and examples of the EPP <poll> command." in the "Introduction" to clarify what is extended and reference [RFC5730] for the EPP <poll> command.
 3. Added missing hyphens to "client-sponsored" and "court-directed".
 4. Removed "changePoll-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:changePoll-1.0" and replaced the paragraph based on what was done in draft-ietf-regext-allocation-token.
 5. Changed normative "SHOULD" to non-normative "should" in "An operation consists of any transform operation that impacts objects that the client sponsors and should be notified of."
 6. Added normative reference to [RFC0020] to define "7-bit US-ASCII".
 7. Added the sentence "The custom operations supported is up to server policy." to the description of the "custom" operation.

8. Broke up the "This extension adds operation detail..." sentence into two separate sentences to address the "does" and the "is" separately.
9. Removed the commas from "Any transform operation to an object..." sentence.
10. Changed to use an IPv6 address from the documentation-only prefix "2001:DB8::/32" in RFC 3849. The IPv6 address 2001:db8:0:0:1:0:0:1 was used.

A.17. Change from REGEXT 10 to REGEXT 11

1. Updates based on the review by Benjamin Kaduk, that include:
 1. Change references of "The enumerated list ... include:" to "The enumerated list ... is:".
 2. In section 2.2, explicitly state what the message is inserted into, with the change of "... MUST be inserted prior to ..." to "... MUST be inserted into the message queue prior to ...".

A.18. Change from REGEXT 11 to REGEXT 12

1. Added clarification for the <changePoll:who> element based on the feedback from Benjamin Kaduk.

Authors' Addresses

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

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

Kal Feher
Neustar
lvl 8/10 Queens Road
Melbourne, VIC 3004
AU

Email: ietf@feherfamily.org
URI: <http://www.neustar.biz>

regext
Internet-Draft
Intended status: Standards Track
Expires: November 5, 2018

J. Latour
CIRA
O. Gudmundsson
Cloudflare, Inc.
P. Wouters
Red Hat
M. Pounsett
Nimbus Operations Inc.
May 4, 2018

Third Party DNS operator to Registrars/Registries Protocol
draft-ietf-regext-dnsoperator-to-rrr-protocol-05

Abstract

There are several problems that arise in the standard Registrant/Registrar/Registry model when the operator of a zone is neither the Registrant nor the Registrar for the delegation. Historically the issues have been minor, and limited to difficulty guiding the Registrant through the initial changes to the NS records for the delegation. As this is usually a one time activity when the operator first takes charge of the zone it has not been treated as a serious issue.

When the domain uses DNSSEC it necessary to make regular (sometimes annual) changes to the delegation, updating DS record(s) in order to track KSK rollover. Under the current model this is prone to delays and errors, as the Registrant must participate in updates to DS records.

This document describes a simple protocol that allows a third party DNS operator to: establish the initial chain of trust (bootstrap DNSSEC) for a delegation; update DS records for a delegation; and, remove DS records from a secure delegation. The DNS operator may do these things in a trusted manner, without involving the Registrant for each operation. This same protocol can be used by Registrants to maintain their own domains if they wish.

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 November 5, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Notional Conventions	4
2.1. Definitions	4
2.2. RFC2119 Keywords	4
3. Process Overview	4
3.1. Identifying the Registration Entity	4
3.2. Establishing a Chain of Trust	5
3.3. Maintaining the Chain of Trust	5
3.4. Acceptance Processing	6
3.5. Bootstrapping DNSSEC	6
4. API Definition	7
4.1. Authentication	7
4.2. RESTful Resources	8
4.2.1. CDS resource	8
4.2.2. Token resource	10
4.3. Customized Error Messages	11
5. Security considerations	11
6. IANA Actions	11
7. Internationalization Considerations	11
8. References	12
8.1. Normative References	12
8.2. Informative References	12
Appendix A. Document History	13
A.1. regext Version 05	13

A.2. regex Version 04	13
A.3. regex Version 03	13
A.4. regex Version 02	14
A.5. regex Version 01	14
A.6. regex Version 00	14
A.7. Version 03	14
A.8. Version 02	14
A.9. Version 01	14
A.10. Version 00	14
Authors' Addresses	14

1. Introduction

After a domain has been registered, one of three parties will maintain the DNS zone loaded on the "primary" DNS servers: the Registrant, the Registrar, or a third party DNS operator. DNS registration systems were originally designed around making registrations easy and fast, however after registration the complexity of making changes to the delegation differs for each of these parties. The Registrar can make changes directly in the Registry systems through some API (typically EPP [RFC5730]). The Registrant is typically limited to using a web interface supplied by the Registrar or Reseller. Typically, a third party DNS Operator must to go through the Registrant to update any delegation information.

Unless the responsible Registration Entity is scanning child zones for CDS records in order to bootstrap or update DNSSEC, the operator must contact and engage the Registrant in updating DS records for the delegation. New information must be communicated to the Registrant, who must submit that information to the Registrar. Typically this involves cutting and pasting between email and a web interface, which is error prone. Furthermore, involving Registrants in this way does not scale for even moderately sized DNS operators. Tracking thousands (or millions) of changes sent to customers, and following up if those changes are not submitted to the Registrar, or are submitted with errors, is itself expensive and error prone.

The current system does not work well, as there are many types of failures that have been reported at all levels in the registration model. The failures result in either the inability to use DNSSEC or in validation failures that cause the domain to become unavailable to users behind validating resolvers.

The goal of this document is to create a protocol for establishing a secure chain of trust that involves parties not in the traditional Registrant/Registrar/Registry (RRR) model, and to reduce the friction in maintaining DNSSEC secured delegations in these cases. It

describes a REST-based [RFC6690] protocol which can be used to establish DNSSEC initial trust (to enable or bootstrap DNSSEC), and to trigger maintenance of DS records.

2. Notional Conventions

2.1. Definitions

For the purposes of this draft, a third-party DNS Operator is any DNS Operator responsible for a zone, where the operator is neither the Registrant nor the Registrar of record for the delegation.

Uses of "child" and "parent" refer to the relationship between DNS zone operators (see [RFC7719] and [I-D.ietf-dnsop-terminology-bis]). In this document, unless otherwise noted, the child is the third-party DNS operator and the parent is the Registry.

Use of the term "Registration Entity" in this document may refer to any party that engages directly in registration activities with the Registrant. Typically this will be a Reseller or Registrar, but in some cases, such as when a Registry directly sells registrations to the public, may apply to the Registry. Even in cases where a Registrar is involved, this term may still apply to a Registry if that Registry normally accepts DS/DNSKEY updates directly from Registrants.

The CDS and CDNSKEY DNS resource records, having substantially the same function but for different record types, are used interchangeably in this document. Unless otherwise noted, any use of "CDS" or "CDNSKEY" can be assumed to also refer to the other.

2.2. RFC2119 Keywords

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. Process Overview

3.1. Identifying the Registration Entity

As of publication of this document, there has never been a standardized or widely deployed method for easily and scalably identifying the Registration Entity for a particular registration.

At this time, WHOIS [RFC3912] is the only widely deployed protocol to carry such information, but WHOIS responses are unstructured text, and each implementor can lay out its text responses differently. In

addition, Registries may include referrals in this unstructured text to the WHOIS interfaces of their Registrars, and those Registrar WHOIS interface in turn have their own layouts. This presents a text parsing problem which is infeasible to solve.

RDAP, the successor to WHOIS, described in [RFC7480], solves the problems of unstructured responses, and a consistently implemented referral system, however at this time RDAP has yet to be deployed at most Registries.

With no current mechanism in place to scalably discover the Registrar for a particular registration, the problem of automatic discovery of the base URL of the API is considered out of scope of this document. The authors recommend standardization of an RDAP extension to obtain this information from the Registry.

3.2. Establishing a Chain of Trust

After signing the zone, the child DNS Operator needs to upload the DS record(s) to the parent. The child can signal its desire to have DNSSEC validation enabled by publishing one of the special DNS records CDS and/or CDNSKEY as defined in [RFC7344] and [RFC8078].

Registration Entities MAY regularly scan the child name servers of unsecured delegations for CDS records in order to bootstrap DNSSEC, and are advised to do so. At the time of publication, some ccTLD Registries are already doing this. A Registration Entity that regularly scans all child zones under its responsibility (both secured and unsecured) for CDS will not require the API described in this document. However, such a Registration Entity should follow the guidelines discussed in Section 3.5 below when using CDS to bootstrap DNSSEC on a previously unsecured delegation.

In the case where the Registration Entity is not normally scanning child zones for CDS records, the Registration Entity SHOULD implement the API from this document, allowing child operators to notify the Registration Entity to begin such a scan.

Once the Registration Entity finds CDS records in a child zone it is responsible for, or receives a signal via this API, it SHOULD start acceptance processing as described below.

3.3. Maintaining the Chain of Trust

Once the secure chain of trust is established, the Registration Entity SHOULD regularly scan the child zone for CDS record changes. If the Registration Entity implements the protocol described in this

document, then it SHOULD also accept signals via this protocol to immediately check the child zone for CDS records.

Server implementations of this protocol MAY include rate limiting to protect their systems and the systems of child operators from abuse.

Each parent operator and Registration Entity is responsible for developing, implementing, and communicating their DNSSEC maintenance policies.

3.4. Acceptance Processing

The Registration Entity, upon receiving a signal or detecting through polling that the child desires to have its delegation updated, SHOULD run a series of tests to ensure that updating the parent zone will not create or exacerbate any problems with the child zone. The basic tests SHOULD include:

- o checks that the child zone is properly signed as per the Registration Entity and parent DNSSEC policies
- o if updating the DS record, a check to ensure the child CDS RRset references a KSK which is present in the child DNSKEY RRset and signs the CDS RRset
- o ensuring all name servers in the apex NS RRset of the child zone agree on the apex NS RRset and CDS RRset contents

The Registration Entity SHOULD NOT make any changes to the DS RRset if the child name servers do not agree on the CDS content.

3.5. Bootstrapping DNSSEC

Registration Entities SHOULD require compliance with additional tests in the case of establishing a new chain of trust.

- o The Registration Entity SHOULD check that all child name servers respond with a consistent CDS RRset for a number of queries over an extended period of time. Any change in DS response or inconsistency between child responses in that time might indicate an attempted Man in the Middle (MITM) attack, and SHOULD reset the test. This minimizes the possibility of an attacker spoofing responses. An example of such a policy might be to scan all child name servers in the delegation NS RRset every two hours for a week.

- o The Registration Entity SHOULD require all of the child name servers in the delegation NS RRset to send the same response to a CDS query whether sent over TCP or UDP.
- o The Registration Entity MAY require the child zone to implement zone delegation best practices as described in [I-D.wallstrom-dnsop-dns-delegation-requirements].
- o The Registration Entity MAY require the child operator to prove they can add data to the zone, for example by publishing a particular token. See Section 4.2.2 below.

4. API Definition

This protocol is partially synchronous, meaning the server can elect to hold connections open until operations have completed, or it can return a status code indicating that it has received a request, and close the connection. It is up to the child to monitor the parent for completion of the operation, and issue possible follow-up calls to the Registration Entity.

Clients may be denied access to change the DS records for domains that are Registry Locked (HTTP Status code 401). Registry Lock is a mechanism provided by certain Registries or Registrars that prevents domain hijacking by ensuring no attributes of the domain are changeable, and no transfer or deletion transactions can be processed against the domain name without manual intervention.

4.1. Authentication

The API does not impose any unique server authentication requirements. The server authentication provided by TLS fully addresses the needs of this protocol. The API MUST be provided over TLS-protected transport (e.g., HTTPS) or VPN.

Client authentication is considered out of scope of this document. The publication of CDS records in the child zone is an indication that the child operator intends to perform DS-record-updating activities (add/delete) in the parent zone. Since this protocol is simply a signal to the Registration Entity that they should examine the child zone for such intentions, additional authentication of the client making the request is considered unnecessary.

Registration Entities MAY implement their own policy to protect access to the API, such as with IP white listing, client TLS certificates, etc.. Registration Entities SHOULD take steps to ensure that a lack of additional authentication does not open up a

denial of service mechanism against the systems of the Registration Entity, the Registry, or the child operator.

4.2. RESTful Resources

In the following text, "{domain}" is the child zone to be operated on.

4.2.1. CDS resource

Path: /domains/{domain}/cds

4.2.1.1. Establishing Initial Trust (Enabling DNSSEC)

4.2.1.1.1. Request

Syntax: POST /domains/{domain}/cds

Request that an initial set of DS records based on the CDS record in the child zone be inserted into the Registry and the parent zone upon the successful completion of the request. If there are multiple CDS records in the CDS RRset, multiple DS records will be added.

The body of the POST SHOULD be empty, however server implementations SHOULD NOT reject nonempty requests.

4.2.1.1.2. Response

- o HTTP Status code 201 indicates a success.
- o HTTP Status code 400 indicates a failure due to validation.
- o HTTP Status code 401 indicates an unauthorized resource access.
- o HTTP Status code 403 indicates a failure due to an invalid challenge token.
- o HTTP Status code 404 indicates the domain does not exist.
- o HTTP Status code 409 indicates the delegation already has a DS RRset.
- o HTTP Status code 429 indicates the client has been rate-limited.
- o HTTP Status code 500 indicates a failure due to unforeseeable reasons.

This request is for setting up initial trust in the delegation. The Registration Entity SHOULD return a status code 409 if it already has a DS RRset for the child zone.

Upon receipt of a 403 response the child operator SHOULD issue a POST for the "token" resource to fetch a challenge token to insert into the zone.

4.2.1.2. Removing DS Records

4.2.1.2.1. Request

Syntax: DELETE /domains/{domain}/cds

Request that the Registration Entity check for a null CDS or CDNSKEY record in the child zone, indicating a request that the entire DS RRset be removed. This will make the delegation insecure.

4.2.1.2.2. Response

- o HTTP Status code 200 indicates a success.
- o HTTP Status code 400 indicates a failure due to validation.
- o HTTP Status code 401 indicates an unauthorized resource access.
- o HTTP Status code 404 indicates the domain does not exist.
- o HTTP Status code 412 indicates the parent does not have a DS RRset
- o HTTP Status code 429 indicates the client has been rate-limited.
- o HTTP Status code 500 indicates a failure due to unforeseeable reasons.

4.2.1.3. Modifying DS Records

4.2.1.3.1. Request

Syntax: PUT /domains/{domain}/cds

Request that the Registration Entity modify the DS RRset based on the CDS/CDNSKEY available in the child zone. As a result of this request the Registration Entity SHOULD add or delete DS or DNSKEY records as indicated by the CDS/CDNSKEY RRset, but MUST NOT delete the entire DS RRset.

4.2.1.3.2. Response

- o HTTP Status code 200 indicates a success.
- o HTTP Status code 400 indicates a failure due to validation.
- o HTTP Status code 401 indicates an unauthorized resource access.
- o HTTP Status code 404 indicates the domain does not exist.
- o HTTP Status code 412 indicates the parent does not have a DS RRset
- o HTTP Status code 429 indicates the client has been rate-limited.
- o HTTP Status code 500 indicates a failure due to unforeseeable reasons.

4.2.2. Token resource

Path: /domains/{domain}/token

4.2.2.1. Establish Initial Trust with Challenge

4.2.2.1.1. Request

Syntax: GET /domains/{domain}/token

The DNSSEC policy of the Registration Entity may require proof that the DNS Operator is in control of the domain. The token API call returns a random token to be included as a TXT record for the `_delegate.@` domain name (where `@` is the apex of the child zone) prior establishing the DNSSEC initial trust. This is an additional trust control mechanism to establish the initial chain of trust.

Once the child operator has received a token, it SHOULD be inserted in the zone and the operator SHOULD proceed with a POST of the cds resource.

The Registration Entity MAY expire the token after a reasonable period. The Registration Entity SHOULD document an explanation of whether and when tokens are expired in their DNSSEC policy.

Note that the `_delegate` TXT record is publicly available and not a secret token.

4.2.2.1.2. Response

- o HTTP Status code 200 indicates a success. A token is included in the body of the response, as a valid TXT record
- o HTTP Status code 404 indicates the domain does not exist.
- o HTTP Status code 500 indicates a failure due to unforeseeable reasons.

4.3. Customized Error Messages

Registration Entities MAY provide a customized error message in the response body in addition to the HTTP status code defined in the previous section. This response MAY include an identifying number/string that can be used to track the request.

5. Security considerations

When zones are properly provisioned, and delegations follow standards and best practices (e.g. [I-D.wallstrom-dnsop-dns-delegation-requirements]), the Registration Entity or Registry can trust the DNS information it receives from multiple child name servers, over time, and/or over TCP to establish the initial chain of trust.

In addition, the Registration Entity or Registry can require the DNS Operator to prove they control the zone by requiring the child operator to navigate additional hurdles, such as adding a challenge token to the zone.

This protocol should increase the adoption of DNSSEC, enabling more zones to become validated thus overall the security gain outweighs the possible drawbacks.

Registrants and DNS Operators always have the option to establish the chain of trust in band via the standard Registrant/Registrar/Registry model.

6. IANA Actions

This document has no actions for IANA

7. Internationalization Considerations

This protocol is designed for machine to machine communications. Clients and servers SHOULD use punycode [RFC3492] when operating on internationalized domain names.

8. References

8.1. Normative References

- [RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, DOI 10.17487/RFC3492, March 2003, <<https://www.rfc-editor.org/info/rfc3492>>.
- [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <<https://www.rfc-editor.org/info/rfc6690>>.
- [RFC7344] Kumari, W., Gudmundsson, O., and G. Barwood, "Automating DNSSEC Delegation Trust Maintenance", RFC 7344, DOI 10.17487/RFC7344, September 2014, <<https://www.rfc-editor.org/info/rfc7344>>.
- [RFC8078] Gudmundsson, O. and P. Wouters, "Managing DS Records from the Parent via CDS/CDNSKEY", RFC 8078, DOI 10.17487/RFC8078, March 2017, <<https://www.rfc-editor.org/info/rfc8078>>.

8.2. Informative References

- [I-D.ietf-dnsop-terminology-bis]
Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", draft-ietf-dnsop-terminology-bis-10 (work in progress), April 2018.
- [I-D.wallstrom-dnsop-dns-delegation-requirements]
Wallstrom, P. and J. Schlyter, "DNS Delegation Requirements", draft-wallstrom-dnsop-dns-delegation-requirements-03 (work in progress), October 2016.
- [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>>.
- [RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, DOI 10.17487/RFC3912, September 2004, <<https://www.rfc-editor.org/info/rfc3912>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<https://www.rfc-editor.org/info/rfc5730>>.

- [RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", RFC 7480, DOI 10.17487/RFC7480, March 2015, <<https://www.rfc-editor.org/info/rfc7480>>.
- [RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <<https://www.rfc-editor.org/info/rfc7719>>.

Appendix A. Document History

A.1. regext Version 05

- o new version to keep the draft alive
- o updating author organization

A.2. regext Version 04

- o changed uses of Registrar to Registration Entity and updated definitions to improve clarity
- o adding note about CDS/CDNSKEY interchangeability in this document
- o added advice to scan all delegations (including insecure delegations) for CDS in order to bootstrap or update DNSSEC
- o removed "Other Delegation Maintenance" section, since we decided a while ago not to use this to update NS

A.3. regext Version 03

- o simplify abstract
- o move all justification text to Intro
- o added HTTP response codes for rate limiting (429), missing DS RRsets (412)
- o expanded on Internationalization Considerations
- o corrected informative/normative document references
- o clarify parent/Registrar references in the draft
- o general spelling/grammar/style cleanup
- o removed references to NS and glue maintenance

- o clarify content of POST body for 'cds' resource
- o change verb for obtaining a 'token' to GET
- o Updated reference to RFC8078

A.4. regext Version 02

- o Clarified based on comments and questions from early implementors (JL)
- o Text edits and clarifications.

A.5. regext Version 01

- o Rewrote Abstract and Into (MP)
- o Introduced code 401 when changes are not allowed
- o Text edits and clarifications.

A.6. regext Version 00

- o Working group document same as 03, just track changed to standard

A.7. Version 03

- o Clarified based on comments and questions from early implementors

A.8. Version 02

- o Reflected comments on mailing lists

A.9. Version 01

- o This version adds a full REST definition this is based on suggestions from Jakob Schlyter.

A.10. Version 00

- o First rough version

Authors' Addresses

Jacques Latour
CIRA
Ottawa, ON

Email: jacques.latour@cira.ca

Olafur Gudmundsson
Cloudflare, Inc.
San Francisco, CA

Email: olafur+ietf@cloudflare.com

Paul Wouters
Red Hat
Toronto, ON

Email: paul@nohats.ca

Matthew Pounsett
Nimbus Operations Inc.
Toronto, ON

Email: matt@conundrum.com

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2020

R. Carney
GoDaddy Inc.
G. Brown
CentralNic Group plc
J. Frakes
October 21, 2019

Registry Fee Extension for the Extensible Provisioning Protocol (EPP)
draft-ietf-regext-epp-fees-20

Abstract

Given the expansion of the DNS namespace, and the proliferation of novel business models, it is desirable to provide a method for Extensible Provisioning Protocol (EPP) clients to query EPP servers for the fees and credits and provide expected fees and credits for certain commands and objects. This document describes an 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 <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 23, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Conventions Used in This Document	3
2. Migrating to Newer Versions of This Extension	4
3. Extension Elements	4
3.1. Client Commands	4
3.2. Currency Codes	5
3.3. Validity Periods	5
3.4. Fees and Credits	6
3.4.1. Refunds	7
3.4.2. Grace Periods	7
3.4.3. Correlation between Refundability and Grace Periods	7
3.4.4. Applicability	7
3.5. Account Balance	8
3.6. Credit Limit	8
3.7. Classification of Objects	9
3.8. Phase and Subphase Attributes	9
3.9. Reason	10
4. Server Handling of Fee Information	11
5. EPP Command Mapping	12
5.1. EPP Query Commands	12
5.1.1. EPP <check> Command	12
5.1.2. EPP Transfer Query Command	16
5.2. EPP Transform Commands	18
5.2.1. EPP <create> Command	18
5.2.2. EPP <delete> Command	20
5.2.3. EPP <renew> Command	21
5.2.4. EPP <transfer> Command	23
5.2.5. EPP <update> Command	25
6. Formal Syntax	27
6.1. Fee Extension Schema	27
7. Security Considerations	32
8. IANA Considerations	32
8.1. XML Namespace	32
8.2. EPP Extension Registry	32
9. Implementation Status	33
9.1. RegistryEngine EPP Service	33
10. Acknowledgements	34
11. Change History	34
11.1. Change from 18 to 19	34
11.2. Change from 18 to 19	34
11.3. Change from 17 to 18	34
11.4. Change from 16 to 17	35

11.5.	Change from 15 to 16	35
11.6.	Change from 14 to 15	35
11.7.	Change from 13 to 14	35
11.8.	Change from 12 to 13	35
11.9.	Change from 11 to 12	35
11.10.	Change from 10 to 11	35
11.11.	Change from 09 to 10	35
11.12.	Change from 08 to 09	36
11.13.	Change from 07 to 08	36
11.14.	Change from 06 to 07	36
11.15.	Change from 05 to 06	36
11.16.	Change from 04 to 05	36
11.17.	Change from 03 to 04	36
11.18.	Change from 02 to 03	37
11.19.	Change from 01 to 02	37
11.20.	Change from 00 to 01	37
11.21.	Change from draft-brown-00 to draft-ietf-regext-fees-00	37
12.	References	37
12.1.	Normative References	37
12.2.	Informative References	39
Authors'	Addresses	39

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 expansion of the DNS namespace, and the proliferation of novel business models, it is 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 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", "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.

"fee" is used as an abbreviation for "urn:ietf:params:xml:ns:epp:fee-1.0". 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.

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. Migrating to Newer Versions of This Extension

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 that is applicable to the given command.

The use of the <fee:command> keys off the use of the "name" attribute to define which transform fees the client is requesting information about. Here is the list of possible values for the "name" attribute:

- o "create" indicating a <create> command as defined in [RFC5730];
- o "delete" indicating a <delete> command as defined in [RFC5730];
- o "renew" indicating a <renew> command as defined in [RFC5730];
- o "update" indicating a <update> command as defined in [RFC5730];
- o "transfer" indicating a <transfer> command as defined in [RFC5730];
- o If the server supports the Registry Grace Period Mapping [RFC3915], then the server MUST also support the "restore" value as defined in [RFC3915];
- o "custom" indicating a custom command that MUST set the "customName" attribute with custom command name. The possible set of custom command name values is up to server policy.

The <fee:command> element MAY have an OPTIONAL "phase" attribute specifying a launch phase as described in [RFC8334]. It may also contain an OPTIONAL "subphase" attribute identifying the custom or sub-phase as described in [RFC8334].

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 [ISO4217:2015].

Note that ISO 4217:2015 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 client commands is OPTIONAL: if a <fee:currency> element is not present in a command, the server MUST determine the currency based on the server default currency or based on the client's account settings which are agreed to 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 "Parameter value range" error instead.

3.3. Validity Periods

When querying for fee information using the <check> command, the <fee:period> element is used to indicate the period measured in years or months, with the appropriate units specified using the "unit"

attribute 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 the server default period. 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. A fee will result in subtracting from the Account Balance (described in Section 3.5) and a credit will result in adding to the Account Balance (described in Section 3.5).

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 zero or greater (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).

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. An OPTIONAL "lang" attribute MAY be present, per the language structure in [RFC5646], to identify the language of the returned text and has a default value of "en" (English). If the "description" attribute is not present, the "lang" attribute can be ignored.

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. An OPTIONAL "lang" attribute MAY be present to identify the language of the returned text and has a default value of "en" (English).

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 [RFC8334]) 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 [RFC3915], 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 and the "refundable" attribute MUST be true. 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" 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" or billable commands (e.g. <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. If the "applied" attribute of the <fee:fee> element is "delayed", then the <fee:balance> MUST reflect the client's account balance without any fees or credits associated with that command.

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 that 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 (e.g. <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 MAY appear in `<check>` and transform 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 that 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.

3.8. Phase and Subphase Attributes

The `<fee:command>` element has two attributes, phase and subphase, that provide additional information related to a specific launch phase as described in [RFC8334]. These attributes are used as filters that should refine the server processing.

If the client `<fee:command>` contains a server supported combination of phase/subphase the server MUST return fee data (including the phase/subphase attribute(s)) for the specific combination.

If the client `<fee:command>` contains no phase/subphase attributes and the server has only one active phase/subphase combination the server MUST return data (including the phase/subphase attribute(s)) of the currently active phase/subphase.

If the client `<fee:command>` contains no phase/subphase attributes and the server has more than one active phase/subphase combination the server MUST respond with a 2003 "Required parameter missing" error.

If the client `<fee:command>` contains no phase/subphase attributes and the server is currently in a "quiet period" (e.g. not accepting registrations or applications) the server MUST return data consistent with the default general availability phase (e.g. "open" or "claims") including the appropriate phase/subphase attribute(s).

If the client `<fee:command>` contains a phase attribute with no subphase and the server has only one active subphase (or no subphase) of this phase, the server MUST return data (including the phase/subphase attribute(s)) of the provided phase and currently active subphase.

If the client `<fee:command>` contains a phase attribute with no subphase and the server has more than one active subphase combination of this phase, the server MUST respond with a 2003 "Required parameter missing" error.

If the client `<fee:command>` contains a subphase with no phase attribute the server MUST respond with a 2003 "Required parameter missing" error.

If the client `<fee:command>` contains a phase attribute not defined in [RFC8334] or not supported by server the server MUST respond with a 2004 "Parameter value range" error.

If the client `<fee:command>` contains a subphase attribute (or phase/subphase combination) not supported by server the server MUST respond with a 2004 "Parameter value range" error.

3.9. Reason

The `<fee:reason>` element is used to provide server specific text in an effort to better explain why a `<check>` command did not complete as the client expected. An OPTIONAL "lang" attribute MAY be present to identify the language, per the language structure in [RFC5646], of the returned text and has a default value of "en" (English).

The `<fee:reason>` element can be used within the server response `<fee:command>` element or within the `<fee:cd>` element. See section 5.1.1 for details on the `<fee:cd>` "check data" element.

If the server cannot calculate the relevant fees, because the object, command, currency, period, class or some combination is invalid per server policy, the server has two ways of handling error processing of `<fee:command>` element(s):

1. Fast-fail - The server, upon error identification, MAY stop processing `<fee:command>` elements and return to the client a `<fee:cd>` containing the `<fee:objID>` and a `<fee:reason>` element detailing the reason for failure.

```
S: <fee:cd avail="0">
S:   <fee:objID>example.xyz</fee:objID>
S:   <fee:reason>Only 1 year registration periods are
S:     valid.</fee:reason>
S: </fee:cd>
```

2. Partial-fail - The server, upon error identification, MAY continue processing `<fee:command>` elements and return to the client a `<fee:cd>` containing successfully processed `<fee:command>`

elements and failed <fee:command> elements. All returned failed <fee:command> elements MUST have a <fee:reason> element detailing the reason for failure, and the server MAY additionally include a <fee:reason> element at the <fee:cd> level.

```
S: <fee:cd avail="0">
S:   <fee:objID>example.xyz</fee:objID>
S:   <fee:command name="create">
S:     <fee:period unit="y">2</fee:period>
S:     <fee:reason>Only 1 year registration periods are
S:       valid.</fee:reason>
S:   </fee:command>
S: </fee:cd>
```

In either failure scenario the server MUST set the <fee:cd> avail attribute to false (0) and the server MUST process all objects in the client request.

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.

The server MUST return avail="0" in its response to a <check> command for any object in the <check> command that does not include the <fee:check> extension for which the server would likewise fail a domain <create> command when no <fee> extension is provided for that same object.

If a server receives a <check> command from a client, which results in no possible fee combination, the server MUST set the "avail" attribute of the <fee:cd> element to false (0) and provide a <fee:reason>.

If a server receives a <check> command from a client, which results in an ambiguous result (i.e. multiple possible fee combinations) the server MUST reject the command with a 2003 "Required parameter missing" error.

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

If the total fee provided by the client is less than the server's own calculation of the fee or the server determines the currency is inappropriate 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 a new command called the Fee Check Command that defines additional elements for the EPP <check> command to provide fee information along with the availability information of the EPP <check> command.

The command MAY contain an <extension> element which MAY contain a <fee:check> element. The <fee:check> element MAY contain one <fee:currency> element and MUST contain one or more <fee:command> elements.

The <fee:command> element(s) MUST contain(s) a "name" attribute (see Section 3.1), an OPTIONAL "phase" attribute, and an OPTIONAL "subphase" attribute (see Section 3.8). The <fee:command> element(s) MAY have the following child elements:

- o An OPTIONAL <fee:period> element (as described in Section 3.3).

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:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:command name="create">
C:           <fee:period unit="y">2</fee:period>
C:         </fee:command>
C:         <fee:command name="renew"/>
C:         <fee:command name="transfer"/>
C:         <fee:command name="restore"/>
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 contain an <extension> element, which MUST contain a child <fee:chkData> element. The <fee:chkData> element MUST contain a <fee:currency> element and a <fee:cd> element for each object referenced in the client <check> command.

Each <fee:cd> (check data) element MUST contain the following child elements:

- o A <fee:objID> element, which MUST match an element referenced in the client <check> command.
- o An OPTIONAL <fee:class> element (as described in Section 3.7).
- o A <fee:command> element matching each <fee:command> (unless the "avail" attribute of the <fee:cd> is false) that appeared in the corresponding <fee:check> of the client command. This element MAY have the OPTIONAL "standard" attribute, with a default value of "0" (or "false"), which indicates whether the fee matches the fee of the "standard" classification (see section 3.7). This element MAY have the OPTIONAL "phase" and "subphase" attributes, which

will match the same attributes in the corresponding <fee:command> element of the client command if sent by the client.

The <fee:cd> element also has an OPTIONAL "avail" attribute which is a boolean. If the value of this attribute evaluates to false, this indicates that the server cannot calculate the relevant fees, because the object, command, currency, period, class or some combination is invalid per server policy. If "avail" is false then the <fee:cd> or the <fee:command> element MUST contain a <fee:reason> element (as described in Section 3.9) and the server MAY eliminate some or all of the <fee:command> element(s).

The <fee:command> element(s) MAY have the following child elements:

- o An OPTIONAL <fee:period> element (as described in Section 3.3), which contains the same unit, if present, that appeared in the <fee:period> element of the command. If the value of the parent <fee:command> element is "restore", this element MUST NOT be included, otherwise it MUST be included. If no <fee:period> appeared in the client command (and the command is not "restore") then the server MUST return its default period value.
- o Zero or more <fee:fee> elements (as described in Section 3.4).
- o Zero or more <fee:credit> elements (as described in Section 3.4).
- o An OPTIONAL <fee:reason> element (as described in Section 3.9).

If the "avail" attribute of the <fee:cd> element is true (1) and if no <fee:fee> elements are present in a <fee:command> 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 true (1), then the <fee:command> 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:epp:fee-1.0">
S:   <fee:currency>USD</fee:currency>
S:   <fee:cd avail="1">
S:     <fee:objID>example.com</fee:objID>
S:     <fee:class>Premium</fee:class>
S:     <fee:command name="create">
S:       <fee:period unit="y">2</fee:period>
S:       <fee:fee
S:         description="Registration Fee"
S:         refundable="1"
S:         grace-period="P5D">10.00</fee:fee>
S:     </fee:command>
S:     <fee:command name="renew">
S:       <fee:period unit="y">1</fee:period>
S:       <fee:fee
S:         description="Renewal Fee"
S:         refundable="1"
S:         grace-period="P5D">10.00</fee:fee>
S:     </fee:command>
S:     <fee:command name="transfer">
S:       <fee:period unit="y">1</fee:period>
S:       <fee:fee
S:         description="Transfer Fee"
S:         refundable="1"
S:         grace-period="P5D">10.00</fee:fee>
S:     </fee:command>
S:     <fee:command name="restore">
S:       <fee:fee
S:         description="Redemption Fee">15.00</fee:fee>
S:     </fee:command>
S:   </fee:cd>
S: <fee:cd avail="1">
S:   <fee:objID>example.net</fee:objID>
S:   <fee:class>standard</fee:class>
S:   <fee:command name="create" standard="1">
S:     <fee:period unit="y">2</fee:period>
S:     <fee:fee
S:       description="Registration Fee"
S:       refundable="1"

```

```

S:         grace-period="P5D">5.00</fee:fee>
S:     </fee:command>
S:     <fee:command name="renew" standard="1">
S:         <fee:period unit="y">1</fee:period>
S:         <fee:fee
S:             description="Renewal Fee"
S:             refundable="1"
S:             grace-period="P5D">5.00</fee:fee>
S:     </fee:command>
S:     <fee:command name="transfer" standard="1">
S:         <fee:period unit="y">1</fee:period>
S:         <fee:fee
S:             description="Transfer Fee"
S:             refundable="1"
S:             grace-period="P5D">5.00</fee:fee>
S:     </fee:command>
S:     <fee:command name="restore" standard="1">
S:         <fee:fee
S:             description="Redemption Fee">5.00</fee:fee>
S:     </fee:command>
S: </fee:cd>
S: <fee:cd avail="0">
S:     <fee:objID>example.xyz</fee:objID>
S:     <fee:command name="create">
S:         <fee:period unit="y">2</fee:period>
S:         <fee:reason>Only 1 year registration periods are
S:             valid.</fee:reason>
S:     </fee:command>
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>

```

5.1.2. 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 is included in the <login> command service extensions.

When the <transfer> query command has been processed successfully, if the client has included the extension in the <login> command service <svcExtension> element, and if the client is authorized by the server to view information about the transfer, then the server MAY include

in the <extension> section of the EPP response a <fee:trnData> element, which contains the following child elements:

- o A <fee:currency> element (as described in Section 3.2).
- o A <fee:period> element (as described in Section 3.3).
- o Zero or more <fee:fee> elements (as described in Section 3.4) containing the fees that will be charged to the gaining client.
- o Zero or more <fee:credit> elements (as described in Section 3.4) 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>2019-06-08T22:00:00.0Z</domain:reDate>
S:         <domain:acID>ClientY</domain:acID>
S:         <domain:acDate>2019-06-13T22:00:00.0Z</domain:acDate>
S:         <domain:exDate>2021-09-08T22:00:00.0Z</domain:exDate>
S:       </domain:trnData>
S:     </resData>
S:     <extension>
S:       <fee:trnData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
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 is included in the <login> command service extensions.

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

- o An OPTIONAL <fee:currency> element (as described in Section 3.2);
- o One or more <fee:fee> elements (as described in Section 3.4).

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

- o A <fee:currency> element (as described in Section 3.2);
- o Zero or more <fee:fee> elements (as described in Section 3.4);
- o Zero or more <fee:credit> elements (as described in Section 3.4);
- o An OPTIONAL <fee:balance> element (as described in Section 3.5);
- o An OPTIONAL <fee:creditLimit> element (as described in Section 3.6).

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:epp:fee-1.0">
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>2019-04-03T22:00:00.0Z</domain:crDate>
S:         <domain:exDate>2021-04-03T22:00:00.0Z</domain:exDate>
S:       </domain:creData>
S:     </resData>
S:     <extension>
S:       <fee:creData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee
S:           description="Registration Fee"
S:           lang="en"
S:           refundable="1"
S:           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 is included in the <login> command service extensions.

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

- o A <fee:currency> element (as described in Section 3.2);

- o Zero or more <fee:fee> elements (as described in Section 3.4);
- o Zero or more <fee:credit> elements (as described in Section 3.4);
- o An OPTIONAL <fee:balance> element (as described in Section 3.5);
- o An OPTIONAL <fee:creditLimit> element (as described in Section 3.6).

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:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:credit
S:           description="AGP Credit"
S:           lang="en">-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 is included in the <login> command service extensions.

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

- o An OPTIONAL <fee:currency> element (as described in Section 3.2);
- o One or more <fee:fee> elements (as described in Section 3.4).

When the <renew> command has been processed successfully, and the client included the extension in the <login> command service extensions, the server MAY include in the <extension> section of the

EPP response a <fee:renData> element, which contains the following child elements:

- o A <fee:currency> element (as described in Section 3.2);
- o Zero or more <fee:fee> elements (as described in Section 3.4);
- o Zero or more <fee:credit> elements (as described in Section 3.4);
- o An OPTIONAL <fee:balance> element (as described in Section 3.5);
- o An OPTIONAL <fee:creditLimit> element (as described in Section 3.6).

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>2019-04-03</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:epp:fee-1.0">
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>2024-04-03T22:00:00.0Z</domain:exDate>
S:         </domain:renData>
S:       </resData>
S:       <extension>
S:         <fee:renData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:           <fee:currency>USD</fee:currency>
S:           <fee:fee
S:             refundable="1"
S:             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 is included in the <login> command service extensions.

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

- o An OPTIONAL <fee:currency> element (as described in Section 3.2);
- o One or more <fee:fee> elements (as described in Section 3.4).

When the <transfer> command has been processed successfully, and the client included the extension in the <login> command service extensions, the server MAY include in the <extension> section of the

EPP response a <fee:trnData> element, which contains the following child elements:

- o A <fee:currency> element (as described in Section 3.2);
- o Zero or more <fee:fee> elements (as described in Section 3.4);
- o Zero or more <fee:credit> elements (as described in Section 3.4);
- o An OPTIONAL <fee:balance> element (as described in Section 3.5);
- o An OPTIONAL <fee:creditLimit> element (as described in Section 3.6).

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:epp:fee-1.0">
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>2019-06-08T22:00:00.0Z</domain:reDate>
S:         <domain:acID>ClientY</domain:acID>
S:         <domain:acDate>2019-06-13T22:00:00.0Z</domain:acDate>
S:         <domain:exDate>2021-09-08T22:00:00.0Z</domain:exDate>
S:       </domain:trnData>
S:     </resData>
S:     <extension>
S:       <fee:trnData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee
S:           refundable="1"
S:           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 is included in the <login> command service extensions.

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

- o An OPTIONAL <fee:currency> element (as described in Section 3.2);
- o One or more <fee:fee> elements (as described in Section 3.4).

When the <update> command has been processed successfully, and the client included the extension in the <login> command service extensions, the server MAY include in the <extension> section of the EPP response a <fee:updData> element, which contains the following child elements:

- o A <fee:currency> element (as described in Section 3.2);
- o Zero or more <fee:fee> elements (as described in Section 3.4);
- o Zero or more <fee:credit> elements (as described in Section 3.4);
- o An OPTIONAL <fee:balance> element (as described in Section 3.5);
- o An OPTIONAL <fee:creditLimit> element (as described in Section 3.6).

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:epp:fee-1.0">
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:epp:fee-1.0">
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>
```

6. Formal Syntax

One schema is presented here that is the EPP Fee 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.

6.1. Fee 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.

BEGIN

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0"
  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:epp:fee-1.0"
  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 Fee Extension
  </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:transformResultType" />
<element name="update" type="fee:transformCommandType" />
<element name="updData" type="fee:transformResultType" />
<element name="delData" type="fee:transformResultType" />

<!-- client <check> command -->
<complexType name="checkType">
  <sequence>
    <element name="currency" type="fee:currencyType"
      minOccurs="0" />
    <element name="command" type="fee:commandType"
      minOccurs="1" maxOccurs="unbounded" />
  </sequence>
</complexType>

<complexType name="objectIdentifierType">
  <simpleContent>
    <extension base="eppcom:labelType">
      <attribute name="element"
        type="NMTOKEN" default="name" />
    </extension>
  </simpleContent>
</complexType>

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

```
</complexType>

<complexType name="objectCDType">
  <sequence>
    <element name="objID" type="fee:objectIdentifierType" />
    <element name="class" type="token" minOccurs="0" />
    <element name="command" type="fee:commandDataType"
      minOccurs="0" maxOccurs="unbounded" />
    <element name="reason" type="fee:reasonType" 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"
      minOccurs="0" />
    <element name="period" type="domain:periodType"
      minOccurs="0" />
    <element name="fee" type="fee:feeType"
      minOccurs="0" maxOccurs="unbounded" />
    <element name="credit" type="fee:creditType"
      minOccurs="0" maxOccurs="unbounded" />
    <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>
```

```
</simpleType>

<complexType name="commandType">
  <sequence>
    <element name="period" type="domain:periodType"
      minOccurs="0" maxOccurs="1" />
  </sequence>
  <attribute name="name" type="fee:commandEnum" use="required"/>
  <attribute name="customName" type="token"/>
  <attribute name="phase" type="token" />
  <attribute name="subphase" type="token" />
</complexType>

<complexType name="commandDataType">
  <complexContent>
    <extension base="fee:commandType">
      <sequence>
        <element name="fee" type="fee:feeType"
          minOccurs="0" maxOccurs="unbounded" />
        <element name="credit" type="fee:creditType"
          minOccurs="0" maxOccurs="unbounded" />
        <element name="reason" type="fee:reasonType"
          minOccurs="0" />
      </sequence>
      <attribute name="standard" type="boolean" default="0" />
    </extension>
  </complexContent>
</complexType>

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

<simpleType name="commandEnum">
  <restriction base="token">
    <enumeration value="create"/>
    <enumeration value="delete"/>
    <enumeration value="renew"/>
    <enumeration value="update"/>
    <enumeration value="transfer"/>
    <enumeration value="restore"/>
    <enumeration value="custom"/>
  </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="lang" type="language" default="en"/>
      <attribute name="refundable" type="boolean" />
      <attribute name="grace-period" type="duration" />
      <attribute name="applied">
        <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"/>
      <attribute name="lang" type="language" default="en"/>
    </extension>
  </simpleContent>
</complexType>

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

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

</schema>
END

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. This extension passes financial information using the EPP protocol, so confidentiality and integrity protection must be provided by the transport mechanism. All transports compliant with [RFC5730] provide the needed level of confidentiality and integrity protections. The server will only provide information, including financial information, that is relevant to the authenticated client.

8. IANA Considerations

8.1. 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 fee namespace:

URI: urn:ietf:params:xml:ns:epp:fee-1.0

Registrant Contact: IESG

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

Registration request for the fee schema:

URI: urn:ietf:params:xml:schema:epp:fee-1.0

Registrant Contact: IESG

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

8.2. 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: Registry Fee 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

9. Implementation Status

Note to RFC Editor: Please remove this section and the reference to [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 [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 [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".

9.1. 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>

10. Acknowledgements

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

- o James Gould of Verisign Inc
- o Luis Munoz of ISC
- o Michael Young of Architelos
- o Ben Levac and Jeff Eckhaus of Demand Media
- o Seth Goldman of Google
- o Klaus Malorny and Michael Bauland of Knipp
- o Jody Kolker, Joe Snitker and Kevin Allendorf of Go Daddy
- o Michael Holloway of Com Laude
- o Santosh Kalsangrah of Impetus Infotech
- o Alex Mayrhofer of Nic.at
- o Thomas Corte of Knipp Medien und Kommunikation GmbH

11. Change History

11.1. Change from 18 to 19

Added normative reference for XML Schema.

11.2. Change from 18 to 19

Updated per IESG review, all updates (except for one schema change) were just textual for clarity and correctness. The schema change was to require the name attribute of the commandType element.

11.3. Change from 17 to 18

Corrected erroneous edit left in place in previous revision (17), reverted text back to original text (revision 16) in section 3.4.

11.4. Change from 16 to 17

Updated per AD review, all updates were just textual for clarity and correctness.

11.5. Change from 15 to 16

Updated per AD review and list comments: several grammar corrections; clarification text added to section 3.4.3 and 3.5; and a schema update for consistency by providing a "lang" attribute to the <fee:fee> and <fee:credit> "description" attribute detailed in section 3.4.

11.6. Change from 14 to 15

Updated schema, moving the "standard" attribute of the "commandDataType" inside the <extension> block.

11.7. Change from 13 to 14

Moved RFC 7451 reference from Normative to Informative section.

11.8. Change from 12 to 13

Updated XML namespace and schema registration to be "epp" scoped - global replace of XML namespace from urn:ietf:params:xml:ns:fee-1.0 to urn:ietf:params:xml:ns:epp:fee-1.0 and the XML schema registration from urn:ietf:params:xml:schema:fee-1.0 to urn:ietf:params:xml:schema:epp:fee-1.0.

11.9. Change from 11 to 12

Updated references to current version of documents and moved the "standard" attribute from the check command (commandType) to the check response (commandDataType).

11.10. Change from 10 to 11

Updated document per Working Group Last Call comments. Made minor textual changes throughout for enhanced clarity per WGLC comments.

11.11. Change from 09 to 10

Updated document per Working Group Last Call comments. Updated schema to version 1.0 in anticipation of standardization, no changes were made to the latest, 0.25, schema. Made minor textual changes throughout for enhanced clarity per WGLC comments.

11.12. Change from 08 to 09

Updated scheme to version 0.25 to allow tighter checking on `<fee:command>` by splitting the client and server definitions, moved the class element from the command to the object level and added an optional standard attribute to the command element. Also updated section 3.1 for clarity on name attribute; updated section 3.9 for clarity on uses of `<fee:reason>`; removed second paragraph in section 5.2.1 as it was duplicative of second to last paragraph in 4.0; and updated section 5.1.1 to add section references.

11.13. Change from 07 to 08

Updated section 3.8 and 5.1.1 to provide clarity on server processing and response of various scenarios (i.e. "quiet" period processing).

11.14. Change from 06 to 07

Updated section 3.8 and 4.0 to provide clarity on server processing and response of various scenarios.

11.15. Change from 05 to 06

Updated scheme to version 0.23 to allow the return of no `<fee:command>` element(s) if an error situation occurs. Edited section 3.8 extensively after input from interim meeting and REGEXT F2F meeting at IETF-99. Added normative reference for draft-ietf-eppext-launchphase.

11.16. Change from 04 to 05

Updated scheme to version 0.21 to support the lang attribute for the reason element of the objectCDType and the commandType types as well as to add the update command to the commandEnum type. Updated section 3.1 to include language for the custom command. Added section 3.9 to provide a description of the `<fee:reason>` element. Fixed typos and added clarification text on when client fee is less than server fee in section 4. Additionally, I added description pointers to appropriate Section 3 definitions for element clarity throughout the document.

11.17. Change from 03 to 04

Updated scheme to version 0.19 to correct typos and to replace the commandTypeValue type with the commandEnum type and customName attribute for stricter validation. Updated various text for grammar and clarity. Added text to section 4 clarifying the `<check>` response

when the client provided no fee extension but the server was expecting the extension.

11.18. Change from 02 to 03

Updated scheme to version 0.17 to simplify the check command syntax. Moved fee avail to objectCDType to allow fast failing on error situations. Removed the objectCheckType as it was no longer being used. Updated examples to reflect these scheme changes. Added language for server failing a <create> if the <fee:fee> passed by the client is less than the server fee.

11.19. Change from 01 to 02

Updated scheme to version 0.15 to fix errors in CommandType, objectCDType, transformCommandType and transformResultType definitions.

11.20. Change from 00 to 01

Added Roger Carney as author to finish draft. Moved Formal Syntax section to main level numbering. Various grammar, typos, and administrative edits for clarity. Removed default value for the "applied" attribute of <fee:fee> so that it can truly be optional. Added support for the <delete> command to return a <fee:fee> element as well. Modified default response on the <check> command for the optional <fee:period> when it was not provided in the command, leaving it to the server to provide the default period value. Extensive edits were done to the <check> command, the <check> response and to the fee extension schema (checkType, objectCheckType, objectIdentifierType, objectCDType, commandType) to support requesting and returning multiple transformation fees in a single call. Added section on Phase/Subphase to provide more context on the uses.

11.21. Change from draft-brown-00 to draft-ietf-regext-fees-00

Updated to be REGEXT WG document.

12. References

12.1. Normative References

[ISO4217:2015]

International Organization for Standardization, "Codes for the representation of currencies", August 2015, <<https://www.iso.org/standard/64758.html>>.

- [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>>.
- [RFC3915] Hollenbeck, S., "Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP)", RFC 3915, DOI 10.17487/RFC3915, September 2004, <<https://www.rfc-editor.org/info/rfc3915>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC8334] Gould, J., Tan, W., and G. Brown, "Launch Phase Mapping for the Extensible Provisioning Protocol (EPP)", RFC 8334, DOI 10.17487/RFC8334, March 2018, <<https://www.rfc-editor.org/info/rfc8334>>.
- [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>>.

12.2. Informative References

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

Authors' Addresses

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

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

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: <http://www.centralnic.com>

Jothan Frakes

Email: jothan@jothan.com
URI: <http://jothan.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

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	3
1.1. Conventions Used in This Document	4
2. Object Attributes	5
2.1. Application Identifier	5
2.2. Validator Identifier	5
2.3. Launch Phases	6
2.3.1. Trademark Claims Phase	7
2.4. Status Values	9
2.4.1. State Transition	10
2.5. Poll Messaging	12
2.6. Mark Validation Models	15
2.6.1. <launch:codeMark> element	16
2.6.2. <mark:mark> element	17
2.6.3. Digital Signature	17
2.6.3.1. <smd:signedMark> element	17
2.6.3.2. <smd:encodedSignedMark> element	17
3. EPP Command Mapping	17
3.1. EPP <check> Command	18
3.1.1. Claims Check Form	18
3.1.2. Availability Check Form	21
3.1.3. Trademark Check Form	23
3.2. EPP <info> Command	26
3.3. EPP <create> Command	30
3.3.1. Sunrise Create Form	30
3.3.2. Claims Create Form	36
3.3.3. General Create Form	39
3.3.4. Mixed Create Form	40
3.3.5. Create Response	42
3.4. EPP <update> Command	43
3.5. EPP <delete> Command	44
3.6. EPP <renew> Command	45
3.7. EPP <transfer> Command	46
4. Formal Syntax	46
4.1. Launch Schema	46
5. IANA Considerations	54
5.1. XML Namespace	54
5.2. EPP Extension Registry	54
6. Implementation Status	55
6.1. Verisign EPP SDK	55
6.2. Verisign Consolidated Top Level Domain (CTLD) SRS	56
6.3. Verisign .COM / .NET SRS	56
6.4. REngin v3.7	57
6.5. RegistryEngine EPP Service	57

6.6. Neustar EPP SDK	58
6.7. gTLD Shared Registry System	58
7. Security Considerations	58
8. Acknowledgements	59
9. References	59
9.1. Normative References	59
9.2. Informative References	60
Appendix A. Change History	60
A.1. Change from 00 to 01	60
A.2. Change from 01 to 02	60
A.3. Change from 02 to 03	61
A.4. Change from 03 to 04	61
A.5. Change from 04 to 05	61
A.6. Change from 05 to 06	62
A.7. Change from 06 to 07	62
A.8. Change from 07 to 08	62
A.9. Change from 08 to 09	62
A.10. Change from 09 to 10	63
A.11. Change from 10 to 11	64
A.12. Change from 11 to 12	64
A.13. Change from 12 to EPPEXT 00	64
A.14. Change EPPEXT 00 to EPPEXT 01	64
A.15. Change EPPEXT 01 to EPPEXT 02	64
A.16. Change EPPEXT 02 to EPPEXT 03	65
A.17. Change EPPEXT 03 to EPPEXT 04	65
A.18. Change EPPEXT 04 to EPPEXT 05	65
A.19. Change EPPEXT 05 to EPPEXT 06	65
A.20. Change EPPEXT 06 to EPPEXT 07	65
A.21. Change from EPPEXT 07 to REGEXT 00	66
A.22. Change from REGEXT 00 to REGEXT 01	66
A.23. Change from REGEXT 01 to REGEXT 02	66
A.24. Change from REGEXT 02 to REGEXT 03	66
A.25. Change from REGEXT 03 to REGEXT 04	66
A.26. Change from REGEXT 04 to REGEXT 05	66
A.27. Change from REGEXT 05 to REGEXT 06	67
A.28. Change from REGEXT 06 to REGEXT 07	67
Authors' Addresses	70

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:

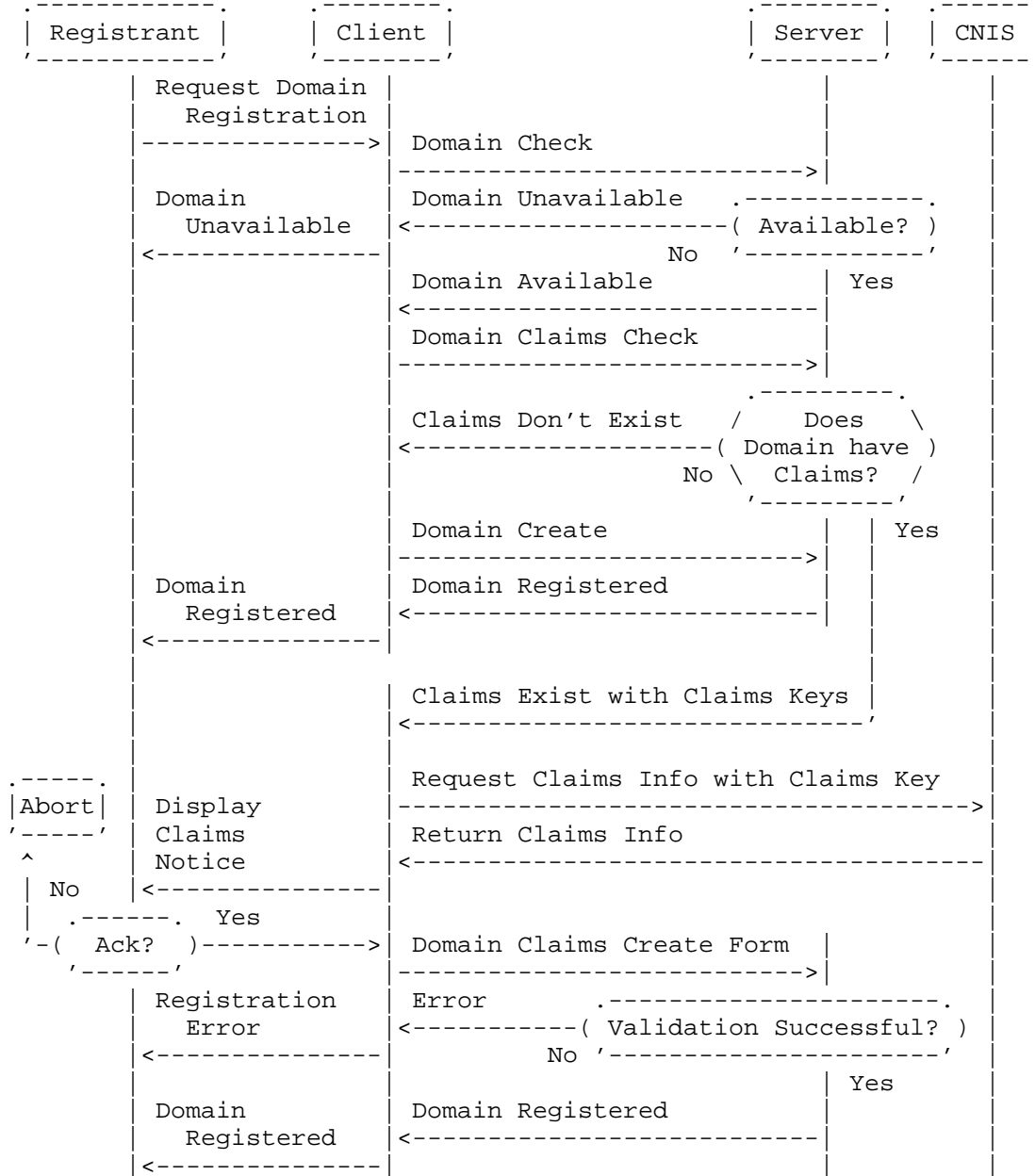


Figure 1

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.

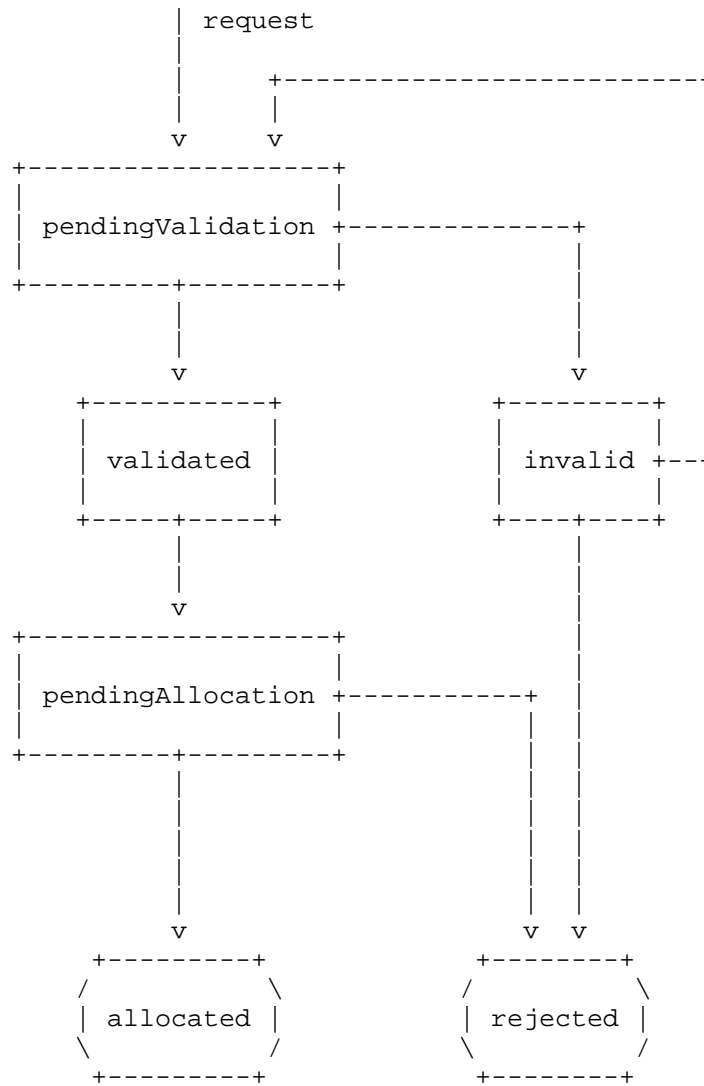


Figure 2

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/rJlNrDO92vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:        </launch:cd>
S:        <launch:cd>
S:          <launch:name exists="1">domain3.example</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJlNrDO92vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:          <launch:claimKey validatorID="custom-tmch">
S:            20140423200/1/2/3/rJlNr2vDsAzasdff7EasdfgjX4R0000000002
S:          </launch:claimKey>
S:        </launch:cd>
S:      </launch:chkData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.1.2. Availability Check Form

The Availability Check Form defines additional elements to extend the EPP <check> command described in the EPP domain name mapping [RFC5731]. No additional elements are defined for the EPP <check>

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

The EPP <check> command is used to determine if an object can be provisioned within a repository. Domain names may be made available only in unique launch phases, whilst remaining unavailable for concurrent launch phases. In addition to the elements expressed in the <domain:check>, the command is extended with the <launch:check> element that contains the following child elements:

<launch:phase>: The launch phase to which domain name availability should be determined. 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/rJlNrDO92vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:        </launch:cd>
S:        <launch:cd>
S:          <launch:name exists="1">domain3.example</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJlNrDO92vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:          <launch:claimKey validatorID="custom-tmch">
S:            20140423200/1/2/3/rJlNr2vDsAzasdff7EasdfgjX4R0000000002
S:          </launch:claimKey>
S:        </launch:cd>
S:      </launch:chkData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.2. 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>jdl1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:        <launch:phase>sunrise</launch:phase>
C:        <launch:codeMark>
C:          <launch:code validatorID="sample1">
C:            49FD46E6C4B45C55D4AC</launch:code>
C:          </launch:codeMark>
C:          <launch:codeMark>
C:            <launch:code>49FD46E6C4B45C55D4AD</launch:code>
C:          </launch:codeMark>
C:          <launch:codeMark>
C:            <launch:code validatorID="sample2">
C:              49FD46E6C4B45C55D4AE</launch:code>
C:            </launch:codeMark>
C:          </launch:codeMark>
C:        </launch:create>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

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

The following is an example <create> domain command using the <launch:create> extension, following the "code with mark" validation model, with a code and mark information:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domain.example</domain:name>
C:        <domain:registrant>jdl1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:        <launch:phase>sunrise</launch:phase>
C:        <launch:codeMark>
C:          <launch:code validatorID="sample">
C:            49FD46E6C4B45C55D4AC</launch:code>
C:          <mark:mark
C:            xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
C:            ...
C:          </mark:mark>
C:        </launch:codeMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

The following is an example <create> domain command using the <launch:create> extension, following the "signed mark" validation model, with the signed mark information for a sunrise application:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domainone.example</domain:name>
C:        <domain:registrant>jdl234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:        type="application">
C:        <launch:phase>sunrise</launch:phase>
C:        <smd:signedMark id="signedMark"
C:          xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
C:          ...
C:        </smd:signedMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domainone.example</domain:name>
C:        <domain:registrant>jdl234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:        <launch:phase>sunrise</launch:phase>
C:        <smd:encodedSignedMark
C:          xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
C:          ...
C:        </smd:encodedSignedMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

3.3.2. Claims Create Form

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

A <launch:create> element is sent along with the regular <create> domain command. The <launch:create> element has an OPTIONAL "type" attribute that defines the expected type of object ("application" or "registration") to create. The server SHOULD validate the "type" attribute, when passed, against the type of object that will be created, 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:registrar>jdl1234</domain:registrar>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:        <launch:phase>claims</launch:phase>
C:        <launch:notice>
C:          <launch:noticeID validatorID="tmch">
C:            370d0b7c9223372036854775807</launch:noticeID>
C:          <launch:notAfter>2014-06-19T10:00:00.0Z
C:          </launch:notAfter>
C:          <launch:acceptedDate>2014-06-19T09:00:00.0Z
C:          </launch:acceptedDate>
C:        </launch:notice>
C:        <launch:notice>
C:          <launch:noticeID validatorID="custom-tmch">
C:            470d0b7c9223654313275808</launch:noticeID>
C:          <launch:notAfter>2014-06-19T10:00:00.0Z
C:          </launch:notAfter>
C:          <launch:acceptedDate>2014-06-19T09:00:30.0Z
C:          </launch:acceptedDate>
C:        </launch:notice>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

3.3.3. General Create Form

The General Create Form of the extension to the EPP domain name mapping [RFC5731] includes the launch phase and optionally the object type to create. The OPTIONAL "type" attribute defines the expected type of object ("application" or "registration") to create. The server SHOULD validate the "type" attribute, when passed, against the type of object that will be created, 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>jdl1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:        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

Copyright (c) 2017 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

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

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace="urn:ietf:params:xml:ns:launch-1.0"
  xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
  xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
  xmlns:mark="urn:ietf:params:xml:ns:mark-1.0"
  xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
>

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

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

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

  <!-- Common container of id (identifier) element -->
  <complexType name="idContainerType">
    <sequence>
      <element
        name="phase"
```

```
        type="launch:phaseType"/>
      <element
        name="applicationID"
        type="launch:applicationIDType"/>
    </sequence>
</complexType>

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

<!-- Definition for launch phase. Name is an
      optional attribute used to extend the phase type.
      For example, when using the phase type value
      of "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 description of the <launch:phase> element, based on feedback from Keith Gaughan on the eppext mailing list.
2. Added the SIDN implementation status information.
3. Fixed a few indentation issues in the samples.

A.19. Change 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 relevant 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 Wissner 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 identifier 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 hanging 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 3, 2019

L. Zhou
CNNIC
N. Kong
Consultant
G. Zhou
J. Yao
CNNIC
J. Gould
Verisign, Inc.
November 30, 2018

Extensible Provisioning Protocol (EPP) Organization Mapping
draft-ietf-regext-org-12

Abstract

This document describes an Extensible Provisioning Protocol (EPP) mapping for provisioning and management of organization objects 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 organizations.

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 3, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions Used in This Document	3
3. Object Attributes	3
3.1. Organization Identifier	4
3.2. Organization Roles	4
3.2.1. Role Type	4
3.2.2. Role Status	4
3.2.3. Role Identifier	4
3.3. Contact and Client Identifiers	5
3.4. Organization Status Values	5
3.5. Role Status Values	6
3.6. Parent Identifier	7
3.7. URL	7
3.8. Dates and Times	7
4. EPP Command Mapping	8
4.1. EPP Query Commands	8
4.1.1. EPP <check> Command	8
4.1.2. EPP <info> Command	10
4.1.3. EPP <transfer> Query Command	16
4.2. EPP Transform Commands	16
4.2.1. EPP <create> Command	16
4.2.2. EPP <delete> Command	20
4.2.3. EPP <renew> Command	21
4.2.4. EPP <transfer> Command	21
4.2.5. EPP <update> Command	22
4.3. Offline Review of Requested Actions	26
5. Formal Syntax	28
6. Internationalization Considerations	37
7. IANA Considerations	37
7.1. XML Namespace	37
7.2. EPP Extension Registry	38
7.3. Role Type Values Registry	38
7.3.1. Registration Template	38
7.3.2. Initial Registry Contents	38
8. Implementation Status	39
8.1. Verisign EPP SDK	40
8.2. CNNIC Implementation	40
9. Security Considerations	41
10. Acknowledgment	41

11. References	41
11.1. Normative References	41
11.2. Informative References	42
Appendix A. Change Log	43
Authors' Addresses	46

1. Introduction

There are many entities, such as registrars, resellers, DNS service operators, or privacy proxies involved in the domain registration business. These kind of entities have not been formally defined as having an object in Extensible Provisioning Protocol (EPP). This document provides a way to specify them as "organization" entities.

This document describes an organization object mapping for version 1.0 of the EPP [RFC5730]. This mapping 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 BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

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.

The XML namespace prefix "org" is used for the namespace "urn:ietf:params:xml:ns:epp:org-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.

3. Object Attributes

An EPP organization 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. Organization Identifier

All EPP organizations are identified by a server-unique identifier. Organization identifiers are character strings with a specified minimum length, a specified maximum length, and a specified format. Organization identifiers use the "clIDType" client identifier syntax described in [RFC5730]. Its corresponding element is <org:id>.

3.2. Organization Roles

The organization roles are used to represent the relationship an organization could have. Its corresponding element is <org:role>. An organization object MUST always have at least one associated role. Roles can be set only by the client that sponsors an organization object. A client can change the role of an organization object using the EPP <update> command.

3.2.1. Role Type

An organization role MUST have a type field. This may have any of the values listed in Section 7.3. An organization could have multiple roles with different role types. Its corresponding element is <org:type>.

3.2.2. Role Status

A role of an organization object MAY have its own statuses. Its corresponding element is <org:status>. The values of the role status are defined in Section 3.5.

3.2.3. Role Identifier

A role MAY have a third-party-assigned identifier such as the IANA ID for registrars. Its corresponding element is <org:roleID>.

Example of organization role identifier:

```
<org:role>
  <org:type>registrar</org:type>
  <org:status>ok</org:status>
  <org:status>linked</org:status>
  <org:roleID>1362</org:roleID>
</org:role>
```

3.3. Contact and Client Identifiers

All EPP contacts are identified by server-unique identifiers. Contact identifiers are character strings with a specified minimum length, a specified maximum length, and a specified format. Contact identifiers use the "clIDType" client identifier syntax described in [RFC5730].

3.4. Organization Status Values

An organization object MUST always have at least one associated status value. Status values can be set only by the client that sponsors an organization object and by the server on which the object resides. A client can change the status of an organization object using the EPP <update> command. Each status value MAY be accompanied by a string of human-readable text that describes the rationale for the status applied to the object.

A client MUST NOT alter server status values set by the server. A server MAY alter or override status values set by a client, subject to local server policies. The status of an object MAY change as a result of either a client-initiated transform command or an action performed by a server operator.

Status values that can be added or removed by a client are prefixed with "client". Corresponding server status values that can be added or removed by a server are prefixed with "server". The "hold" and "terminated" status values are server-managed when the organization has no parent identifier [Section 3.6] and otherwise MAY be client-managed based on server policy. Other status values that do not begin with either "client" or "server" are server-managed.

Status Value Descriptions:

- o ok: This is the normal status value for an object that has no operations pending or active prohibitions. This value is set and removed by the server as other status values are added or removed.
- o hold: Organization transform commands and new links MUST be rejected.
- o terminated: The organization which has been terminated MUST NOT be linked. Organization transform commands and new links MUST be rejected.
- o linked: The organization object has at least one active association with another object. The "linked" status is not

explicitly set by the client. Servers should provide services to determine existing object associations.

- o `clientLinkProhibited`, `serverLinkProhibited`: Requests to add new links to the organization MUST be rejected.
- o `clientUpdateProhibited`, `serverUpdateProhibited`: Requests to update the object (other than to remove this status) MUST be rejected.
- o `clientDeleteProhibited`, `serverDeleteProhibited`: Requests to delete the object MUST be rejected.
- o `pendingCreate`, `pendingUpdate`, `pendingDelete`: A transform command has been processed for the object, but the action has not been completed by the server. Server operators can delay action completion for a variety of reasons, such as to allow for human review or third-party action. A transform command that is processed, but whose requested action is pending, is noted with response code 1001.

"`pendingCreate`", "`ok`", "`hold`", and "`terminated`" are mutually exclusive statuses. Organization MUST have exactly one of these statuses set.

"`ok`" status MAY only be combined with "`linked`" status.

A client or server MAY combine "`linked`" with either "`clientLinkProhibited`" or "`serverLinkProhibited`" if new links must be prohibited.

"`pendingDelete`" status MUST NOT be combined with either "`clientDeleteProhibited`" or "`serverDeleteProhibited`" status.

The `pendingCreate`, `pendingDelete`, and `pendingUpdate` status values MUST NOT be combined with each other.

If "`clientUpdateProhibited`" or "`serverUpdateProhibited`" is set, the client will not be able to update the object. For "`clientUpdateProhibited`", the client will first need to remove "`clientUpdateProhibited`" prior to attempting to update the object. The server can modify the object at any time.

3.5. Role Status Values

A role SHOULD have at least one associated status value. Valid values include "`ok`", "`linked`", "`clientLinkProhibited`", and "`serverLinkProhibited`".

Status Value Descriptions:

- o ok: This is the normal status value for a role that has no operations pending or active prohibitions. This value is set and removed by the server as other status values are added or removed.
- o linked: The role of an organization object has at least one active association with another object. The "linked" status is not explicitly set by the client. Servers SHOULD provide services to determine existing object associations.
- o clientLinkProhibited, serverLinkProhibited: Requests to add new links to the role MUST be rejected.

3.6. Parent Identifier

There can be more than one layer of organizations, such as a reseller. The parent identifier, as defined with the <org:parentId> element, represents the parent organization identifier in a child organization.

The case of reseller organizations provides an example. 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 MUST be prohibited. For example: if organization A has B as its parent identifier, organization B cannot have organization A as its parent identifier. The same is true for larger loops involving three or more organizations.

3.7. URL

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

3.8. Dates and Times

Date and time attribute values MUST be represented in Universal Coordinated Time (UTC) using the Gregorian calendar. The extended date-time form using upper case "T" and "Z" characters defined in [W3C.REC-xmlschema-2-20041028] MUST be used to represent date-time values, as XML Schema does not support truncated date-time forms or lower case "t" and "z" characters.

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 organization information via EPP.

4.1. EPP Query Commands

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

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 an `<org:check>` element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The `<org:check>` element contains the following child elements:

- o One or more `<org:id>` elements that contain the server-unique identifier of the organization 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:      <org:check
C:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
C:          <org:id>res1523</org:id>
C:          <org:id>re1523</org:id>
C:          <org:id>1523res</org:id>
C:        </org: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 `<org:chkData>` element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The `<org:chkData>` element contains one or more `<org:cd>` elements that contain the following child elements:

- o An `<org: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 `<org: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 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).

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 lang="en">Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <org:chkData
S:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
S:        <org:cd>
S:          <org:id avail="1">res1523</org:id>
S:        </org:cd>
S:        <org:cd>
S:          <org:id avail="0">re1523</org:id>
S:          <org:reason lang="en">In use</org:reason>
S:        </org:cd>
S:        <org:cd>
S:          <org:id avail="1">1523res</org:id>
S:        </org:cd>
S:      </org: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 an organization object. In addition to the standard EPP command elements, the <info> command MUST contain a <org:info> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:info> element contains the following child elements:

- o An <org:id> element that contains the server-unique identifier of the organization 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:      <org:info
C:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
C:          <org:id>res1523</org:id>
C:        </org: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 `<org:infData>` element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The `<org:infData>` element contains the following child elements:

- o An `<org:id>` element that contains the server-unique identifier of the organization object, as defined in Section 3.1.
- o An `<org:roid>` element that contains the Repository Object Identifier assigned to the organization object when the object was created.
- o One or more `<org:role>` elements that contain the role type, role statuses and optional role id of the organization.
 - * An `<org:type>` element that contains the type of the organization, as defined in Section 3.2.
 - * One or more `<org:status>` elements that contain the role statuses. The values of the role status are defined in Section 3.5.
 - * An OPTIONAL `<org:roleID>` element that contains a third-party-assigned identifier, such as IANA ID for registrars, as defined in Section 3.2.3.
- o One or more `<org:status>` elements that contain the operational status of the organization, as defined in Section 3.4.
- o An OPTIONAL `<org:parentId>` element that contains the identifier of the parent object, as defined in Section 3.6.
- o Zero to two `<org: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 Unicode in the range U+0020 - U+007E. If a localized form (type="loc") is provided, element content MAY be represented in unrestricted UTF-8. The <org:postalInfo> element contains the following child elements:

- * An <org:name> element that contains the name of the organization.
- * An OPTIONAL <org:addr> element that contains address information associated with the organization. A <org:addr> element contains the following child elements:
 - + One, two, or three <org:street> elements that contain the organization's street address.
 - + An <org:city> element that contains the organization's city.
 - + An OPTIONAL <org:sp> element that contains the organization's state or province.
 - + An OPTIONAL <org:pc> element that contains the organization's postal code.
 - + An <org:cc> element that contains the alpha-2 organization's country code. The detailed format of this element is described in section 2.4.3 of [RFC5733].
- o An OPTIONAL <org:voice> element that contains the organization's voice telephone number. The detailed format of this element is described in Section 2.5 of [RFC5733].
- o An OPTIONAL <org:fax> element that contains the organization's facsimile telephone number.
- o An OPTIONAL <org:email> element that contains the organization's email address. The detailed format of this element is described in section 2.6 of [RFC5733].
- o An OPTIONAL <org:url> element that contains the URL to the website of the organization. The detailed format of this element is described in [RFC3986].
- o Zero or more <org:contact> elements that contain identifiers for the contact objects to be associated with the organization object.

Contact object identifiers MUST be known to the server before the contact object can be associated with the organization object. The required "type" is used to represent contact types. The type values include "admin", "tech", "billing", "abuse", and "custom". The OPTIONAL "typeName" attribute is used to define the name of a "custom" type.

- o An OPTIONAL <org:clID> element that contains the organization identifier of the sponsoring client. There is no <org:clID> element if the organization is managed by the registry.
- o An <org:crID> element that contains the identifier of the client that created the organization object.
- o An <org:crDate> element that contains the date and time of organization object creation.
- o An <org:upID> element that contains the identifier of the client that last updated the organization object. This element MUST NOT be present if the organization has never been modified.
- o An <org:upDate> element that contains the date and time of the most recent organization object modification. This element MUST NOT be present if the organization object has never been modified.

Example <info> response for "Example Registrar Inc." organization organization object with identifier "registrar1362":

```
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">Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <org:infData
S:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
S:        <org:id>registrar1362</org:id>
S:        <org:roid>registrar1362-REP</org:roid>
S:        <org:role>
S:          <org:type>registrar</org:type>
S:          <org:status>ok</org:status>
S:          <org:status>linked</org:status>
S:          <org:roleID>1362</org:roleID>
S:        </org:role>
S:        <org:status>ok</org:status>
S:        <org:postalInfo type="int">
```

```
S:      <org:name>Example Registrar Inc.</org:name>
S:      <org:addr>
S:          <org:street>123 Example Dr.</org:street>
S:          <org:street>Suite 100</org:street>
S:          <org:city>Dulles</org:city>
S:          <org:sp>VA</org:sp>
S:          <org:pc>20166-6503</org:pc>
S:          <org:cc>US</org:cc>
S:      </org:addr>
S:      </org:postalInfo>
S:      <org:voice x="1234">+1.7035555555</org:voice>
S:      <org:fax>+1.7035555556</org:fax>
S:      <org:email>contact@organization.example</org:email>
S:      <org:url>https://organization.example</org:url>
S:      <org:contact type="admin">sh8013</org:contact>
S:      <org:contact type="billing">sh8013</org:contact>
S:      <org:contact type="custom"
S:          typeName="legal">sh8013</org:contact>
S:      <org:crID>ClientX</org:crID>
S:      <org:crDate>1999-04-03T22:00:00.0Z</org:crDate>
S:      <org:upID>ClientX</org:upID>
S:      <org:upDate>1999-12-03T09:00:00.0Z</org:upDate>
S:      </org:infData>
S:  </resData>
S:  <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:  </trID>
S: </response>
S:</epp>
```

Example <info> response for "Example Reseller Inc." organization object of reseller type managed by identifier "registrar1362":


```
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">Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <org:infData
S:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
S:        <org:id>reseller1523</org:id>
S:        <org:roid>reseller1523-REP</org:roid>
S:        <org:role>
S:          <org:type>reseller</org:type>
S:          <org:status>ok</org:status>
S:          <org:status>linked</org:status>
S:        </org:role>
S:        <org:status>ok</org:status>
S:        <org:parentId>registrar1362</org:parentId>
S:        <org:postalInfo type="int">
S:          <org:name>Example Reseller Inc.</org:name>
S:          <org:addr>
S:            <org:street>123 Example Dr.</org:street>
S:            <org:street>Suite 100</org:street>
S:            <org:city>Dulles</org:city>
S:            <org:sp>VA</org:sp>
S:            <org:pc>20166-6503</org:pc>
S:            <org:cc>US</org:cc>
S:          </org:addr>
S:        </org:postalInfo>
S:        <org:fax>+1.7035555556</org:fax>
S:        <org:url>https://organization.example</org:url>
S:        <org:contact type="admin">sh8013</org:contact>
S:        <org:clID>1362</org:clID>
S:        <org:crID>ClientX</org:crID>
S:        <org:crDate>1999-04-03T22:00:00.0Z</org:crDate>
S:        <org:upID>ClientX</org:upID>
S:        <org:upDate>1999-12-03T09:00:00.0Z</org:upDate>
S:      </org: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> Query Command

The transfer semantics does not apply to organization object. No EPP <transfer> query command is defined in this document.

4.2. EPP Transform Commands

This document provides three commands to transform organization object information: <create> to create an instance of an organization object, <delete> to delete an instance of an organization object, and <update> to change information associated with an organization 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, the client 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 an organization object. In addition to the standard EPP command elements, the <create> command MUST contain a <org:create> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:create> element contains the following child elements:

- o An <org:id> element that contains the desired server-unique identifier for the organization to be created, as defined in Section 3.1.
- o One or more <org:role> elements that contain the role type, role statuses and optional role id of the organization.
- * An <org:type> element that contains the type of the organization, as defined in Section 3.2.

- * Zero or more <org:status> elements that contain the role statuses. The values of the role status are defined in Section 3.5.
- * An OPTIONAL <org:roleID> element that contains a third-party-assigned identifier, such as IANA ID for registrars, as defined in Section 3.2.3.
- o Zero or more <org:status> elements that contain the operational status of the organization, as defined in Section 3.4.
- o An OPTIONAL <org:parentId> element that contains the identifier of the parent object, as defined in Section 3.6.
- o Zero to two <org: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 Unicode in the range U+0020 - U+007E. If a localized form (type="loc") is provided, element content MAY be represented in unrestricted UTF-8. The <org:postalInfo> element contains the following child elements:
 - * An <org:name> element that contains the name of the organization.
 - * An OPTIONAL <org:addr> element that contains address information associated with the organization. A <org:addr> element contains the following child elements:
 - + One, two, or three <org:street> elements that contain the organization's street address.
 - + An <org:city> element that contains the organization's city.
 - + An OPTIONAL <org:sp> element that contains the organization's state or province.
 - + An OPTIONAL <org:pc> element that contains the organization's postal code.
 - + An <org:cc> element that contains the alpha-2 organization's country code. The detailed format of this element is described in section 2.4.3 of [RFC5733].

- o An OPTIONAL <org:voice> element that contains the organization's voice telephone number. The detailed format of this element is described in Section 2.5 of [RFC5733]
- o An OPTIONAL <org:fax> element that contains the organization's facsimile telephone number.
- o An OPTIONAL <org:email> element that contains the organization's email address. The detailed format of this element is described in section 2.6 of [RFC5733].
- o An OPTIONAL <org:url> element that contains the URL to the website of the organization. The detailed format of this element is described in [RFC3986].
- o Zero or more <org:contact> elements that contain identifiers for the contact objects associated with the organization object.

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:      <org:create
C:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
C:          <org:id>res1523</org:id>
C:          <org:role>
C:            <org:type>reseller</org:type>
C:          </org:role>
C:          <org:parentId>1523res</org:parentId>
C:          <org:postalInfo type="int">
C:            <org:name>Example Organization Inc.</org:name>
C:            <org:addr>
C:              <org:street>123 Example Dr.</org:street>
C:              <org:street>Suite 100</org:street>
C:              <org:city>Dulles</org:city>
C:              <org:sp>VA</org:sp>
C:              <org:pc>20166-6503</org:pc>
C:              <org:cc>US</org:cc>
C:            </org:addr>
C:          </org:postalInfo>
C:          <org:voice x="1234">+1.7035555555</org:voice>
C:          <org:fax>+1.7035555556</org:fax>
C:          <org:email>contact@organization.example</org:email>
C:          <org:url>https://organization.example</org:url>
C:          <org:contact type="admin">sh8013</org:contact>
C:          <org:contact type="billing">sh8013</org:contact>
C:        </org: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 <org:creData> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:creData> element contains the following child elements:

- o An <org:id> element that contains the server-unique identifier for the created organization, as defined in Section 3.1.
- o An <org:crDate> element that contains the date and time of organization-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 lang="en">Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <org:creData
S:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
S:        <org:id>res1523</org:id>
S:        <org:crDate>1999-04-03T22:00:00.0Z</org:crDate>
S:      </org: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 an organization object. In addition to the standard EPP command elements, the <delete> command MUST contain an <org:delete> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:delete> element MUST contain the following child element:

- o An <org:id> element that contains the server-unique identifier of the organization object to be deleted, as defined in Section 3.1.

An organization object MUST NOT be deleted if it is associated with other known objects. An associated organization MUST NOT be deleted until associations with other known objects have been broken. A server MUST 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:      <org:delete
C:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
C:          <org:id>res1523</org:id>
C:        </org: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 lang="en">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 organization objects, so there is no mapping defined for the EPP <renew> command.

4.2.4. EPP <transfer> Command

Transfer semantics do not apply to organization 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 an organization object. In addition to the standard EPP command elements, the <update> command MUST contain a <org:update> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:update> element contains the following child elements:

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

At least one <org:add>, <org:rem> or <org:chg> 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 OPTIONAL <org:add> and <org:rem> elements contain the following child elements:

- o Zero or more <org:contact> elements that contain the identifiers for contact objects to be associated with or removed from the organization object. Contact object identifiers MUST be known to the server before the contact object can be associated with the organization object.
- o Zero or more <org:role> elements that contain the role type, role statuses and optional role id of the organization.
 - * An <org:type> element that contains the role type of the organization, as defined in Section 3.2. The role type uniquely identifies the role to update.
 - * Zero or more <org:status> elements that contain the role statuses. The values of the role status are defined in Section 3.5.
 - * An OPTIONAL <org:roleID> element that contains a third-party-assigned identifier, such as IANA ID for registrars, as defined in Section 3.2.3.

- o Zero or more <org:status> elements that contain the operational status of the organization.

An OPTIONAL <org:chg> element contains the following child elements, where at least one child element MUST be present:

- o An OPTIONAL <org:parentId> element that contains the identifier of the parent object.
- o Zero to two <org: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 Unicode in the range U+0020 - U+007E. If a localized form (type="loc") is provided, element content MAY be represented in unrestricted UTF-8. The change of the postal info is defined as a replacement of that postal info element with the contents of the sub-elements included in the update command. An empty <org:postalInfo> element is supported to allow a type of postal info to be removed. The <org:postalInfo> element contains the following child elements:
 - * An <org:name> element that contains the name of the organization.
 - * An OPTIONAL <org:addr> element that contains address information associated with the organization. A <org:addr> element contains the following child elements:
 - + One, two, or three <org:street> elements that contain the organization's street address.
 - + An <org:city> element that contains the organization's city.
 - + An OPTIONAL <org:sp> element that contains the organization's state or province.
 - + An OPTIONAL <org:pc> element that contains the organization's postal code.
 - + An <org:cc> element that contains the alpha-2 organization's country code. The detailed format of this element is described in section 2.4.3 of [RFC5733].
- o An OPTIONAL <org:voice> element that contains the organization's voice telephone number. The detailed format of this element is described in Section 2.5 of [RFC5733]

- o An OPTIONAL <org:fax> element that contains the organization's facsimile telephone number.
- o An OPTIONAL <org:email> element that contains the organization's email address. The detailed format of this element is described in section 2.6 of [RFC5733].
- o An OPTIONAL <org:url> element that contains the URL to the website of the organization. The detailed format of this element is described in [RFC3986]

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:      <org:update>
C:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
C:          <org:id>res1523</org:id>
C:          <org:add>
C:            <org:contact type="tech">sh8013</org:contact>
C:            <org:role>
C:              <org:type>privacyproxy</org:type>
C:              <org:status>clientLinkProhibited</org:status>
C:            </org:role>
C:            <org:status>clientLinkProhibited</org:status>
C:          </org:add>
C:          <org:rem>
C:            <org:contact type="billing">sh8014</org:contact>
C:            <org:role>
C:              <org:type>reseller</org:type>
C:            </org:role>
C:          </org:rem>
C:          <org:chg>
C:            <org:postalInfo type="int">
C:              <org:addr>
C:                <org:street>124 Example Dr.</org:street>
C:                <org:street>Suite 200</org:street>
C:                <org:city>Dulles</org:city>
C:                <org:sp>VA</org:sp>
C:                <org:pc>20166-6503</org:pc>
C:                <org:cc>US</org:cc>
C:              </org:addr>
C:            </org:postalInfo>
C:            <org:voice>+1.7034444444</org:voice>
C:            <org:fax/>
C:          </org:chg>
C:        </org: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 lang="en">Command completed successfully</msg>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

4.3. Offline Review of Requested Actions

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

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

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1001">
S:      <msg lang="en">Command completed successfully;
S:        action pending</msg>
S:    </result>
S:    <resData>
S:      <org:creData
S:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
S:        <org:id>res1523</org:id>
S:        <org:crDate>1999-04-03T22:00:00.0Z</org:crDate>
S:      </org:creData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The status of the organization object after returning this response MUST include "pendingCreate". The server operator reviews the request offline, and informs the client of the outcome of the review by queuing a service message for retrieval via the <poll> command; it MAY additionally use an out-of-band mechanism to inform the client of the outcome.

The service message MUST contain text that describes the notification in the child <msg> element of the response <msgQ> element. In addition, the EPP <resData> element MUST contain a child <org:panData> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:panData> element contains the following child elements:

- o An <org:id> element that contains the server-unique identifier of the organization object. The <org:id> element contains a REQUIRED "paResult" attribute. A positive boolean value indicates that the request has been approved and completed. A negative boolean value indicates that the request has been denied and the requested action has not been taken.
- o An <org:paTRID> element that contains the client transaction identifier and server transaction identifier returned with the original response to process the command. The client transaction

identifier is OPTIONAL and will only be returned if the client provided an identifier with the original <create> command.

- o An <org:paDate> element that contains the date and time describing when review of the requested action was completed.

Example "review completed" service message:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg lang="en">Command completed successfully;
S:        ack to dequeue</msg>
S:    </result>
S:    <msgQ count="5" id="12345">
S:      <qDate>1999-04-04T22:01:00.0Z</qDate>
S:      <msg>Pending action completed successfully.</msg>
S:    </msgQ>
S:    <resData>
S:      <org:panData
S:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
S:        <org:id paResult="1">res1523</org:id>
S:        <org:paTRID>
S:          <clTRID>ABC-12345</clTRID>
S:          <svTRID>54321-XYZ</svTRID>
S:        </org:paTRID>
S:        <org:paDate>1999-04-04T22:00:00.0Z</org:paDate>
S:      </org:panData>
S:    </resData>
S:    <trID>
S:      <clTRID>BCD-23456</clTRID>
S:      <svTRID>65432-WXY</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5. Formal Syntax

An EPP object 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:epp:org-1.0"
  xmlns:org="urn:ietf:params:xml:ns:epp:org-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"/>
  <import namespace="urn:ietf:params:xml:ns:epp-1.0"/>

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

  <!--
  Child elements found in EPP commands.
  -->
  <element name="create" type="org:createType"/>
  <element name="delete" type="org:sIDType"/>
  <element name="update" type="org:updateType"/>
  <element name="check" type="org:mIDType"/>
  <element name="info" type="org:infoType"/>
  <element name="panData" type="org:panDataType"/>

  <!--
  Utility types.
  -->
  <simpleType name="statusType">
    <restriction base="token">
      <enumeration value="ok"/>
      <enumeration value="hold"/>
      <enumeration value="terminated"/>
      <enumeration value="clientDeleteProhibited"/>
      <enumeration value="clientUpdateProhibited"/>
      <enumeration value="clientLinkProhibited"/>
      <enumeration value="linked"/>
      <enumeration value="pendingCreate"/>
      <enumeration value="pendingUpdate"/>
      <enumeration value="pendingDelete"/>
    </restriction>
  </simpleType>
</schema>
```

```
        <enumeration value="serverDeleteProhibited"/>
        <enumeration value="serverUpdateProhibited"/>
        <enumeration value="serverLinkProhibited"/>
    </restriction>
</simpleType>

<simpleType name="roleStatusType">
    <restriction base="token">
        <enumeration value="ok"/>
        <enumeration value="clientLinkProhibited"/>
        <enumeration value="linked"/>
        <enumeration value="serverLinkProhibited"/>
    </restriction>
</simpleType>

<complexType name="roleType">
    <sequence>
        <element name="type" type="token"/>
        <element name="status" type="org:roleStatusType"
            minOccurs="0" maxOccurs="3"/>
        <element name="roleID" type="token" minOccurs="0"/>
    </sequence>
</complexType>

<complexType name="postalInfoType">
    <sequence>
        <element name="name"
            type="org:postalLineType"/>
        <element name="addr"
            type="org:addrType" minOccurs="0"/>
    </sequence>
    <attribute name="type"
        type="org:postalInfoEnumType"
        use="required"/>
</complexType>

<complexType name="contactType">
    <simpleContent>
        <extension base="eppcom:clIDType">
            <attribute name="type" type="org:contactAttrType"
                use="required"/>
            <attribute name="typeName" type="token"/>
        </extension>
    </simpleContent>
</complexType>

<simpleType name="contactAttrType">
    <restriction base="token">
```



```
        <enumeration value="admin"/>
        <enumeration value="billing"/>
        <enumeration value="tech"/>
        <enumeration value="abuse"/>
        <enumeration value="custom"/>
    </restriction>
</simpleType>

<complexType name="e164Type">
    <simpleContent>
        <extension base="org:e164StringType">
            <attribute name="x" type="token" />
        </extension>
    </simpleContent>
</complexType>

<simpleType name="e164StringType">
    <restriction base="token">
        <pattern value="(\+[0-9]{1,3}\.[0-9]{1,14})?" />
        <maxLength value="17" />
    </restriction>
</simpleType>

<simpleType name="postalLineType">
    <restriction base="normalizedString">
        <minLength value="1" />
        <maxLength value="255" />
    </restriction>
</simpleType>

<simpleType name="optPostalLineType">
    <restriction base="normalizedString">
        <maxLength value="255" />
    </restriction>
</simpleType>

<simpleType name="pcType">
    <restriction base="token">
        <maxLength value="16" />
    </restriction>
</simpleType>

<simpleType name="ccType">
    <restriction base="token">
        <length value="2" />
    </restriction>
</simpleType>
```

```

<complexType name="addrType">
  <sequence>
    <element name="street" type="org:optPostalLineType"
      minOccurs="0" maxOccurs="3" />
    <element name="city" type="org:postalLineType" />
    <element name="sp" type="org:optPostalLineType"
      minOccurs="0" />
    <element name="pc" type="org:pcType"
      minOccurs="0" />
    <element name="cc" type="org:ccType" />
  </sequence>
</complexType>

<simpleType name="postalInfoEnumType">
  <restriction base="token">
    <enumeration value="loc" />
    <enumeration value="int" />
  </restriction>
</simpleType>

<!--
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>

<!--
Pending action notification response elements.
-->
<complexType name="panDataType">
  <sequence>
    <element name="id" type="org:paCLIDType"/>
    <element name="paTRID" type="epp:trIDType"/>
    <element name="paDate" type="dateTime"/>
  </sequence>

```

```
</complexType>

<complexType name="paCLIDType">
  <simpleContent>
    <extension base="eppcom:clIDType">
      <attribute name="paResult" type="boolean"
        use="required"/>
    </extension>
  </simpleContent>
</complexType>

<!--
Child elements of the <info> commands.
-->
<complexType name="infoType">
  <sequence>
    <element name="id"
      type="eppcom:clIDType"/>
  </sequence>
</complexType>

<!--
Child elements of the <create> command.
-->
<complexType name="createType">
  <sequence>
    <element name="id"
      type="eppcom:clIDType"/>
    <element name="role"
      type="org:roleType" maxOccurs="unbounded"/>
    <element name="status"
      type="org:statusType" minOccurs="0" maxOccurs="4"/>
    <element name="parentId"
      type="eppcom:clIDType" minOccurs="0"/>
    <element name="postalInfo"
      type="org:postalInfoType" minOccurs="0" maxOccurs="2"/>
    <element name="voice"
      type="org:e164Type" minOccurs="0"/>
    <element name="fax"
      type="org:e164Type" minOccurs="0"/>
    <element name="email"
      type="eppcom:minTokenType" minOccurs="0"/>
    <element name="url"
      type="anyURI" minOccurs="0"/>
    <element name="contact"
      type="org:contactType"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
```

```
</complexType>

<!--
Child elements of the <update> command.
-->
<complexType name="updateType">
  <sequence>
    <element name="id"
      type="eppcom:clIDType"/>
    <element name="add"
      type="org:addRemType" minOccurs="0"/>
    <element name="rem"
      type="org:addRemType" minOccurs="0"/>
    <element name="chg"
      type="org:chgType" minOccurs="0"/>
  </sequence>
</complexType>

<!--
Data elements that can be added or removed.
-->
<complexType name="addRemType">
  <sequence>
    <element name="contact"
      type="org:contactType" minOccurs="0" maxOccurs="unbounded"/>
    <element name="role" type="org:roleType"
      minOccurs="0" maxOccurs="unbounded"/>
    <element name="status" type="org:statusType"
      minOccurs="0" maxOccurs="9"/>
  </sequence>
</complexType>

<!--
Data elements that can be changed.
-->
<complexType name="chgType">
  <sequence>
    <element name="parentId"
      type="eppcom:clIDType" minOccurs="0"/>
    <element name="postalInfo"
      type="org:chgPostalInfoType"
      minOccurs="0" maxOccurs="2"/>
    <element name="voice"
      type="org:e164Type" minOccurs="0"/>
    <element name="fax"
      type="org:e164Type" minOccurs="0"/>
    <element name="email"
      type="eppcom:minTokenType" minOccurs="0"/>
  </sequence>
</complexType>
```

```
        <element name="url"
            type="anyURI" minOccurs="0"/>
    </sequence>
</complexType>

<complexType name="chgPostalInfoType">
    <sequence>
        <element name="name"
            type="org:postalLineType" minOccurs="0"/>
        <element name="addr"
            type="org:addrType" minOccurs="0"/>
    </sequence>
    <attribute name="type"
        type="org:postalInfoEnumType" use="required"/>
</complexType>

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

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

<complexType name="checkType">
    <sequence>
        <element name="id" type="org:checkIDType" />
        <element name="reason" type="eppcom:reasonType"
            minOccurs="0" />
    </sequence>
</complexType>

<complexType name="checkIDType">
    <simpleContent>
        <extension base="eppcom:clIDType">
            <attribute name="avail" type="boolean"
                use="required" />
        </extension>
    </simpleContent>
</complexType>
```

```
</complexType>

<!--
<info> response elements.
-->
<complexType name="infDataType">
  <sequence>
    <element name="id"
      type="eppcom:clIDType"/>
    <element name="roid"
      type="eppcom:roidType"/>
    <element name="role"
      type="org:roleType" maxOccurs="unbounded"/>
    <element name="status"
      type="org:statusType" maxOccurs="9"/>
    <element name="parentId"
      type="eppcom:clIDType" minOccurs="0"/>
    <element name="postalInfo"
      type="org:postalInfoType" minOccurs="0" maxOccurs="2"/>
    <element name="voice"
      type="org:e164Type" minOccurs="0"/>
    <element name="fax"
      type="org:e164Type" minOccurs="0"/>
    <element name="email"
      type="eppcom:minTokenType" minOccurs="0"/>
    <element name="url"
      type="anyURI" minOccurs="0"/>
    <element name="contact"
      type="org:contactType" minOccurs="0" maxOccurs="unbounded"/>
    <element name="clID"
      type="eppcom:clIDType" minOccurs="0"/>
    <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"/>
  </sequence>
</complexType>

<!--
<create> response elements.
-->
<complexType name="creDataType">
  <sequence>
    <element name="id" type="eppcom:clIDType" />
    <element name="crDate" type="dateTime" />
  </sequence>
</complexType>
```

```
    </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 [RFC3629] and UTF-16 [RFC2781]. 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 organization object mapping, the elements and element content described in this document MUST inherit the internationalization conventions used to represent higher-layer domain and core protocol structures present in an XML instance that includes this extension.

7. IANA Considerations

7.1. XML Namespace

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

Registration request for the organization namespace:

URI: urn:ietf:params:xml:ns:epp:org-1.0

Registrant Contact: IESG

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

Registration request for the organization XML schema:

URI: urn:ietf:params:xml:schema:epp:org-1.0

Registrant Contact: IESG

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

7.2. EPP Extension Registry

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

Name of Extension: Extensible Provisioning Protocol (EPP)
Organization Mapping

Document status: Standards Track

Reference: RFCXXXX (please replace "XXXX" with the RFC number for this document after a number is assigned by the RFC Editor)

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

7.3. Role Type Values Registry

IANA has created a new category of protocol registry for values of the organization roles. The name of this registry is "EPP Organization Role Values". The registration policy for this registry is "Expert Review" [RFC8126].

7.3.1. Registration Template

Value: the string value being registered.

Description: Brief description of the organization role values.

Registrant Name: For IETF RFCs, state "IESG". For others, give the name of the responsible party.

Registrant Contact Information: an email address, postal address, or some other information to be used to contact the registrant.

7.3.2. Initial Registry Contents

Followings are the initial registry contents:

Value: registrar

Description: The entity object instance represents the authority responsible for the registration in the registry.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: reseller

Description: The entity object instance represents a third party through which the registration was conducted (i.e., not the registry or registrar).

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: privacyproxy

Description: The entity object instance represents a third-party who could help to register a domain without exposing the registrants' private information.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: dns-operator

Description: The entity object instance represents a third-party DNS operator that maintains the name servers and zone data on behalf of a registrant.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

8. Implementation Status

Note to RFC Editor: Please remove this section and the reference to [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 [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 [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".

8.1. 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-org.

Level of maturity: Development

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

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

8.2. CNNIC Implementation

Organization: CNNIC

Name: EPP Organization Mapping

Description: CNNIC is trying to update EPP organization mapping from previous reseller mapping according to this document.

Level of maturity: Development

Coverage: EPP organization mapping

Contact: zhouguiqing@cnnic.cn

9. Security Considerations

The organization object may have personally identifiable information, such as <org:contact>. This information is not a required element in this document which can be provided on a voluntary basis. If it is provided, both client and server MUST ensure that authorization information is stored and exchanged with high-grade encryption mechanisms to provide privacy services, which is specified in [RFC5733]. The security considerations described in [RFC5730] or those caused by the protocol layers used by EPP will apply to this specification as well.

10. Acknowledgment

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

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>>.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <<https://www.rfc-editor.org/info/rfc3629>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [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>>.

- [RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.17487/RFC5733, August 2009, <<https://www.rfc-editor.org/info/rfc5733>>.
- [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>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [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 First Edition REC-xml-20040204, February 2004, <<http://www.w3.org/TR/2004/REC-xml-20040204>>.
- [W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028, October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>>.
- [W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>>.

11.2. Informative References

- [RFC2781] Hoffman, P. and F. Yergeau, "UTF-16, an encoding of ISO 10646", RFC 2781, DOI 10.17487/RFC2781, February 2000, <<https://www.rfc-editor.org/info/rfc2781>>.
- [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 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.
- * Refactoried the XSD file. Added <chgPostalInfoType> element.
- * Added acknowledgment.

WG document-00: WG document submitted

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

Organization WG document-00: Change to a generic organization object mapping.

Organization WG document-01: Added "Implementation Status" section.

Organization WG document-02: Accepted some of the feedbacks on the mailing list.

Organization WG document-03:

- * Updated section 3.2, changed the structure of organization role.
- * Updated section 4.2.5 for the "add", "rem" and "chg" example.
- * Updated section 5 of formal syntax.
- * Updated section 7.2 for the registration template and initial values.
- * Updated section 8 of implementation status.

Organization WG document-04:

- * Updated section 3.2, changed the structure of organization role.
- * Updated references.
- * Updated section 8 of implementation status.

Organization WG document-05:

- * Updated the description of <org:status> of a role.
- * Removed the third paragraph of "Implementation Status".
- * Remove the Informative Reference to draft-ietf-regext-reseller from the draft.

Organization WG document-06:

- * Updated typos.
- * Added "Query" for "<Transfer> Query Command".

- * Change "Registrant Contact" to IESG in section 7.1.
- * Modified section 7.2.

Organization WG document-07:

- * Updated typos.
- * Added dns-operator in section 7.1.
- * Added "OPTIONAL" for <org:addr>

Organization WG document-08:

- * Updated "Offline Review of Requested Actions".

Organization WG document-09:

- * Updated "This element or its ancestor element MUST identify the organization namespace." in section 4.1.1 and other parts of this document.
- * Updated text in section 2 match RFC 8174.
- * Modified "roleid" to "roleID".
- * Updated text about loops in section 3.6.
- * Referred section 2.5 of RFC5733 for voice format.
- * Updated XML schema for the maxOccurs value of "reason" element.
- * Updated section 7.3.
- * Replaced "http" with "https" in the examples.
- * Updated writing typos.
- * Modified XML namespace and schema.

Organization WG document-10:

- * Modified XML namespace and schema.
- * Removed the maxOccurs value of "reason" element.

Organization WG document-11:

- * Typo of RFC2781 and moved this reference in "Informative References".
- * "Loops MUST be prohibited." in section 3.6.

Organization WG document-12:

- * Removed "OPTIONAL" when "zero or more" or "zero to two" appears.
- * Updated the "Organization Status Values" text.
- * Updated the full xml namespace.
- * Updated the text in "Offline review".
- * Updated the text in "Security Considerations".
- * Added "Document satus" and "Reference" in section "EPP Extension Registry".
- * Added references of RFC3688, RFC3986 and RFC5646.

Authors' Addresses

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

Email: zhoulinlin@cnnic.cn

Ning Kong
Consultant

Email: ietfing@gmail.com

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

Email: zhouguiqing@cnnic.cn

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

Email: yaojk@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 8, 2019

L. Zhou
CNNIC
N. Kong
Consultant
J. Wei
J. Yao
CNNIC
J. Gould
Verisign, Inc.
December 5, 2018

Organization Extension for the Extensible Provisioning Protocol (EPP)
draft-ietf-regext-org-ext-11

Abstract

This document describes an extension to Extensible Provisioning Protocol (EPP) object mappings, which is designed to support assigning an organization to any existing object (domain, host, contact) as well as any future 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 8, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions Used in This Document	3
3. Object Attributes	4
3.1. Organization Identifier	4
4. EPP Command Mapping	4
4.1. EPP Query Commands	4
4.1.1. EPP <check> Command	4
4.1.2. EPP <info> Command	4
4.1.3. EPP <transfer> Query Command	7
4.2. EPP Transform Commands	8
4.2.1. EPP <create> Command	8
4.2.2. EPP <delete> Command	10
4.2.3. EPP <renew> Command	10
4.2.4. EPP <transfer> Command	11
4.2.5. EPP <update> Command	11
5. Formal Syntax	15
6. Internationalization Considerations	18
7. IANA Considerations	18
7.1. XML Namespace	18
7.2. EPP Extension Registry	18
8. Implementation Status	19
8.1. Verisign EPP SDK	19
8.2. CNNIC Implementation	20
9. Security Considerations	20
10. Acknowledgment	20
11. References	20
11.1. Normative References	20
11.2. Informative References	22
Appendix A. Change Log	22
Authors' Addresses	25

1. Introduction

In the business model of domain registration, we usually have three roles of entities: a registrant, a registrar and a registry, as defined in section 9 of [ID.draft-ietf-dnsop-terminology-bis]. There may be other roles of entities involved in the domain registration process, such as resellers, DNS operators in section 9 of [ID.draft-ietf-dnsop-terminology-bis], privacy proxies, etc.

A domain reseller is an individual or a company that acts as an agent for accredited registrars. DNS operator is defined in section 9 of [ID.draft-ietf-dnsop-terminology-bis]. A privacy proxy is an entity used for domain registrations to protect the private information of the individuals and organizations. These kind of entities are defined as "organizations" with different role types in this document.

In order to facilitate provisioning and management of organization information in a shared central repository, this document proposes an organization extension mapping for any Extensible Provisioning Protocol (EPP) object like domain names in [RFC5731], hosts in [RFC5732] and contacts in [RFC5733]. The examples provided in this document are used for the domain object for illustration purpose. The host and contact object could be extended in the same way with the domain object.

Organization object identifiers defined in [ID.draft-ietf-regext-org] MUST be known to the server before the organization object can be associated with the EPP object.

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 BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

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.

The XML namespace prefix "orgext" is used for the namespace "urn:ietf:params:xml:ns:epp:orgext-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.

3. Object Attributes

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

3.1. Organization Identifier

Organization identifier provides the ID of an organization. Its corresponding element is `<orgext:id>` which refers to the `<org:id>` element defined in [ID.draft-ietf-regext-org]. All organization objects are identified by a server-unique identifier. A "role" attribute is used to represent the relationship that the organization has to the EPP object. Any given object MUST have at most one associated organization ID for any given role value.

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 assigning organizations to EPP objects.

4.1. EPP Query Commands

EPP provides three commands to retrieve EPP object information: `<check>` to determine if an object can be provisioned within a repository, `<info>` to retrieve detailed information associated with an object, and `<transfer>` to retrieve 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 object mapping.

4.1.2. EPP `<info>` Command

This extension does not add any elements to the EPP `<info>` command described in the EPP object mapping. However, additional elements are defined for the `<info>` response in the EPP object mapping.

When an `<info>` command has been processed successfully, the EPP `<resData>` element MUST contain child elements as described in the EPP object extensions. In addition, the EPP `<extension>` element SHOULD contain a child `<orgext:infData>` element. This element or its ancestor element MUST identify the extension namespace "urn:ietf:params:xml:ns:epp:orgext-1.0" if the object has data

associated with this extension and based on server policy. The <orgext:infData> element contains the following child elements:

- o Zero or more <orgext:id> elements are allowed that contain the identifier of the organization, as defined in Section 3.1. The "role" attribute is used to represent the relationship that the organization has to the object. See Section 7.3 in [ID.draft-ietf-regext-org] for a list of values.

Example <info> response for an authorized client with multiple organizations:

```
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
S:        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>jdl1234</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:      <orgext:infData
S:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
S:        <orgext:id role="reseller">reseller1523</orgext:id>
S:        <orgext:id role="privacyproxy">proxy2935</orgext:id>
S:      </orgext:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ngcl-IvJjzMZc</clTRID>
S:      <svTRID>test142AWQONJZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> response for an authorized client with no organization:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg lang="en-US">Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrar>jd1234</domain:registrar>
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:      <orgext:infData
S:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0"/>
S:      </extension>
S:    <trID>
S:      <clTRID>ngcl-IvJjzMZc</clTRID>
S:      <svTRID>test142AWQONJZ</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> Query Command

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

4.2. EPP Transform Commands

EPP provides five commands to transform EPP 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 the object sponsorship changes, and `<update>` to change information associated with an object.

4.2.1. EPP `<create>` Command

This extension defines additional elements for the EPP `<create>` command described in the EPP object extensions. No additional elements are defined for the EPP `<create>` response.

The EPP `<create>` command provides a transform operation that allows a client to create an object. In addition to the EPP command elements described in the EPP object extensions, the command MUST contain an `<extension>` element, and the `<extension>` element MUST contain a child `<orgext:create>` element. This element or its ancestor element MUST identify the extension namespace "urn:ietf:params:xml:ns:epp:orgext-1.0" if the client wants to associate data defined in this extension to the object. The `<orgext:create>` element contains the following child elements:

- o One or more `<orgext:id>` elements that contain the identifier of the organization, as defined in Section 3.1. The "role" attribute is used to represent the relationship that the organization has to the object. See Section 7.3 in [ID.draft-ietf-regext-org] for a list of values.

Example `<create>` Command with only one organization:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <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">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>fooBAR</domain:pw>
C:          </domain:authInfo>
C:        </domain:create>
C:      </create>
C:    <extension>
C:      <orgext:create
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:          <orgext:id role="reseller">reseller1523</orgext:id>
C:        </orgext:create>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <create> Command with multiple organizations:

```
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">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>fooBAR</domain:pw>
C:          </domain:authInfo>
C:        </domain:create>
C:      </create>
C:    <extension>
C:      <orgext:create
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:          <orgext:id role="reseller">reseller1523</orgext:id>
C:          <orgext:id role="privacyproxy">proxy2935</orgext:id>
C:        </orgext: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 object extension.

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 object mapping.

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 object mapping.

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 object mapping, but after a successful transfer of an object with an assigned organization, the handling of the assigned organization is dependent on the organization roles and server policy.

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 an object. In addition to the EPP <update> command elements, the command MUST contain an <extension> element, and the <extension> element MUST contain a child <orgext:update> element. This element or its ancestor element MUST identify the extension namespace "urn:ietf:params:xml:ns:epp:orgext-1.0" if the client wants to update the object with data defined in this extension. The <orgext:update> element contains the following child elements:

- o An OPTIONAL <orgext:add> element that contains one or more <orgext:id> elements, as defined in Section 3.1, that add non-existent organization roles to the object. The <orgext:id> element MUST have a non-empty organization identifier value. The server SHOULD validate that the <orgext:id> element role does not exist.
- o An OPTIONAL <orgext:rem> element that contains one or more <orgext:id> elements, as defined in Section 3.1, that remove organization roles from the object. The <orgext:id> element MAY have an empty organization identifier value. The server SHOULD validate the existence of the <orgext:id> element role and the organization identifier if provided.
- o An OPTIONAL <orgext:chg> element that contains one or more <orgext:id> elements, as defined in Section 3.1, that change organization role identifiers for the object. The existing organization identifier value will be replaced for the defined role. The server SHOULD validate the existence of the <orgext:id> element role.

At least one <orgext:add>, <orgext:rem> or <orgext:chg> element MUST be provided.

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>
C:        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:      <orgext:update>
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:          <orgext:add>
C:            <orgext:id role="reseller">reseller1523</orgext:id>
C:          </orgext:add>
C:        </orgext:update>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <update> command, adding multiple organizations:

```
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:update>
C:      </update>
C:    <extension>
C:      <orgext:update>
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:          <orgext:add>
C:            <orgext:id role="reseller">reseller1523</orgext:id>
C:            <orgext:id role="privacyproxy">proxy2935</orgext:id>
C:          </orgext:add>
C:        </orgext: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>
C:        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:    <orgext:update>
C:      xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:      <orgext:rem>
C:        <orgext:id role="reseller"/>
C:      </orgext:rem>
C:    </orgext:update>
C:  </extension>
C:  <clTRID>ABC-12345</clTRID>
C: </command>
C:</epp>
```

Example <update> command, removing multiple organizations:

```
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:update>
C:    </update>
C:  <extension>
C:    <orgext:update>
C:      xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:      <orgext:rem>
C:        <orgext:id role="reseller"/>
C:        <orgext:id role="privacyproxy"/>
C:      </orgext:rem>
C:    </orgext: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>
C:        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:      <orgext:update>
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:          <orgext:chg>
C:            <orgext:id role="reseller">reseller1523</orgext:id>
C:          </orgext:chg>
C:        </orgext:update>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <update> command, updating multiple organization identifiers:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update>
C:        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:      <orgext:update>
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:          <orgext:chg>
C:            <orgext:id role="reseller">reseller1523</orgext:id>
C:            <orgext:id role="privacyproxy">proxy2935</orgext:id>
C:          </orgext:chg>
C:        </orgext: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 object extension.

An EPP error response MUST be returned if an <update> command cannot be processed for any reason. An attempt to add one organization ID or multiple organization IDs with a particular role value when at least one of them already exists does not change the object at all. A server SHOULD notify clients that object relationships exist by sending a 2305 error response code. An attempt to remove an organization ID or multiple organization IDs with a particular role value when at least one of them does not exist does not change the object at all. A server SHOULD notify clients that object relationships does not exist by sending a 2305 error response code. An attempt to change an organization ID or multiple organization IDs with a particular role value when at least one of them does not exist does not change the object at all. A server SHOULD notify clients that object relationships does not exist by sending a 2305 error response code. Response format with error value elements is defined in Section 2.6 of [RFC5730].

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:epp:orgext-1.0"
  xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
>

  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
      Organization Extension Schema v1.0
    </documentation>
  </annotation>

  <!-- Child elements found in EPP commands. -->
```



```
<element
  name="create"
  type="orgext:createType"/>
<element
  name="update"
  type="orgext:updateType"/>

<!--
  Organization identifier with required role
-->
<complexType name="orgIdType">
  <simpleContent>
    <extension base="token">
      <attribute
        name="role"
        type="token"
        use="required"/>
    </extension>
  </simpleContent>
</complexType>

<!--
  Child elements of the <orgext:create> command
  All elements must be present at time of creation
-->
<complexType name="createType">
  <sequence>
    <!-- agent identifier or the organization,
         e.g. registrar, reseller, privacy proxy, etc. -->
    <element
      name="id"
      type="orgext:orgIdType"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>

<!--
  Child elements of <orgext:update> command
-->
<complexType name="updateType">
  <sequence>
    <element
      name="add"
      type="orgext:addRemChgType"
      minOccurs="0"
    />
    <element
      name="rem"
```

```
        type="orgext:addRemChgType"
        minOccurs="0"
    />
    <element
        name="chg"
        type="orgext:addRemChgType"
        minOccurs="0"
    />
</sequence>
</complexType>

<complexType name="addRemChgType">
    <sequence>
        <!-- agent identifier of the organization,
             e.g. registrar, reseller, privacy proxy, etc. -->
        <element
            name="id"
            type="orgext:orgIdType"
            maxOccurs="unbounded"/>
    </sequence>
</complexType>

<!-- Child response element -->
<element
    name="infData"
    type="orgext:infDataType"/>

<!-- <orgext:infData> response elements -->
<complexType name="infDataType">
    <sequence>
        <!-- agent identifier the organization,
             e.g. registrar, reseller, privacy proxy, etc. -->
        <element
            name="id"
            type="orgext:orgIdType"
            minOccurs="0"
            maxOccurs="unbounded"/>
    </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 object mapping, the elements, element content described in this document MUST inherit the internationalization conventions used to represent higher-layer domain and core protocol structures present in an XML instance that includes this extension.

7. IANA Considerations

7.1. XML Namespace

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

Registration request for the organization extension namespace:

URI: urn:ietf:params:xml:ns:epp:orgext-1.0

Registrant Contact: IESG

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

Registration request for the organization XML schema:

URI: urn:ietf:params:xml:schema:epp:orgext-1.0

Registrant Contact: IESG

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: Organization Extension for the Extensible Provisioning Protocol (EPP)

Document status: Standards Track

Reference: RFCXXXX (please replace "XXXX" with the RFC number for this document after a number is assigned by the RFC Editor)

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

8. Implementation Status

Note to RFC Editor: Please remove this section and the reference to [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 [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 [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".

8.1. 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-org-ext.

Level of maturity: Development

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

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

8.2. CNNIC Implementation

Organization: CNNIC

Name: Organization Extension for EPP

Description: CNNIC is trying to update organization extension from previous reseller extension according to this document.

Level of maturity: Development

Coverage: Organization extension for EPP

Contact: zhouguiqing@cnnic.cn

9. 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.

10. Acknowledgment

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

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>>.

- [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>>.
- [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>>.
- [RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.17487/RFC5733, August 2009, <<https://www.rfc-editor.org/info/rfc5733>>.
- [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>>.
- [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>>.

11.2. Informative References

[ID.draft-ietf-dnsop-terminology-bis]

Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", September 2018, <<http://tools.ietf.org/html/draft-ietf-dnsop-terminology-bis>>.

[ID.draft-ietf-regext-org]

Zhou, L., Kong, N., Zhou, G., Yao, J., and J. Gould, "Extensible Provisioning Protocol (EPP) Reseller Mapping", November 2018, <<http://tools.ietf.org/html/draft-ietf-regext-org>>.

[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 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 acknowledgment.
- * 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.

Organization WG document-00: Change to a generic organization object extension.

Organization WG document-01: Added "Implementation Status" section.

Organization WG document-02: Accepted some of the feedbacks on the mailing list. Modified the examples in the document.

Organization WG document-03:

- * Updated typos.
- * Changed some descriptions about <orgext:id> and role attribute.
- * Modified the example of "domain with no organization".
- * Updated section 8, adding implementation status of Verisign.

Organization WG document-04:

- * Updated typos.
- * Removed the example of <update> command, domain with no organization.
- * Updated references.

- * Updated section 8 of implementation status.

Organization WG document-05:

- * Removed the minOccurs="0" from the addRemChgType type of the XML schema
- * Removed the third paragraph of "Implementation Status".
- * Remove the Informative Reference to draft-ietf-regext-reseller-ext from the draft.

Organization WG document-06:

- * Updated "Abstraction".
- * Added "Query" for "<Transfer> Query Command".
- * Change "Registrant Contact" to IESG in section 7.1.
- * Modified section 7.2.

Organization WG document-07:

- * Updated "Abstraction".

Organization WG document-08:

- * Updated error codes of <update> response.
- * Modified XML namespace and schema.

Organization WG document-09:

- * Modified XML namespace and schema.
- * Changed "Exactly one" to "At least one" in section 4.2.5.

Organization WG document-10:

- * Updated the reseller id and dns proxy id in the document.
- * Updated the full xml namespace.
- * Updated the text of EPP <orgext:add>, <orgext:rem> and <orgext:chg>.

- * Added "Document satus" and "Reference" in section "EPP Extension Registry".

Organization WG document-11:

- * Added the reference of draft-ietf-dnsop-terminology-bis.

Authors' Addresses

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

Email: zhoulinlin@cnnic.cn

Ning Kong
Consultant

Email: ietfing@gmail.com

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

Email: weijunkai@cnnic.cn

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

Email: yaojk@cnnic.cn

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

Email: jgould@verisign.com

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: April 14, 2019

R. Carney
J. Snitker
GoDaddy Inc.
October 11, 2018

Validate Mapping for the Extensible Provisioning Protocol (EPP)
draft-ietf-regext-validate-04

Abstract

This document describes an Extensible Provisioning Protocol (EPP) 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 <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 14, 2019.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
1.1. Conventions Used in This Document	3
2. Object Attributes	3
2.1. Key Value	3
2.2. Validate Id	4
2.3. Validate PostalInfo, Voice, Fax, Email, AuthInfo	4
3. EPP Command Mapping	4
3.1. EPP Query Commands	4
3.1.1. EPP <check> Command	4
3.1.2. EPP <info> Command	8
3.1.3. EPP <transfer> Command	8
3.2. EPP Transform Commands	9
3.2.1. EPP <create> Command	9
3.2.2. EPP <delete> Command	9
3.2.3. EPP <renew> Command	9
3.2.4. EPP <transfer> Command	9
3.2.5. EPP <update> Command	9
4. Formal Syntax	9
4.1. Validate Schema	9
5. Security Considerations	13
6. IANA Considerations	13
6.1. XML Namespace	13
7. Implementation Status	13
7.1. To Be Filled In	14
8. Acknowledgements	14
9. Change History	14
9.1. Change from 03 to 04	14
9.2. Change from 02 to 03	14
9.3. Change from 01 to 02	14
9.4. Change from 00 to 01	14
9.5. Change from carney-regext 01 to ietf-regext 00	15
10. Normative References	15
Authors' Addresses	15

1. Introduction

This document describes a Validate 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", "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.

"validate" is used as an abbreviation for "urn:ietf:params:xml:ns:validate-0.2". The XML namespace prefix "validate" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

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

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

The XML namespace prefix above contains a version number, specifically "0.2". 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. Object Attributes

A EPP validation object has attributes and associated values that can be viewed by the client. This section describes each attribute type in detail.

2.1. Key Value

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.

o An example <validate:kv key="VATID" value="0123456789"/>.

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

2.2. Validate Id

The `<validate:id>` element is used in two different scenarios.

First if the `<validate:id>` is passed by itself with no other elements (e.g. `>validate:postalInfo</validate:postalInfo>`) then the client intent is that this is an already existing contact and the server should handle the request by looking up the associated data in its system and using that data to validate against.

Second scenario would be if the request includes additional elements then the server should treat the `<validate:id>` as a temporary contact handle and should not perform a look on the contact but use the data that is passed in the request to validate against.

2.3. Validate PostalInfo, Voice, Fax, Email, AuthInfo

These elements are intended to mirror the definitions in [RFC5733].

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 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

The EPP `<check>` command is used to validate a list of contact information. The `<check>` command MUST contain a `<validate:check>` element that identifies the validate namespace. The `<validate:check>` element contains the following child elements:

- o 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 has two required attributes: `contactType`, which describes the role (registrant, admin, tech,

billing, etc.) of the contact that the contact should be validated for; and tld, which provides the top level domain to be validated against. The <validate:contact> element contains the following child elements:

- o one <validate:id> element (as described in section 2.2).
- o an OPTIONAL <validate:postalInfo> element (as described in section 2.3).
- o an OPTIONAL <validate:voice> element (as described in section 2.3).
- o an OPTIONAL <validate:fax> element (as described in section 2.3).
- o an OPTIONAL <validate:email> element (as described in section 2.3).
- o an OPTIONAL <validate:authInfo> element (as described in section 2.3).
- o zero or more <validate:kv> elements (as described in section 2.1).

The following is an example <check> command where "sh8013" and "sh8014" are both "new" contacts and "sh8012" is an existing contact on the server.

```
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-0.2"
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: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:kv key="VAT" value="1234567890"/>
C:    </validate:contact>
C:    <validate:contact contactType="tech" tld="COM">
C:      <validate:id>sh8012</validate:id>
C:    </validate:contact>
C:    <validate:contact contactType="admin" tld="COM">
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:contact>
C:    <validate:contact contactType="billing" tld="COM">
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:contact>
C:  </validate:check>
```

```
C:    </check>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When the server receives a <check> command with a <validate:contact> element that contains only a <validate:id> element the server will process this as an existing contact. If the contact does not exist the server MUST return an EPP error response for that specific <validate:contact>.

When a <check> command has been processed successfully, the EPP <resData> element MUST contain a child <validate:chkData> element that identifies the validate namespace. The <validate:chkData> element MUST contain a <validate:cd> element for each <validate:check> element contained in the <check> command. The <validate:cd> element contains the following child elements:

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

The following is an example of the <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:      <validate:chkData
S:        xmlns:validate="urn:ietf:params:xml:ns:validate-0.2">
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

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

3.1.3. EPP <transfer> Command

Transfer semantics do not apply to validate 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 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

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

3.2.2. EPP <delete> Command

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

3.2.3. EPP <renew> Command

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

3.2.4. EPP <transfer> Command

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

3.2.5. EPP <update> Command

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

4. Formal Syntax

One schema is presented here that is the EPP Validate 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 Schema

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

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

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

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

BEGIN

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace="urn:ietf:params:xml:ns:validate-0.2"
  xmlns:validate="urn:ietf:params:xml:ns:validate-0.2"
  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
    </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="id"
        type="eppcom:clIDType" />
      <element name="postalInfo"
        type="contact:postalInfoType"
        minOccurs="0" maxOccurs="2" />
      <element name="voice"
        type="contact:el64Type" minOccurs="0" />
      <element name="fax"
        type="contact:el64Type" minOccurs="0" />
      <element name="email"
        type="eppcom:minTokenType" minOccurs="0"/>
      <element name="authInfo"
        type="contact:authInfoType"
        minOccurs="0"/>
      <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="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. Security Considerations

The mapping 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.

6. IANA Considerations

6.1. 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 validate namespace:

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

Registrant Contact: IESG

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

Registration request for the validate schema:

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

Registrant Contact: IESG

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

7. Implementation Status

Note to RFC Editor: Please remove this section and the reference to [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 [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 [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".

7.1. To Be Filled In

Add implementation details once available.

8. Acknowledgements

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

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

9. Change History

9.1. Change from 03 to 04

Removed the <validate:cd> element from the <check> command, moving all sub-elements to the <validate:contact> element to simplify. Also removed the <disclose> element as it was not needed in this context. Also updated references to current versions of documents.

9.2. Change from 02 to 03

Corrected some formatting issues.

9.3. Change from 01 to 02

Corrected some formatting issues.

9.4. 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 using the <check> command and response framework. Stubbed in an Implementation section for later documentation.

9.5. Change from carney-regext 01 to ietf-regext 00

Updated miscellaneous verbiage to reflect the change from an extension and changed to ietf naming as REGEXT WG will assume this work.

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, <<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>>.
- [RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.17487/RFC5733, August 2009, <<https://www.rfc-editor.org/info/rfc5733>>.
- [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>>.
- [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>>.

Authors' Addresses

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

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

Internet-Draft

Validate

October 2018

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

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

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: March 25, 2019

M. Loffredo
M. Martinelli
IIT-CNR/Registro.it
S. Hollenbeck
Verisign Labs
September 21, 2018

Registration Data Access Protocol (RDAP) Query Parameters for Result
Sorting and Paging
draft-loffredo-regext-rdap-sorting-and-paging-05

Abstract

The Registration Data Access Protocol (RDAP) does not include core functionality for clients to provide sorting and paging parameters for control of large result sets. This omission can lead to unpredictable server processing of queries and client processing of responses. This unpredictability can be greatly reduced if clients can provide servers with their preferences for managing response values. This document describes RDAP query extensions that allow clients to specify their preferences for sorting and paging result sets.

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 March 25, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	2
1.1. Conventions Used in This Document	4
2. RDAP Query Parameter Specification	4
2.1. Sorting and Paging Metadata	4
2.2. "count" Parameter	6
2.3. "sort" Parameter	7
2.3.1. Representing Sorting Links	11
2.4. "limit" and "offset" Parameters	12
2.4.1. Representing Paging Links	13
3. Negative Answers	14
4. RDAP Conformance	15
5. Implementation Considerations	15
5.1. Considerations about Paging Implementation	16
6. Implementation Status	19
6.1. IIT-CNR/Registro.it	19
6.2. Google Registry	19
7. Security Considerations	20
8. IANA Considerations	20
9. Acknowledgements	20
10. References	21
10.1. Normative References	21
10.2. Informative References	22
Appendix A. Change Log	24
Authors' Addresses	25

1. Introduction

The availability of functionality for result sorting and paging provides benefits to both clients and servers in the implementation of RESTful services [REST]. These benefits include:

- o reducing the server response bandwidth requirements;
- o improving server response time;
- o improving query precision and, consequently, obtaining more reliable results;
- o decreasing server query processing load;
- o reducing client response processing time.

Approaches to implementing features for result sorting and paging can be grouped into two main categories:

1. Sorting and paging are implemented through the introduction of additional parameters in the query string (i.e. ODATA protocol [OData-Part1]);
2. Information related to the number of results and the specific portion of the result set to be returned, in addition to a set of ready-made links for the result set scrolling, are inserted in the HTTP header of the request/response.

However, there are some drawbacks associated with use of the HTTP header. First, the header properties cannot be set directly from a web browser. Moreover, in an HTTP session, the information on the status (i.e. the session identifier) is usually inserted in the header or in the cookies, while the information on the resource identification or the search type is included in the query string. The second approach is therefore not compliant with the HTTP standard [RFC7230]. As a result, this document describes a specification based on use of query parameters.

Currently the RDAP protocol [RFC7482] defines two query types:

- o lookup: the server returns only one object;
- o search: the server returns a collection of objects.

While the lookup query does not raise issues in the management of large result sets, the search query can potentially generate a large result set that could be truncated according to the limits of the server. In addition, it is not possible to obtain the total number of the objects found that might be returned in a search query response [RFC7483]. Lastly, there is no way to specify sort criteria to return the most relevant objects at the beginning of the result set. Therefore, the client could traverse the whole result set to find the relevant objects or, due to truncation, could not find them at all.

The protocol described in this specification extends RDAP query capabilities to enable result sorting and paging, by adding new query parameters that can be applied to RDAP search path segments. The service is implemented using the Hypertext Transfer Protocol (HTTP) [RFC7230] and the conventions described in RFC 7480 [RFC7480].

The implementation of these parameters is technically feasible, as operators for counting, sorting and paging rows are currently supported by the major RDBMSs.

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 [RFC2119].

2. RDAP Query Parameter Specification

The new query parameters are OPTIONAL extensions of path segments defined in RFC 7482 [RFC7482]. They are as follows:

- o "count": a boolean value that allows a client to request the total number of objects found (that due to truncation can be different from the number of returned objects);
- o "sort": a string value that allows a client to request a specific sort order for the result set;
- o "limit" and "offset": numeric values that allow a client to request a specific portion of the entire result set.

Augmented Backus-Naur Form (ABNF) [RFC5234] is used in the following sections to describe the formal syntax of these new parameters.

2.1. Sorting and Paging Metadata

According to most advanced principles in REST design, collectively known as HATEOAS (Hypermedia as the Engine of Application State) ([HATEOAS]), a client entering a REST application through an initial URI should use the server-provided links to dynamically discover available actions and access the resources it needs. In this way, the client is not requested to have prior knowledge of the service and, consequently, to hard code the URIs of different resources. This would allow the server to make URI changes as the API evolves without breaking the clients. Definitively, a REST service should be self-descriptive as much as possible.

Therefore, the implementation of the query parameters described in this specification recommends servers to provide additional information in their responses about both the available sorting criteria and the possible pagination. Such information is collected in two new data structures named, respectively, "sorting_metadata" and "paging_metadata".

Obviously, both the new data structures are OPTIONAL because their presence in the response not only depends on the implementation of sorting and paging query capabilities but also on some situations related to the results. For example, it is quite natural to expect

that the "paging_metadata" section will not be present at the last result page when the server implements only the forward pagination.

The "sorting_metadata" structure contains the following fields:

- o "currentSort": the value of sort parameter as specified in the query string;
- o "availableSorts": an array of objects each one describing an available sorting criterion:
 - * "property": the name that can be used by the client to request the sorting criterion;
 - * "jsonPath": the JSON Path of the RDAP field corresponding to the property;
 - * "default": whether the sorting criterion is applied by default;
 - * "links": an array of links as described in RFC 8288 [RFC8288] containing the query string that applies the sorting criterion.

Both "currentSort" and "availableSorts" are OPTIONAL fields of the "sorting_metadata" structure. In particular, the "currentSort" field is provided when the query string contains a valid value for sort parameter, while the "availableSorts" field SHOULD be provided when the sort parameter is missing in the query string or when it is present and the server implements more than a sorting criterion for the RDAP object. At least the "property" field is REQUIRED in each item of the "availableSorts" array while the other fields are RECOMMENDED.

The "paging_metadata" structure contains the following fields:

- o "totalCount": a numeric value representing the total number of objects found;
- o "pageCount": a numeric value representing the number of objects returned in the current page;
- o "offset": a numeric value identifying the start of current page in the result set;
- o "nextOffset": a numeric value identifying the start of the next page in the result set or null if the result set has been completely scrolled;
- o "links": an array of links as described in RFC 8288 [RFC8288] containing the reference to next page.

Only the "pageCount" field is REQUIRED in the "paging_metadata" structure. The other fields appear when pagination occurs. In this specification, only the forward pagination is dealt because it is considered satisfactory in order to traverse the result set. If a server should also implement backward pagination, an appropriate field (e.g. "prevOffset") identifying the start of the previous page is RECOMMENDED. Finally, the "totalCount" field is provided if the query string contains the count parameter.

FOR DISCUSSION: Should the metadata described in this specification be part of a more general "metadata" section including other contents (e.g rate limits, information about the server, information about the response, metadata information related to other parameters)?

2.2. "count" Parameter

Currently the RDAP protocol does not allow a client to determine the total number of the results in a query response when the result set is truncated. This is rather inefficient because the user cannot evaluate the query precision and, at the same time, cannot receive information that could be relevant.

The count parameter provides additional functionality (Figure 1) that allows a client to request information from the server that specifies the total number of elements matching a particular search pattern.

`https://example.com/rdap/domains?name=*nr.com&count=true`

Figure 1: Example of RDAP query reporting the count parameter

The ABNF syntax is the following:

```
count = "count" EQ ( trueValue / falseValue )
trueValue = ("true" / "yes" / "1")
falseValue = ("false" / "no" / "0")
EQ = "="
```

A trueValue means that the server MUST provide the total number of the objects in the "totalCount" field of the "paging_metadata" section (Figure 2). A falseValue means that the server MUST NOT provide this number.

```
{
  "rdapConformance": [
    "rdap_level_0",
    "paging_level_0"
  ],
  ...
  "paging_metadata": {
    "totalCount": 73
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 2: Example of RDAP response with "paging_metadata" section containing the "totalCount" field

2.3. "sort" Parameter

The RDAP protocol does not provide any capability to specify results sort criteria. A server could implement a default sorting scheme according to the object class, but this feature is not mandatory and might not meet user requirements. Sorting can be addressed by the client, but this solution is rather inefficient. Sorting and paging features provided by the RDAP server could help avoid truncation of relevant results and allow for scrolling the result set using subsequent queries.

The sort parameter allows the client to ask the server to sort the results according to the values of one or more properties and according to the sort direction of each property. The ABNF syntax is the following:

```
sort = "sort" EQ sortItem *( "," sortItem )
sortItem = property-ref [ ":" ( "a" / "d" ) ]
```

"a" means that the ascending sort MUST be applied, "d" means that the descending sort MUST be applied. If the sort direction is absent, an ascending sort MUST be applied (Figure 3).

In the sort parameter ABNF syntax, property-ref represents a reference to a property of an RDAP object. Such a reference could be expressed by using a JSON Path. The JSON Path in a JSON document [RFC8259] is equivalent to the XPath [W3C.CR-xpath-31-20161213] in a XML document. For example, the JSON Path to select the value of the ASCII name inside an RDAP domain object is "\$.ldhName", where \$ identifies the root of the document (DOM). Another way to select a value inside a JSON document is the JSON Pointer [RFC6901]. While

JSON Path or JSON Pointer are both standard ways to select any value inside JSON data, neither is particularly easy to use (e.g. "\$.events[?(@.eventAction='registration')].eventDate" is the JSON Path expression of the registration date in a RDAP domain object).

Therefore, this specification provides a definition of property-ref in terms of RDAP properties. However, not all the RDAP properties are suitable to be used in sort criteria, such as:

- o properties providing service information (e.g. links, notices, remarks, etc.);
- o multivalued properties (e.g. status, roles, variants, etc.);
- o properties modeling relationships to other objects (e.g. entities).

On the contrary, some properties expressed as values of other properties (e.g. registration date) could be used in such a context.

In the following, a list of the proposed properties for sort criteria is presented. The properties are divided in two groups: object common properties and object specific properties.

- o Object common properties. Object common properties are derived from the merge of the "eventAction" and the "eventDate" properties. The following values of the sort parameter are defined:

- * registrationDate
- * reregistrationDate
- * lastChangedDate
- * expirationDate
- * deletionDate
- * reinstantiationDate
- * transferDate
- * lockedDate
- * unlockedDate

- o Object specific properties. With regard to the specific properties, some of them are already defined among the query paths. In the following the list of the proposed sorting properties, grouped by objects, is shown:

- * Domain: ldhName
- * Nameserver: ldhName, ipv4, ipv6.
- * Entity: fn, handle, org, email, voice, country, city.

In the following, the correspondence between the sorting properties and the RDAP fields is shown (Table 1):

Object class	Sorting property	RDAP property	Reference in RFC 7483	Reference in RFC 6350
Searchable objects	Common properties	eventAction values suffixed by "Date"	4.5.	
Domain	ldhName	ldhName	5.3.	
Nameserver	ldhName	ldhName	5.2.	
	ipV4	v4 ipAddress	5.2.	
	ipV6	v6 ipAddress	5.2.	
Entity	handle	handle	5.1.	
	fn	vcard fn	5.1.	6.2.1
	org	vcard org	5.1.	6.6.4
	voice	vcard tel with type="voice"	5.1.	6.4.1
	email	vcard email	5.1.	6.4.2
	country	country name in vcard adr	5.1.	6.3.1
	city	locality in vcard adr	5.1.	6.3.1

Table 1: Sorting properties definition

With regard to the definitions in Table 1, some further considerations must be made to disambiguate cases where the RDAP property is multivalued:

- o Even if a nameserver can have multiple IPv4 and IPv6 addresses, the most common configuration includes one address for each IP version. Therefore, the assumption of having a single IPv4 and/or IPv6 value for a nameserver cannot be considered too stringent.
- o With the exception of handle values, all the sorting properties defined for entity objects can be multivalued according to the definition of vCard as given in RFC6350 [RFC6350]. When more than a value is reported, sorting can be applied to the preferred value identified by the parameter pref="1".

Each RDAP provider MAY define other sorting properties than those shown in this document.

The "jsonPath" field in the "sorting_metadata" section is used to clarify the RDAP field the sorting property refers to. In the following, the mapping between the sorting properties and the JSON Paths of the RDAP fields is shown (Table 2). The JSON Paths are provided according to the Goessner v.0.8.0 specification ([GOESSNER-JSON-PATH]):

Object class	Sorting property	JSON Path
Searchable objects	registrationDate	"\$.domainSearchResults[*].events[?(@.eventAction=="registration")].eventDate"
	reregistrationDate	"\$.domainSearchResults[*].events[?(@.eventAction=="reregistration")].eventDate"
	lastChangedDate	"\$.domainSearchResults[*].events[?(@.eventAction=="lastChanged")].eventDate"
	expirationDate	"\$.domainSearchResults[*].events[?(@.eventAction=="expiration")].eventDate"
	deletionDate	"\$.domainSearchResults[*].events[?(@.eventAction=="deletion")].eventDate"
	reinstantiationDate	"\$.domainSearchResults[*].events[?(@.eventAction=="reinstantiation")].eventDate"
	transferDate	"\$.domainSearchResults[*].events[?(@.eventAction=="transfer")].eventDate"
	lockedDate	"\$.domainSearchResults[*].events[?(@.eventAction=="locked")].eventDate"
	unlockedDate	"\$.domainSearchResults[*].events[?(@.eventAction=="unlocked")].eventDate"
Domain	ldhName	\$.domainSearchResults[*].ldhName
Nameserver	ldhName	\$.nameserverSearchResults[*].ldhName
	ipV4	\$.nameserverSearchResults[*].ipAddresses.v4[0]
	ipV6	\$.nameserverSearchResults[*].ipAddresses.v6[0]
Entity	handle	\$.entitySearchResults[*].handle
	fn	\$.entitySearchResults[*].vcardArray[1][?(@[0]="fn")][3]
	org	\$.entitySearchResults[*].vcardArray[1][?(@[

	voice	0]="org")][3]
	email	\$.entitySearchResults[*].vcardArray[1][?(@[0]="tel" && @[1].type=="voice")][3]
	country	\$.entitySearchResults[*].vcardArray[1][?(@[0]="email")][3]
	city	\$.entitySearchResults[*].vcardArray[1][?(@[0]="adr")][3][6]
		\$.entitySearchResults[*].vcardArray[1][?(@[0]="adr")][3][3]

Table 2: Sorting properties - JSON Path Mapping

If the sort parameter reports an allowed sorting property, it MUST be provided in the "currentSort" field of the "sorting_metadata" structure.

https://example.com/rdap/domains?name=*nr.com&sort=ldhName

https://example.com/rdap/domains?name=*nr.com&sort=registrationDate:d

https://example.com/rdap/domains?name=*nr.com&sort=lockedDate,ldhName

Figure 3: Examples of RDAP query reporting the sort parameter

2.3.1. Representing Sorting Links

An RDAP server MAY use the "links" array of the "sorting_metadata" section to provide ready-made references [RFC8288] to the available sort criteria (Figure 4). Each link represents a reference to an alternate view of the results.

```

{
  "rdapConformance": [
    "rdap_level_0",
    "sorting_level_0"
  ],
  ...
  "sorting_metadata": {
    "currentSort": "ldhName",
    "availableSorts": [
      {
        "property": "registrationDate",
        "jsonPath": "$.domainSearchResults[*].events[?(@.eventAction==\"registratio
n\")].eventDate",
        "default": false,
        "links": [
          {
            "value": "https://example.com/rdap/domains?name=*nr.com
&sort=ldhName",
            "rel": "alternate",
            "href": "https://example.com/rdap/domains?name=*nr.com
&sort=registrationDate",
            "title": "Result Ascending Sort Link",
            "type": "application/rdap+json"
          },
          {
            "value": "https://example.com/rdap/domains?name=*nr.com
&sort=ldhName",
            "rel": "alternate",
            "href": "https://example.com/rdap/domains?name=*nr.com
&sort=registrationDate:d",
            "title": "Result Descending Sort Link",
            "type": "application/rdap+json"
          }
        ]
      }
    ],
    "domainSearchResults": [
      ...
    ]
  }
}

```

Figure 4: Example of a "sorting_metadata" instance to implement result sorting

2.4. "limit" and "offset" Parameters

An RDAP query could return a response with hundreds of objects, especially when partial matching is used. For that reason, two parameters addressing result pagination are defined to make responses easier to handle:

- o "limit": means that the server MUST return the first N objects of the result set in the response;
- o "offset": means that the server MUST skip the first N objects and MUST return objects starting from position N+1.

The ABNF syntax is the following:

```
EQ = "="  
limit = "limit" EQ positive-number  
offset = "offset" EQ positive-number  
positive-number = non-zero-digit *digit  
non-zero-digit = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" /  
"9"  
digit = "0" / non-zero-digit
```

When limit and offset are used together, they allow implementation of result pagination. The following examples illustrate requests to return, respectively, the first 5 objects, the set of objects starting from position 6, and first 5 objects starting from position 11 of the result set (Figure 5).

```
https://example.com/rdap/domains?name=*nr.com&limit=5
```

```
https://example.com/rdap/domains?name=*nr.com&offset=5
```

```
https://example.com/rdap/domains?name=*nr.com&limit=5&offset=10
```

Figure 5: Examples of RDAP query reporting the limit and offset parameters

2.4.1. Representing Paging Links

An RDAP server MAY use the "links" array of the "paging_metadata" section to provide a ready-made reference [RFC8288] to the next page of the result set (Figure 6). Examples of additional "rel" values are "first", "last", "prev".


```
{
  "rdapConformance": [
    "rdap_level_0",
    "paging_level_0"
  ],
  ...
  "notices": [
    {
      "title": "Search query limits",
      "type": "result set truncated due to excessive load",
      "description": [
        "search results for domains are limited to 10"
      ]
    }
  ],
  "paging_metadata": {
    "totalCount": 73,
    "pageCount": 10,
    "offset": 10,
    "nextOffset": 20,
    "links": [
      {
        "value": "https://example.com/rdap/domains?name=*nr.com",
        "rel": "next",
        "href": "https://example.com/rdap/domains?name=*nr.com&limit=10
              &offset=10",
        "title": "Result Pagination Link",
        "type": "application/rdap+json"
      }
    ]
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 6: Example of a "paging_metadata" instance to implement result pagination based on offset and limit

3. Negative Answers

The value constraints for the parameters are defined by their ABNF syntax. Therefore, each request providing an invalid value for a parameter SHOULD obtain an HTTP 400 (Bad Request) response code. The same response SHOULD be returned if the client provides an unsupported value for the sort parameter in both single and multi sort.

The server can provide a different response when it supports the limit and/or offset parameters and the client submits values that are out of the valid ranges. The possible cases are:

- o If the client submits a value for the limit parameter that is greater than the number of objects to be processed, it is RECOMMENDED that server returns a response including only the processed objects.
- o If the client submits a value for the offset parameter that is greater than the number of objects to be processed, it is RECOMMENDED that server returns an HTTP 404 (Not Found) response code.

Optionally, the response MAY include additional information regarding the negative answer in the HTTP entity body.

4. RDAP Conformance

Servers returning the "paging_metadata" section in their responses MUST include "paging_level_0" in the rdapConformance array as well as servers returning the "sorting_metadata" section MUST include "sorting_level_0".

5. Implementation Considerations

The implementation of the new parameters is technically feasible, as operators for counting, sorting and paging are currently supported by the major RDBMSs.

In the following, the match between the new defined parameters and the SQL operators is shown (Table 3):

New query parameter	SQL operator
count	count(*) query without offset, limit and order by [MYSQL-COUNT],[POSTGRES-COUNT],[ORACLE-COUNT]
sort	order by [MYSQL-SORT],[POSTGRES-SORT],[ORACLE-SORT]
limit	limit n (in MySql [MYSQL-LIMIT] and Postgres [POSTGRES-LIMIT]) FETCH FIRST n ROWS ONLY (in Oracle [ORACLE-LIMIT])
offset	offset m (in Postgres) OFFSET m ROWS (in Oracle)
limit + offset	limit n offset m (in MySql and Postgres) OFFSET m ROWS FETCH NEXT n ROWS ONLY (in Oracle)

Table 3: New query parameters vs. SQL operators

With regard to Oracle, Table 3 reports only one of the three methods that can be used to implement limit and offset parameters. The others are described in [ORACLE-ROWNUM] and [ORACLE-ROW-NUMBER].

In addition, similar operators are completely or partially supported by the most known NoSQL databases (MongoDB, CouchDB, HBase, Cassandra, Hadoop) so the implementation of the new parameters seems to be practicable by servers working without the use of an RDBMS.

5.1. Considerations about Paging Implementation

The use of limit and offset operators represents the most common way to implement results pagination. However, when offset has a high value, scrolling the result set could take some time. In addition, offset pagination may return inconsistent pages when data are frequently updated (i.e. real-time data) but this is not the case of registration data. An alternative approach to offset pagination is the keyset pagination, a.k.a. seek-method [SEEK] or cursor based pagination. This method has been taken as the basis for the implementation of a cursor parameter [CURSOR] by some REST API providers (e.g. [CURSOR-API1],[CURSOR-API2]). The cursor parameter is an opaque URL-safe string representing a logical pointer to the first result of the next page (Figure 7).

```
{
  "rdapConformance": [
    "rdap_level_0",
    "paging_level_0"
  ],
  ...
  "notices": [
    {
      "title": "Search query limits",
      "type": "result set truncated due to excessive load",
      "description": [
        "search results for domains are limited to 10"
      ]
    }
  ],
  "paging_metadata": {
    "totalCount": 73,
    "pageCount": 10,
    "links": [
      {
        "value": "https://example.com/rdap/domains?name=*nr.com",
        "rel": "next",
        "href": "https://example.com/rdap/domains?name=*nr.com&limit=10
          &cursor=wJlCDLil6KTWypN7T6vc6nWEmEYe99HjflXYlXmqV-M=",
        "title": "Result Pagination Link",
        "type": "application/rdap+json"
      }
    ]
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 7: Example of a "paging_metadata" instance to implement keyset pagination

But keyset pagination raises some drawbacks with respect to offset pagination:

- o it needs at least one key field;
- o it does not allow to sort by any field and paginate the results because sorting has to be made on the key field;
- o it does not allow to skip pages because they have to be scrolled in sequential order starting from the initial page;
- o it makes very hard the navigation of the result set in both directions because all comparison and sort operations have to be reversed.

Furthermore, in the RDAP context, some additional considerations can be made:

- o an RDAP object is a conceptual aggregation of information collected from more than one data structure (e.g. table) and this makes even harder for the developers the implementation of the seek-method that is already quite difficult. In fact, for example, the entity object can gather information from different data structures (registrars, registrants, contacts, resellers, and so on), each one with its own key field mapping the RDAP entity handle;
- o depending on the number of the page results as well as the number and the complexity of the properties of each RDAP object in the response, the time required by offset pagination to skip the previous pages could be much faster than the processing time needed to build the current page. In fact, RDAP objects are usually formed by information belonging to multiple data structures and containing multivalued properties (e.g. arrays) and, therefore, data selection is a time consuming process. This situation occurs even though the data selection process makes use of indexes;
- o depending on the access levels defined by each RDAP operator, the increase of complexity and the decrease of flexibility of keyset pagination with respect to the offset pagination could be considered impractical.

Finally, the keyset pagination is not fully compliant with the additional RDAP capabilities proposed by this document. In fact, the presence of a possible cursor parameter does not seem to be consistent with both the sorting capability and the possibility to implement additional ready-made links besides the classic "next page" link. But, while the provisioning of more paging links can be superfluous, dropping the sorting capability seems quite unreasonable.

If pagination is implemented by using a cursor, both "offset" and "nextOffset" fields MUST not be included in the "paging_metadata" section.

FOR DISCUSSION: Should RDAP specification reports both offset and cursor parameters and let operators to implement pagination according to their needs, the user access levels, the submitted queries?

6. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

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, "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. IIT-CNR/Registro.it

Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it
Location: <https://rdap.pubtest.nic.it/>
Description: This implementation includes support for RDAP queries using data from the public test environment of .it ccTLD. The RDAP server does not implement any security policy because data returned by this server are only for experimental testing purposes. The RDAP server implements both offset and cursor based pagination (the latter only when sort and offset parameters are not present in the query string).
Level of Maturity: This is a "proof of concept" research implementation.
Coverage: This implementation includes all of the features described in this specification.
Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

6.2. Google Registry

Responsible Organization: Google Registry
Location: <https://www.registry.google/rdap/>

Description: This implementation includes support for RDAP queries for TLDs such as .GOOGLE, .HOW, .SOY, and .xn--q9jyb4c . The RDAP server implements cursor based pagination (the number of objects per page is fixed so the limit parameter is not available). The link used to request the next page is included in the notice section of the response.

Level of Maturity: Production.

Coverage: This implementation includes the cursor parameter described in this specification.

Contact Information: Brian Mountford, mountford@google.com

7. Security Considerations

Security services for the operations specified in this document are described in RFC 7481 [RFC7481].

Search query typically requires more server resources (such as memory, CPU cycles, and network bandwidth) when compared to lookup query. This increases the risk of server resource exhaustion and subsequent denial of service due to abuse. This risk can be mitigated by either restricting search functionality and limiting the rate of search requests. Servers can also reduce their load by truncating the results in the response. However, this last security policy can result in a higher inefficiency if the RDAP server does not provide any functionality to return the truncated results.

The new parameters presented in this document provide the RDAP operators with a way to implement a secure server without penalizing its efficiency. The "count" parameter gives the user a measure to evaluate the query precision and, at the same time, return a significant information. The sort parameter allows the user to obtain the most relevant information at the beginning of the result set. In both cases, the user doesn't need to submit further unnecessary search requests. Finally, the limit and offset parameters enable the user to scroll the result set by submitting a sequence of sustainable queries according to the server limits.

8. IANA Considerations

This document has no actions for IANA.

9. Acknowledgements

The authors would like to acknowledge Brian Mountford for his contribution to the development of this document.

10. References

10.1. Normative References

- [ISO.3166.1988]
International Organization for Standardization, "Codes for the representation of names of countries, 3rd edition", ISO Standard 3166, August 1988.
- [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>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.
- [RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <<https://www.rfc-editor.org/info/rfc6350>>.
- [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>>.
- [RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", RFC 7480, DOI 10.17487/RFC7480, March 2015, <<https://www.rfc-editor.org/info/rfc7480>>.
- [RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", RFC 7481, DOI 10.17487/RFC7481, March 2015, <<https://www.rfc-editor.org/info/rfc7481>>.
- [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>>.

- [RFC7483] Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", RFC 7483, DOI 10.17487/RFC7483, March 2015, <<https://www.rfc-editor.org/info/rfc7483>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <<https://www.rfc-editor.org/info/rfc8288>>.

10.2. Informative References

- [CURSOR] Nimesh, R., "Paginating Real-Time Data with Keyset Pagination", July 2014, <<https://www.sitepoint.com/paginating-real-time-data-cursor-based-pagination/>>.
- [CURSOR-API1] facebook.com, "facebook for developers - Using the Graph API", July 2017, <<https://developers.facebook.com/docs/graph-api/using-graph-api>>.
- [CURSOR-API2] twitter.com, "Pagination", 2017, <<https://developer.twitter.com/en/docs/ads/general/guides/pagination.html>>.
- [GOESSNER-JSON-PATH] Goessner, S., "JSONPath - XPath for JSON", 2007, <<http://goessner.net/articles/JsonPath/>>.
- [HATEOAS] Jedrzejewski, B., "HATEOAS - a simple explanation", 2018, <<https://www.e4developer.com/2018/02/16/hateoas-simple-explanation/>>.
- [MYSQL-COUNT] mysql.com, "MySQL 5.7 Reference Manual, Counting Rows", October 2015, <<https://dev.mysql.com/doc/refman/5.7/en/counting-rows.html>>.
- [MYSQL-LIMIT] mysql.com, "MySQL 5.7 Reference Manual, SELECT Syntax", October 2015, <<https://dev.mysql.com/doc/refman/5.7/en/select.html>>.

[MYSQL-SORT]

mysql.com, "MySQL 5.7 Reference Manual, Sorting Rows", October 2015, <<https://dev.mysql.com/doc/refman/5.7/en/sorting-rows.html>>.

[OData-Part1]

Pizzo, M., Handl, R., and M. Zurmuehl, "OData Version 4.0. Part 1: Protocol Plus Errata 03", June 2016, <<http://docs.oasis-open.org/odata/odata/v4.0/errata03/os/complete/part1-protocol/odata-v4.0-errata03-os-part1-protocol-complete.pdf>>.

[ORACLE-COUNT]

Oracle Corporation, "Database SQL Language Reference, COUNT", March 2016, <<http://docs.oracle.com/database/122/SQLRF/COUNT.htm>>.

[ORACLE-LIMIT]

Oracle Corporation, "Database SQL Language Reference, SELECT, Row limiting clause", March 2016, <<http://docs.oracle.com/database/122/SQLRF/SELECT.htm>>.

[ORACLE-ROW-NUMBER]

Oracle Corporation, "Database SQL Language Reference, SELECT, ROW_NUMBER", March 2016, <http://docs.oracle.com/database/122/SQLRF/ROW_NUMBER.htm#SQLRF06100>.

[ORACLE-ROWNUM]

Oracle Corporation, "Database SQL Language Reference, SELECT, ROWNUM Pseudocolumn", March 2016, <<http://docs.oracle.com/database/122/SQLRF/ROWNUM-Pseudocolumn.htm#SQLRF00255>>.

[ORACLE-SORT]

Oracle Corporation, "Database SQL Language Reference, SELECT, Order by clause", March 2016, <<http://docs.oracle.com/database/122/SQLRF/SELECT.htm>>.

[POSTGRES-COUNT]

postgresql.org, "PostgreSQL, Aggregate Functions", September 2016, <<https://www.postgresql.org/docs/9.6/static/functions-aggregate.html>>.

- [POSTGRES-LIMIT] postgresql.org, "PostgreSQL, LIMIT and OFFSET", September 2016, <<https://www.postgresql.org/docs/9.6/static/queries-limit.html>>.
- [POSTGRES-SORT] postgresql.org, "PostgreSQL, Sorting Rows", September 2016, <<https://www.postgresql.org/docs/9.6/static/queries-order.html>>.
- [REST] Fredrich, T., "RESTful Service Best Practices, Recommendations for Creating Web Services", April 2012, <http://www.restapitutorial.com/media/RESTful_Best_Practices-v1_1.pdf>.
- [RFC6901] Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Pointer", RFC 6901, DOI 10.17487/RFC6901, April 2013, <<https://www.rfc-editor.org/info/rfc6901>>.
- [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>>.
- [SEEK] EverSQL.com, "Faster Pagination in Mysql - Why Order By With Limit and Offset is Slow?", July 2017, <<https://www.eversql.com/faster-pagination-in-mysql-why-order-by-with-limit-and-offset-is-slow/>>.
- [W3C.CR-xpath-31-20161213] Robie, J., Dyck, M., and J. Spiegel, "XML Path Language (XPath) 3.1", World Wide Web Consortium CR CR-xpath-31-20161213, December 2016, <<https://www.w3.org/TR/2016/CR-xpath-31-20161213>>.

Appendix A. Change Log

- 00: Initial version.
- 01: Added the paragraph "Considerations about Paging Implementation" to "Implementation Considerations" section. Added "Implementation Status" section. Added acknowledgements. Renamed the property reporting the paging links.
- 02: Corrected the value of "title" field in "paging_links" property. Updated references to RFC5988 (obsoleted by RFC 8288) and RFC7159 (obsoleted by RFC 8259). Revised some sentences.
- 03: Added the paragraph "Google Registry" to "Implementation Status" section.

- 04: Rearranged the information about pagination included in RDAP responses. Added the section "Paging Metadata". Replaced the wrong reference to RFC 5266 with the correct reference to RFC 5226.
- 05: Renamed "sortby" parameter in "sort". Removed "country" from the list of sorting properties. Added "sorting_level_0" into the "rdapConformance" array. Changed the title of section "Paging Metadata" in "Sorting and Paging Metadata". Changed the "IANA Considerations" section. Added "Representing Sorting Links" section. Changed the name of some sorting properties to be compliant with EPP.

Authors' Addresses

Mario Loffredo
IIT-CNR/Registro.it
Via Moruzzi,1
Pisa 56124
IT

Email: mario.loffredo@iit.cnr.it
URI: <http://www.iit.cnr.it>

Maurizio Martinelli
IIT-CNR/Registro.it
Via Moruzzi,1
Pisa 56124
IT

Email: maurizio.martinelli@iit.cnr.it
URI: <http://www.iit.cnr.it>

Scott Hollenbeck
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
USA

Email: shollenbeck@verisign.com
URI: <https://www.verisignlabs.com/>

Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 26, 2018

A. Newton
ARIN
January 22, 2018

A Description of RDAP JSON Messages Using JSON Content Rules
draft-newton-rdap-jcr-06

Abstract

This document describes the JSON responses in the Registration Data Access Protocol with the formal notation of JSON Content Rules.

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 26, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. Response	3
3. Object Classes	4
3.1. Entity Object Class	4
3.2. Nameserver Object Class	15
3.3. Domain Object Class	15
3.4. IP Network Object Class	16
3.5. Autnum Object Class	17
4. Search Results	17
5. Error Response	18
6. Common Structures	18
6.1. RDAP Conformance	19
6.2. Links	19
6.3. Notices And Remarks	19
6.4. Language Identifier	20
6.5. Events	20
6.6. Status	21
6.7. Port 43	21
6.8. Public IDs	22
7. Validating Responses	22
8. Stricter Validation	22
9. Complete Rulesets for RDAP	30
10. Normative References	50
Appendix A. Acknowledgements	51
Author's Address	51

1. Introduction

The JSON [RFC7159] responses of the Registration Data Access Protocol [RFC7483] are officially defined with English prose. Those definitions contain imprecise or ambiguous JSON structures and require lengthy, tedious examples in the attempt to offer clarification. The English prose can be difficult for non-native English readers, and the examples create their own confusion.

This document describes the JSON found in RDAP with JSON Content Rules [I-D.newton-json-content-rules] (JCR).

JCR overcomes some of the obstacles of describing JSON with English prose, reducing the tediousness of the prose and accompanying lengthy examples to understandable data structures. Additionally, JCR has mechanisms which can be used by software developers to create test harnesses and technology compatibility kits.

Though this document describes all of the JSON found in [RFC7483], it presents the structures in a different order. The rules defined here

use the JCR mixin style of specification, where common structures are defined in group rules instead of separately, distinct objects.

2. Response

[RFC7483] describes ten distinct JSON response: five entity class response, an error response, a help response, and three search responses.

```
@{root} $entity_response = {
    $response_mixin,
    $entity_mixin
}

@{root} $nameserver_response = {
    $response_mixin,
    $nameserver_mixin
}

@{root} $domain_response = {
    $response_mixin,
    $domain_mixin
}

@{root} $network_response = {
    $response_mixin,
    $network_mixin
}

@{root} $autnum_response = {
    $response_mixin,
    $autnum_mixin
}

@{root} $error_response = {
    $response_mixin,
    $error_mixin
}

@{root} $help_response = {
    $response_mixin,
    $lang ?
}

@{root} $domainSearch_response = {
    $response_mixin,
    $lang ?,
    $domainSearchResult
}
```

```
}  
  
@{root} $nameserverSearch_response = {  
    $response_mixin,  
    $lang ?,  
    $nameserverSearchResult  
}  
  
@{root} $entitySearch_response = {  
    $response_mixin,  
    $lang ?,  
    $entitySearchResult  
}
```

Figure 1

All of the responses have a common set of object members described by response_mixin.

```
$response_mixin = (  
    $rdapConformance ?,  
    "notices" : $notices ?  
)
```

Figure 2

3. Object Classes

The primary data structures in RDAP are called object classes. These are first order object instances with identifiers. They are JSON objects which contain other JSON data types.

3.1. Entity Object Class

The Entity object class represents persons or organizations.


```
$entities = "entities" : [ $entity_oc * ]

$entity_oc = {
  $entity_mixin
}

$entity_mixin = (
  "objectClassName" : "entity",
  $common_mixin,
  "vcardArray"      : [ "vcard", $vcard * ] ?,
  "asEventActor"    : [ $noActorEvent * ] ?,
  $roles ?,
  $publicIds ?,
  $entities ?,
  "networks"        : [ $network_oc * ] ?,
  "autnums"         : [ $autnum_oc * ] ?
)

$roles = "roles" : [ string * ]
```

Figure 3

The Entity object class incorporates jCard [RFC7095] (vCard in JSON) for contact information.

Each jCard is made up of vCard properties in an array. The very first property must be "version", and the "fn" property must appear once and only once.

```
$vcard = [  
  [ "version", {}, "text", "4.0" ],  
  $vcard_not_fn_properties * ,  
  [ "fn", { $vcard_type_param?, $vcard_language_param? ,  
            $vcard_altid_param? , $vcard_pid_param? ,  
            $vcard_pref_param?, $vcard_any_param? },  
    "text", string ],  
  $vcard_not_fn_properties *  
]  
  
$vcard_not_fn_properties =  
(  
  $vcard_source |  
  $vcard_kind |  
  $vcard_n |  
  $vcard_nickname |  
  $vcard_photo |  
  $vcard_bday |  
  $vcard_anniversary |  
  $vcard_gender |  
  $vcard_adr |  
  $vcard_tel |  
  $vcard_email |  
  $vcard_impp |  
  $vcard_lang |  
  $vcard_tz |  
  $vcard_geo |  
  $vcard_title |  
  $vcard_role |  
  $vcard_logo |  
  $vcard_org |  
  $vcard_member |  
  $vcard_related |  
  $vcard_categories |  
  $vcard_note |  
  $vcard_prodid |  
  $vcard_rev |  
  $vcard_sound |  
  $vcard_uid |  
  $vcard_clientpidmap |  
  $vcard_url |  
  $vcard_key |  
  $vcard_fburl |  
  $vcard_caladruri |  
  $vcard_caluri |  
  $vcard_x10  
)
```

Figure 4

Each vCard property is composed of a property name, a set of parameters, and data values.

```

$vcardsource = [ "source", $vcardsource_prop ]
$vcardsource_prop = (
    { $vcardsource_pid_param? , $vcardsource_pref_param? ,
      $vcardsource_mediatype_param? , $vcardsource_any_param? } ,
    "uri", uri
)

$vcardsourcekind = [ "kind", $vcardsourcekind_prop ]
$vcardsourcekind_prop = (
    { $vcardsourcekind_any_param? },
    "text", $vcardsourcekind_type
)

$vcardsourcen = [ "n", $vcardsourcen_prop ]
$vcardsourcen_prop = (
    { $vcardsourcen_sort_as_param? , $vcardsourcen_language_param? ,
      $vcardsourcen_altid_param? , $vcardsourcen_any_param? },
    "text", $vcardsourcen_string_type_array
)

$vcardsourcenickname = [ "nickname", $vcardsourcenickname_prop ]
$vcardsourcenickname_prop = (
    { $vcardsourcenickname_type_param? , $vcardsourcenickname_language_param? ,
      $vcardsourcenickname_altid_param? , $vcardsourcenickname_pid_param? ,
      $vcardsourcenickname_pref_param? , $vcardsourcenickname_any_param? },
    "text", $vcardsourcenickname_string_type_array
)

$vcardsourcephoto = [ "photo", $vcardsourcephoto_prop ]
$vcardsourcephoto_prop = (
    { $vcardsourcephoto_altid_param? , $vcardsourcephoto_type_param? ,
      $vcardsourcephoto_mediatype_param? , $vcardsourcephoto_pref_param? ,
      $vcardsourcephoto_pid_param? , $vcardsourcephoto_any_param? },
    "uri", uri
)

$vcardsourcebirthday = [ "bday", $vcardsourcebirthday_prop ]
$vcardsourcebirthday_prop = (
    ( { $vcardsourcebirthday_language_param? , $vcardsourcebirthday_altid_param? ,
        $vcardsourcebirthday_calscale_param? , $vcardsourcebirthday_any_param? },
      "text", string
    ) |
    ( { $vcardsourcebirthday_altid_param? , $vcardsourcebirthday_calscale_param? ,

```

```

        $vcard_any_param? },
        "date-and-or-time", $vcard_date_and_or_time_type
    )
)

$vcard_anniversary = [ "anniversary", $vcard_anniversary_prop ]
$vcard_anniversary_prop = (
    { $vcard_altid_param?, $vcard_calscale_param?, $vcard_any_param? },
    "date-and-or-time", $vcard_date_and_or_time_type
)

$vcard_gender = [ "gender", $vcard_gender_prop ]
$vcard_gender_prop = (
    { $vcard_any_param? },
    "text",
    ( $vcard_gender_type | [ $vcard_gender_type, string + ] )
)

$vcard_adr = [ "adr", $vcard_adr_prop ]
$vcard_adr_prop = (
    { $vcard_label_param? , $vcard_language_param? ,
      $vcard_geo_param ? , $vcard_tz_param? ,
      $vcard_altid_param? , $vcard_pid_param? ,
      $vcard_pref_param? , $vcard_type_param? ,
      $vcard_any_param? },
    "text", $vcard_string_type_array
)

$vcard_tel = [ "tel", $vcard_tel_prop ]
$vcard_tel_prop = (
    ( { $vcard_type_param? , $vcard_pid_param? , $vcard_pref_param? ,
        $vcard_altid_param?, $vcard_any_param? }, "text", string
      ) |
    ( { $vcard_type_param? , $vcard_pid_param? , $vcard_pref_param? ,
        $vcard_altid_param?, $vcard_mediatype_param?,
        $vcard_any_param? },
      "uri", uri..tel
    )
)

$vcard_email = [ "email", $vcard_email_prop ]
$vcard_email_prop = (
    { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param? ,
      $vcard_altid_param? , $vcard_any_param ? },
    "text", string
)

$vcard_impp = [ "impp", $vcard_impp_prop ]

```

```
$vcard_impp_prop = (  
  { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param? ,  
    $vcard_mediatype_param? , $vcard_altid_param? ,  
    $vcard_any_param? },  
  "uri", uri..impp  
)  
  
$vcard_lang = [ "lang", $vcard_lang_prop ]  
$vcard_lang_prop = (  
  { $vcard_pid_param? , $vcard_pref_param? , $vcard_altid_param? ,  
    $vcard_type_param? , $vcard_any_param? },  
  "language-tag", $vcard_language_tag_type  
)  
  
$vcard_tz = [ "tz", $vcard_tz_prop ]  
$vcard_tz_prop = (  
  { $vcard_altid_param? , $vcard_pid_param? , $vcard_pref_param? ,  
    $vcard_type_param? , $vcard_mediatype_param? ,  
    $vcard_any_param? },  
  ( ( "text", string ) |  
    ( "uri" , uri ) |  
    ( "utc-offset" , $vcard_utc_offset_type ) )  
)  
  
$vcard_geo = [ "geo", $vcard_geo_prop ]  
$vcard_geo_prop = (  
  { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param? ,  
    $vcard_mediatype_param? , $vcard_altid_param? ,  
    $vcard_any_param? },  
  "uri", uri..geo  
)  
  
$vcard_title = [ "title", $vcard_title_prop ]  
$vcard_title_prop = (  
  { $vcard_language_param? , $vcard_pid_param? , $vcard_pref_param? ,  
    $vcard_altid_param? , $vcard_type_param? , $vcard_any_param? },  
  "text", string  
)  
  
$vcard_role = [ "role", $vcard_role_prop ]  
$vcard_role_prop = (  
  { $vcard_language_param? , $vcard_pid_param? , $vcard_pref_param? ,  
    $vcard_altid_param? , $vcard_type_param? , $vcard_any_param? },  
  "text", string  
)  
  
$vcard_logo = [ "logo", $vcard_logo_prop ]  
$vcard_logo_prop = (  

```

```

    { $vcard_altid_param? , $vcard_type_param? ,
      $vcard_mediatype_param? , $vcard_pref_param? ,
      $vcard_pid_param? , $vcard_language_param? ,
      $vcard_any_param? },
    "uri", uri
  )

$vcard_org = [ "org", $vcard_org_prop ]
$vcard_org_prop = (
  { $vcard_sort_as_param? , $vcard_language_param? ,
    $vcard_pid_param? , $vcard_pref_param? , $vcard_altid_param? ,
    $vcard_type_param? , $vcard_any_param? },
  "text", $vcard_string_type_array
)

$vcard_member = [ "member", $vcard_member_prop ]
$vcard_member_prop = (
  { $vcard_altid_param? , $vcard_mediatype_param? ,
    $vcard_pref_param? , $vcard_pid_param? , $vcard_any_param? },
  "uri", uri
)

$vcard_related = [ "related", $vcard_related_prop ]
$vcard_related_prop = (
  ( { $vcard_mediatype_param? , $vcard_pref_param? ,
      $vcard_altid_param? , $vcard_type_param? ,
      $vcard_any_param? },
    "uri", uri
  ) |
  ( { $vcard_language_param? , $vcard_pref_param? ,
      $vcard_altid_param? , $vcard_type_param? ,
      $vcard_any_param? },
    "text", $vcard_related_type_array
  )
)

$vcard_categories = [ "categories", $vcard_categories_prop ]
$vcard_categories_prop = (
  { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param? ,
    $vcard_altid_param? , $vcard_any_param? },
  "text", $vcard_string_type_array
)

$vcard_note = [ "note", $vcard_note_prop ]
$vcard_note_prop = (
  { $vcard_language_param? , $vcard_pid_param? ,
    $vcard_pref_param? , $vcard_type_param? ,
    $vcard_altid_param? , $vcard_any_param? },

```

```
    "text", string
  )

  $vcard_prodid = [ "prodid", $vcard_prodid_prop ]
  $vcard_prodid_prop = ( { $vcard_any_param ? }, "text", string )

  $vcard_rev = [ "rev", $vcard_rev_prop ]
  $vcard_rev_prop = (
    { $vcard_any_param ? }, "timestamp", datetime
  )

  $vcard_sound = [ "sound", $vcard_sound_prop ]
  $vcard_sound_prop = (
    { $vcard_altid_param? , $vcard_mediatype_param? ,
      $vcard_language_param? , $vcard_pref_param? ,
      $vcard_pid_param? , $vcard_any_param? },
    "uri", uri
  )

  $vcard_uid = [ "uid", $vcard_uid_prop ]
  $vcard_uid_prop = (
    ( { $vcard_any_param? }, "uri", uri ) |
    ( { $vcard_any_param? }, "text", string )
  )

  $vcard_clientpidmap = [ "clientpidmap", $vcard_clientpidmap_prop ]
  $vcard_clientpidmap_prop = ( {}, "text", [ /^[0-9]+$/, uri ] )

  $vcard_url = [ "url", $vcard_url_prop ]
  $vcard_url_prop = (
    { $vcard_altid_param? , $vcard_mediatype_param? ,
      $vcard_pref_param? , $vcard_pid_param? , $vcard_any_param? },
    "uri", uri
  )

  $vcard_key = [ "key", $vcard_key_prop ]
  $vcard_key_prop = (
    ( { $vcard_mediatype_param? , $vcard_pref_param? ,
      $vcard_altid_param? , $vcard_type_param? ,
      $vcard_any_param? },
      "uri", uri
    ) |
    ( { $vcard_pref_param? , $vcard_altid_param? ,
      $vcard_type_param? , $vcard_any_param? },
      "text", string
    )
  )
)
```

```

$vcards_furl = [ "furl", $vcards_furl_prop ]
$vcards_furl_prop = (
    { $vcards_altid_param? , $vcards_mediatype_param? ,
      $vcards_altid_param? , $vcards_pref_param? ,
      $vcards_pid_param? , $vcards_any_param? },
    "uri", uri
)

$vcards_caladruri = [ "caladruri", $vcards_caladruri_prop ]
$vcards_caladruri_prop = (
    { $vcards_altid_param? , $vcards_mediatype_param? ,
      $vcards_altid_param? , $vcards_pref_param? ,
      $vcards_pid_param? , $vcards_any_param? },
    "uri", uri
)

$vcards_caluri = [ "caluri", $vcards_caluri_prop ]
$vcards_caluri_prop = (
    { $vcards_altid_param? , $vcards_mediatype_param? ,
      $vcards_altid_param? , $vcards_pref_param? ,
      $vcards_pid_param? , $vcards_any_param? },
    "uri", uri
)

$vcards_x10 = [ $vcards_x_name_type, $vcards_x10_prop ]
$vcards_x10_prop = (
    { $vcards_language_param? , $vcards_pref_param? ,
      $vcards_altid_param? , $vcards_pid_param? ,
      $vcards_type_param? , $vcards_mediatype_param? ,
      $vcards_calscale_param? , $vcards_sort_as_param? ,
      $vcards_geo_param? , $vcards_tz_param? ,
      $vcards_label_param? , $vcards_any_param? },
    $vcards_x_name_type, $vcards_string_type_array
)

```

Figure 5

vCard parameters are grouped into a JSON object, where each member of the object is a parameter.

```
$vcard_language_param = "language" : $vcard_language_tag_type
$vcard_pref_param = "pref" : $vcard_pref_type
$vcard_altid_param = "altid" : string
$vcard_pid_param = "pid" : $vcard_pid_type_array
$vcard_type_param = "type" : $vcard_type_type_array
$vcard_mediatype_param = "mediatype" : $vcard_mediatype_type
$vcard_calscale_param = "calscale" : $vcard_calscale_type
$vcard_sort_as_param = "sort-as" : $vcard_string_type_array
$vcard_geo_param = "geo" : uri..geo
$vcard_tz_param = "tz" : $vcard_utc_offset_type
$vcard_label_param = "label" : $vcard_string_type_array
$vcard_any_param = /^x\[A-Za-z0-9\-\]*$/ : $vcard_string_type_array
```

Figure 6

vCard values are determined by a string denoting the value type, followed by the value or an array of those values.

```
$vcard_string_type_array =
( string | [ ( string * ) * ] )
$vcard_language_tag_type =:
/[a-z]{2}(-[A-Z][a-zA-Z]*(\[A-Z]{2})?)?$/
$vcard_mediatype_type =: /\w+\/[\-. \w]+(?:\[ \w]+)?$/
$vcard_utc_offset_type =: /^((\[ \- \+ ] \d{1,2}) :? ( \d{2} )?)?$/
$vcard_date_or_time =: (
/^(\d{4})-(\d{2})-(\d{2})$/ |
/^(\d{4})$/ |
/^--(\d{2})-(\d{2})$/ |
/^--(\d{2})$/ |
/^---(\d{2})$/ |
/^(\d{2})(\[ \- \+ ] \d{1,2}) :? ( \d{2} )?)?$/ |
/^(\d{2}) :? ( \d{2} ) ( [ \- \+ ] \d{1,2} ) :? ( \d{2} )?)?$/ |
/^(\d{2}) :? ( \d{2} ) :? ( \d{2} ) ( Z | [ \- \+ ] \d{1,2} ) :? ( \d{2} )?)?$/ |
/^-(\d{2}) :? ( \d{2} ) ( Z | [ \- \+ ] \d{1,2} ) :? ( \d{2} )?)?$/ |
/^-(\d{2}) ( [ \- \+ ] \d{1,2} ) :? ( \d{2} )?)?$/ |
/^--(\d{2}) ( [ \- \+ ] \d{1,2} ) :? ( \d{2} )?)?$/
)
$vcard_date_and_or_time_type =: (
datetime | $vcard_date_or_time
)
$vcard_group_type =: /^[a-zA-Z0-9\-\-]+$/
$vcard_type_type =: (
"home" | "work" | $vcard_tel_type | $vcard_related_type |
$vcard_x_name_type
```

```

)
$vccard_type_type_array = (
    $vccard_type_type | [ $vccard_type_type * ]
)
$vccard_tel_type =: (
    "text" | "voice" | "fax" | "cell" | "video" |
    "pager" | "textphone" | "main-number"
)
$vccard_tel_type_array = ( $vccard_tel_type | [ $vccard_tel_type * ] )
$vccard_related_type =: (
    "contact" | "acquaintance" | "friend" | "met" |
    "co-worker" | "colleague" | "co-resident" |
    "neighbor" | "child" | "parent" | "sibling" |
    "spouse" | "kin" | "muse" | "crush" | "date" |
    "sweetheart" | "me" | "agent" | "emergency"
)
$vccard_related_type_array = (
    $vccard_related_type | [ $vccard_related_type * ]
)
$vccard_index_type =: /^[1-9]?[0-9]*$/
$vccard_expertise_level_type =: (
    "beginner" | "average" | "expert"
)
$vccard_hobby_type =: ( "high" | "medium" | "low" )
$vccard_pref_type =: /^[1-9]?[0-9]{1}$|^100$/
$vccard_pid_type =: /^[0-9]+(\.[0-9]+)?$/
$vccard_pid_type_array = (
    $vccard_pid_type | [ $vccard_pid_type * ]
)
$vccard_gender_type =: ( "M" | "F" | "O" | "N" | "U" )
$vccard_gender_type_array = (
    $vccard_gender_type | [ $vccard_gender_type * ]
)
$vccard_kind_type =: (
    "individual" | "group" | "org" | "location" |
    "application" | "device"
)
$vccard_iana_token_type =: /^[A-Za-z0-9\-]*$/
$vccard_x_name_type =: /^x\-[A-Za-z0-9\-]*$/
$vccard_calscale_type =: (
    "gregorian" | $vccard_iana_token_type | $vccard_x_name_type
)

```

Figure 7

3.2. Nameserver Object Class

The nameserver object class represents DNS nameservers in registries.

```
$nameservers = "nameservers" : [ $nameserver_oc * ]

$nameserver_oc = {
    $nameserver_mixin
}

$nameserver_mixin = (
    "objectClassName" : "nameserver",
    $common_mixin,
    "ldhName"          : fqdn,
    "unicodeName"      : idn ?,
    "ipAddresses"      : {
        "v4" : [ ipv4 + ] ?,
        "v6" : [ ipv6 + ] ?
    } ?,
    $entities ?
)
```

Figure 8

3.3. Domain Object Class

The Domain object class is the most complex of all the object classes defined in RDAP. It represents both forward and reverse DNS delegations. It's complexity is mostly due to the DNSSEC provisions of the object class.

```
$domain_oc = {
    $domain_mixin
}

$domain_mixin = (
    "objectClassName" : "domain",
    $common_mixin,
    "ldhName"          : fqdn,
    "unicodeName"      : idn ?,
    "variants"         : [ $variant * ] ?,
    $nameservers ?,
    $secureDNS ?,
    $entities ?,
    $publicIds ?,
    "network"          : $network_oc ?
)
```

```

$variant = {
  $variantRelation ?,
  "relation"      : [ string * ] ?,
  "idnTable"      : string ?,
  "variantNames" : [
    { "ldhName" : fqdn, "unicodeName" : idn } *
  ]
}

$variantRelation = "relation" : [ string * ]

$securedDNS = "securedDNS" : {
  "zoneSigned"      : boolean ?,
  "delegationSigned" : boolean ?,
  "maxSigLife"      : integer ?,
  "dsData"          : [ $dsData_obj * ] ?,
  "keyData"         : [ $keyData_obj * ] ?
}

$dsData_obj = {
  "keyTag"      : integer,
  "algorithm"   : integer,
  "digest"      : string,
  "digestType" : integer,
  $events ?,
  $links ?
}

$keyData_obj = {
  "flags"      : integer,
  "protocol"   : integer,
  "publicKey"  : string,
  "algorithm"  : integer,
  $events ?,
  $links ?
}

```

Figure 9

3.4. IP Network Object Class

The IP Network object class represents IP network registrations in RIRs.

```

$network_oc = {
    $network_mixin
}

$network_mixin = (
    "objectClassName" : "ip network",
    $common_mixin,
    "startAddress"     : ( ipv4 | ipv6 ) ?,
    "endAddress"       : ( ipv4 | ipv6 ) ?,
    "ipVersion"        : ( "v4" | "v6" ) ?,
    "name"             : string ?,
    "type"             : string ?,
    "country"          : /[A-Z]{2}/ ?,
    "parentHandle"     : string ?,
    $entities ?
)

```

Figure 10

3.5. Autnum Object Class

The Autnum object class represents an autonomous system number or blocks of autonomous system numbers in an RIR.

```

$autnum_oc = {
    $autnum_mixin
}

$autnum_mixin = (
    "objectClassName" : "autnum",
    $common_mixin,
    "startAutnum"     : int32 ?,
    "endAutnum"       : int32 ?,
    "name"            : string ?,
    "type"            : string ?,
    "country"         : string ?,
    $entities ?
)

```

Figure 11

4. Search Results

Search results in RDAP are merely arrays of object classes.

```
$domainSearchResult =  
  "domainSearchResults"      : [ $domain_oc + ]  
  
$nameserverSearchResult =  
  "nameserverSearchResults" : [ $nameserver_oc + ]  
  
$entitySearchResult =  
  "entitySearchResults"      : [ $entity_oc + ]
```

Figure 12

5. Error Response

Section 6 of [RFC7483] describes RDAP error responses.

```
$error_mixin = (  
  "errorCode"    : integer,  
  "title"        : string ?,  
  "description"  : [ string * ] ?  
)
```

Figure 13

6. Common Structures

Section 4 of [RFC7483] describes eight common structures used throughout the JSON in RDAP.

Most of these common structures are grouped together in a rule called `common_mixin`.

```
$common_mixin = (  
  "handle"      : string ?,  
  "remarks"     : [ $notice * ] ?,  
  $links ?,  
  $events ?,  
  $status ?,  
  $port43 ?,  
  $lang ?  
)
```

Figure 14

6.1. RDAP Conformance

The `rdapConformance` array is the versioning and capabilities negotiation mechanism of RDAP.

```
$rdapConformance = "rdapConformance" : [ string * ]
```

Figure 15

6.2. Links

Structures in RDAP may link to information in other data systems using links. Additionally, RDAP uses "self" links to identify instances of RDAP object classes. The data found in each link is described by [RFC5988].

RDAP links are an array of distinct objects, each representing a separate link.

```
$links = ( "links" : [ $link * ] )

; see RFC 5988
$link = {
  "value"      : uri ?,
  "rel"        : string ?,
  "href"       : uri,
  "hreflang"   : [ $lang_value * ] ?,
  "title"      : string ?,
  "media"      : string ?,
  "type"       :
    /[a-zA-Z][a-zA-Z0-9]*\[a-zA-Z][a-zA-Z0-9]* / ?
}
```

Figure 16

6.3. Notices And Remarks

In RDAP, notices and remarks share the same structure. The difference is that notices are meta-data regarding the entirety of a response whereas remarks are meta-data covering a specific instance of an object class.

```
$notices = [ $notice * ]

$notice = {
  "title"      : string ?,
  "description" : [ string * ],
  $noticeRemarkType ?,
  $links ?,
  $lang ?
}

$noticeRemarkType = "type" : string
```

Figure 17

6.4. Language Identifier

The "lang" member occurs many RDAP data structures. And the same construct is used in the links structures.

```
; the language value as defined in RFC 5646
$lang_value =: /[a-z]{2}(\-[A-Z][a-zA-Z]*(\-[A-Z]{2}))?/?

$lang = ( "lang" : $lang_value )
```

Figure 18

6.5. Events

RDAP events note when a specific action has occurred on an object instance, and by whom. The same structure appears in all object classes, as well as being re-used by entities embedded by other objects.


```
$events = "events" : [ $event * ]

$noActorEvent_mixin = (
    $eventAction,
    "eventDate"    : datetime,
    $links ?,
    $lang ?
)

$eventAction = "eventAction" : string

$noActorEvent = {
    $noActorEvent_mixin
}

$event = {
    $noActorEvent_mixin,
    "eventActor" : string ?
}
```

Figure 19

6.6. Status

The status of RDAP object instances is indicated by an array of strings, where the value of the strings are registered in an IANA registry.

```
$status = "status" : [ string * ]
```

Figure 20

6.7. Port 43

RDAP object classes reference their corresponding Whois representation using the "port43" object member. This is simply a string holding the hostname of the Whois service.

```
$port43 = "port43" : string
```

Figure 21

6.8. Public IDs

Some RDAP services are required to identify entities and domains by public identifiers, such as ICANN Registrar IDs. The `publicIds` object member is an array of objects to represent these identifiers.

```
$publicIds = "publicIds" : [ $publicId * ]

$publicId = {
  "type"      : string,
  "identifier" : string
}
```

Figure 22

7. Validating Responses

Many JSON roots for RDAP are defined in Figure 1. For applications where an RDAP query must yield a specific response, the appropriate root must be used for validating the response.

For example, if the RDAP query `https://example.com/rdap/ip/2001:db8::0` is expected to yield an IP network response, then the validation must only use the `$network_response` root.

8. Stricter Validation

RDAP has a very lenient JSON model where information and structures not strictly forbidden are allowed. However, this lenient model allows information from RDAP help and error responses to be found in other responses, and responses for single objects to be found in responses from searches.

For applications where these structures cannot be mixed and validation is desired, override rules may be used.

For readability, two new rules are created to group the single objects and the search results.

The responses with a single object returned represented by mixins. This rule groups those mixins.

```
$object_class = (  
    $entity_mixin |  
    $nameserver_mixin |  
    $domain_mixin |  
    $network_mixin |  
    $autnum_mixin  
)
```

Figure 23

The responses to searches are objects containing a single search member which is an array containing the object class mixins. This rule groups the search members.

```
$search_results = (  
    $domainSearchResult |  
    $nameserverSearchResult |  
    $entitySearchResult  
)
```

Figure 24

Using the new rules in Figure 23 and Figure 24, override rules (rules rewriting existing rules) can be written for greater strictness.

These rules for single object responses prevent search and error response elements to be present.

```
@{root} $entity_response = {
    $response_mixin,
    $entity_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}

@{root} $nameserver_response = {
    $response_mixin,
    $nameserver_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}

@{root} $domain_response = {
    $response_mixin,
    $domain_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}

@{root} $network_response = {
    $response_mixin,
    $network_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}

@{root} $autnum_response = {
    $response_mixin,
    $autnum_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}
```

Figure 25

These rules for help and error responses prevent single object and search arrays.

```
@{root} $error_response = {  
    $response_mixin,  
    $error_mixin,  
    @{not} $object_class,  
    @{not} $search_results  
}  
  
@{root} $help_response = {  
    $response_mixin,  
    $lang ?,  
    @{not} $error_mixin,  
    @{not} $object_class,  
    @{not} $search_results  
}
```

Figure 26

These rules for search responses prevent single object and error data structures.

```
@{root} $domainSearch_response = {
    $response_mixin,
    $lang ?,
    $domainSearchResult,
    @{not} $error_mixin,
    @{not} $object_class
}

@{root} $nameserverSearch_response = {
    $response_mixin,
    $lang ?,
    $nameserverSearchResult,
    @{not} $error_mixin,
    @{not} $object_class
}

@{root} $entitySearch_response = {
    $response_mixin,
    $lang ?,
    $entitySearchResult,
    @{not} $error_mixin,
    @{not} $object_class
}
```

Figure 27

Stricter validation is also possible by overriding rules that take generic strings and overriding them with enumerated lists of values from the RDAP JSON values registry.

These rules override the generic domain variant relations in RDAP with values found in the IANA registry.

```
$variantRelation = "relation" : [ $variantRelation_values * ]

$variantRelation_values = (
    "registered" |
    "unregistered" |
    "registration restricted" |
    "open registration" |
    "conjoined"
)
```

Figure 28

These rules override the generic event actions in RDAP with values found in the IANA registry.

```
$eventAction = "eventAction" : $eventAction_values

$eventAction_values = (
    "registration" |
    "reregistration" |
    "last changed" |
    "expiration" |
    "deletion" |
    "reinstantiation" |
    "transfer" |
    "locked" |
    "unlocked" |
    "last update of RDAP database" |
    "registrar expiration" |
    "enum validation expiration"
)
```

Figure 29

These rules override the generic notice and remark types in RDAP with values found in the IANA registry.

```
$noticeRemarkType = "type" : $noticeRemarkType_values

$noticeRemarkType_values = (
  "result set truncated due to authorization" |
  "result set truncated due to excessive load" |
  "result set truncated due to unexplainable reasons" |
  "object truncated due to authorization" |
  "object truncated due to excessive load" |
  "object truncated due to unexplainable reasons"
)
```

Figure 30

These rules override the generic entity roles in RDAP with values found in the IANA registry.

```
$roles = "roles" : [ $role_values * ]

$role_values = (
  "registrant" |
  "technical" |
  "administrative" |
  "abuse" |
  "billing" |
  "registrar" |
  "reseller" |
  "sponsor" |
  "proxy" |
  "notifications" |
  "noc"
)
```

Figure 31

These rules override the generic status values in RDAP with values found in the IANA registry.

```
$status = "status" : [ $status_values * ]

$status_values = (
  "validated" |
  "renew prohibited" |
  "update prohibited" |
  "transfer prohibited" |
  "delete prohibited" |
  "proxy" |
  "private" |
  "removed" |
  "obscured" |
  "associated" |
  "active" |
  "inactive" |
  "locked" |
  "pending create" |
  "pending renew" |
  "pending transfer" |
  "pending update" |
  "pending delete" |
  "add period" |
  "auto renew period" |
  "client delete prohibited" |
  "client hold" |
  "client renew prohibited" |
  "client transfer prohibited" |
  "client update prohibited" |
  "pending restore" |
  "redemption period" |
  "renew period" |
  "server delete prohibited" |
  "server renew prohibited" |
  "server transfer prohibited" |
  "server update prohibited" |
  "server hold" |
  "transfer period"
)
```

Figure 32

9. Complete Rulesets for RDAP

The following rulesets, along with a test framework and examples of good and bad RDAP JSON instances, may be found at <https://github.com/arineng/draft-rdap-jcr>.

The following is the complete ruleset of JSON Content Rules for RDAP.

```
;
; JSON Content Rules (JCR) ruleset
; for the Registry Data Access Protocol (RDAP)
;
; Specified in RFC 7483
;

# ruleset-id rdap_level_0

;
; The various types of responses
;

@{root} $entity_response = {
    $response_mixin,
    $entity_mixin
}

@{root} $nameserver_response = {
    $response_mixin,
    $nameserver_mixin
}

@{root} $domain_response = {
    $response_mixin,
    $domain_mixin
}

@{root} $network_response = {
    $response_mixin,
    $network_mixin
}

@{root} $autnum_response = {
    $response_mixin,
    $autnum_mixin
}

@{root} $error_response = {
    $response_mixin,
```

```
        $error_mixin
    }

    @{{root}} $help_response = {
        $response_mixin,
        $lang ?
    }

    @{{root}} $domainSearch_response = {
        $response_mixin,
        $lang ?,
        $domainSearchResult
    }

    @{{root}} $nameserverSearch_response = {
        $response_mixin,
        $lang ?,
        $nameserverSearchResult
    }

    @{{root}} $entitySearch_response = {
        $response_mixin,
        $lang ?,
        $entitySearchResult
    }

    $response_mixin = (
        $rdapConformance ?,
        "notices" : $notices ?
    )

;
; RFC 7483 Section 4.1 - RDAP Conformance
;

$rdapConformance = "rdapConformance" : [ string * ]

;
; RFC 7483 Section 4.2 - Links
;

$links = ( "links" : [ $link * ] )

; see RFC 5988
$link = {
    "value"      : uri ?,
    "rel"        : string ?,
    "href"       : uri,
```

```

    "hreflang" : [ $lang_value * ] ?,
    "title"    : string ?,
    "media"    : string ?,
    "type"     :
        /[a-zA-Z][a-zA-Z0-9]*\[a-zA-Z][a-zA-Z0-9]* / ?
}

;
; RFC 7483 Section 4.3 - Notices
;

$notices = [ $notice * ]

$notice = {
    "title"      : string ?,
    "description" : [ string * ],
    $noticeRemarkType ?,
    $links ?,
    $lang ?
}

$noticeRemarkType = "type" : string

;
; RFC 7483 Section 4.4 - Language Identifier
;

; the language value as defined in RFC 5646
$lang_value =: /[a-z]{2}(\-[A-Z][a-zA-Z]*(\-[A-Z]{2}))?/?

$lang = ( "lang" : $lang_value )

;
; RFC 7483 Section 4.5 - Events
;

$events = "events" : [ $event * ]

$noActorEvent_mixin = (
    $eventAction,
    "eventDate" : datetime,
    $links ?,
    $lang ?
)

$eventAction = "eventAction" : string

$noActorEvent = {

```

```
    $noActorEvent_mixin
  }

  $event = {
    $noActorEvent_mixin,
    "eventActor" : string ?
  }

;
; RFC 7483 Section 4.6 - Status
;

$status = "status" : [ string * ]

;
; RFC 7483 Section 4.7 - Port43
;

$port43 = "port43" : string

;
; RFC 7482 Section 4.8 - Public Ids
;

$publicIds = "publicIds" : [ $publicId * ]

$publicId = {
  "type"      : string,
  "identifier" : string
}

;
; Common Object Class Structures
;

$common_mixin = (
  "handle" : string ?,
  "remarks" : [ $notice * ] ?,
  $links ?,
  $events ?,
  $status ?,
  $port43 ?,
  $lang ?
)

;
; RFC 7483 Section 5.1 - Entity Object Class
```

```

;

$entities = "entities" : [ $entity_oc * ]

$entity_oc = {
    $entity_mixin
}

$entity_mixin = (
    "objectClassName" : "entity",
    $common_mixin,
    "vcardArray"       : [ "vcard", $vcard * ] ?,
    "asEventActor"     : [ $noActorEvent * ] ?,
    $roles ?,
    $publicIds ?,
    $entities ?,
    "networks"         : [ $network_oc * ] ?,
    "autnums"          : [ $autnum_oc * ] ?
)

$roles = "roles" : [ string * ]

; jCard (see RFC 7095) is an algorithm for converting
; vCard to JSON. Each jCard is represented by an array of
; vCard properties. The "version" property must be the
; first property, and there must be one and only one "fn"
; property in some part of the array.
;
; Here we represent this by an ordered array, with "version"
; being the first property, then listing all the properties
; that match by property name, then matching
; the "fn" property, then matching any other property again.
$vcard = [
    [ "version", {}, "text", "4.0" ],
    $vcard_not_fn_properties * ,
    [ "fn", { $vcard_type_param?, $vcard_language_param? ,
              $vcard_altid_param? , $vcard_pid_param? ,
              $vcard_pref_param?, $vcard_any_param? },
      "text", string ],
    $vcard_not_fn_properties *
]

$vcard_not_fn_properties =
(
    $vcard_source |
    $vcard_kind |
    $vcard_n |
    $vcard_nickname |

```

```

    $vcard_photo |
    $vcard_bday |
    $vcard_anniversary |
    $vcard_gender |
    $vcard_adr |
    $vcard_tel |
    $vcard_email |
    $vcard_impp |
    $vcard_lang |
    $vcard_tz |
    $vcard_geo |
    $vcard_title |
    $vcard_role |
    $vcard_logo |
    $vcard_org |
    $vcard_member |
    $vcard_related |
    $vcard_categories |
    $vcard_note |
    $vcard_prodid |
    $vcard_rev |
    $vcard_sound |
    $vcard_uid |
    $vcard_clientpidmap |
    $vcard_url |
    $vcard_key |
    $vcard_fburl |
    $vcard_caladruri |
    $vcard_caluri |
    $vcard_x10
)

; vCard properties. Each one is defined with a common
; definition, and then a group that matches the property name.
$vcard_source = [ "source", $vcard_source_prop ]
$vcard_source_prop = (
    { $vcard_pid_param? , $vcard_pref_param?, $vcard_altid_param? ,
      $vcard_mediatype_param?, $vcard_any_param? } ,
    "uri", uri
)

$vcard_kind = [ "kind", $vcard_kind_prop ]
$vcard_kind_prop = (
    { $vcard_any_param? },
    "text", $vcard_kind_type
)

$vcard_n = [ "n", $vcard_n_prop ]

```

```

$vcards_n_prop = (
  { $vcards_sort_as_param? , $vcards_language_param? ,
    $vcards_altid_param? , $vcards_any_param? },
  "text", $vcards_string_type_array
)

$vcards_nickname = [ "nickname", $vcards_nickname_prop ]
$vcards_nickname_prop = (
  { $vcards_type_param? , $vcards_language_param? ,
    $vcards_altid_param? , $vcards_pid_param? ,
    $vcards_pref_param? , $vcards_any_param? },
  "text", $vcards_string_type_array
)

$vcards_photo = [ "photo", $vcards_photo_prop ]
$vcards_photo_prop = (
  { $vcards_altid_param? , $vcards_type_param? ,
    $vcards_mediatype_param? , $vcards_pref_param? ,
    $vcards_pid_param? , $vcards_any_param? },
  "uri", uri
)

$vcards_bday = [ "bday", $vcards_bday_prop ]
$vcards_bday_prop = (
  ( { $vcards_language_param?, $vcards_altid_param?,
    $vcards_calscale_param?, $vcards_any_param? },
    "text", string
  ) |
  ( { $vcards_altid_param?, $vcards_calscale_param?,
    $vcards_any_param? },
    "date-and-or-time", $vcards_date_and_or_time_type
  )
)

$vcards_anniversary = [ "anniversary", $vcards_anniversary_prop ]
$vcards_anniversary_prop = (
  { $vcards_altid_param?, $vcards_calscale_param?, $vcards_any_param? },
  "date-and-or-time", $vcards_date_and_or_time_type
)

$vcards_gender = [ "gender", $vcards_gender_prop ]
$vcards_gender_prop = (
  { $vcards_any_param? },
  "text",
  ( $vcards_gender_type | [ $vcards_gender_type, string + ] )
)

$vcards_adr = [ "adr", $vcards_adr_prop ]

```



```
$vcard_adr_prop = (  
  { $vcard_label_param? , $vcard_language_param? ,  
    $vcard_geo_param ? , $vcard_tz_param? ,  
    $vcard_altid_param? , $vcard_pid_param? ,  
    $vcard_pref_param? , $vcard_type_param? ,  
    $vcard_any_param? },  
  "text", $vcard_string_type_array  
)  
  
$vcard_tel = [ "tel", $vcard_tel_prop ]  
$vcard_tel_prop = (  
  ( { $vcard_type_param? , $vcard_pid_param? , $vcard_pref_param? ,  
      $vcard_altid_param?, $vcard_any_param? }, "text", string  
    ) |  
  ( { $vcard_type_param? , $vcard_pid_param? , $vcard_pref_param? ,  
      $vcard_altid_param?, $vcard_mediatype_param?,  
      $vcard_any_param? },  
    "uri", uri..tel  
  )  
)  
  
$vcard_email = [ "email", $vcard_email_prop ]  
$vcard_email_prop = (  
  { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param? ,  
    $vcard_altid_param? , $vcard_any_param ? },  
  "text", string  
)  
  
$vcard_impp = [ "impp", $vcard_impp_prop ]  
$vcard_impp_prop = (  
  { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param? ,  
    $vcard_mediatype_param? , $vcard_altid_param? ,  
    $vcard_any_param },  
  "uri", uri..impp  
)  
  
$vcard_lang = [ "lang", $vcard_lang_prop ]  
$vcard_lang_prop = (  
  { $vcard_pid_param? , $vcard_pref_param? , $vcard_altid_param? ,  
    $vcard_type_param? , $vcard_any_param? },  
  "language-tag", $vcard_language_tag_type  
)  
  
$vcard_tz = [ "tz", $vcard_tz_prop ]  
$vcard_tz_prop = (  
  { $vcard_altid_param?, $vcard_pid_param? , $vcard_pref_param? ,  
    $vcard_type_param? , $vcard_mediatype_param?,  
    $vcard_any_param? },
```

```

    ( ( "text", string ) |
      ( "uri" , uri ) |
      ( "utc-offset" , $vcard_utc_offset_type ) )
  )

  $vcard_geo = [ "geo", $vcard_geo_prop ]
  $vcard_geo_prop = (
    { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param? ,
      $vcard_mediatype_param? , $vcard_altid_param? ,
      $vcard_any_param? },
    "uri", uri..geo
  )

  $vcard_title = [ "title", $vcard_title_prop ]
  $vcard_title_prop = (
    { $vcard_language_param? , $vcard_pid_param? , $vcard_pref_param? ,
      $vcard_altid_param? , $vcard_type_param? , $vcard_any_param? },
    "text", string
  )

  $vcard_role = [ "role", $vcard_role_prop ]
  $vcard_role_prop = (
    { $vcard_language_param? , $vcard_pid_param? , $vcard_pref_param? ,
      $vcard_altid_param? , $vcard_type_param? , $vcard_any_param? },
    "text", string
  )

  $vcard_logo = [ "logo", $vcard_logo_prop ]
  $vcard_logo_prop = (
    { $vcard_altid_param? , $vcard_type_param? ,
      $vcard_mediatype_param? , $vcard_pref_param? ,
      $vcard_pid_param? , $vcard_language_param? ,
      $vcard_any_param? },
    "uri", uri
  )

  $vcard_org = [ "org", $vcard_org_prop ]
  $vcard_org_prop = (
    { $vcard_sort_as_param? , $vcard_language_param? ,
      $vcard_pid_param? , $vcard_pref_param? , $vcard_altid_param? ,
      $vcard_type_param? , $vcard_any_param? },
    "text", $vcard_string_type_array
  )

  $vcard_member = [ "member", $vcard_member_prop ]
  $vcard_member_prop = (
    { $vcard_altid_param? , $vcard_mediatype_param? ,
      $vcard_pref_param? , $vcard_pid_param? , $vcard_any_param? },

```

```
    "uri", uri
  )

  $vcard_related = [ "related", $vcard_related_prop ]
  $vcard_related_prop = (
    ( { $vcard_mediatype_param? , $vcard_pref_param? ,
        $vcard_altid_param? , $vcard_type_param? ,
        $vcard_any_param? },
      "uri", uri
    ) |
    ( { $vcard_language_param? , $vcard_pref_param? ,
        $vcard_altid_param? , $vcard_type_param? ,
        $vcard_any_param? },
      "text", $vcard_related_type_array
    )
  )

  $vcard_categories = [ "categories", $vcard_categories_prop ]
  $vcard_categories_prop = (
    { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param ? ,
      $vcard_altid_param? , $vcard_any_param? },
    "text", $vcard_string_type_array
  )

  $vcard_note = [ "note", $vcard_note_prop ]
  $vcard_note_prop = (
    { $vcard_language_param? , $vcard_pid_param? ,
      $vcard_pref_param? , $vcard_type_param? ,
      $vcard_altid_param? , $vcard_any_param ? },
    "text", string
  )

  $vcard_prodid = [ "prodid", $vcard_prodid_prop ]
  $vcard_prodid_prop = ( { $vcard_any_param ? }, "text", string )

  $vcard_rev = [ "rev", $vcard_rev_prop ]
  $vcard_rev_prop = (
    { $vcard_any_param ? }, "timestamp", datetime
  )

  $vcard_sound = [ "sound", $vcard_sound_prop ]
  $vcard_sound_prop = (
    { $vcard_altid_param? , $vcard_mediatype_param? ,
      $vcard_language_param? , $vcard_pref_param? ,
      $vcard_pid_param? , $vcard_any_param? },
    "uri", uri
  )
```

```
$vcard_uid = [ "uid", $vcard_uid_prop ]
$vcard_uid_prop = (
  ( { $vcard_any_param? }, "uri", uri ) |
  ( { $vcard_any_param? }, "text", string )
)

$vcard_clientpidmap = [ "clientpidmap", $vcard_clientpidmap_prop ]
$vcard_clientpidmap_prop = ( {}, "text", [ /^[0-9]+$/, uri ] )

$vcard_url = [ "url", $vcard_url_prop ]
$vcard_url_prop = (
  { $vcard_altid_param? , $vcard_mediatype_param? ,
    $vcard_pref_param? , $vcard_pid_param? , $vcard_any_param? },
  "uri", uri
)

$vcard_key = [ "key", $vcard_key_prop ]
$vcard_key_prop = (
  ( { $vcard_mediatype_param? , $vcard_pref_param? ,
    $vcard_altid_param? , $vcard_type_param? ,
    $vcard_any_param? },
    "uri", uri
  ) |
  ( { $vcard_pref_param? , $vcard_altid_param? ,
    $vcard_type_param? , $vcard_any_param? },
    "text", string
  )
)

$vcard_fburl = [ "fburl", $vcard_fburl_prop ]
$vcard_fburl_prop = (
  { $vcard_altid_param? , $vcard_mediatype_param? ,
    $vcard_altid_param? , $vcard_pref_param? ,
    $vcard_pid_param? , $vcard_any_param? },
  "uri", uri
)

$vcard_caladruri = [ "caladruri", $vcard_caladruri_prop ]
$vcard_caladruri_prop = (
  { $vcard_altid_param? , $vcard_mediatype_param? ,
    $vcard_altid_param? , $vcard_pref_param? ,
    $vcard_pid_param? , $vcard_any_param? },
  "uri", uri
)

$vcard_caluri = [ "caluri", $vcard_caluri_prop ]
$vcard_caluri_prop = (
  { $vcard_altid_param? , $vcard_mediatype_param? ,
```

```

    $vcard_altid_param? , $vcard_pref_param? ,
    $vcard_pid_param? , $vcard_any_param? },
    "uri", uri
)

$vcard_x10 = [ $vcard_x_name_type, $vcard_x10_prop ]
$vcard_x10_prop = (
    { $vcard_language_param? , $vcard_pref_param? ,
      $vcard_altid_param? , $vcard_pid_param? ,
      $vcard_type_param? , $vcard_mediatype_param? ,
      $vcard_calscale_param? , $vcard_sort_as_param? ,
      $vcard_geo_param? , $vcard_tz_param? ,
      $vcard_label_param? , $vcard_any_param? },
    $vcard_x_name_type, $vcard_string_type_array
)

; Each vCard property can have parameters.
$vcard_language_param = "language" : $vcard_language_tag_type
$vcard_pref_param = "pref" : $vcard_pref_type
$vcard_altid_param = "altid" : string
$vcard_pid_param = "pid" : $vcard_pid_type_array
$vcard_type_param = "type" : $vcard_type_type_array
$vcard_mediatype_param = "mediatype" : $vcard_mediatype_type
$vcard_calscale_param = "calscale" : $vcard_calscale_type
$vcard_sort_as_param = "sort-as" : $vcard_string_type_array
$vcard_geo_param = "geo" : uri..geo
$vcard_tz_param = "tz" : $vcard_utc_offset_type
$vcard_label_param = "label" : $vcard_string_type_array
$vcard_any_param = /^x\[A-Za-z0-9\-\]*$/ : $vcard_string_type_array

; Each vCard property has a value type.
$vcard_string_type_array =
    ( string | [ ( string | [ string * ] ) * ] )
$vcard_language_tag_type =:
    /[a-z]{2}(-[A-Z][a-zA-Z]*(-[A-Z]{2}))?$/
$vcard_mediatype_type =: /^w+\/[\.w]+(?:\[\.w\])?$/
$vcard_utc_offset_type =: /^((\[+\]\d{1,2}):?(\d{2}))?$/
$vcard_date_or_time =: (
    /^(\d{4})-?(\d{2})-?(\d{2})$/ |
    /^(\d{4})$/ |
    /^--(\d{2})-?(\d{2})$/ |
    /^--(\d{2})$/ |
    /^---(\d{2})$/ |
    /^(\d{2})(([-+]\d{1,2}):?(\d{2}))?$/ |
    /^(\d{2}):?(\d{2})((\[+\]\d{1,2}):?(\d{2}))?$/ |
    /^(\d{2}):?(\d{2}):?(\d{2})(Z|([+\-]\d{1,2}):?(\d{2}))?$/ |
    /^-(\d{2}):?(\d{2})(Z|([+\-]\d{1,2}):?(\d{2}))?$/ |
    /^-(\d{2})((\[+\]\d{1,2}):?(\d{2}))?$/ |

```

```

    /^--(\d{2})(([\-\\+]\d{1,2}):?(\d{2})?)?$/
)
$vcard_date_and_or_time_type =: (
    datetime | $vcard_date_or_time
)
$vcard_group_type =: /^[a-zA-Z0-9\\-]+$/
$vcard_type_type =: (
    "home" | "work" | $vcard_tel_type | $vcard_related_type |
    $vcard_x_name_type
)
$vcard_type_type_array = (
    $vcard_type_type | [ $vcard_type_type * ]
)
$vcard_tel_type =: (
    "text" | "voice" | "fax" | "cell" | "video" |
    "pager" | "textphone" | "main-number"
)
$vcard_tel_type_array = ( $vcard_tel_type | [ $vcard_tel_type * ] )
$vcard_related_type =: (
    "contact" | "acquaintance" | "friend" | "met" |
    "co-worker" | "colleague" | "co-resident" |
    "neighbor" | "child" | "parent" | "sibling" |
    "spouse" | "kin" | "muse" | "crush" | "date" |
    "sweetheart" | "me" | "agent" | "emergency"
)
$vcard_related_type_array = (
    $vcard_related_type | [ $vcard_related_type * ]
)
$vcard_index_type =: /^[1-9]?[0-9]*$/
$vcard_expertise_level_type =: (
    "beginner" | "average" | "expert"
)
$vcard_hobby_type =: ( "high" | "medium" | "low" )
$vcard_pref_type =: /^[1-9]?[0-9]{1}$|^100$/
$vcard_pid_type =: /^[0-9]+(\\.[0-9]+)?$/
$vcard_pid_type_array = (
    $vcard_pid_type | [ $vcard_pid_type * ]
)
$vcard_gender_type =: ( "M" | "F" | "O" | "N" | "U" )
$vcard_gender_type_array = (
    $vcard_gender_type | [ $vcard_gender_type * ]
)
$vcard_kind_type =: (
    "individual" | "group" | "org" | "location" |
    "application" | "device"
)
$vcard_iana_token_type =: /^[A-Za-z0-9\\-]*$/
$vcard_x_name_type =: /^x\\-[A-Za-z0-9\\-]*$/

```

```
$vcard_calscale_type =: (
    "gregorian" | $vcard_iana_token_type | $vcard_x_name_type
)

;
; RFC 7483 Section 5.2 - Nameserver Object Class
;

$nameservers = "nameservers" : [ $nameserver_oc * ]

$nameserver_oc = {
    $nameserver_mixin
}

$nameserver_mixin = (
    "objectClassName" : "nameserver",
    $common_mixin,
    "ldhName"         : fqdn,
    "unicodeName"     : idn ?,
    "ipAddresses"     : {
        "v4" : [ ipv4 + ] ?,
        "v6" : [ ipv6 + ] ?
    } ?,
    $entities ?
)

;
; RFC 7483 Section 5.3 - Domain Object Class
;

$domain_oc = {
    $domain_mixin
}

$domain_mixin = (
    "objectClassName" : "domain",
    $common_mixin,
    "ldhName"         : fqdn,
    "unicodeName"     : idn ?,
    "variants"        : [ $variant * ] ?,
    $nameservers ?,
    $secureDNS ?,
    $entities ?,
    $publicIds ?,
    "network"         : $network_oc ?
)

$variant = {
```

```

    $variantRelation ?,
    "relation"      : [ string * ] ?,
    "idnTable"      : string ?,
    "variantNames" : [
        { "ldhName" : fqdn, "unicodeName" : idn } *
    ]
}

$variantRelation = "relation" : [ string * ]

$secureDNS = "secureDNS" : {
    "zoneSigned"      : boolean ?,
    "delegationSigned" : boolean ?,
    "maxSigLife"      : integer ?,
    "dsData"          : [ $dsData_obj * ] ?,
    "keyData"          : [ $keyData_obj * ] ?
}

$dsData_obj = {
    "keyTag"      : integer,
    "algorithm"   : integer,
    "digest"      : string,
    "digestType"  : integer,
    $events ?,
    $links ?
}

$keyData_obj = {
    "flags"      : integer,
    "protocol"   : integer,
    "publicKey"  : string,
    "algorithm"  : integer,
    $events ?,
    $links ?
}

;
; RFC 7483 Section 5.4 - IP Network Object Class
;

$network_oc = {
    $network_mixin
}

$network_mixin = (
    "objectClassName" : "ip network",
    $common_mixin,
    "startAddress"    : ( ipv4 | ipv6 ) ?,

```



```
    "endAddress"      : ( ipv4 | ipv6 ) ?,
    "ipVersion"       : ( "v4" | "v6" ) ?,
    "name"            : string ?,
    "type"            : string ?,
    "country"         : /[A-Z]{2}/ ?,
    "parentHandle"    : string ?,
    $entities ?
)

;
; RFC 7483 Section 5.5 - Autnum Object Class
;

$autnum_oc = {
    $autnum_mixin
}

$autnum_mixin = (
    "objectClassName" : "autnum",
    $common_mixin,
    "startAutnum"     : int32 ?,
    "endAutnum"       : int32 ?,
    "name"            : string ?,
    "type"            : string ?,
    "country"         : string ?,
    $entities ?
)

;
; RFC 7483 Section 6 - Error
;

$error_mixin = (
    "errorCode"       : integer,
    "title"           : string ?,
    "description"     : [ string * ] ?
)

;
; RFC 7483 Section 8 - Search Results
;

$domainSearchResult =
    "domainSearchResults" : [ $domain_oc + ]

$nameserverSearchResult =
    "nameserverSearchResults" : [ $nameserver_oc + ]
```

```
$entitySearchResult =
  "entitySearchResults"      : [ $entity_oc + ]
```

Figure 33: Complete Ruleset for RDAP

The following is the complete ruleset of override rules for stricter validation of RDAP.

```
;
; Override rules for strict RDAP checking.
;

;
; Object class response
;

@{root} $entity_response = {
    $response_mixin,
    $entity_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}

@{root} $nameserver_response = {
    $response_mixin,
    $nameserver_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}

@{root} $domain_response = {
    $response_mixin,
    $domain_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}

@{root} $network_response = {
    $response_mixin,
    $network_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}

@{root} $autnum_response = {
    $response_mixin,
    $autnum_mixin,
```

```
        @{{not}} $error_mixin,
        @{{not}} $search_results
    }
;
; Help and error response
;

@{{root}} $error_response = {
    $response_mixin,
    $error_mixin,
    @{{not}} $object_class,
    @{{not}} $search_results
}

@{{root}} $help_response = {
    $response_mixin,
    $lang ?,
    @{{not}} $error_mixin,
    @{{not}} $object_class,
    @{{not}} $search_results
}
;
; Search responses
;

@{{root}} $domainSearch_response = {
    $response_mixin,
    $lang ?,
    $domainSearchResult,
    @{{not}} $error_mixin,
    @{{not}} $object_class
}

@{{root}} $nameserverSearch_response = {
    $response_mixin,
    $lang ?,
    $nameserverSearchResult,
    @{{not}} $error_mixin,
    @{{not}} $object_class
}

@{{root}} $entitySearch_response = {
    $response_mixin,
    $lang ?,
    $entitySearchResult,
    @{{not}} $error_mixin,
    @{{not}} $object_class
}
```

```
;
; Object class mixins
;

$object_class = (
    $entity_mixin |
    $nameserver_mixin |
    $domain_mixin |
    $network_mixin |
    $autnum_mixin
)
;
; All search results
;

$search_results = (
    $domainSearchResult |
    $nameserverSearchResult |
    $entitySearchResult
)
;
; IANA Status Values
;

$status = "status" : [ $status_values * ]

$status_values = (
    "validated" |
    "renew prohibited" |
    "update prohibited" |
    "transfer prohibited" |
    "delete prohibited" |
    "proxy" |
    "private" |
    "removed" |
    "obscured" |
    "associated" |
    "active" |
    "inactive" |
    "locked" |
    "pending create" |
    "pending renew" |
    "pending transfer" |
    "pending update" |
    "pending delete" |
    "add period" |
    "auto renew period" |
    "client delete prohibited" |
```

```
    "client hold" |
    "client renew prohibited" |
    "client transfer prohibited" |
    "client update prohibited" |
    "pending restore" |
    "redemption period" |
    "renew period" |
    "server delete prohibited" |
    "server renew prohibited" |
    "server transfer prohibited" |
    "server update prohibited" |
    "server hold" |
    "transfer period"
)
;
; IANA Notice and Remark Types
;

$noticeRemarkType = "type" : $noticeRemarkType_values

$noticeRemarkType_values = (
    "result set truncated due to authorization" |
    "result set truncated due to excessive load" |
    "result set truncated due to unexplainable reasons" |
    "object truncated due to authorization" |
    "object truncated due to excessive load" |
    "object truncated due to unexplainable reasons"
)
;
; IANA Roles
;

$roles = "roles" : [ $role_values * ]

$role_values = (
    "registrant" |
    "technical" |
    "administrative" |
    "abuse" |
    "billing" |
    "registrar" |
    "reseller" |
    "sponsor" |
    "proxy" |
    "notifications" |
    "noc"
)
```

```
;
; IANA Domain Variant Relations
;

$variantRelation = "relation" : [ $variantRelation_values * ]

$variantRelation_values = (
    "registered" |
    "unregistered" |
    "registration restricted" |
    "open registration" |
    "conjoined"
)
;
; IANA Event Actions
;

$eventAction = "eventAction" : $eventAction_values

$eventAction_values = (
    "registration" |
    "reregistration" |
    "last changed" |
    "expiration" |
    "deletion" |
    "reinstantiation" |
    "transfer" |
    "locked" |
    "unlocked" |
    "last update of RDAP database" |
    "registrar expiration" |
    "enum validation expiration"
)
```

Figure 34: Override Ruleset for RDAP

10. Normative References

- [I-D.newton-json-content-rules]
Newton, A. and P. Cordell, "A Language for Rules Describing JSON Content", draft-newton-json-content-rules-09 (work in progress), September 2017.
- [RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <<https://www.rfc-editor.org/info/rfc5988>>.

- [RFC7095] Kewisch, P., "jCard: The JSON Format for vCard", RFC 7095, DOI 10.17487/RFC7095, January 2014, <<https://www.rfc-editor.org/info/rfc7095>>.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<https://www.rfc-editor.org/info/rfc7159>>.
- [RFC7483] Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", RFC 7483, DOI 10.17487/RFC7483, March 2015, <<https://www.rfc-editor.org/info/rfc7483>>.

Appendix A. Acknowledgements

Bernhard Reutner-Fischer and participants of the IETF REGEXT mailing list contributed to the JCR found in this document.

Mario Laffredo provided significant feedback on the rulesets in this document, especially with regard to jCard validation, and provided helpful guidance on validation based on his own experience with the implementation of an RDAP validator.

Author's Address

Andrew Lee Newton
American Registry for Internet Numbers
3635 Concorde Parkway
Chantilly, VA 20151
US

Email: andy@arin.net
URI: <http://www.arin.net>