

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2019

M. Blanchet
Viagenie
March 11, 2019

Finding Additional Registration Data (RDAP) Service
draft-blanchet-regext-entityid2rdapserver-00

Abstract

This document specifies a method to find which Registration Data Access Protocol (RDAP) server is authoritative to answer additional information for a query already answered by another server. It is based on an entity id to RDAP server location mapping registry managed by IANA. One use case of this specification is the domain registry RDAP server providing a referral URL to the registrar RDAP server, based on the registrar entity id, for information that the registrar is authoritative for such as the contact or reseller information.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 September 12, 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 (<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. High-Level Functional Description | 3 |
| 4. Registry of Entity to RDAP server location | 3 |
| 5. Identifying the Entity | 4 |
| 6. Structure of the Entity to RDAP Server Location Registry | 4 |
| 7. Formal Definition | 6 |
| 7.1. Imported JSON Terms | 6 |
| 7.2. Registry Syntax | 6 |
| 8. Recursive Referrals | 7 |
| 9. Merging the Data Received from Multiple RDAP Servers | 7 |
| 10. Security Considerations | 7 |
| 11. IANA Considerations | 8 |
| 12. Discussion of this draft | 8 |
| 13. References | 9 |
| 13.1. Normative References | 9 |
| 13.2. Informative References | 10 |
| 13.3. URIs | 10 |
| Acknowledgements | 11 |
| Author's Address | 11 |

1. Introduction

Finding the authoritative Registration Data Access Protocol (RDAP) server is specified in [RFC7484]. In some use cases, the authoritative server answering an RDAP query may not have all the information, but instead another server has the missing information. However, the first server may not know the location (URL) of that other server, but just an organization identifier, therefore it can not send a link or redirect, as described in [RFC7483]. Operationally, the location of the other server will need to be known to many servers, where storing the mapping centrally enables the scalable management of the locations..

The typical use case is for domain registries where the RDAP server of the domain registry is not authoritative for or does not have some information for the query, but the registrar, a separate entity from the domain registry, is authoritative and does have that additional information. The information may include contact, reseller, expiration date information. The registry RDAP server needs to provide a referral location (URL) to the client, or provide the

organization identifier for the client to map to a location (URL), to enable the client to retrieve the information from the registrar RDAP server.

This specification is generic to include other possible current or future cases, so it does not focus on the specific thin domain registry-registrar case while it uses that use case for examples.

2. Conventions Used in This Document

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

3. High-Level Functional Description

The functional description of this proposal is as follows:

1. an RDAP client finds the authoritative RDAP server using [RFC7484],
2. the client sends its query to the authoritative RDAP server,
3. the authoritative RDAP server returns an answer as described in [RFC7483] with a reference to the identifier of an entity who has more data for this query,
4. The client finds the RDAP server of the entity by either:
 - A. Using a referral URL returned by the server based on the server using this specification,
 - B. Using this specification to find the RDAP server of the entity, based on the entity identifier,
5. the client sends the same query to the RDAP server of that entity,
6. the server returns an answer as described in [RFC7483],
7. the client shows all the information received from both servers.

4. Registry of Entity to RDAP server location

While it is expected that the RDAP servers will be managed by organizations, this specification uses the term "entity" to support any generic case. This specification defines a registry managed by IANA which maps an entity Id to its RDAP server location (URL). The

RDAP server location information description is similar to [RFC7484] so that RDAP client can parse similarly for both registries.

5. Identifying the Entity

The organization identifier used in the RDAP answer is the key to the entry of the RDAP server registry specified in this document. This key should be unique in the registry. For the specific case of gTLD domain registries, ICANN through IANA has created a registry of gTLD accredited registrars [1]. In that registry, a registrar is identified by a number. For ccTLD domain registries, some registrars may not be in this registry, as they do not need to be accredited by ICANN. In a generic way not related to domain registries, there should be a registry of entities providing a unique number for these entities. IANA already have a registry of organizations identifiers, as numbers, the Private Enterprise Numbers [2] registry, with a policy of first come first serve without any limitation, with easy registration procedure [3], used in multiple contexts for IETF protocols. This document suggests to use this registry for the assignment of unique numbers to entities. Therefore, this document specifies two namespaces for the entity identification: one for accredited gTLD registrars and one from the IANA private enterprise numbers registry. In the case of domain registries where a registrar is not in the first list, that registrar can easily get a unique organization number from the IANA organizations registry in a timely manner. This specification defines a registry which maps an entity id to its RDAP server location (URL). Therefore, the entity id with its namespace creates a unique key to the registry.

6. Structure of the Entity to RDAP Server Location Registry

The Entity to RDAP Server Location registry, as specified in Section 11 below, have been made available as JSON [RFC7159] objects, which can be retrieved via HTTP from locations specified by IANA. The JSON object for each registry contains a series of members containing metadata about the registry such as a version identifier, a timestamp of the publication date of the registry, and a description. Additionally, a "services" member contains the registry items themselves, as JSON objects. Each object has a key which uniquely defines the entity and the value is an array of its RDAP server URLs.

An example structure of the JSON output of the registry is illustrated:

```
{
  "version": "1.0",
  "publication": "YYYY-MM-DDTHH:MM:SSZ",
  "description": "Some text",
  "services": {
    "entry1-accregids":
      [
        "https://registrar2.example.com/myrdap/",
        "http://registrar2.example.com/myrdap/"
      ],
    "entry2-pen":
      [
        "https://registrar4.example.com/rdap/"
      ],
    "entry3-accregids":
      [
        "https://myregistrar.example.com/rdap/"
      ]
  }
}
```

The formal syntax is described in Section 7.

The "version" corresponds to the format version of the registry. This specification defines version "1.0".

The syntax of the "publication" value conforms to the Internet date/time format [RFC3339]. The value is the latest update date of the registry by IANA.

The optional "description" string can contain a comment regarding the content of the registry.

Per [RFC7258], in each array of base RDAP URLs, the secure versions of the transport protocol SHOULD be preferred and tried first. For example, if the base RDAP URLs array contains both HTTPS and HTTP URLs, the client SHOULD try the HTTPS version first.

Base RDAP URLs MUST have a trailing "/" character because they are concatenated to the various segments defined in [RFC7482].

JSON names MUST follow the format recommendations of [RFC7480]. Any unrecognized JSON object properties or values MUST be ignored by implementations.

The syntax of the keys is as follows:

- o entity: a unsigned integer encoded in ASCII

- o separator: the 0x2d ASCII hyphen
- o namespace: either "accregids" for an entity id from the IANA accredited registrar Ids registry or "pen" for an entity id from the IANA Private Enterprise Numbers registry

7. Formal Definition

This section is the formal definition of the registries. The structure of JSON objects and arrays using a set of primitive elements is defined in [RFC7159]. Those elements are used to describe the JSON structure of the registries.

7.1. Imported JSON Terms

- o OBJECT: a JSON object, defined in Section 4 of [RFC7159]
- o MEMBER: a member of a JSON object, defined in Section 4 of [RFC7159]
- o MEMBER-NAME: the name of a MEMBER, defined as a "string" in Section 4 of [RFC7159]
- o MEMBER-VALUE: the value of a MEMBER, defined as a "value" in Section 4 of [RFC7159]
- o ARRAY: an array, defined in Section 5 of [RFC7159]
- o ARRAY-VALUE: an element of an ARRAY, defined in Section 5 of [RFC7159]
- o STRING: a "string", as defined in Section 7 of [RFC7159]

7.2. Registry Syntax

Using the above terms for the JSON structures, the syntax of a registry is defined as follows: TBD

- o rdap-entity2server-registry: an OBJECT containing a MEMBER version and a MEMBER publication, an optional MEMBER description, and a MEMBER services-list
- o version: a MEMBER with MEMBER-NAME "version" and MEMBER-VALUE a STRING
- o publication: a MEMBER with MEMBER-NAME "publication" and MEMBER-VALUE a STRING

- o description: a MEMBER with MEMBER-NAME "description" and MEMBER-VALUE a STRING
- o services-list: a MEMBER with MEMBER-NAME "services" and
- o TDB
- o service-uri-list: an ARRAY, where each ARRAY-VALUE is a service-uri
- o service-uri: a STRING

8. Recursive Referrals

This specification does not restrict the use of recursive links. For example, the answer from the additional RDAP server may itself contain reference to other servers, hence the possibility of recursion. However, without limits, this may end up with infinite recursion. Based on its use case, the RDAP client should set a limit on the number of referrals it will follow. In the specific case of thin domain registries with registrars RDAP servers, there should be a limit of 2 levels: the domain registry RDAP server and the registrar RDAP server.

9. Merging the Data Received from Multiple RDAP Servers

The answer from the additional RDAP server may contain data that overlaps with the answer from the initial authoritative RDAP server. This document does not specify which data should be chosen in case of overlaps or conflicts, since it depends on the use case. In the specific case of thin domain registries with registrars RDAP servers, the data received by all RDAP servers should be additive and shown appropriately to the user. For example, if the domain registry RDAP server answer contains an expiration date for the domain queried, and if the registrar RDAP server answer also contains an expiration date, then the two expiration dates are shown to the user of the RDAP client.

10. Security Considerations

By providing a method to find an entity RDAP servers, this document helps to ensure that the end users will get the RDAP data from an authoritative source, instead of from rogue sources. The method has the same security properties as the RDAP protocols themselves. The transport used to access the registries can be more secure by using TLS [RFC5246], which IANA supports.

Additional considerations on using RDAP are described in [RFC7481].

11. IANA Considerations

IANA has created the RDAP Entity to RDAP Server Location Registry, listed below, and made them available as JSON objects. The contents of these registries are described in Section 6 with the formal syntax specified in Section 7.

Because this registry will be accessed by software, the download demand may be unusually high compared to normal IANA registries. The technical infrastructure by which registries are published will need to be reviewed and might need to be augmented.

Software accessing these registries will depend on the HTTP Expires header field to limit their query rate. It is, therefore, important for that header field to be properly set to provide timely information as the registries change, while maintaining a reasonable load on the IANA servers.

The HTTP Content-Type returned to clients accessing these JSON-formatted registries MUST be "application/json", as defined in [RFC7159].

The registry entries may not be sorted.

12. Discussion of this draft

this is a todo list for the author on topics to be done/resolved, or comments received. This section will disappear when draft is finished.

- o should this document merge with RFC7484 and become "RFC7484-bis"
- o identify the exact field the first server refers to the entity
- o Additional namespaces may be added with updates of this specification. we don't want to setup a registry of namespace, do we?
- o The process for adding or updating entries in these registries should be defined here

alternate structure proposed by James Gould:


```
{
  "description": "RDAP bootstrap file for registry to registrar referrals",
  "publication": "2019-02-14T02:00:02Z",
  "repositories": [
    {
      "id": "ICANN",
      "description": "ICANN registrar repository for ICANN accredited registrars",
      "registrars": [
        {
          "id": "292",
          "description": "MarkMonitor",
          "url": "https://rdap.markmonitor.com/rdap/"
        }
      ]
    },
    {
      "id": "US",
      "description": "US registry repository for US registrars",
      "registrars": [
        {
          "id": "9999",
          "description": "Example non-ICANN accredited registrar for .US ccTLD",
          "url": "https://rdap.registrar.example/rdap/"
        }
      ]
    }
  ]
}
```

13. References

13.1. 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>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [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>>.

13.2. Informative References

- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.
- [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>>.
- [RFC7484] Blanchet, M., "Finding the Authoritative Registration Data (RDAP) Service", RFC 7484, DOI 10.17487/RFC7484, March 2015, <<https://www.rfc-editor.org/info/rfc7484>>.

13.3. URIs

- [1] <https://www.iana.org/assignments/registrar-ids/registrar-ids.xhtml>
- [2] <https://www.iana.org/assignments/enterprise-numbers/enterprise-numbers>
- [3] <https://pen.iana.org/pen/PenApplication.page>

Acknowledgements

The following people have provided comments and reviews improving the document significantly (in no particular order): Audric Schiltnknecht, Julien Bernard, James Gould.

Author's Address

Marc Blanchet
Viagenie
246 Aberdeen
Quebec, QC G1R 2E1
Canada

EMail: Marc.Blanchet@viagenie.ca
URI: <http://viagenie.ca>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 19, 2019

J. Gould
M. Pozun
VeriSign, Inc.
January 15, 2019

Login Security Extension for the Extensible Provisioning Protocol (EPP)
draft-gould-regext-login-security-03

Abstract

The Extensible Provisioning Protocol (EPP) includes a client authentication scheme that is based on a user identifier and password. The structure of the password field is defined by an XML Schema data type that specifies minimum and maximum password length values, but there are no other provisions for password management other than changing the password. This document describes an EPP extension that allows longer passwords to be created and adds additional security features to the EPP login command and response.

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 19, 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 | 2 |
| 1.1. Conventions Used in This Document | 3 |
| 2. Migrating to Newer Versions of This Extension | 3 |
| 3. Object Attributes | 4 |
| 3.1. Event | 4 |
| 3.2. "[LOGIN-SECURITY]" Password | 5 |
| 3.3. Dates and Times | 6 |
| 4. EPP Command Mapping | 6 |
| 4.1. EPP <login> Command | 6 |
| 5. Formal Syntax | 13 |
| 5.1. Login Security Extension Schema | 13 |
| 6. IANA Considerations | 15 |
| 6.1. XML Namespace | 15 |
| 6.2. EPP Extension Registry | 16 |
| 7. Implementation Status | 16 |
| 8. Security Considerations | 16 |
| 9. Acknowledgements | 17 |
| 10. References | 17 |
| 10.1. Normative References | 17 |
| 10.2. Informative References | 17 |
| 10.3. URIs | 17 |
| Appendix A. Change History | 18 |
| A.1. Change from 00 to 01 | 18 |
| A.2. Change from 01 to 02 | 18 |
| A.3. Change from 02 to 03 | 18 |
| Authors' Addresses | 18 |

1. Introduction

This document describes an Extensible Provisioning Protocol (EPP) extension for enhancing the security of the EPP login command in EPP RFC 5730. The enhancements include supporting longer passwords (or passphrases) than the 16-character maximum and providing a list of security events in the login response. The password (current and new) in EPP RFC 5730 can be overridden by the password included in the extension to extend past the 16-character maximum. The security events supported include: password expiry, client certificate expiry, insecure cipher, insecure TLS protocol, new password complexity, login security statistical warning, and a custom event. The attributes supported by the security events include identifying the event type or sub-type, indicating the security level of warning or error, a future or past-due expiration date, the value that resulted in the

event, the duration of the statistical event, and a free-form description with an optional language.

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.

"loginSec-0.3" is used as an abbreviation for "urn:ietf:params:xml:ns:epp:loginSec-0.3". The XML namespace prefix "loginSec" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

2. Migrating to Newer Versions of This Extension

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

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

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

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

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

3. Object Attributes

This extension adds additional elements to [RFC5730] login command and response. Only those new elements are described here.

3.1. Event

A security event, using the <loginSec:event> element, represents either a warning or error identified by the server after the client has connected and submitted the login command. There MAY be multiple events returned that provides information for the client to address. The <loginSec:event> MAY include a free form description. All of the security events use a consistent set of attributes, where the exact set of applicable attributes is based on the event type. The supported set of <loginSec:event> element attributes include:

"type": A REQUIRED attribute that defines the type of security event. The enumerated list of "type" values include:

- "password": Identifies a password expiry event, where the password expires in the future or has expired based on the "exDate" date and time.
- "certificate": Identifies a client certificate expiry event, where the client certificate will expire at the "exDate" date and time.
- "cipher": Identifies the use of an insecure or deprecated TLS cipher suite.
- "tlsProtocol": Identifies the use of an insecure or deprecated TLS protocol.
- "newPW": The new password does not meet the server password complexity requirements.
- "stat": Provides a login security statistical warning that MUST set the "name" of the statistic.
- "custom": Custom event type that MUST set the "name" attribute with the custom event type name.
- "name": Used to define a sub-type or the type name when the "type" attribute is "custom".
- "level": Defines the level of the event as either "warning" for a warning event that needs action, or "error" for an error event that requires immediate action.
- "exDate": Contains the date and time that a "warning" level has or will become an "error" level. At expiry there MAY be an error to connect or MAY be an error to login. An example is an expired certificate that will result in a error to connect or an expired password that may result in a failed login.

"value": Identifies the value that resulted in the login security event. An example is the negotiated insecure cipher suite or the negotiated insecure TLS protocol.

"duration": Defines the duration that a statistical event is associated with.

"lang": Identifies the language of the free form description if the negotiated language is something other than the default value of "en" (English).

Example login security event for a password expiring in a week:

```
<loginSec:event
  type="password"
  level="warning"
  exDate="2018-04-01T22:00:00.0Z"
  lang="en">
  Password expiration soon
</loginSec:event>
```

Example login security event for identifying 100 failed logins over the last day, using the "stat" sub-type of "failedLogins":

```
<loginSec:event
  type="stat"
  name="failedLogins"
  level="warning"
  value="100"
  duration="P1D">
  Excessive invalid daily logins
</loginSec:event>
```

3.2. "[LOGIN-SECURITY]" Password

The <loginSec:pw> element MUST override the [RFC5730] <pw> element only if the <pw> contains the predefined value of "[LOGIN-SECURITY]", which is a constant value for the server to use the <loginSec:pw> element for the password. Similarly, the <loginSec:newPW> element MUST override the [RFC5730] <newPW> element only if the <newPW> contains the predefined value of "[LOGIN-SECURITY]", which is a constant value for the server to use the <loginSec:newPW> element for the new password. The "[LOGIN-SECURITY]" pre-defined string MUST be supported by the server for the client to explicitly indicate to the server whether to use <loginSec:pw> element in place of the [RFC5730] <pw> element or to use the <loginSec:newPW> in place of the [RFC5730] <newPW> element.

3.3. 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].

4.1. EPP <login> Command

This extension defines additional elements to extend the EPP <login> command and response to be used in conjunction with [RFC5730].

The EPP <login> command is used to establish a session with an EPP server. This extension overrides the password that is passed with the [RFC5730] <pw> or the <newPW> element as defined in Section 3.2. A <loginSec:loginSec> element is sent along with the [RFC5730] <login> command and MUST contain at least one of the following child elements:

<loginSec:userAgent>: OPTIONAL client user agent that identifies the client software and platform used by the server to identify functional or security constraints, current security issues, and potential future functional or security issues for the client.

<loginSec:pw>: OPTIONAL plain text password that is case sensitive, has a minimum length of 6 characters, and has a maximum length that is up to server policy. All leading and trailing whitespace is removed, and all internal contiguous whitespace that includes #x9 (tab), #xA (linefeed), #xD (carriage return), and #x20 (space) is replaced with a single #x20 (space). This element MUST only be used if the [RFC5730] <pw> element is set to the "[LOGIN-SECURITY]" value.

<loginSec:newPW>: OPTIONAL plain text new password that is case sensitive, has a minimum length of 6 characters, and has a maximum length that is up to server policy. All leading and trailing whitespace is removed, and all internal contiguous whitespace that includes #x9 (tab), #xA (linefeed), #xD (carriage return), and #x20 (space) is replaced with a single #x20 (space). This element MUST only be used if the [RFC5730] <newPW> element is set to the "[LOGIN-SECURITY]" value.

Example login command that uses the <loginSec:pw> element instead of the [RFC5730] <pw> element to establish the session and includes the <loginSec:userAgent> element:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <login>
C:      <clID>ClientX</clID>
C:      <pw>[LOGIN-SECURITY]</pw>
C:      <options>
C:        <version>1.0</version>
C:        <lang>en</lang>
C:      </options>
C:      <svcs>
C:        <objURI>urn:ietf:params:xml:ns:obj1</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj2</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj3</objURI>
C:        <svcExtension>
C:          <extURI>urn:ietf:params:xml:ns:epp:loginSec-0.3</extURI>
C:        </svcExtension>
C:      </svcs>
C:    </login>
C:    <extension>
C:      <loginSec:loginSec
C:        xmlns:loginSec=
C:          "urn:ietf:params:xml:ns:epp:loginSec-0.3">
C:        <loginSec:userAgent>EPP SDK/1.0.0
C:          (Java 1.7.0_15; x86_64 Mac OS X 10.11.6)
C:        </loginSec:userAgent>
C:        <loginSec:pw>this is a long password</loginSec:pw>
C:      </loginSec:loginSec>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example login command that uses the <loginSec:pw> element instead of the [RFC5730] <pw> element to establish the session, and uses the <loginSec:newPW> element instead of the [RFC5730] <newPW> element to set the new password:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <login>
C:      <clID>ClientX</clID>
C:      <pw>[LOGIN-SECURITY]</pw>
C:      <newPW>[LOGIN-SECURITY]</newPW>
C:      <options>
C:        <version>1.0</version>
C:        <lang>en</lang>
C:      </options>
C:      <svcs>
C:        <objURI>urn:ietf:params:xml:ns:obj1</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj2</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj3</objURI>
C:        <svcExtension>
C:          <extURI>urn:ietf:params:xml:ns:epp:loginSec-0.3</extURI>
C:        </svcExtension>
C:      </svcs>
C:    </login>
C:    <extension>
C:      <loginSec:loginSec
C:        xmlns:loginSec=
C:          "urn:ietf:params:xml:ns:epp:loginSec-0.3">
C:        <loginSec:pw>this is a long password
C:        </loginSec:pw>
C:        <loginSec:newPW>new password that is still long
C:        </loginSec:newPW>
C:      </loginSec:loginSec>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example login command that uses the [RFC5730] <pw> element to establish the session, and uses the <loginSec:newPW> element instead of the [RFC5730] <newPW> element to set the new password:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <login>
C:      <clID>ClientX</clID>
C:      <pw>shortpassword</pw>
C:      <newPW>[LOGIN-SECURITY]</newPW>
C:      <options>
C:        <version>1.0</version>
C:        <lang>en</lang>
C:      </options>
C:      <svcs>
C:        <objURI>urn:ietf:params:xml:ns:obj1</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj2</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj3</objURI>
C:        <svcExtension>
C:          <extURI>urn:ietf:params:xml:ns:epp:loginSec-0.3</extURI>
C:        </svcExtension>
C:      </svcs>
C:    </login>
C:    <extension>
C:      <loginSec:loginSec
C:        xmlns:loginSec=
C:          "urn:ietf:params:xml:ns:epp:loginSec-0.3">
C:        <loginSec:newPW>new password that is still long
C:      </loginSec:newPW>
C:    </loginSec:loginSec>
C:  </extension>
C:  <clTRID>ABC-12345</clTRID>
C: </command>
C:</epp>
```

Upon a completed login command (success or failed), the extension MUST be included in the response based on the following conditions:

Client supports extension: client supports the extension based on the <svcExtension> element of the <login> command.

At least one login security event: The server has identified at least one login security event to communicate to the client.

The extension to the EPP response uses the <loginSec:loginSecData> element that contains the following child elements:

<loginSec:event>: One or more <loginSec:event> elements defined in Section 3.1.

Example EPP response to a successful login command where the password will expire in a week:

```
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:      <loginSec:loginSecData
S:        xmlns:loginSec=
S:          "urn:ietf:params:xml:ns:epp:loginSec-0.3">
S:        <loginSec:event
S:          type="password"
S:          level="warning"
S:          exDate="2018-04-01T22:00:00.0Z"
S:          lang="en">
S:          Password expiring in a week
S:        </loginSec:event>
S:      </loginSec:loginSecData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example EPP response to a failed login command where the password has expired and the new password does not meet the server complexity requirements:

```
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="2200">
S:      <msg>Authentication error</msg>
S:    </result>
S:    <extension>
S:      <loginSec:loginSecData
S:        xmlns:loginSec=
S:          "urn:ietf:params:xml:ns:epp:loginSec-0.3">
S:        <loginSec:event
S:          type="password"
S:          level="error"
S:          exDate="2018-03-26T22:00:00.0Z">
S:          Password has expired
S:        </loginSec:event>
S:        <loginSec:event
S:          type="newPW"
S:          level="error">
S:          New password does not meet complexity requirements
S:        </loginSec:event>
S:      </loginSec:loginSecData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example EPP response to a successful login command where there is a set of login security events:

```
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:      <loginSec:loginSecData
S:        xmlns:loginSec=
S:          "urn:ietf:params:xml:ns:epp:loginSec-0.3">
S:        <loginSec:event
```

```
S:         type="password"
S:         level="warning"
S:         exDate="2018-04-01T22:00:00.0Z"
S:         lang="en">
S:         Password expiration soon
S:     </loginSec:event>
S:     <loginSec:event
S:         type="certificate"
S:         level="warning"
S:         exDate="2018-04-02T22:00:00.0Z"/>
S:     <loginSec:event
S:         type="cipher"
S:         level="warning"
S:         value="TLS_RSA_WITH_AES_128_CBC_SHA">
S:         Non-PFS Cipher negotiated
S:     </loginSec:event>
S:     <loginSec:event
S:         type="tlsProtocol"
S:         level="warning"
S:         value="TLSv1.0">
S:         Insecure TLS protocol negotiated
S:     </loginSec:event>
S:     <loginSec:event
S:         type="stat"
S:         name="failedLogins"
S:         level="warning"
S:         value="100"
S:         duration="P1D">
S:         Excessive invalid daily logins
S:     </loginSec:event>
S:     <loginSec:event
S:         type="custom"
S:         name="myCustomEvent"
S:         level="warning">
S:         A custom login security event occurred
S:     </loginSec:event>
S: </loginSec:loginSecData>
S: </extension>
S: <trID>
S:     <clTRID>ABC-12345</clTRID>
S:     <svTRID>54321-XYZ</svTRID>
S: </trID>
S: </response>
S: </epp>
```

5. Formal Syntax

One schema is presented here that is the EPP Login Security 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.

5.1. Login Security Extension Schema

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
  <schema targetNamespace="urn:ietf:params:xml:ns:epp:loginSec-0.3"
    xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
    xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
    xmlns:loginSec="urn:ietf:params:xml:ns:epp:loginSec-0.3"
    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
        Login Security Extension Schema.
      </documentation>
    </annotation>

    <!-- login command extension elements -->
    <element name="loginSec" type="loginSec:loginSecType"/>

    <!--
    Attributes associated with the login command extension.
    -->
    <complexType name="loginSecType">
      <sequence>
        <element name="userAgent" type="token"
          minOccurs="0"/>
        <element name="pw" type="loginSec:pwType"
          minOccurs="0"/>
      </sequence>
    </complexType>
  </schema>
END
```



```
        <element name="newPW" type="loginSec:pwType"
          minOccurs="0"/>
      </sequence>
    </complexType>

    <simpleType name="pwType">
      <restriction base="token">
        <minLength value="6"/>
      </restriction>
    </simpleType>

    <!-- login response extension elements -->
    <element name="loginSecData" type="loginSec:loginSecDataType"/>

    <!--
      Attributes associated with the change.
    -->
    <complexType name="loginSecDataType">
      <sequence>
        <element name="event" type="loginSec:eventType"
          minOccurs="1" maxOccurs="unbounded"/>
      </sequence>
    </complexType>

    <complexType name="eventType">
      <simpleContent>
        <extension base="normalizedString">
          <attribute name="type"
            type="loginSec:typeEnum" use="required"/>
          <attribute name="name" type="token"/>
          <attribute name="level"
            type="loginSec:levelEnum" use="required"/>
          <attribute name="exDate" type="dateTime"/>
          <attribute name="value" type="token"/>
          <attribute name="duration"
            type="duration"/>
          <attribute name="lang"
            type="language" default="en"/>
        </extension>
      </simpleContent>
    </complexType>

    <!--
      Enumerated list of event types, with extensibility via "custom".
    -->
    <simpleType name="typeEnum">
      <restriction base="token">
```

```
        <enumeration value="password"/>
        <enumeration value="certificate"/>
        <enumeration value="cipher"/>
        <enumeration value="tlsProtocol"/>
        <enumeration value="newPW"/>
        <enumeration value="stat"/>
        <enumeration value="custom"/>
    </restriction>
</simpleType>

<!--
Enumerated list of levels.
-->
<simpleType name="levelEnum">
    <restriction base="token">
        <enumeration value="warning"/>
        <enumeration value="error"/>
    </restriction>
</simpleType>

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

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]. The following URI assignment is requested of IANA:

Registration request for the loginSec namespace:

URI: urn:ietf:params:xml:ns:epp:loginSec-0.3
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.

Registration request for the loginSec XML schema:

URI: urn:ietf:params:xml:schema:epp:loginSec-0.3
Registrant Contact: IESG
XML: See the "Formal Syntax" section of this document.

6.2. 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: "Login Security 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

7. Implementation Status

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

TBD

8. Security Considerations

The extension leaves the password (<pw> element) and new password (<newPW> element) minimum length beyond 6 characters and the maximum length up to sever policy. The server SHOULD enforce minimum and maximum length requirements that are appropriate for their operating environment. One example of a guideline for password length policies can be found in section 5 of NIST Special Publication 800-63B [1].

The client SHOULD NOT decrease the security of a new password by decreasing the length of the current password. For example, a client with a 20 character password set using the extension, should not use the login command in [RFC5730] without using the extension, to set a new password that is less than or equal to 16 characters.

The extension provides an extensible list of login security events to inform clients of connection and login warnings and errors.

9. Acknowledgements

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

- o Patrick Mevzek
- o Scott Hollenbeck

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<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>>.
- [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>>.

10.2. 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>>.

10.3. URIs

- [1] <https://pages.nist.gov/800-63-3/sp800-63b.html>

Appendix A. Change History

A.1. Change from 00 to 01

1. Based on the feedback from Patrick Mevzek and a proposal from Scott Hollenbeck, changed the minimum length of the password from 8 to 6, revised the description of the password, and added text in the Security Considerations section for the server password length policy.

A.2. Change from 01 to 02

1. Changed the XML namespace from urn:ietf:params:xml:ns:loginSec-0.3 to urn:ietf:params:xml:ns:epp:loginSec-0.3, and changed the XML schema registration from urn:ietf:params:xml:ns:loginSec-0.3 to urn:ietf:params:xml:schema:epp:loginSec-0.3 based on a request from IANA with draft-ietf-regext-allocation-token.

A.3. Change from 02 to 03

Updates based on the review by Patrick Mevzek, that include:

1. Fix the inconsistent case for newPW, that required a global change in the draft text and an update to the XML schema to "urn:ietf:params:xml:ns:loginSec-0.3".
2. Changed "contains the following child elements" to "MUST contain at least one of the following child elements", section "EPP <login> Command" to ensure that an empty <loginSec:loginSec> element is not passed.
3. Add "The client SHOULD NOT decrease the security of a new password by decreasing the length of the current password." along with an example to the "Security Considerations" section.

Authors' Addresses

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

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

Matthew Pozun
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
US

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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: August 5, 2019

T. Harrison
G. Michaelson
APNIC
A. Newton
ARIN
February 1, 2019

RDAP Mirroring Protocol (RMP)
draft-harrison-regext-rdap-mirroring-00

Abstract

The Registration Data Access Protocol (RDAP) is used by Regional Internet Registries (RIRs) and Domain Name Registries (DNRs) to provide access to their resource registration information. While most clients can retrieve the information they need on an ad hoc basis from the public services maintained by each registry, there are instances where local copies of those remote data sources need to be maintained, for various reasons (e.g. performance requirements). This document defines a protocol for transferring bulk RDAP response data and for keeping a local copy of that data up to date.

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 August 5, 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 (<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. Requirements Language | 3 |
| 2. RDAP Mirroring Protocol Implementation | 3 |
| 2.1. Overview | 3 |
| 2.2. File Definitions | 4 |
| 2.2.1. Update Notification File | 4 |
| 2.2.2. Snapshot File | 5 |
| 2.2.3. Delta File | 6 |
| 2.3. RDAP Objects | 7 |
| 2.4. Serial Numbers | 9 |
| 2.5. Server Use | 9 |
| 2.5.1. Initialization | 9 |
| 2.5.2. Publishing Updates | 10 |
| 2.5.3. Consolidation | 10 |
| 2.6. Client Use | 11 |
| 2.6.1. Processing the Update Notification File | 11 |
| 2.6.1.1. Initial | 11 |
| 2.6.1.2. Subsequent | 12 |
| 3. Operational Considerations | 12 |
| 4. Security Considerations | 12 |
| 5. Acknowledgements | 13 |
| 6. IANA Considerations | 13 |
| 7. References | 13 |
| 7.1. Normative References | 13 |
| 7.2. Informative References | 14 |
| Authors' Addresses | 14 |

1. Introduction

The Registration Data Access Protocol (RDAP) [RFC7480] is used by Regional Internet Registries (RIRs) and Domain Name Registries (DNRs) to provide access to their resource registration information. For a client, this typically involves following the bootstrap process [RFC7484] to determine the base URL for the query, constructing an RDAP request, sending it, and then processing the response.

This mode of operation is appropriate for many use cases. However, some clients may need local access to the whole data set:

their performance requirements may be such that the time required for sending/receiving HTTP requests to arbitrary remote servers is not acceptable;

they may be conducting analysis of the data set as a whole; or

they may be providing access to the data set in their own right, as an alternative to redirecting to the authoritative source for the data.

This document defines a protocol that can be used by a client to retrieve a local copy of a remote RDAP data set, as well as to maintain that local copy as further remote updates occur.

1.1. Requirements Language

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

2. RDAP Mirroring Protocol Implementation

2.1. Overview

A registry that wants to make use of this protocol publishes an Update Notification File to a specified URL. That file in turn links to a Snapshot File and a series of Delta Files. The Snapshot File contains all of the registry's RDAP state as at a given point in time. The Delta Files contain changes that have been made to the registry's RDAP state since the Snapshot File was generated.

As further changes are made to the registry's RDAP state, the registry publishes new Delta Files, amends the Update Notification File to include links to the new Delta Files, and then republishes the Update Notification File. Periodically, the registry regenerates and republishes the Snapshot File, which in turn allows for older Delta Files to be removed from the Update Notification File.

All files in the protocol are signed by the server using JSON Web Signature (JWS) [RFC7515]. The JSON Web Key (JWK) [RFC7517] used by the server to sign the files is distributed out-of-band.

A client that wants to make use of this protocol needs to learn the server's Update Notification File URL and JSON Web Key out-of-band. Once these are known, the client retrieves the Update Notification File, validates its signature, and follows the links in it to retrieve the Snapshot File and Delta Files. It validates the signatures on these files, and then uses them to initialize its local

state. It then records the serial number of the most-recently-issued Delta File, or of the Snapshot File if no Delta Files are present.

The client will then periodically retrieve the Update Notification File, determine the Delta Files that have been added since it was last retrieved by the client, retrieve those Delta Files, and update its local state accordingly.

A server may opt not to publish a Snapshot File in the Update Notification File. Such servers will only publish Delta Files in their Update Notification File, and must distribute the Snapshot File out-of-band.

2.2. File Definitions

2.2.1. Update Notification File

Example Update Notification File:

```
{
  "version": 1,
  "refresh": 3600,
  "snapshot": { "uri": "https://example.com/1/snapshot.json",
                 "serial": 1 },
  "deltas": [
    { "uri": "https://example.com/2/delta.json",
      "serial": 2 },
    { "uri": "https://example.com/3/delta.json",
      "serial": 3 },
  ]
}
```

The following validation rules MUST be observed when creating or parsing Update Notification Files:

Update Notification Files MUST be well-formed JSON [RFC8259].

The "version" attribute in the root element MUST be present, with a value of "1".

A "refresh" attribute MAY be present. If it is present, it is an integer representing how long the client should wait (in seconds) after retrieving the Update Notification File before attempting to retrieve it again.

A "snapshot" attribute containing a link to a Snapshot File MAY be present.

A "deltas" attribute containing an array of links to Delta Files MUST be present. If no Delta Files have been published by the server, this array will be empty.

The Delta File entries in the "deltas" attribute MUST be in serial number order, and the serial numbers MUST form a contiguous sequence.

If a Snapshot File is included, its serial number MUST either be equal to that of one of the Delta Files, or one less than the smallest Delta File serial number.

2.2.2. Snapshot File

Example Snapshot File:

```
{
  "version": 1,
  "serial": 3,
  "defaults": { "port43": "whois.example.com",
    ... },
  "objects": [
    { "id": "https://example.org/I1",
      "object": { "rdapConformance": [ "rdap_level_0" ],
        "objectClassName": "ip network",
        ... } },
    { "id": "https://example.org/D2",
      "object": { "rdapConformance": [ "rdap_level_0" ],
        "objectClassName": "domain",
        ... } },
    ...
  ]
}
```

The following validation rules MUST be observed when creating or parsing Snapshot Files:

Snapshot Files MUST be well-formed JSON [RFC8259].

The "version" attribute in the root element MUST be present, with a value of "1".

The "serial" attribute in the root element MUST be present, with a value that is an unsigned 32-bit integer.

An "objects" attribute containing an array of RDAP object (see [RFC7483]) and identifier pairs MUST be present. If no RDAP objects have been published by the server, this array will be empty.

A "defaults" attribute MAY be present. For each object received from the server, the client should treat the object as also having each attribute from the "defaults" attribute, except where the object already contains an attribute with that name. This applies to objects received both before and after the Snapshot File is processed.

2.2.3. Delta File

Example Delta File:

```
{
  "version": 1,
  "serial": 4,
  "defaults": { "port43": "whois-2.example.org",
    ... },
  "removed_objects": [ "1" ],
  "added_or_updated_objects": [
    { "id": "https://example.org/I3",
      "object": { "rdapConformance": [ "rdap_level_0" ],
        "objectClassName": "ip network",
        ... } },
    { "id": "https://example.org/D4",
      "object": { "rdapConformance": [ "rdap_level_0" ],
        "objectClassName": "domain",
        ... } },
    ...
  ]
}
```

The following validation rules MUST be observed when creating or parsing Delta Files:

Snapshot Files MUST be well-formed JSON [RFC8259].

The "version" attribute in the root element MUST be present, with a value of "1".

The "serial" attribute in the root element MUST be present, with a value that is an unsigned 32-bit integer.

A "defaults" attribute MAY be present. For each object received from the server, the client should treat the object as also having each attribute from the "defaults" attribute, except where the object already contains an attribute with that name. This applies to objects received both before and after the Delta File is processed.

A "removed_objects" attribute containing an array of RDAP object identifiers MUST be present. If no RDAP objects have been removed since the previous Delta File was generated by the server, this array will be empty.

An "added_or_updated_objects" attribute containing an array of RDAP object (see [RFC7483]) and identifier pairs MUST be present. If no RDAP objects have been added or updated since the previous Delta File was generated by the server, this array will be empty.

2.3. RDAP Objects

The base RDAP object definitions from [RFC7483] do not contain any mandatory attributes. For the purposes of this protocol, each RDAP object MUST include an "rdapConformance" attribute, so that a client can determine whether the object is one that it is able to process.

Each RDAP object is paired with an identifier (the "id" attribute in the parent object). The "id" attribute value MUST be a URI ([RFC3986]). Its value uniquely identifies an RDAP object within the data set, so as to support internal linking and for subsequent removal of the object by a Delta File.

In each RDAP object, each link to an RDAP object that is a member of the data set for which the server is providing mirroring MUST have as the value of its "href" attribute the identifier for that object (i.e. the value of the "id" attribute from the target's parent object). This includes self-references (i.e. links with the "self" relation type).

If a server includes a link object with the "self" relation type with each of its RDAP objects, then using the value of the "href" attribute of that link object as the identifier for each RDAP object is RECOMMENDED.

For example, a Delta File containing an entity object, along with an IP network object that links to it:

```
{
  "version": 1,
  "serial": 5,
  "removed_objects": [],
  "added_or_updated_objects": [
    { "id": "https://example.org/E5",
      "object": { "rdapConformance": [ "rdap_level_0" ],
                  "objectClassName": "entity",
                  "handle": "E5",
                  "links": [
                    { "rel": "self",
                      "href": "https://example.org/E5",
                      ... }
                  ],
                  ... } },
    { "id": "https://example.org/I6",
      "object": { "rdapConformance": [ "rdap_level_0" ],
                  "objectClassName": "ip network",
                  "links": [
                    { "rel": "self",
                      "href": "https://example.org/I6",
                      ... }
                  ],
                  "entities": [
                    { "handle": "E5",
                      "links": [
                        { "rel": "self",
                          "href": "https://example.org/E5",
                          ... }
                      ],
                      ... }
                  ],
                  ... } }
    ...
  ]
}
```

A server MAY omit from an object data that it returns as part of its corresponding public service response, when that data can be determined by reference to another object in the data set. In such cases, the server MUST include a "links" attribute containing a link object with a "self" relation, so that the target object can be resolved by the client.

2.4. Serial Numbers

Serial numbers in the files defined in this protocol are unsigned 32-bit integers. For the purposes of this protocol, the serial number arithmetic defined in [RFC1982] applies.

2.5. Server Use

2.5.1. Initialization

If a server is publishing a Snapshot File via the Update Notification File, then initialization is like so:

- generate an initial Snapshot File, with a serial number selected by the server;

- sign the Snapshot File using JWS, and publish the result using JWS Compact Serialization at a URL that is unique to this serial number;

- generate an initial Update Notification File, with a serial number equal to that of the Snapshot File, and containing a link to the Snapshot File; and

- sign the Update Notification File using JWS, and publish the result using JWS Compact Serialization.

To avoid doubt, any RDAP object that is part of the data set for which the server is providing mirroring, as well as being the target of a link contained within another RDAP object, MUST be present within the Snapshot File.

If a server is not publishing a Snapshot File via the Update Notification File, then initialization is like so:

- generate an initial Snapshot File, with a serial number selected by the server;

- sign the Snapshot File using JWS, and distribute the result to clients out-of-band using JWS Compact Serialization;

- generate an initial Update Notification File, containing no Snapshot File link or Delta File links, with a serial number equal to that of the Snapshot File (i.e. the one that will be distributed out-of-band); and

- sign the Update Notification File using JWS, and publish the result using JWS Compact Serialization.

2.5.2. Publishing Updates

The server periodically publishes changes that have been made to its RDAP state as Delta Files. The timing and frequency of publication is a local policy matter for the server. The process is like so:

if a Delta File has been generated previously: generate a new Delta File, containing the changes that have been made to the RDAP state since the last Delta File was generated, with a serial number that is one greater than the serial number of the last Delta File;

if no Delta File has been generated previously: generate a new Delta File, containing the changes that have been made to the RDAP state since the last Snapshot File was generated, with a serial number that is one greater than the serial number of the last Snapshot File;

sign the Delta File using JWS, and publish the result using JWS Compact Serialization at a URL that is unique to its serial number;

take the currently-published Update Notification File, increment its serial number, add a link to the new Delta File, and optionally perform the steps described in the "Consolidation" section below; and

sign the Update Notification File using JWS, and publish the result using JWS Compact Serialization.

Delta Files MUST NOT remove an RDAP object that would cause a relative reference link within the client's local state to become unresolvable.

2.5.3. Consolidation

On publishing an update, the server may optionally consolidate the Snapshot File and Delta Files that it is publishing. The process is like so:

if the server is publishing a Snapshot File: generate a new Snapshot File based on the server's current state with a serial number equal to that of the new Delta File, publish the new Snapshot File, and replace the link in the Update Notification File to the previous Snapshot File with a link to the new Snapshot File; and

remove Delta Files from the Update Notification File that have become stale.

Whether a given Delta File is 'stale' is a local policy matter for the server.

2.6. Client Use

2.6.1. Processing the Update Notification File

2.6.1.1. Initial

The client downloads the signed Update Notification File using the URL provided by the server (out-of-band) and validates the signature against the server's JWK.

The client validates the signature of the Snapshot File against the server's JWK, and then uses that file to initialize its local state by adding all of the objects from the "objects" attribute. It then records the serial number of the Snapshot File. The signed Snapshot File is either accessible from the Update Notification File, or made available to the client out-of-band.

The client then processes each Delta File from the Update Notification File in order, from the Delta File with a serial number one greater than the client's recorded serial number, through to the Delta File with the largest serial number, in order to update its local state.

Processing a Delta File involves three steps:

- verify the signature against the server's JWK;

- for each entry in the "removed_objects" attribute, remove from the local state any object with an "id" attribute value equal to the entry;

- for each entry in the "added_or_updated_objects" attribute: if an object with the given values for the "id" attribute exists in the local state, then replace that object with the new object; otherwise, add the new object to the local state.

Once this is complete, the client records the serial number of the last Delta File that it processed.

2.6.1.2. Subsequent

The client downloads the signed Update Notification File using the URL provided by the server (out-of-band), and validates the signature against the server's JWK. If the frequency with which the client should do this has been suggested by the server via the "refresh" attribute, the client SHOULD honor that suggestion. If the "refresh" attribute is not present, retrieval frequency is a local policy matter for the client.

The client then processes each Delta File from the Update Notification File in order, from the Delta File with a serial number one greater than that which has been recorded, through to the Delta File with the largest serial number. Processing of the Delta Files is otherwise as per the instructions for initial processing.

If the Update Notification File retrieved by the client does not contain a Delta File with a serial number one greater than that which has been recorded, the client MUST delete all of its local state and reinitialize itself. If the Update Notification File contains a Snapshot File, then that Snapshot File can be used for reinitialization. If it does not, then a new Snapshot File must be located out-of-band.

3. Operational Considerations

A server may omit previously-published Delta Files from its Update Notification File as a matter of local policy. If a server is publishing its Snapshot Files out-of-band, then omitting a Delta File that a client needs will result in the client needing to perform an out-of-band action in order to reinitialize its state. Even if a server is linking to its Snapshot Files from the Update Notification File, reinitialization may be an expensive operation for a client. Servers should consider adopting local policy that limits the chance of reinitialization happening: for example, by using the "refresh" attribute value in the Update Notification File.

4. Security Considerations

[RFC7481] describes security requirements and considerations for RDAP generally. Those requirements and considerations also apply to the use of this protocol.

This protocol requires the use of JWS ([RFC7515]) and JWK ([RFC7517]), which in turn refer to JSON Web Algorithms (JWA) ([RFC7518]). Implementations MUST support ES256 as defined in JWA ([RFC7518], section 3.4) for signing and validating files in this protocol. Implementations MAY support other algorithms from the

"JSON Web Signature and Encryption Algorithms" registry created by [RFC7518].

5. Acknowledgements

This protocol is largely a repurposing of the RPKI Repository Delta Protocol (RRDP) [RFC8182] for RDAP. Much of the terminology (e.g. Update Notification File, Snapshot File, Delta File) is taken from that document, and the structure is also quite similar.

Experience with the Near Real Time Mirroring (NRTM) [NRTM] protocol, which serves a similar purpose for databases that are based on the Routing Policy Specification Language (RPSL) [RFC2622], helped to inform this corresponding effort in RDAP.

6. IANA Considerations

TBD

7. References

7.1. Normative References

- [RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <<https://www.rfc-editor.org/info/rfc1982>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.

- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <<https://www.rfc-editor.org/info/rfc7515>>.
- [RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <<https://www.rfc-editor.org/info/rfc7517>>.
- [RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <<https://www.rfc-editor.org/info/rfc7518>>.
- [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>>.

7.2. Informative References

- [NRTM] "Near Real Time Mirroring", December 2010, <<https://www.ripe.net/manage-ips-and-asns/db/support/documentation/mirroring>>.
- [RFC2622] Alaettinoglu, C., Villamizar, C., Gerich, E., Kessens, D., Meyer, D., Bates, T., Karrenberg, D., and M. Terpstra, "Routing Policy Specification Language (RPSL)", RFC 2622, DOI 10.17487/RFC2622, June 1999, <<https://www.rfc-editor.org/info/rfc2622>>.
- [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>>.
- [RFC7484] Blanchet, M., "Finding the Authoritative Registration Data (RDAP) Service", RFC 7484, DOI 10.17487/RFC7484, March 2015, <<https://www.rfc-editor.org/info/rfc7484>>.
- [RFC8182] Bruijnzeels, T., Muravskiy, O., Weber, B., and R. Austein, "The RPKI Repository Delta Protocol (RRDP)", RFC 8182, DOI 10.17487/RFC8182, July 2017, <<https://www.rfc-editor.org/info/rfc8182>>.

Authors' Addresses

Tom Harrison
Asia-Pacific Network Information Centre
6 Cordelia St
South Brisbane, QLD 4101
Australia

Email: tomh@apnic.net

George G. Michaelson
Asia-Pacific Network Information Centre
6 Cordelia St
South Brisbane, QLD 4101
Australia

Email: ggm@apnic.net

Andrew Lee Newton
American Registry for Internet Numbers
PO Box 232290
Centreville, VA 20120
United States of America

Email: andy@arin.net

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>

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 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

REGEXT Working Group
Internet-Draft
Intended status: Standards Track
Expires: 24 September 2022

S. Hollenbeck
Verisign Labs
23 March 2022

Federated Authentication for the Registration Data Access Protocol
(RDAP) using OpenID Connect
draft-ietf-regext-rdap-openid-12

Abstract

The Registration Data Access Protocol (RDAP) provides "RESTful" web services to retrieve registration metadata from domain name and regional internet registries. RDAP allows a server to make access control decisions based on client identity, and as such it includes support for client identification features provided by the Hypertext Transfer Protocol (HTTP). Identification methods that require clients to obtain and manage credentials from every RDAP server operator present management challenges for both clients and servers, whereas a federated authentication system would make it easier to operate and use RDAP without the need to maintain server-specific client credentials. This document describes a federated authentication system for RDAP based on OpenID Connect.

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 24 September 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 3 |
| 1.1. Problem Statement | 3 |
| 1.2. Proposal | 4 |
| 2. Conventions Used in This Document | 4 |
| 3. Federated Authentication for RDAP | 4 |
| 3.1. RDAP and OpenID Connect | 5 |
| 3.1.1. Terminology | 5 |
| 3.1.2. Overview | 5 |
| 3.1.3. RDAP Authentication and Authorization Steps | 6 |
| 3.1.3.1. Provider Discovery | 7 |
| 3.1.3.2. Authentication Request | 7 |
| 3.1.3.3. End-User Authorization | 8 |
| 3.1.3.4. Authorization Response and Validation | 8 |
| 3.1.3.5. Token Processing | 8 |
| 3.1.3.6. Delivery of User Information | 8 |
| 3.1.4. Specialized Claims for RDAP | 9 |
| 3.1.4.1. Stated Purpose | 9 |
| 3.1.4.2. Do Not Track | 10 |
| 4. Protocol Parameters | 10 |
| 4.1. Data Structures | 11 |
| 4.1.1. Session | 11 |
| 4.1.2. Device Info | 12 |
| 4.2. Client Login | 13 |
| 4.2.1. Clients with Limited User Interfaces | 15 |
| 4.2.1.1. UI-constrained Client Login | 15 |
| 4.2.1.2. UI-constrained Client Login Polling | 17 |
| 4.3. Session Status | 17 |
| 4.4. Session Refresh | 18 |
| 4.5. Client Logout | 20 |
| 4.6. Parameter Processing | 21 |
| 5. Token Exchange | 21 |
| 6. RDAP Query Processing | 21 |
| 7. RDAP Conformance | 22 |
| 8. IANA Considerations | 22 |
| 8.1. RDAP Extensions Registry | 22 |
| 8.2. JSON Web Token Claims Registry | 23 |
| 8.3. RDAP Query Purpose Registry | 23 |

| | |
|---|----|
| 9. Implementation Status | 26 |
| 9.1. Editor Implementation | 27 |
| 9.2. Verisign Labs | 27 |
| 9.3. Viagenie | 28 |
| 10. Security Considerations | 28 |
| 10.1. Authentication and Access Control | 29 |
| 11. Acknowledgments | 29 |
| 12. References | 29 |
| 12.1. Normative References | 29 |
| 12.2. Informative References | 31 |
| Appendix A. Change Log | 32 |
| Author's Address | 32 |

1. Introduction

The Registration Data Access Protocol (RDAP) provides "RESTful" web services to retrieve registration metadata from domain name and regional internet registries. RDAP allows a server to make access control decisions based on client identity, and as such it includes support for client identification features provided by the Hypertext Transfer Protocol (HTTP) [RFC7230].

RDAP is specified in multiple documents, including "HTTP Usage in the Registration Data Access Protocol (RDAP)" [RFC7480], "Security Services for the Registration Data Access Protocol (RDAP)" [RFC7481], "Registration Data Access Protocol Query Format" [RFC9082], and "JSON Responses for the Registration Data Access Protocol (RDAP)" [RFC9083]. RFC 7481 describes client identification and authentication services that can be used with RDAP, but it does not specify how any of these services can (or should) be used with RDAP.

1.1. Problem Statement

The traditional "user name and password" authentication method does not scale well in the RDAP ecosystem. Assuming that all domain name and address registries will eventually provide RDAP service, it is impractical and inefficient for users to secure login credentials from the hundreds of different server operators. Authentication methods based on user names and passwords do not provide information that describes the user in sufficient detail (while protecting the personal privacy of the user) for server operators to make fine-grained access control decisions based on the user's identity. The authentication system used for RDAP needs to address all of these needs.

1.2. Proposal

A basic level of RDAP service can be provided to users who possess an identifier issued by a recognized provider who is able to authenticate and validate the user. The identifiers issued by social media services, for example, can be used. Users who require higher levels of service (and who are willing to share more information about them self to gain access to that service) can secure identifiers from specialized providers who are or will be able to provide more detailed information about the user. Server operators can then make access control decisions based on the identification information provided by the user.

A federated authentication system in which an RDAP server outsources identification and authentication services to a trusted OpenID Provider would make it easier to operate and use RDAP by re-using existing identifiers to provide a basic level of access. It can also provide the ability to collect additional user identification information, and that information can be shared with the consent of the user. This type of system allows an RDAP server to make access control decisions based on the nature of a query and the identity, authentication, and authorization information that is received from the OpenID Provider. This document describes a federated authentication system for RDAP based on OpenID Connect [OIDC] that meets all of these needs.

2. 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.

3. Federated Authentication for RDAP

RDAP itself does not include native security services. Instead, RDAP relies on features that are available in other protocol layers to provide needed security services including access control, authentication, authorization, availability, data confidentiality, data integrity, and identification. A description of each of these security services can be found in "Internet Security Glossary, Version 2" [RFC4949]. This document focuses on a federated authentication system for RDAP that provides services for authentication, authorization, and identification, allowing a server operator to make access control decisions. Section 3 of RFC 7481 [RFC7481] describes general considerations for RDAP access control, authentication, and authorization.

The traditional client-server authentication model requires clients to maintain distinct credentials for every RDAP server. This situation can become unwieldy as the number of RDAP servers increases. Federated authentication mechanisms allow clients to use one credential to access multiple RDAP servers and reduce client credential management complexity.

3.1. RDAP and OpenID Connect

OpenID Connect 1.0 [OIDCC] is a decentralized, single sign-on (SSO) federated authentication system that allows users to access multiple web resources with one identifier instead of having to create multiple server-specific identifiers. Users acquire identifiers from OpenID Providers, or OPs. Relying Parties, or RPs, are applications (such as RDAP) that outsource their user authentication function to an OP. OpenID Connect is built on top of the authorization framework provided by the OAuth 2.0 [RFC6749] protocol.

The OAuth authorization framework describes a method for users to access protected web resources without having to hand out their credentials. Instead, clients are issued Access Tokens by authorization servers with the permission of the resource owners. Using OpenID Connect and OAuth, multiple RDAP servers can form a federation and clients can access any server in the federation by providing one credential registered with any OP in that federation. The OAuth authorization framework is designed for use with HTTP and thus can be used with RDAP.

3.1.1. Terminology

This document uses the terms "client" and "server" defined by RDAP [RFC7480]. An RDAP client performs the role of an OpenID Connect Core [OIDCC] Entity or End-User. An RDAP server performs the role of an OpenID Connect Core Relying Party (RP). Additional terms from Section 1.2 of the OpenID Connect Core specification are incorporated by reference.

3.1.2. Overview

At a high level, RDAP authentication of a browser-like client using OpenID Connect requires completion of the following steps:

1. An RDAP client sends an RDAP "help" query to an RDAP server to determine the type of OpenID Authorization Server that's used by the RDAP server. This information is returned in the `rdapConformance` section of the response. A value of `"rdap_openidc_local_level_0"` indicates that the server uses a local Authorization Server. A value of `"rdap_openidc_remote_level_0"` indicates that the server uses a remote Authorization Server.
2. An RDAP client (acting as an OpenID End-User) sends an RDAP "login" request to an RDAP server as described in Section 4.2.
3. The RDAP server (acting as an OpenID Relying Party (RP)) prepares an Authentication Request containing the desired request parameters.
4. The RDAP server sends the RDAP client and Authentication Request to an Authorization Server operated by an OpenID Provider (OP) using an HTTP redirect.
5. The Authorization Server authenticates the End-User.
6. The Authorization Server obtains End-User consent/authorization.
7. The Authorization Server sends the RDAP Client back to the RDAP server with an Authorization Code using an HTTP redirect.
8. The RDAP server requests a response using the Authorization Code at the Token Endpoint.
9. The RDAP server receives a response that contains an ID Token and Access Token in the response body.
10. The RDAP server validates the ID Token and retrieves the claims associated with the End-User's identity.

The RDAP server can then make identification, authorization, and access control decisions based on End-User identity information and local policies. Note that OpenID Connect describes different process flows for other types of clients, such as script-based or command line clients.

3.1.3. RDAP Authentication and Authorization Steps

End-Users MUST possess an identifier (an OpenID) issued by an OP to use OpenID Connect with RDAP. An OP SHOULD include support for the claims described in Section 3.1.4 to provide additional information needed for RDAP End-User authorization. OpenID Connect requires RPs to register with OPs to use OpenID Connect services for an End-User. The registration process is often completed using out-of-band methods, but it is also possible to use the automated method described by the "OpenID Connect Dynamic Client Registration" protocol [OIDCR]. The parties involved can use any method that is mutually acceptable.

3.1.3.1. Provider Discovery

An RDAP server/RP needs to be able to map an End-User's identifier to an OP. This can be accomplished using the OPTIONAL "OpenID Connect Discovery" protocol [OIDCD], but that protocol is not widely implemented. Out-of-band methods are also possible and can be more dependable. For example, an RP can support a limited number of OPs and maintain internal associations of those identifiers with the OPs that issued them. An RP can also ask an End-User to identify the OP that issued their identifier as part of an RDAP query workflow. In this case, the RP will need to maintain state for the association between the user's identifier and the OP in order to process later queries that rely on passing the access token and user identifier as authorization parameters. An RP MAY use any provider discovery approach that is suitable for its operating environment.

3.1.3.2. Authentication Request

Once the OP is known, an RP MUST form an Authentication Request and send it to the OP as described in Section 3 of the OpenID Connect Core protocol [OIDCC]. The authentication path followed (authorization, implicit, or hybrid) will depend on the Authentication Request `response_type` set by the RP. The remainder of the processing steps described here assume that the Authorization Code Flow is being used by setting "`response_type=code`" in the Authentication Request.

The benefits of using the Authorization Code Flow for authenticating a human user are described in Section 3.1 of the OpenID Connect Core protocol. The Implicit Flow is more commonly used by clients implemented in a web browser using a scripting language; it is described in Section 3.2 of the OpenID Connect Core protocol. The Hybrid Flow (described in Section 3.3 of the OpenID Connect Core protocol) combines elements of the Authorization and Implicit Flows by returning some tokens from the Authorization Endpoint and others from the Token Endpoint.

An Authentication Request can contain several parameters. REQUIRED parameters are specified in Section 3.1.2.1 of the OpenID Connect Core protocol [OIDCC]. Apart from these parameters, it is RECOMMENDED that the RP include the optional "`login_hint`" parameter in the request, with the value being that of the "`id`" query parameter of the End-User's RDAP "login" request. Passing the "`login_hint`" parameter allows a client to pre-fill login form information, so logging in can be more convenient for users. Other parameters MAY be included.

The OP receives the Authentication Request and attempts to validate it as described in Section 3.1.2.2 of the OpenID Connect Core protocol [OIDCC]. If the request is valid, the OP attempts to authenticate the End-User as described in Section 3.1.2.3 of the OpenID Connect Core protocol [OIDCC]. The OP returns an error response if the request is not valid or if any error is encountered.

3.1.3.3. End-User Authorization

After the End-User is authenticated, the OP MUST obtain authorization information from the End-User before releasing information to the RDAP Server/RP. This process is described in Section 3.1.2.4 of the OpenID Connect Core protocol [OIDCC].

3.1.3.4. Authorization Response and Validation

After the End-User is authenticated, the OP will send a response to the RP that describes the result of the authorization process in the form of an Authorization Grant. The RP MUST validate the response. This process is described in Sections 3.1.2.5 – 3.1.2.7 of the OpenID Connect Core protocol [OIDCC].

3.1.3.5. Token Processing

The RP sends a Token Request using the Authorization Grant to a Token Endpoint to obtain a Token Response containing an Access Token, ID Token, and an OPTIONAL Refresh Token. The RP MUST validate the Token Response. This process is described in Section 3.1.3 of the OpenID Connect Core protocol [OIDCC].

3.1.3.6. Delivery of User Information

The set of claims can be retrieved by sending a request to a UserInfo Endpoint using the Access Token. The claims MAY be returned in the ID Token. The process of retrieving claims from a UserInfo Endpoint is described in Section 5.3 of the OpenID Connect Core protocol [OIDCC].

OpenID Connect specifies a set of standard claims in Section 5.1. Additional claims for RDAP are described in Section 3.1.4.

3.1.4. Specialized Claims for RDAP

OpenID Connect claims are pieces of information used to make assertions about an entity. Section 5 of the OpenID Connect Core protocol [OIDCC] describes a set of standard claims that can be used to identify a person. Section 5.1.2 notes that additional claims MAY be used, and it describes a method to create them. The set of claims that are specific to RDAP are associated with an OAuth scope request parameter value (see Section 3.3 of RFC 6749 ([RFC6749])) of "rdap".

3.1.4.1. Stated Purpose

There are communities of RDAP users and operators who wish to make and validate claims about a user's "need to know" when it comes to requesting access to a resource. For example, a law enforcement agent or a trademark attorney may wish to be able to assert that they have a legal right to access a protected resource, and a server operator will need to be able to receive and validate that claim. These needs can be met by defining and using an additional "purpose" claim.

The "purpose" claim identifies the purpose for which access to a protected resource is being requested. Use of the "purpose" claim is OPTIONAL; processing of this claim is subject to server acceptance of the purpose and successful authentication of the End-User. Unrecognized purpose values MUST be ignored and the associated query MUST be processed as if the unrecognized purpose value was not present at all.

The "purpose" value is a case-sensitive string containing a StringOrURI value as specified in Section 2 of the JSON Web Token (JWT) specification ([RFC7519]). An example:

```
{"purpose" : "domainNameControl"}
```

Purpose values are themselves registered with IANA. Each entry in the registry contains the following fields:

Value: the purpose string value being registered. Value strings can contain upper case characters from "A" to "Z", lower case ASCII characters from "a" to "z", and the underscore ("_") character. Value strings contain at least one character and no more than 64 characters.

Description: a one- or two-sentence description of the meaning of the purpose value, how it might be used, and/or how it should be interpreted by clients and servers.

This registry is operated under the "Specification Required" policy defined in RFC 5226 ([RFC5226]). The set of initial values used to populate the registry as described in Section 8.3 are taken from the final report (<https://www.icann.org/en/system/files/files/final-report-06jun14-en.pdf>) produced by the Expert Working Group on gTLD Directory Services chartered by the Internet Corporation for Assigned Names and Numbers (ICANN).

3.1.4.2. Do Not Track

There are also communities of RDAP users and operators who wish to make and validate claims about a user's wish to not have their queries logged, tracked, or recorded. For example, a law enforcement agent may wish to be able to assert that their queries are part of a criminal investigation and should not be tracked due to a risk of query exposure compromising the investigation, and a server operator will need to be able to receive and validate that claim. These needs can be met by defining and using an additional "do not track" claim.

The "do not track" ("dnt") claim can be used to identify an End-User that is authorized to perform queries without the End-User's association with those queries being logged, tracked, or recorded by the server. Client use of the "dnt" claim is OPTIONAL. Server operators MUST NOT log, track, or record any association of the query and the End-User's identity if the End-User is successfully identified and authorized, the "dnt" claim is present, the value of the claim is "true", and accepting the claim complies with local regulations regarding logging and tracking.

The "dnt" value is represented as a JSON boolean literal. An example:

```
{"dnt" : true}
```

No special query tracking processing is required if this claim is not present or if the value of the claim is "false". Use of this claim MUST be limited to End-Users who are granted "do not track" privileges in accordance with service policies and regulations. Specification of these policies and regulations is beyond the scope of this document.

4. Protocol Parameters

This specification adds the following protocol parameters to RDAP:

1. Data structures to return information that describes an established session and the information needed to establish a session for a UI-constrained device.

2. A query parameter to request authentication for a specific End-User identity.
3. Path segments to start, stop, refresh, and determine the status of an authenticated session for a specific End-User identity.

4.1. Data Structures

This specification describes two new data structures that are used to return information to a client: a "session" data structure that contains information that describes an established session, and a "deviceInfo" data structure that contains information that describes an active attempt to establish a session on a UI-constrained device.

4.1.1. Session

The "session" data structure is an object that contains two sub-objects:

1. A "userClaims" object that contains the set of claims associated with the End-User's identity as used/requested by the RDAP server to make access control decisions. The set of possible values is determined by OP policy.
2. A "sessionInfo" object that contains two members:
 - a. "tokenExpiration": an integer value that represents the number of seconds from the current time for which the Access Token remains valid, and
 - b. "tokenRefresh": A boolean value that indicates if the OP supports refresh tokens. As described in RFC 6749 [RFC6749], support for refresh tokens is OPTIONAL.

An example of a "session" data structure:

```
"session": {
  "userClaims": {
    "sub": "103892603076825016132",
    "name": "User Person",
    "given_name": "User",
    "family_name": "Person",
    "picture": "https://lh3.example.com/a-/AOh14=s96-c",
    "email": "user@example.com",
    "email_verified": true,
    "locale": "en",
    "purpose": "domainNameControl",
    "dnt": false
  },
  "sessionInfo": {
    "tokenExpiration": 3599,
    "tokenRefresh": true
  }
}
```

Figure 1

4.1.2. Device Info

The flow described in Section 3.1.3 requires an End-User to interact with a server using a user interface that can process HTTP. This will not work well in situations where the client is automated or an End-User is using a command line user interface such as curl (<http://curl.haxx.se/>) or wget (<https://www.gnu.org/software/wget/>). This limitation can be addressed using a web browser on a second device. The information that needs to be entered using the web browser is contained in the "deviceInfo" data structure.

The "deviceInfo" data structure is an object that contains three members:

1. "verification_url": the URL that the End-User needs to visit using the web browser,
2. "user_code": the string value that the End-User needs to enter on the form presented in the web browser, and
3. "expires_in": an integer value that represents the number of seconds after which the opportunity to visit the URL and enter the user_code will expire.

An example of a "deviceInfo" data structure:

```
"deviceInfo": {  
  "verification_url": "https://www.example.com/device",  
  "user_code": "NJJQ-GJFC",  
  "expires_in": "1800"  
}
```

Figure 2

4.2. Client Login

Client authentication is requested by sending a "session/login" request to an RDAP server. If the RDAP server supports only remote Authorization Servers, the "session/login" request MUST include an End-User identifier that's delivered using one of two methods: by adding a query component to an RDAP request URI using the syntax described in Section 3.4 of RFC 3986 [RFC3986], or by including an HTTP authorization header for the Basic authentication scheme as described in RFC 7617 [RFC7617]. Clients can use either of these methods to deliver the End-User identifier to a server that supports remote Authorization Servers. Servers that support remote Authorization Servers MUST accept both methods. If the RDAP server supports a local Authorization Server, the End-User identifier MAY be omitted.

The query used to request client authentication is represented as an OPTIONAL "key=value" pair using a key value of "id" and a value component that contains the client identifier issued by an OP. An example for client identifier "user.idp.example":

```
https://example.com/rdap/session/login?id=user.idp.example
```

The authorization header for the Basic authentication scheme contains a Base64-encoded representation of the client identifier issued by an OP. No password is provided. An example for client identifier "user.idp.example":

```
https://example.com/rdap/session/login
```

```
Authorization: Basic dXNlci5pZHAuZXhhbXBsZQ==
```

An example for use with a local Authorization Server:

```
https://example.com/rdap/session/login
```

The response to this request MUST use the response structures specified in RFC 9083 [RFC9083]. In addition, the response MUST include an indication of the requested operation's success or failure in the "notices" data structure (including the client identifier), and, if successful, a "session" data structure.

An example of a successful "session/login" response:

```
{
  "rdapConformance": [
    "rdap_openidc_remote_level_0"
  ],
  "lang": "en-US",
  "notices": {
    "title": "Login Result",
    "description": [
      "Login succeeded",
      "user.idp.example"
    ],
  },
  "session": {
    "userClaims": {
      "sub": "103892603076825016132",
      "name": "User Person",
      "given_name": "User",
      "family_name": "Person",
      "picture": "https://lh3.example.com/a-/AOh14=s96-c",
      "email": "user@example.com",
      "email_verified": true,
      "locale": "en",
      "purpose": "domainNameControl",
      "dnt": false
    },
    "sessionInfo": {
      "tokenExpiration": 3599,
      "tokenRefresh": true
    }
  }
}
```

Figure 3

An example of a failed "session/login" response:

```
{
  "rdapConformance": [
    "rdap_openidc_remote_level_0"
  ],
  "lang": "en-US",
  "notices": {
    "title": "Login Result",
    "description": [
      "Login failed",
      "user.idp.example"
    ]
  }
}
```

Figure 4

4.2.1. Clients with Limited User Interfaces

The "OAuth 2.0 Device Authorization Grant" [RFC8628] provides an OPTIONAL method to request user authorization from devices that have an Internet connection, but lack a suitable browser for a more traditional OAuth flow. This method requires an End-User to use a second device (such as a smart telephone) that has access to a web browser for entry of a code sequence that is presented on the UI-constrained device.

4.2.1.1. UI-constrained Client Login

Client authentication is requested by sending a "session/device" request to an RDAP server. If the RDAP server supports only remote Authorization Servers, the "session/device" request MUST include an End-User identifier that's delivered using one of two methods: by adding a query component to an RDAP request URI using the syntax described in Section 3.4 of RFC 3986 [RFC3986], or by including an HTTP authorization header for the Basic authentication scheme as described in RFC 7617 [RFC7617]. If the RDAP server supports a local Authorization Server, the End-User identifier MAY be omitted. Clients can use either of these methods. Servers MUST support both methods.

The query used to request client authentication is represented as an OPTIONAL "key=value" pair using a key value of "id" and a value component that contains the client identifier issued by an OP.

An example using wget for client identifier "user.idp.example":

```
wget -qO- --keep-session-cookies --save-cookies\
https://example.com/rdap/session/device?id=user.idp.example
```


Figure 5

The authorization header for the Basic authentication scheme contains a Base64-encoded representation of the client identifier issued by an OP. No password is provided.

An example using curl and an authorization header:

```
curl -H "Authorization: Bearer dXNlci5pZHAuZXhhbXBsZQ=="\  
-c cookies.txt https://example.com/rdap/session/device
```

Figure 6

The response to this request MUST use the response structures specified in RFC 9083 [RFC9083]. In addition, the response MUST include an indication of the requested operation's success or failure in the "notices" data structure (including the client identifier), and, if successful, a "deviceInfo" data structure.

An example of a "session/device" response:

```
{  
  "rdapConformance": [  
    "rdap_openidc_remote_level_0"  
  ],  
  "lang": "en-US",  
  "notices": {  
    "title": "Device Login Result",  
    "description": [  
      "Login succeeded",  
      "user.idp.example"  
    ]  
  },  
  "deviceInfo": {  
    "verification_url": "https://www.example.com/device",  
    "user_code": "NJJQ-GJFC",  
    "expires_in": 1800  
  }  
}
```

Figure 7

4.2.1.2. UI-constrained Client Login Polling

After successful processing of the "session/device" request, the client MUST send a "session/devicepoll" request to the RDAP server to continue the login process. This request performs the polling function described in RFC 8628 [RFC8628], allowing the RDAP server to wait for the End-User to enter the information returned from the "session/device" request using the interface on their second device. After the End-User has completed that process, or if the process fails or times out, the OP will respond to the polling requests with an indication of success or failure.

An example using wget:

```
wget -qO- --load-cookies cookies.txt\  
https://example.com/rdap/session/devicepoll
```

Figure 8

An example using curl:

```
curl -b cookies.txt https://example.com/rdap/session/devicepoll
```

Figure 9

The response to this request MUST use the response structures described in Section 4.2. RDAP query processing can continue normally on the UI-constrained device once the "login" process has been completed.

4.3. Session Status

Clients MAY send a query to an RDAP server to determine the status of an existing login session using a "session/status" path segment. An example "session/status" request:

```
https://example.com/rdap/session/status
```

The response to this query MUST use the response structures specified in RFC 9083 [RFC9083]. In addition, the response MUST include an indication of the requested operation's success or failure in the "notices" data structure (including the client identifier), and, if successful, a "session" data structure.

An example of a "session/status" response:

```
{
  "rdapConformance": [
    "rdap_openidc_remote_level_0"
  ],
  "lang": "en-US",
  "notices": {
    "title": "Session Status Result",
    "description": [
      "Session status succeeded",
      "user.idp.example"
    ]
  },
  "session": {
    "userClaims": {
      "sub": "103892603076825016132",
      "name": "User Person",
      "given_name": "User",
      "family_name": "Person",
      "picture": "https://lh3.example.com/a-/AOh14=s96-c",
      "email": "user@example.com",
      "email_verified": true,
      "locale": "en",
      "purpose": "domainNameControl",
      "dnt": false
    },
    "sessionInfo": {
      "tokenExpiration": 3490,
      "tokenRefresh": true
    }
  }
}
```

Figure 10

4.4. Session Refresh

Clients MAY send a request to an RDAP server to refresh, or extend, an existing login session using a "session/refresh" path segment. The RDAP server MAY attempt to refresh the access token associated with the current session as part of extending the session for a period of time determined by the RDAP server. As described in RFC 6749 [RFC6749], OP support for refresh tokens is OPTIONAL. An RDAP server MUST determine if the OP supports token refresh and process the refresh request by either requesting refresh of the access token or by returning a response that indicates that token refresh is not supported by the OP in the "notices" data structure. An example "session/refresh" request:

`https://example.com/rdap/session/refresh`

The response to this request MUST use the response structures specified in RFC 9083 [RFC9083]. In addition, the response MUST include an indication of the requested operation's success or failure in the "notices" data structure (including the client identifier), and, if successful, a "session" data structure.

An example of a "session/refresh" response:

```
{
  "rdapConformance": [
    "rdap_openidc_remote_level_0"
  ],
  "lang": "en-US",
  "notices": {
    "title": "Session Refresh Result",
    "description": [
      "Session refresh succeeded",
      "user.idp.example",
      "Token refresh succeeded."
    ]
  },
  "session": {
    "userClaims": {
      "sub": "103892603076825016132",
      "name": "User Person",
      "given_name": "User",
      "family_name": "Person",
      "picture": "https://lh3.example.com/a-/AOh14=s96-c",
      "email": "user@example.com",
      "email_verified": true,
      "locale": "en",
      "purpose": "domainNameControl",
      "dnt": false
    },
    "sessionInfo": {
      "tokenExpiration": 3599,
      "tokenRefresh": true
    }
  }
}
```

Figure 11

4.5. Client Logout

Clients MAY send a request to an RDAP server to terminate an existing login session. Termination of a session is requested using a "session/logout" path segment. Access and refresh tokens can be revoked during the "session/logout" process as described in RFC 7009 [RFC7009] if supported by the OP (token revocation endpoint support is OPTIONAL per RFC 8414 [RFC8414]). If supported, this feature SHOULD be used to ensure that the tokens are not mistakenly associated with a future RDAP session. Alternatively, an RDAP server MAY attempt to logout from the OP using the "OpenID Connect RP-Initiated Logout" protocol ([OIDCL]) if that protocol is supported by the OP.

An example "session/logout" request:

`https://example.com/rdap/session/logout`

The response to this request MUST use the response structures specified in RFC 9083 [RFC9083]. In addition, the response MUST include an indication of the requested operation's success or failure in the "notices" data structure (including the client identifier). The "notices" data structure MUST also include an indication of the success or failure of any attempt to logout from the OP or to revoke the tokens issued by the OP.

An example of a "session/logout" response:

```
{
  "rdapConformance": [
    "rdap_openidc_remote_level_0"
  ],
  "lang": "en-US",
  "notices": {
    "title": "Logout Result",
    "description": [
      "Logout succeeded",
      "user.idp.example",
      "Provider logout failed: Not supported by provider.",
      "Token revocation successful."
    ]
  }
}
```

Figure 12

In the absence of a "logout" request, an RDAP session MUST be terminated by the RDAP server after a server-defined period of time. The server should also take appropriate steps to ensure that the tokens associated with the terminated session cannot be reused. This SHOULD include revoking the tokens or logging out from the OP if either operation is supported by the OP.

4.6. Parameter Processing

Unrecognized query parameters MUST be ignored. An RDAP server that processes an authenticated query MUST determine if the End-User identification information is associated with an OP that is recognized and supported by the server. Servers MUST reject queries that include identification information that is not associated with a supported OP by returning an HTTP 501 (Not Implemented) response. An RDAP server that receives a query containing identification information associated with a recognized OP MUST perform the steps required to authenticate the user with the OP, process the query, and return an RDAP response that is appropriate for the End-User's level of authorization and access.

5. Token Exchange

ID tokens include an audience parameter that contains the OAuth 2.0 client_id of the RP as an audience value. In some operational scenarios (such as a client that is providing a proxy service), an RP can receive tokens with an audience value that does not include the RP's client_id. These tokens might not be trusted by the RP, and the RP might refuse to accept the tokens. This situation can be remedied by having the RP exchange these tokens with the OP for a set of trusted tokens that reset the audience parameter. This token exchange protocol is described in RFC 8693 [RFC8693]. This issue is not visible to the RDAP client and should be managed by the OpenID implementation used by the RDAP server.

6. RDAP Query Processing

Once an RDAP session is active, an RDAP server MUST determine if the End-User is authorized to perform any queries that are received during the duration of the session. This MAY include rejecting queries outright, and it MAY include omitting or otherwise redacting information that the End-User is not authorized to receive. Specific processing requirements are beyond the scope of this document. A client can end a session explicitly by sending a "session/logout" request to the RDAP server. A session can also be ended implicitly by the server after a server-defined period of time. The status of a session can be determined at any time by sending a "session/status" query to the RDAP server.

An RDAP server MUST maintain session state information for the duration of an active session. This is commonly done using HTTP cookies as described in RFC 6265 [RFC6265]. Doing so allows End-User to submit queries without having to explicitly identify and authenticate themselves for each and every query.

7. RDAP Conformance

RDAP responses that contain values described in this document MUST indicate conformance with this specification by including an `rdapConformance` ([RFC9083]) value of `"rdap_openidc_remote_level_0"` (to indicate support for one or more remote Authorization Servers), `"rdap_openidc_local_level_0"` (to indicate support for a local Authorization Server), or both values if the server supports both remote and local OpenID Authorization Servers. The information needed to register these values in the RDAP Extensions Registry is described in Section 8.1.

Example `rdapConformance` structure with extension specified:

```
"rdapConformance" :  
  [  
    "rdap_level_0",  
    "rdap_openidc_remote_level_0"  
  ]
```

Figure 13

8. IANA Considerations

8.1. RDAP Extensions Registry

IANA is requested to register the following values in the RDAP Extensions Registry:

Extension identifier: `rdap_openidc_remote_level_0`
Registry operator: Any
Published specification: This document.
Contact: IESG <iesg@ietf.org>
Intended usage: This extension describes a federated authentication method for RDAP using OAuth 2.0, OpenID Connect, and a remote Authorization Server.

Extension identifier: `rdap_openidc_local_level_0`
Registry operator: Any
Published specification: This document.
Contact: IESG <iesg@ietf.org>

Intended usage: This extension describes a federated authentication method for RDAP using OAuth 2.0, OpenID Connect, and a local Authorization Server.

8.2. JSON Web Token Claims Registry

IANA is requested to register the following values in the JSON Web Token Claims Registry:

Claim Name: "purpose"

Claim Description: This claim describes the stated purpose for submitting a request to access a protected RDAP resource.

Change Controller: IESG

Specification Document(s): Section 3.1.4.1 of this document.

Claim Name: "dnt"

Claim Description: This claim contains a JSON boolean literal that describes an End-User's "do not track" preference for identity tracking, logging, or recording when accessing a protected RDAP resource.

Change Controller: IESG

Specification Document(s): Section 3.1.4.2 of this document.

8.3. RDAP Query Purpose Registry

IANA is requested to create a new protocol registry to manage RDAP query purpose values. This registry should be named "Registration Data Access Protocol (RDAP) Query Purpose Values" and should appear under the "Registration Data Access Protocol (RDAP)" section of IANA's protocol registries. The information to be registered and the procedures to be followed in populating the registry are described in Section 3.1.4.1.

Section at <http://www.iana.org/protocols>: Registration Data Access Protocol (RDAP)

Name of registry: Registration Data Access Protocol (RDAP) Query Purpose Values

Registration Procedure: Specification Required

Reference: This document

Required information: See Section 3.1.4.1.

Review process: "Specification Required" as described in RFC 5226 [RFC5226].

Size, format, and syntax of registry entries: See Section 3.1.4.1.

Initial assignments and reservations:

-----BEGIN FORM-----

Value: domainNameControl

Description: Tasks within the scope of this purpose include creating and managing and monitoring a registrant's own domain name, including creating the domain name, updating information about the domain name, transferring the domain name, renewing the domain name, deleting the domain name, maintaining a domain name portfolio, and detecting fraudulent use of the Registrant's own contact information.

-----END FORM-----

-----BEGIN FORM-----

Value: personalDataProtection

Description: Tasks within the scope of this purpose include identifying the accredited privacy/proxy provider associated with a domain name and reporting abuse, requesting reveal, or otherwise contacting the provider.

-----END FORM-----

-----BEGIN FORM-----

Value: technicalIssueResolution

Description: Tasks within the scope of this purpose include (but are not limited to) working to resolve technical issues, including email delivery issues, DNS resolution failures, and web site functional issues.

-----END FORM-----

-----BEGIN FORM-----

Value: domainNameCertification

Description: Tasks within the scope of this purpose include a Certification Authority (CA) issuing an X.509 certificate to a subject identified by a domain name.

-----END FORM-----

-----BEGIN FORM-----

Value: individualInternetUse

Description: Tasks within the scope of this purpose include identifying the organization using a domain name to instill consumer trust, or contacting that organization to raise a customer complaint to them or file a complaint about them.

-----END FORM-----

-----BEGIN FORM-----

Value: businessDomainNamePurchaseOrSale

Description: Tasks within the scope of this purpose include making purchase queries about a domain name, acquiring a domain name from a registrant, and enabling due diligence research.

-----END FORM-----

-----BEGIN FORM-----

Value: academicPublicInterestDNSRRResearch

Description: Tasks within the scope of this purpose include academic public interest research studies about domain names published in the registration data service, including public information about the registrant and designated contacts, the domain name's history and status, and domain names registered by a given registrant (reverse query).

-----END FORM-----

-----BEGIN FORM-----

Value: legalActions

Description: Tasks within the scope of this purpose include investigating possible fraudulent use of a registrant's name or address by other domain names, investigating possible trademark infringement, contacting a registrant/licensee's legal representative prior to taking legal action and then taking a legal action if the concern is not satisfactorily addressed.

-----END FORM-----

-----BEGIN FORM-----

Value: regulatoryAndContractEnforcement

Description: Tasks within the scope of this purpose include tax authority investigation of businesses with online presence, Uniform Dispute Resolution Policy (UDRP) investigation, contractual compliance investigation, and registration data escrow audits.

-----END FORM-----

-----BEGIN FORM-----

Value: criminalInvestigationAndDNSAbuseMitigation

Description: Tasks within the scope of this purpose include reporting abuse to someone who can investigate and address that abuse, or contacting entities associated with a domain name during an offline criminal investigation.

-----END FORM-----

-----BEGIN FORM-----

Value: dnsTransparency

Description: Tasks within the scope of this purpose involve querying the registration data made public by registrants to satisfy a wide variety of use cases around informing the general public.

-----END FORM-----

9. 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".

Version -09 of this specification introduced changes that are incompatible with earlier implementations. Implementations that are consistent with this specification will be added as they are identified.

9.1. Editor Implementation

Location: <https://procuratus.net/rdap/>

Description: This implementation is a functionally-limited RDAP server that supports only the path segments described in this specification. It uses the "jumbojett/OpenID-Connect-PHP" library found on GitHub, which appears to no longer be under active development. The library was modified to add support for the device authorization grant. Session variable management is still a little buggy. Supported OPs include Google (Gmail) and Yahoo.
Level of Maturity: This is a "proof of concept" research implementation.

Coverage: This implementation includes all of the features described in this specification.

Version compatibility: Version -11+ of this specification.

Contact Information: Scott Hollenbeck, shollenbeck@verisign.com

9.2. Verisign Labs

Responsible Organization: Verisign Labs

Location: <https://rdap.verisignlabs.com/>

Description: This implementation includes support for domain registry RDAP queries using live data from the .cc and .tv country code top-level domains and the .career generic top-level domain. Three access levels are provided based on the authenticated identity of the client:

1. Unauthenticated: Limited information is returned in response to queries from unauthenticated clients.

2. Basic: Clients who authenticate using a publicly available identity provider like Google Gmail or Microsoft Hotmail will receive all of the information available to an unauthenticated client plus additional registration metadata, but no personally identifiable information associated with entities.
3. Advanced: Clients who authenticate using a more restrictive identity provider will receive all of the information available to a Basic client plus whatever information the server operator deems appropriate for a fully authorized client. Currently supported identity providers include those developed by Verisign Labs (<https://testprovider.rdap.verisignlabs.com/>) and CZ.NIC (<https://www.mojeid.cz/>).

Level of Maturity: This is a "proof of concept" research implementation.

Coverage: This implementation includes all of the features described in this specification.

Version compatibility: Version -07 of this specification.

Contact Information: Scott Hollenbeck, shollenbeck@verisign.com

9.3. Viagenie

Responsible Organization: Viagenie

Location: <https://auth.viagenie.ca>

Description: This implementation is an OpenID identity provider enabling users and registries to connect to the federation. It also includes a barebone RDAP client and RDAP server in order to test the authentication framework. Various level of purposes are available for testing.

Level of Maturity: This is a "proof of concept" research implementation.

Coverage: This implementation includes most features described in this specification as an identity provider.

Version compatibility: Version -07 of this specification.

Contact Information: Marc Blanchet, marc.blanchet@viagenie.ca

10. Security Considerations

Security considerations for RDAP can be found in RFC 7481 [RFC7481]. Security considerations for OpenID Connect Core [OIDCC] and OAuth 2.0 [RFC6749] can be found in their reference specifications. OpenID Connect defines optional mechanisms for robust signing and encryption that can be used to provide data integrity and data confidentiality services as needed.

10.1. Authentication and Access Control

Having completed the client identification, authorization, and validation process, an RDAP server can make access control decisions based on a comparison of client-provided information and local policy. For example, a client who provides an email address (and nothing more) might be entitled to receive a subset of the information that would be available to a client who provides an email address, a full name, and a stated purpose. Development of these access control policies is beyond the scope of this document.

11. Acknowledgments

The author would like to acknowledge the following individuals for their contributions to the development of this document: Marc Blanchet, Tom Harrison, Russ Housley, Jasdip Singh, Rhys Smith, Jaromir Talir, Rick Wilhelm, and Alessandro Vesely. In addition, the Verisign Registry Services Lab development team of Joseph Harvey, Andrew Kaizer, Sai Mogali, Anurag Saxena, Swapneel Sheth, Nitin Singh, and Zhao Zhao provided critical "proof of concept" implementation experience that helped demonstrate the validity of the concepts described in this document.

Mario Loffredo provided significant feedback based on implementation experience that led to welcome improvements in several sections of this document. His contributions are greatly appreciated.

12. References

12.1. Normative References

- [OIDC] OpenID Foundation, "OpenID Connect",
<<http://openid.net/connect/>>.
- [OIDCC] OpenID Foundation, "OpenID Connect Core incorporating
errata set 1", November 2014,
<http://openid.net/specs/openid-connect-core-1_0.html>.
- [OIDCD] OpenID Foundation, "OpenID Connect Discovery 1.0
incorporating errata set 1", November 2014,
<http://openid.net/specs/openid-connect-discovery-1_0.html>.
- [OIDCL] OpenID Foundation, "OpenID Connect RP-Initiated Logout 1.0
- draft 01", August 2020, <[https://openid.net/specs/
openid-connect-rpinitiated-1_0.html](https://openid.net/specs/openid-connect-rpinitiated-1_0.html)>.

- [OIDCR] OpenID Foundation, "OpenID Connect Dynamic Client Registration 1.0 incorporating errata set 1", November 2014, <http://openid.net/specs/openid-connect-registration-1_0.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>>.
- [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>>.
- [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>>.
- [RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <<https://www.rfc-editor.org/info/rfc6265>>.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <<https://www.rfc-editor.org/info/rfc6749>>.
- [RFC7009] Lodderstedt, T., Ed., Dronia, S., and M. Scurtescu, "OAuth 2.0 Token Revocation", RFC 7009, DOI 10.17487/RFC7009, August 2013, <<https://www.rfc-editor.org/info/rfc7009>>.
- [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)", STD 95, 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)", STD 95, RFC 7481, DOI 10.17487/RFC7481, March 2015, <<https://www.rfc-editor.org/info/rfc7481>>.

- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <<https://www.rfc-editor.org/info/rfc7519>>.
- [RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <<https://www.rfc-editor.org/info/rfc7617>>.
- [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>>.
- [RFC8628] Denniss, W., Bradley, J., Jones, M., and H. Tschofenig, "OAuth 2.0 Device Authorization Grant", RFC 8628, DOI 10.17487/RFC8628, August 2019, <<https://www.rfc-editor.org/info/rfc8628>>.
- [RFC8693] Jones, M., Nadalin, A., Campbell, B., Ed., Bradley, J., and C. Mortimore, "OAuth 2.0 Token Exchange", RFC 8693, DOI 10.17487/RFC8693, January 2020, <<https://www.rfc-editor.org/info/rfc8693>>.
- [RFC9082] Hollenbeck, S. and A. Newton, "Registration Data Access Protocol (RDAP) Query Format", STD 95, RFC 9082, DOI 10.17487/RFC9082, June 2021, <<https://www.rfc-editor.org/info/rfc9082>>.
- [RFC9083] Hollenbeck, S. and A. Newton, "JSON Responses for the Registration Data Access Protocol (RDAP)", STD 95, RFC 9083, DOI 10.17487/RFC9083, June 2021, <<https://www.rfc-editor.org/info/rfc9083>>.

12.2. Informative References

- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <<https://www.rfc-editor.org/info/rfc4949>>.
- [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>>.
- [RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <<https://www.rfc-editor.org/info/rfc8414>>.

Appendix A. Change Log

- 00: Initial working group version ported from draft-hollenbeck-regext-rdap-openid-10.
- 01: Modified ID Token delivery approach to note proper use of an HTTP bearer authorization header.
- 02: Modified token delivery approach (Access Token is the bearer token) to note proper use of an HTTP bearer authorization header, fixing the change made in -01.
- 03: Updated OAuth 2.0 Device Authorization Grant description and reference due to publication of RFC 8628.
- 04: Updated OAuth 2.0 token exchange description and reference due to publication of RFC 8693. Corrected the RDAP conformance identifier to be registered with IANA.
- 05: Keepalive refresh.
- 06: Keepalive refresh.
- 07: Added "login_hint" description to Section 3.1.3.2. Added some text to Section 3.1.4.2 to note that "do not track" requires compliance with local regulations.
- 08: Rework of token management processing in Sections 4 and 5.
- 09: Updated RDAP specification references. Added text to describe both local and remote Authorization Server processing. Removed text that described passing of ID Tokens as query parameters.
- 10: Updated Section 3.1.3.1. Replaced token processing queries with "login", "session", and "logout" queries.
- 11: Replaced queries with "session/*" queries. Added description of "rdap" OAuth scope. Added implementation status information.
- 12: Updated data structure descriptions. Updated Section 8. Minor formatting changes due to a move to xml2rfc-v3 markup.

Author's Address

Scott Hollenbeck
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
United States of America
Email: shollenbeck@verisign.com
URI: <http://www.verisignlabs.com/>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 14, 2019

J. Gould
VeriSign, Inc.
January 10, 2019

Verification Code Extension for the Extensible Provisioning Protocol
(EPP)
draft-ietf-regext-verificationcode-06

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on July 14, 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. Verification Code | 4 |
| 2.1.1. Signed Code | 4 |
| 2.1.2. Encoded Signed Code | 7 |
| 2.2. Verification Profile | 11 |
| 3. EPP Command Mapping | 12 |
| 3.1. EPP Query Commands | 12 |
| 3.1.1. EPP <check> Command | 12 |
| 3.1.2. EPP <info> Command | 12 |
| 3.1.3. EPP <transfer> Command | 24 |
| 3.2. EPP Transform Commands | 25 |
| 3.2.1. EPP <create> Command | 25 |
| 3.2.2. EPP <delete> Command | 27 |
| 3.2.3. EPP <renew> Command | 28 |
| 3.2.4. EPP <transfer> Command | 28 |
| 3.2.5. EPP <update> Command | 28 |
| 4. Formal Syntax | 28 |
| 4.1. Verification Code Extension Schema | 28 |
| 5. IANA Considerations | 32 |
| 5.1. XML Namespace | 32 |
| 5.2. EPP Extension Registry | 32 |
| 6. Implementation Status | 33 |
| 6.1. Verisign EPP SDK | 33 |
| 6.2. Net::DRI | 34 |
| 7. Security Considerations | 34 |
| 8. References | 35 |
| 8.1. Normative References | 35 |
| 8.2. Informative References | 36 |
| Appendix A. Acknowledgements | 36 |
| Appendix B. Change History | 36 |
| B.1. Change from 00 to 01 | 36 |
| B.2. Change from 01 to 02 | 36 |
| B.3. Change from 02 to 03 | 36 |
| B.4. Change from 03 to 04 | 36 |
| B.5. Change from 04 to REGEXT 00 | 37 |
| B.6. Change from REGEXT 00 to REGEXT 01 | 37 |
| B.7. Change from REGEXT 01 to REGEXT 02 | 37 |
| B.8. Change from REGEXT 02 to REGEXT 03 | 37 |
| B.9. Change from REGEXT 03 to REGEXT 04 | 37 |

| | |
|--|----|
| B.10. Change from REGEXT 04 to REGEXT 05 | 37 |
| B.11. Change from REGEXT 05 to REGEXT 06 | 37 |
| Author's Address | 38 |

1. Introduction

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This mapping, an extension to EPP object mappings like the EPP domain name mapping [RFC5731], EPP host mapping [RFC5732], and EPP contact mapping [RFC5733], can be used to pass a verification code to one of the EPP transform commands. The domain name object is used for examples in the document. The verification code is signed using XML Signature [W3C.CR-xmlsig-core2-20120124] and is "base64" encoded. The "base64" encoded text of the verification code MUST conform to [RFC2045]. The verification code demonstrates that verification was done by a Verification Service Provider (VSP).

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

1.1. Conventions Used in This Document

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

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

2. Object Attributes

This extension adds additional elements to EPP object mappings like the EPP domain name mapping [RFC5731], EPP host mapping [RFC5732], and EPP contact mapping [RFC5733]. Only those new elements are described here.

2.1. Verification Code

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

Verification Code Token ABNF

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

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

2.1.1. Signed Code

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

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

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

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

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

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

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

```
</Transforms>
  <DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
  <DigestValue>wgyW3nZPoEfpptlhRILKnOQnbdU6ArM7ShrAfHgDFg=
  </DigestValue>
  </Reference>
</SignedInfo>
  <SignatureValue>
jMu4PfyQGgiJBF0GWSEPFcJjmywCEqR2h4LD+ge6XQ+JnmKFFCuCZS/3SLKAx0L1w
QDFO2e0Y69k2G7/LGE37X3vOflobFMloGwja8+GMVraoto5xAd4/AF7eHukgAymD
o9toxo2h0yV4A4PmXzsU6S86XtCcUE+S/WM72nyn47zoUCzzPKHZBryeWehVFQ+
jYRMIAMzM57HHQA+6eaXefRvtPETgUO4aVIVSugc4OUAZZwbYcZrC6wOaQqqqAZi
30aPOBYbAvHMSmWSS+hFkbshomJfHxb97TD2grlYNrQIzqXk7WbHWy2SYda+sI/Z
ipJsXNa6ostUw1CzA7jfwA==
  </SignatureValue>
  <KeyInfo>
    <X509Data>
      <X509Certificate>
MI IESTCCAzGgAwIBAgIBAgjANBgkqhkiG9w0BAQsFADBiMQswCQYDVQQGEwJVUzEL
MAkGA1UECBMCQ0ExFDASBgNVBACTC0xvcyBBbmdlbGVzMRRMwEQYDVQQKEwJQ0FO
TiBUTUNIMRswGQYDVQQDEwJJQ0FOTiBUTUNIIFRFU1QgQ0EwHhcNMTMwMjA4MDAw
MDAwWhcNMjgwMjA3MjM1OTU5WjBsmQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ0Ex
FDASBgNVBACTC0xvcyBBbmdlbGVzMRRMwEQYDVQQKEw5WYWxpZGF0b3IgcVE1DSDEh
MB8GA1UEAxMYVmFsaWRhdG9yIFRnQ0ggVEVTVCBDRVJUMIIBIjANBgkqhkiG9w0B
AQEFAAOCAQ8AMIIBCgKCAQEAo/cwvXhbVY10RDWWvoveZpETVZVVcMCovUVNg/sw
WinuMgEWgVQFrz0xA04pEhXCFVv4evbUpekJ5buqU1gmQy0sCKQ1hOHTdPjvkC5u
pDqa51Flk0TMAmKIQjs7aUKCmA4RG4tTTGK/EjR1ix8/D0gHYVRldy1YPrMP+ou7
5bOVnIos+HifrAtrIv4qEqwLL4FTZAUpaCa2BmgXfy2CSRQbxD5OrlgcSa3vurh5
sPMCNxqaXmIXmQipS+DuEBqMM8tldaN7RYojUEKRGVsNk5i9y2/7sjnlzzyUPf7v
L4GgDYqhJYWV61DnXgx/Jd6CWxvsndf6scscQzUTE1+hywIDAQABO4H/MIH8MAwG
A1UdEwEB/wQCMAAwHQYDVR0OBBYEFpZEcIQcD/Bj2IFz/LEruo2ADJviMIGMBgNV
HSMEGyQwgYGAFO0/7kEh3FuEKS+Q/kYHaD/W6wihoWakZDBiMQswCQYDVQQGEwJV
UzELMAkGA1UECBMCQ0ExFDASBgNVBACTC0xvcyBBbmdlbGVzMRRMwEQYDVQQKEwJQ
Q0FOTiBUTUNIMRswGQYDVQQDEwJJQ0FOTiBUTUNIIFRFU1QgQ0GCAQEWdG9YDVR0P
AQH/BAQDAgeAMC4GA1UdHwQnMCUwI6AhoB+GHWh0dHA6Ly9jcmwuaWNhbm4ub3Jn
L3RtY2guY3JsMA0GCSqGSIb3DQEBCwUAA4IBAQB2qSy7ui+43cebKUKWWPrrzz9y/
IkrMeJGKjo40n+9uekaw3DJ5EqiOf/qZ4pjBD++oR6BJCb6NQuQKwnoAz51E4Ssu
y5+i93oT3HfyVc4gNMIoHm1PS1917DBKrbwbzAea/0jKWVzrvM77TBfjxD3AQo1R
bU5dBr6IjbdLFlnO5x0G0mrG7x5OUPuurihyiURpFDpWH8KAH1wMcCpXGXFRtGKk
wydgyVYAty7otkl/z3bZkCVT34gPvF70sR6+QxUy8u0LzF5A/beYaZpxSYG31amL
AdXitTWFipaIGea9lEGFM0L9+Bg7XzNn4nVLXokyEB3bgS4scG6QznX23FGk
      </X509Certificate>
    </X509Data>
  </KeyInfo>
</Signature>
</verificationCode:signedCode>
```

2.1.1.2. Encoded Signed Code

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

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

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

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

```
<verificationCode:encodedSignedCode
  xmlns:verificationCode=
    "urn:ietf:params:xml:ns:verificationCode-1.0">
  <verificationCode:code>
ICAgICAgPHZlcmlmaWNhdGlvbkNvZGU6c2lnbmVkd29kZQogICAgICAgIHhtbG5z
OnZlcmlmaWNhdGlvbkNvZGU9CiAgICAgICAgICAidXJuOmllldGY6cGFyYW1zOnht
bDpuczp2ZXJpZmljYXRpb25Db2RlLTEuMCIKICAgICAgICAgIGlkPSJzaWduZWZl
b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMTIzPC92ZXJp
ZmljYXRpb25Db2RlOmNvZGU+CiAgPFNpZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3
dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CiAgIDxTaWduZWZlbnZvPogICAg
PENhbm9uaWNhbG16YXRpb25NZXR0b2QKIEFsZ29yaXR0bT0iaHR0cDovL3d3dy53
My5vcmcvMjAwMS8xMC94bWwtZXhjlWmxNG4jIi8+CiAgICA8U2lnbmF0dXJlTWV0
aG9kCiBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZHNp
Zy1tb3JlI3JzYS1zaGEyNTYiLz4KICAgIDxSZWZlcmluY2UgVVJPSIjc2lnbmVkd29k
ZSI+CiAgICAgPFYyZW5zZm9ybXNpZm9kZm9kZm9kZm9kZm9kZm9kZm9kZm9kZm9k
aXR0bT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3B1
ZC1zaWduYXRlcmUiLz4KICAgICA8L1RyYW5zZm9ybXNpZm9kZm9kZm9kZm9kZm9k
dGhvZAogQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVu
YyNzaGEyNTYiLz4KIDxEaWdlc3RlYXNpZm9kZm9kZm9kZm9kZm9kZm9kZm9kZm9k
UW5iZHRVnkFyTtdTaHJBZkhREZnPTwvRGlzXN0VmFsdWU+CiAgICA8L1JlZmVy
ZW5jZT4KICAgPC9TaWduZWZlbnZvPogICAgU2lnbmF0dXJlVmFsdWU+CiBqTXU0
UGZ5UUpSkJGMEU0VQRkNkaml5d0NFcVIyaDRMRCTnZTZyUStKbm1LRkZDdUNa
Uy8zU0xLQXgwTDF3CiBRREZPMmUwWTY5azJHNy9MR0UzN1gzdk9mbG9iRk0xb0d3
amE4K0dNvNjhb3RvNjBvZDQvQUY3ZUhl1a2dBeW1ECiBvOXRveG9hMmgweVY0QTRQ
bVh6c1U2Uzg2WHRDY1VFK1MvV003Mm55bjQ3em9VQ3p6UETiWkJSWVXZWhWR1Er
CiBqVWJNSUFNek01N0hIUUErNmVhWGVmUnZ0UEVUZ1VPNGFWSVZTdWdjNE9VQVpa
d2JZY1pyQzZ3T2FRcXFxQVppCiAzMGFQT0JZYkF2SE1TbVdTUytoRmtic2hvbUpm
```



```

SHhiOTdURDJncmxZTnJRSXpxWGs3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z
VfV3MUN6QTdqZndBPT0KICAgPC9TaWduYXR1cmVWYX1ZT4KICAgPetleUluZm8+
CiAgICA8WUWOUrhGE+CiAgICA8WUWOUNlcnRpZmljYXRlPgogTUlJRvNUQ0NB
ekdnQXdJQkFnSUJBakFOQmdrcWhraUc5dzBCQVFzRkFEQmlNUXN3Q1FZRFZRUUdF
d0pWVXpFTaogTUFrR0ExVUVDQk1DUTBFeEZEQVNCZ05WQkFjVEMweHZjeUJCYm1k
bGJHVnpNUk13RVFZRFZRUUtFd3BKUTBGTWogVG1CVVRVTklNUUN3R1FZRFZRUURF
eEpKUTBGT1RpQ1VUVU5JSUZSRlUxUWdRMEV3SGhjTk1UTXdNakE0TURBdwogTURB
dlldoY05NVGd3TWpBM01qTTFPVFU1V2pCc01Rc3dDUVlEVlFRR0V3SlZVekVMTUFr
R0ExVUVDQk1DUTBFeAogRkRBU0JnTlZCQWNUQzB4dmN5Qk1ibWRsYkdWek1SY3dG
UVlEVlFRS0V3NVdZV3hwWkdGMGIzSWdWRTFEU0RfAaogTUI4R0ExVUUVBeE1ZVmlG
c2FXUmhkRz15SUZSTlEwZ2dWRVZUVkNCRFJWSlVNSUlCSWpBTkJna3Foa2lHOXcw
QgogQVFRkFBT0NBUThtBTUlJQkNnS0NBUEVBYy9jd3ZYaGJWWWWwUkRXV3ZveWVw
cEVUVlplWVmnNQ292VVZ0Zy9zdWogV2ludUlnRVdnVlFGcnoweEEwNHBFaFhDRlZ2
NGV2Y1VwZWTkNWJ1cVUxZ21ReU9zQ0tRbGhPSFRkUGp2a0M1dQogcERxYTUxRmxr
MFRNYU1rSVFqcZdhVUtdbUE0Ukc0dFRUR0svRWpSMWl4OC9EMGdIWVZSbGR5M1VlQ
ck1QK291NwogNWJpVm5Jb3MrSGlmcKf0ck12NHFFcXdMTDRGVFpBVXBhQ2EyQm1n
WGZ5MkNTU1FieEQ1T3IxZ2NTYTN2dXJoNogc1BNQ054cWFYbUlYbVFpcFMrRHVF
QnFNTTh0bGRhtjdSWW9qVUVLckdWc05rNwK5eTivN3Nqbjf6eXlVUGY3dgogTDRH
Z0RZcWhKWVdWNjFieb1hneC9KZDZDV3h2c25ERjZzY3NjUxXpVVEVSK2h5d01EQVFB
Qm80SC9NSUG4TUF3RwogQTFVZEV3RUIvd1FDTUFBd0hRWURWUjBPQkJZRUZQWkvj
SVFjRC9CaJJRnovTEVSdW8yQURKdmlNSUdNQmdOVgogSFNNRWdZUXdnWUdBRk8w
LzdrRWgzRnVFS1MrUS9rWUhhRC9XNndpaG9XYWtareJpTVFzd0NRWURWUWVHRXdk
VgogV6XpFTE1Ba0dBmVVFQ0JNQ1EwRXhGREFTQmdOVkJBY1RDMHh2Y3lCQmJtZGxi
RlZ6TVJNd0VRWURWUWVFLRXdwSgogUTBGT1RpQ1VUVU5JTVJZd0dRWURWUWVFERXhK
SlEwRk9UaUJVVVFVOSU1GUkZVMVFfUTBHQ0FRXDEZ11EVlIwUAogQVFILOJBUURB
Z2VBTUM0R0ExVVRId1FuTUNvd0k2QWhvQitHSFdoMGRIQTZMeTlqY213dWFXThhi
bTRlYjNkbgogTDNSdFkyZ3VZM0pzTUEwR0NTcUdTSWIzRFFFQkN3VUFBNElCQVFC
MnFTeTdlaSs0M2N1YktVS3dXUHJ6ejl5LwogSWtyTWVKR0tqbzQwbis5dWVrYXcz
REolRXFpT2YvcVo0cGpCRCSrb1I2QkpDYjZOUXVRS3dub0F6NWwFNFNzdQogeTUR
aTkzb1QzSGZ5VmM0Z05NSW9IbTFQUZe5bDdEQktyYndiekFlYS8waktXVnpydm1W
N1RCZmp4RDNBWU8xUgogY1U1ZEJyNk1qYmRMRmxuTzV4MEcwbXJHN3g1T1VQdXVy
aWh5aVVSceZEchdIOEtBSDF3TWNDcFhHWEZSdEdLawogd3lkZ3lWWUF0eTdvdGts
L3ozYlprQ1ZUMzRnUHZGNzBzUjYrUXhVeThlMEx6RjVBL2JlWWFachHtWUczMWFt
TAogQWRYaXRUV0ZpcGFJR2VhOWxFR0ZNMEw5K0JnN1h6Tm40blZMWG9reUVCm2Jn
UzRzY0c2UXpuWDIzRkdrCiAgIDwvWDUwOUNlcnRpZmljYXRlPgogICA8L1g1MD1E
YXRhPgogICA8L0tleUluZm8+CiAgPC9TaWduYXR1cmU+CgkJPjZlZm8+CiAgPC9TaWduYXR1cmU+CgkJPjZlZm8+
b25Db2RlOnNpZ25lZENvZGU+Cg==
</verificationCode:code>
</verificationCode:encodedSignedCode>

```

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

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <domain:create>

```

```

    xmlns:domain="urn:iETF:params:xmL:ns:domain-1.0">
      <domain:name>domain.example</domain:name>
      <domain:registrant>jd1234</domain:registrant>
      <domain:contact type="admin">sh8013</domain:contact>
      <domain:contact type="tech">sh8013</domain:contact>
      <domain:authInfo>
        <domain:pw>2fooBAR</domain:pw>
      </domain:authInfo>
    </domain:create>
  </create>
  <extension>
    <verificationCode:encodedSignedCode
      xmlns:verificationCode=
        "urn:iETF:params:xmL:ns:verificationCode-1.0">
        <verificationCode:code>
ICAgICAgPHZlcmImaWNhdGlvbkNvZGU6c2lnbmVkQ29kZQogICAgICAgIHhtbG5z
OnZlcmImaWNhdGlvbkNvZGU9CiAgICAgICAgICAidXJuOmlldGY6cGFyYWlwZOnht
bDpuczp2ZXJpZm1jYXRpb25Db2RlLTUeMCIKICAgICAgICAgIGlkPSJzaWduZWRR
b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTptjb2RlPjEtYWJjMTIzPC92ZXJp
Zm1jYXRpb25Db2RlOmNvZGU+CiAgPFNP2Z5hdHVyZSB4bWxucz0iaHR0cDovL3d3
dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CiaGIDXTaWduZWRRJbmZvPgogICAg
PENhbm9uaWNhbGl6YXRpb25NZXRob2QKIEFsZ29yaXRobT0iaHR0cDovL3d3dy53
My5vcmcvMjAwMS8xM94bWwtZXhjLWMxNG4jIi8+CiaGICA8U2lnbmF0dXJlTWV0
aG9kCiBBBgdvcm10aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDExMDQveGlsZHNp
Zy1tb3JlI3JzYS1zaGEyNTYiLz4KICAgIDxsZWZlcmVuY2UgVVJJPSIjc2lnbmVk
Q29kZSI+CiaGICAgPFfryYW5zM9ybXM+CiaGICAgIDxuCMFuc2ZvcmlkeF5sZ29y
aXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3Bl
ZC1zaWduYXRlcmlz4KICAgICA8L1RyYW5zM9ybXM+CiaGICAgPERpZ2VzdE1l
dGhvZAogQWxnbn3JpdGhtPSJodHRwOi8vd3d3LnclLm9yZy8yMDAxLzA0L3htbGVu
YyNzaGEyNTYiLz4KIDxEaWdlc3RWYXwx1ZT53Z3lXM25aUG9FZnBwdGxoUklMS25P
UW5iZHRVNkFyTTdTahJBZkhREZnPTwvRGlnZXN0VmFsZWU+CiaGICA8L1JlZmVy
ZW5jZT4KICAgPC9TaWduZWRRJbmZvPgogICA8U2lnbmF0dXJlVmFsZWU+CibqTXU0
UGZ5UUdpSkJGMEDXU0VQRkNKam15d0NFcVIyaDRMRCTnZTZYUSTKbm1LRkZDdUNA
Uy8zU0xLQXgwTDF3CiBRREZPMWWTY5azJHNY9MR0Uzn1gzdk9mbG9irk0xb0d3
amE4K0dNVnJhb3RvNXhBZDQvQUY3ZUhla2dBewlECiBvOXRveG9hMmgweVY0QTRQ
bVh6c1U2Uzg2WHRDY1VFk1MvV003Mm55bjQ3em9VQ3p6UEtIWkJSeWVXZWWhWRlEr
CiBqVWJSUFNEk01N0hiIUUERnmVhWGVMUnZ0UEVUZ1VPNGFWSVZTdWdjNE9VQVpa
d2JZY1pyQzZ3T2FRcXFxQVppCiAzMGFQT0JZYkF2SE1TbvDUytoRmtic2hvbUpm
SHhiOTdURDJncmxZTnJRSPxwGS3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z
VFV3MUN6QTDqZndBPT0KICAgPC9TaWduYXRlcmlzVWYXwx1ZT4KICAgPEtleUluZm8+
CiAgICA8WDUwOURhdGE+CiaGICA8WDUwOUNlcnRpZm1jYXRlPgogTU1JRvNUQ0NB
ekdnQXdJQkFnSUJBakFOQmdrcWhraUc5dzBCQVZzRkFEQmLNUXN3Q1FZRFRZRUUDf
d0pWVXpFTaogTUFrrR0EXVUVDQk1DUTBFEEZEQVNCZ05WQkfjVEMweHZjeUJCym1k
bGJHVnpNUk13RVFRZFZRuUtdf3BKUTBGTwogVG1CVVRVTklNUUnN3R1FZRFRZRUURF
eEpKUTBGTrlpQ1VUVU5JSUSZR1UxUWdRMev3SGhjTk1UTXDNaKe0TURBdwogTURB
dlldoY05NVGD3TWpBM01qtTFPVFU1Z2pCc01Rc3dUdEV1FRR0V3SlZVekVMTUfr
R0EXVUVDQk1DUTBFEEQARkRBu0JnTlZCQNWUzB4dmN5QkjibWRSYkdWek1SY3dG
UV1EV1FRS0V3NVdZV3hwWkdGMGIzSWdWRTFEU0RFaAoqTUI4R0EXVUVBeElZVm1G

```

c2FXUmhkRz15SUZSt1EwZ2dWRVZUVkNCRFJWS1VNSU1CSWpBTkJna3Foa21lHOXcw
QgogQVFRkFBT0NBUTHBTU1JQkNnS0NBuUVBby9jd3ZYaGJWWwwUkRXV3ZveWVa
cEVUv1pWmNNQ292VvZOzy9zdwogV21udU1nRVdnVlFGcnoweEEwNHBFaFhDR1Z2
NGV2Y1VwZWtKNWJ1cVUxZ21ReU9zQ0tRbGhPSFRkUGp2a0M1dQogcERxYTUxRmxr
MFRNYU1rSVFqczdhVUTDbUE0Ukc0dFRUR0svRWpSMW14OC9EMGdIWVZSbGR5MV10
ck1QK291NwogNWJPVm5Jb3MrSG1mckF0ck12NHFFcXdmTRDGVFpBVXBhQ2EYQm1n
Wgz5MkNtU1FieEQ1T31xZ2NTY2nd2xJoNqogc1BNQ054cWFYbU1YbVfpcFMrRHVF
QnFNTHt0bGrHTjdsSWW9qVUVLckdWc05rNwK5eTivN3Nqbjf6eX1VUGY3dgogTDRH
Z0RZcWhKWdWNjFEb1hneC9KZDZDV3h2c25ERjZzY3NjUxpvVEVsK2h5d01EQVFB
Qm80SC9NSUg4TUF3RwogQTFVZE3V3RUIvd1FDTUFBd0hRWURWUjBPQkJZRUZQWkVj
SVfjRC9CaJJJRnovTEVSDw8yQURKdmlNSUdNqmdOVgogSFNNRWdZUXdnWUDBrk8w
LzdrRWgzRnVFS1MrUS9rWUhhRC9XNndpaG9XYWtaREJpTVFzd0NRWURWUVFHRXdk
VgogVXpFTE1Ba0dBMVVFQ0JNQ1EwRXhGREFTQmdOVkjbY1RDMHh2Y3lCQmJtZGxi
R1Z6TVJNd0VRWURWUVFLRXdwSgogUTBGT1RpQ1VUVU5JTVJZzd0dRWURWUVFERXhk
S1EwRk9UaUJVVVFVOSU1GukZVMVfnUTBHQ0FRRXdEZ11EV1IwUAogQVF1L0JBURB
Z2VBUTM0R0ExVVRId1FuTUNVd0kTQWvhvQitHSFdomGRIQTZMETlqY213dWFXtMhi
bTR1YjNkbgogTDNSdFkyZ3VZM0pzTUeWR0NTcUdTSWIzRFFFQkN3VUFBNE1CQVFC
MnFTeTd1aSs0M2N1YktVS3dXUHJ6ej15LwogSWtyTWVKR0tqbzQwbis5dWVrYXcz
REolRXFpT2YvcVo0cGpCRCSrb1I2QkpDYjZOUXVRS3dub0F6NWxfNFnzdQogeTUR
aTkzb1QzSGZ5VmM0Z05NSW9IbTFQUZE5bDdEQktyYndiekFlYS8waktXVnpydm1W
N1RCZmp4RDNBUW8xUgogYlU1ZEJyNklqYmRMRmxuTzV4MEcwbXJHN3g1T1VQdXVy
aWh5aVVSCEZEchdIOEtBSDF3TWNDcFhHWEZSDedLawogd3lkZ3lWWUF0eTdvdGts
L3ozYlprQ1ZUMzRnUHZGNzBzUjYrUXhVeTh1MEx6RjVBL2JlWWFfacHhTWUczMWFt
TAogQWRyYXRUV0ZpcGFJR2VhOWFRX0ZNMWE5K0JnN1h6Tm40b1ZMWG9reUVCm2Jn
UzRzY0c2UXpuWDIzRkdrCiAgIdwvWDUwOUNLcnRpZmljYXRlPogogICAgL1glMD1E
YXRhPgogICAgL0tleUluZm8-CiAgPC9TaWduYXR1cmU-CgkJPc92ZXJpZmljYXRp
b25Db2RlOnNpZ25lZENvZGU-Cg==

</verificationCode:code>

<verificationCode:code>

Pd94bWwgdmVyc2l2bj0iMS4wIiBlbnNvZGluz0iVVRGLTgiPz48dmVyaWZpY2F0aW9uQ29kZTpzaWduZWRRDb2RlIHhtbG5zOnZlcmlmaWNhdGlvbknvZGU9InVybjppZXRMOnBhcmFtczcp4bW6bnM6dmVyaWZpY2F0aW9uQ29kZS0xLjAiIGlkPSJzaWduZWRRDb2RlIiB0eXB1PSJyZWdp3c3RyYW50Ij48dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMjIyPC92ZXJpZmlljYXRpb25Db2RlOmNvZGU+PGRzaWc6U2lnbmF0dXJlIHhtbG5zOmRzaWc9Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyMiPjxkc2lnOlNpZ25leEluZm8+PGRzaWc6Q2Fub25pY2FsaXphdGlvbklldGhvZCBbbGdvcmllOAG09Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyNyNjc2Etc2hhMSIvPjxkc2lnOlJlZmVyZW5jZSBVUkk9IiInZaWduZWRRDb2RlIj48ZHNpZzpUcmFuc2ZvcmlzPjxkc2lnOlRyYW5zM9ybSBbbGdvcmllOAG09Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyNlb3Zlbg9wZWQt c2lnbmF0dXJlIi8+PC9kc2lnOlRyYW5zM9ybXM+PGRzaWc6RGlnZXNO TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjIyPC92ZXJpZmlljYXRpb25Db2RlOmNvZGU+PGRzaWc6RGlnZXNOVmFsdWU+SfG2TUlWUWdnSSZtNG9tT3haYjBGTVlVS1BRdk15WmUYbDVEdEhh Q1ZMNND08LRzaWc6RGlnZXNOVmFsdWU+PC9kc2lnOlJlZmVyZW5jZTp48L2RzeA Wc6U2lnbmVksW5mbz48ZHNpZzpTaWduYXRlc3RlcnVWYXwx1ZT5VOUhPNVlVYWE0ZUsyYXRz U1RuQk1DU3dXM0dWUZmVyc2l2bj0iMS4wIiBlbnNvZGluz0iVVRGLTgiPz48dmVyaWZpY2F0aW9uQ29kZTpzaWduZWRRDb2RlIHhtbG5zOnZlcmlmaWNhdGlvbknvZGU9InVybjppZXRMOnBhcmFtczcp4bW6bnM6dmVyaWZpY2F0aW9uQ29kZS0xLjAiIGlkPSJzaWduZWRRDb2RlIiB0eXB1PSJyZWdp3c3RyYW50Ij48dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMjIyPC92ZXJpZmlljYXRpb25Db2RlOmNvZGU+PGRzaWc6U2lnbmF0dXJlIHhtbG5zOmRzaWc9Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyMiPjxkc2lnOlNpZ25leEluZm8+PGRzaWc6Q2Fub25pY2FsaXphdGlvbklldGhvZCBbbGdvcmllOAG09Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyNyNjc2Etc2hhMSIvPjxkc2lnOlJlZmVyZW5jZSBVUkk9IiInZaWduZWRRDb2RlIj48ZHNpZzpUcmFuc2ZvcmlzPjxkc2lnOlRyYW5zM9ybSBbbGdvcmllOAG09Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyNlb3Zlbg9wZWQt c2lnbmF0dXJlIi8+PC9kc2lnOlRyYW5zM9ybXM+PGRzaWc6RGlnZXNO TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjIyPC92ZXJpZmlljYXRpb25Db2RlOmNvZGU+PGRzaWc6RGlnZXNOVmFsdWU+SfG2TUlWUWdnSSZtNG9tT3haYjBGTVlVS1BRdk15WmUYbDVEdEhh Q1ZMNND08LRzaWc6RGlnZXNOVmFsdWU+PC9kc2lnOlJlZmVyZW5jZTp48L2RzeA Wc6U2lnbmVksW5mbz48ZHNpZzpTaWduYXRlc3RlcnVWYXwx1ZT5VOUhPNVlVYWE0ZUsyYXRz U1RuQk1DU3dXM0dWUZmVyc2l2bj0iMS4wIiBlbnNvZGluz0iVVRGLTgiPz48dmVyaWZpY2F0aW9uQ29kZTpzaWduZWRRDb2RlIHhtbG5zOnZlcmlmaWNhdGlvbknvZGU9InVybjppZXRMOnBhcmFtczcp4bW6bnM6dmVyaWZpY2F0aW9uQ29kZS0xLjAiIGlkPSJzaWduZWRRDb2RlIiB0eXB1PSJyZWdp3c3RyYW50Ij48dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMjIyPC92ZXJpZmlljYXRpb25Db2RlOmNvZGU+PGRzaWc6U2lnbmF0dXJlIHhtbG5zOmRzaWc9Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyMiPjxkc2lnOlNpZ25leEluZm8+PGRzaWc6Q2Fub25pY2FsaXphdGlvbklldGhvZCBbbGdvcmllOAG09Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyNyNjc2Etc2hhMSIvPjxkc2lnOlJlZmVyZW5jZSBVUkk9IiInZaWduZWRRDb2RlIj48ZHNpZzpUcmFuc2ZvcmlzPjxkc2lnOlRyYW5zM9ybSBbbGdvcmllOAG09Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyNlb3Zlbg9wZWQt c2lnbmF0dXJlIi8+PC9kc2lnOlRyYW5zM9ybXM+PGRzaWc6RGlnZXNO TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjIyPC92ZXJpZmlljYXRpb25Db2RlOmNvZGU+PGRzaWc6RGlnZXNOVmFsdWU+SfG2TUlWUWdnSSZtNG9tT3haYjBGTVlVS1BRdk15WmUYbDVEdEhh Q1ZMNND08LRzaWc6RGlnZXNOVmFsdWU+PC9kc2lnOlJlZmVyZW5jZTp48L2RzeA Wc6U2lnbmVksW5mbz48ZHNpZzpTaWduYXRlc3RlcnVWYXwx1ZT5VOUhPNVlVYWE0ZUsyYXRz U1RuQk1DU3dXM0dWUZmVyc2l2bj0iMS4wIiBlbnNvZGluz0iVVRGLTgiPz48dmVyaWZpY2F0aW9uQ29kZTpzaWduZWRRDb2RlIHhtbG5zOnZlcmlmaWNhdGlvbknvZGU9InVybjppZXRMOnBhcmFtczcp4bW6bnM6dmVyaWZpY2F0aW9uQ29kZS0xLjAiIGlkPSJzaWduZWRRDb2RlIiB0eXB1PSJyZWdp3c3RyYW50Ij48dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMjIyPC92ZXJpZmlljYXRpb25Db2RlOmNvZGU+PGRzaWc6U2lnbmF0dXJlIHhtbG5zOmRzaWc9Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyMiPjxkc2lnOlNpZ25leEluZm8+PGRzaWc6Q2Fub25pY2FsaXphdGlvbklldGhvZCBbbGdvcmllOAG09Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyNyNjc2Etc2hhMSIvPjxkc2lnOlJlZmVyZW5jZSBVUkk9IiInZaWduZWRRDb2RlIj48ZHNpZzpUcmFuc2ZvcmlzPjxkc2lnOlRyYW5zM9ybSBbbGdvcmllOAG09Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyNlb3Zlbg9wZWQt c2lnbmF0dXJlIi8+PC9kc2lnOlRyYW5zM9ybXM+PGRzaWc6RGlnZXNO TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjIyPC92ZXJpZmlljYXRpb25Db2RlOmNvZGU+PGRzaWc6RGlnZXNOVmFsdWU+SfG2TUlWUWdnSSZtNG9tT3haYjBGTVlVS1BRdk15WmUYbDVEdEhh Q1ZMNND08LRzaWc6RGlnZXNOVmFsdWU+PC9kc2lnOlJlZmVyZW5jZTp48L2RzeA Wc6U2lnbmVksW5mbz48ZHNpZzpTaWduYXRlc3RlcnVWYXwx1ZT5VOUhPNVlVYWE0ZUsyYXRz U1RuQk1DU3dXM0dWUZmVyc2l2bj0iMS4wIiBlbnNvZGluz0iVVRGLTgiPz48dmVyaWZpY2F0aW9uQ29kZTpzaWduZWRRDb2RlIHhtbG5zOnZlcmlmaWNhdGlvbknvZGU9InVybjppZXRMOnBhcmFtczcp4bW6bnM6dmVyaWZpY2F0aW9uQ29kZS0xLjAiIGlkPSJzaWduZWRRDb2RlIiB0eXB1PSJyZWdp3c3RyYW50Ij48dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMjIyPC92ZXJpZmlljYXRpb25Db2RlOmNvZGU+PGRzaWc6U2lnbmF0dXJlIHhtbG5zOmRzaWc9Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyMiPjxkc2lnOlNpZ25leEluZm8+PGRzaWc6Q2Fub25pY2FsaXphdGlvbklldGhvZCBbbGdvcmllOAG09Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyNyNjc2Etc2hhMSIvPjxkc2lnOlJlZmVyZW5jZSBVUkk9IiInZaWduZWRRDb2RlIj48ZHNpZzpUcmFuc2ZvcmlzPjxkc2lnOlRyYW5zM9ybSBbbGdvcmllOAG09Imh0dHA6Ly93d3cudzMub3JnLzlwMDAvMDkveGlsZHNPZyNlb3Zlbg9wZWQt c2lnbmF0dXJlIi8+PC9kc2lnOlRyYW5zM9ybXM+PGRzaWc6RGlnZXNO TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjIyPC92ZXJpZmlljYXRpb25Db2RlOmNvZGU+PGRzaWc6RGlnZXNOVmFsdWU+SfG2TUlWUWdnSSZtNG9tT3haYjBGTVlVS1BRdk15WmUYbDVEdEhh Q1ZMNND08LRzaWc6RGlnZXNOVmFsdWU+PC9kc2lnOlJlZmVyZW5jZTp48L2RzeA Wc6U2lnbmVksW5mbz48ZHNpZzpTaWduYXRlc3RlcnVWYXwx1ZT5VOUhPNVlVYWE0ZUsyYXRz U1RuQk1DU3dXM0dWUZmVyc2l2bj0iMS4wIiBlbnNvZGluz0iVVRGLTgiPz48dmVyaWZpY2F0aW9uQ29kZTpzaWduZWRRDb2RlIHhtbG5zOnZlcmlmaWNhdGlvbknvZGU9InVybjppZXRMOnBhcmFtczcp4bW6bnM6dmVyaWZpY2F0aW9uQ29kZS0xLjAiIGlkPSJzaWduZWRRDb2RlIiB0eXB1PSJyZWdp3c3RyYW50Ij48dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMjIyPC92ZXJpZmlljYXRpb25Db2RlOmNvZGU+PGRzaWc6U2lnbmF0dXJ

WXB3RkROMmZLY3JVCk1YV0hncE56K0oycTh6MWpTcVJMUEw0UmpnRWw0eGhiOXl5
 cExOZC8xQXJXRv1hWWZEdUc1S3FYV05MRG5YVzJoQkEzK0R5Wk82MFQKcTVPd0R5
 ZVFSVlNPVWNXVE9FOTJsSlZ4M014Q1V6d1hoL0ZOSTlPbGtXK0ZPNVZNNTZlTmZq
 UEhkU1JVdjdZQzRmM0NnWmFaSWFXNQp2RmJnTmJodFJVa0hsSVhnYVNGWDgvcFdV
 RXFIY0dLTUxnRU1nbHBnQ3RtOf1IcXVqb0tXUk0yUDNiK2h3ZTRsU0hSWVRjK0pB
 eEluC1U4RDc1WnliWThnSWFuZUprS2dwVTk2T0tJTGQ5L0l0UVhaeHZnPT08L2Rz
 aWc6U2lnbmF0dXJlVmFsdWU+PGRzaWc6S2V5SW5mbz48ZHNPZzpYNTA5RGF0YT48
 ZHNpZzpYNTA5Q2VydGlmaWNhdGU+TU1JRGlUQ0NBbkdnQXdJQkFnSUVmcXE2SFRB
 TkJna3Foa2lHOXcwQkFRc0ZBREIXTVJBd0RnWURWUVFHRXdkVmJtdHVIM2R1TVJB
 dwpEZ1lEVlFRSUV3ZFZibXR1YjNkdU1SQXdEZ1lEVlFRSEV3ZFZibXR1YjNkdU1S
 QXdEZ1lEVlFRS0V3ZFZibXR1YjNkdU1SQXdEZ1lEC1ZRUUxXd2RWYm10dWIzZHVN
 Umt3RndZRFZRUURFeEIyWlhKcFptbGpZWfJwYjI1RG1yUmxNQjRFRFRFMU1EWXhO
 VE14TURBeU1sb1gKRFRNMU1EWXhNREl4TURBeU1sb3dkVEVRTUE0R0ExVUVCaE1I
 VlclcmJtOTNiakVRTUE0R0ExVUVDQk1IVlclcmJtOTNiakVRTUE0RwpBMVVFQnhN
 SFZXNXJibTkzYmpFUU1BNEdBMVVFQ2hNSFZXNXJibTkzYmpFUU1BNEdBMVVFQ3hN
 SFZXNXJibTkzYmpFWk1CY0dBMVVFckF4TVFkbVZ5YVdacFkyRjBhVz11UTI5a1pU
 Q0NBU013RFFZSktvWklodmNOQVFFQkJRQURnZ0VQQURDQ0FRb0NnZ0VCQUpjY2pY
 cmsKUWFJL2lHUEZ3WmVITjFnRFVhcTltVnJmQis2eWR5Qmdoc2FHVfZoaERIOFNO
 TmtpamxIMkxQC3J3TjhjVjhQZ1BPOXRwbG9rR2F5UwpXNktFaHZtTk03b1dsZk5L
 SkdSdGNidGMzTnJuYzhiUUJacU1xcFo0U1NRTmh5QWh6Ri85UmErd3Rfc0JWeGF3
 VDC1L2J0SDZ1Yytmc1J0de5FcmhJdVlJUUmN0WTZIRmRaR3B1S3cxYn1YK0RsNkJP
 L3ZLdnQ4ND1lY1R3aEZIcDUwWGH2NFVTL0Z5aWVLaGs3dDdHRnJGRlQKL2NCTGsy
 WmxFallLcFlEU2dlc2lseFg2QkptZVdCbXZLQz1TL2pBZDhNWmRHVUg2aHNHRXB1
 U1BmZkZQV3FWcXl6V0p5bG91OXF4ZQpnUTZjOFo2SVpXZkUzakxSOUVySDhzOTFD
 MmlpTFZrQ0F3RUFBYU1oTUI4d0hRWURWUjBPQkJZRUZiY0JLdk03dmk3dUZNTUx5
 ZE43CmVGXVF2YzVVTUEwR0NTcUdTSWIZRFFFQkN3VUFBNELCQVFVBVjB2cm1rSWRB
 d2l4THZ0NUx5eXpTNFdTU1d0dVlWL2JQMVg3NzVMRmYKSWH3a2xoMENidk5rYXlK
 Tms2Tnp0eDlSc1AwNWZndkxrZER1N0V5cnRzY3I1ZVdETG1WMGtKMWE1N1Z4bnJh
 aEdLTnM2Wit1Ui9pSAPMatJXb3liWEpFT2N0NWtJSjFzL05CeUURdkdGdjFoTmJz
 dVVVUEVCYwVtaWpYUFR0OWxxZE9uM1FIbktobXhsalczYS9KbmhtT20vCkrWYTE0
 NDJXTVVUS1UyVf1WVldtdUs2NFkwQXFfRn2F1dzkvVzIzZEcrT2xhOW9VYnBrSXJr
 dDRDN3hRa0d5SXN2eUo3bi9lOFhBRDIKbno1T1cvek5GWnlrZDAzT2N3M240NkZx
 c1IwVD1BbFBEBWHQxUjlmMjZMd11xdjk3dWtVNEcrMVRJNHOrV0F2TctVRk9FVnNu
 PC9kc2lnOlglMD1DZXJ0aWZpY2F0ZT48L2RzaWc6WDUwOURhdGE+PC9kc2lnOktl
 eUluZm8+PC9kc2lnOlNpZ25hdHVyZT48L3Zlcm1maWNhdGlvbknVZGU6c2lnbmVk
 Q29kZT4=

```

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

```

2.2. Verification Profile

A Verification Profile defines the set of verification code types, the commands that the verification code types are required, supported, or not supported, and the grace period by which the

verification code types MUST be set. It is up to server policy what action to take if the verification code type is not set by the grace period. A server MAY support many verification profiles, each with a unique name and a unique verification policy that is implemented by the server. Each client MAY have zero or more server assigned verification profiles that will enforce the required verification policies. Most likely a client will be assigned zero or one server assigned verification profile, but overlapping profiles is possible. Overlapping verification profiles MUST be treated as a logical "and" of the policies by the server. If no verification profile is assigned to the client, no additional verification is required by the client.

3. EPP Command Mapping

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

3.1. EPP Query Commands

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

3.1.1. EPP <check> Command

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

3.1.2. EPP <info> Command

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

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

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

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

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <domain:info
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>domain.example</domain:name>
C:          <domain:authInfo>
C:            <domain:pw>2fooBAR</domain:pw>
C:          </domain:authInfo>
C:        </domain:info>
C:      </info>
C:    <extension>
C:      <verificationCode:info
C:        xmlns:verificationCode=
C:          "urn:ietf:params:xml:ns:verificationCode-1.0"/>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

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

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

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

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

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

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

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

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

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

compliant The object is compliant with the verification profile.

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

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

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

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

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

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:          "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>compliant
S:      </verificationCode:status>
S:      <verificationCode:profile name="sample">
S:        <verificationCode:status>compliant
S:      </verificationCode:status>
S:      <verificationCode:set>
S:        <verificationCode:code type="domain"
```

```

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

```

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

```

S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">

```

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

Example <info> domain response using the <verificationCode:infData> extension for a compliant domain using the "sample" profile, and with the two verification code types, from the non-sponsoring client:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>compliant
S:        </verificationCode:status>
S:        <verificationCode:profile name="sample">
S:          <verificationCode:status>compliant
S:          </verificationCode:status>
S:          <verificationCode:set>
S:            <verificationCode:code type="domain"
S:              date="2010-04-03T22:00:00.0Z"/>
S:            <verificationCode:code type="registrant"
S:              date="2010-04-03T22:00:00.0Z"/>
S:          </verificationCode:set>
S:        </verificationCode:profile>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="serverHold"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:          "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>nonCompliant
S:        </verificationCode:status>
S:        <verificationCode:profile name="sample">
S:          <verificationCode:status>nonCompliant
S:          </verificationCode:status>
S:          <verificationCode:missing>
S:            <verificationCode:code
S:              type="domain"
S:              due="2010-04-03T22:00:00.0Z"/>
S:            <verificationCode:code
S:              type="registrant"
S:              due="2010-04-08T22:00:00.0Z"/>
S:          </verificationCode:missing>
S:        </verificationCode:profile>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

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

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>pendingCompliance
S:        </verificationCode:status>
S:        <verificationCode:profile name="sample">
S:          <verificationCode:status>pendingCompliance
S:          </verificationCode:status>
S:          <verificationCode:missing>
S:            <verificationCode:code
S:              type="registrant"
S:              due="2010-04-08T22:00:00.0Z"/>
S:          </verificationCode:missing>
S:          <verificationCode:set>
S:            <verificationCode:code type="domain"
S:              date="2010-04-03T22:00:00.0Z">1-abc333
S:            </verificationCode:code>
S:          </verificationCode:set>
S:        </verificationCode:profile>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```


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

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>notApplicable
S:        </verificationCode:status>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.1.3. EPP <transfer> Command

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

3.2. EPP Transform Commands

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

3.2.1. EPP <create> Command

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

The EPP <create> command provides a transform operation that allows a client to create an object. In addition to the EPP command elements described in an object mapping like [RFC5731], the command MAY contain a child <verificationCode:encodedSignedCode> element, as defined in Section 2.1.2, that identifies the extension namespace for the client to provide proof of verification by a Verification Service Provider (VSP). The server MAY support multiple policies for the passing of the <verificationCode:encodedSignedCode> element based on the client profile, which include:

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

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
```

```
C:      <create>
C:      <domain:create
C:          xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:              <domain:name>domain.example</domain:name>
C:              <domain:registrant>jdl234</domain:registrant>
C:              <domain:contact type="admin">sh8013</domain:contact>
C:              <domain:contact type="tech">sh8013</domain:contact>
C:              <domain:authInfo>
C:                  <domain:pw>2fooBAR</domain:pw>
C:              </domain:authInfo>
C:          </domain:create>
C:      </create>
C:      <extension>
C:          <verificationCode:encodedSignedCode
C:              xmlns:verificationCode=
C:                  "urn:ietf:params:xml:ns:verificationCode-1.0">
C:              <verificationCode:code>
C:ICAgICAgPHZlcmhmaWNhdGlvbkNvZGU6c2lnbmVkQ29kZQogICAgICAgIHhtbG5z
C:OnZlcmlmaWNhdGlvbkNvZGU9CiAgICAgICAgICAidXJuOmlldGY6cGFyYWlwOnht
C:bDpuczp2ZXJpZmlljYXRpb25Db2RlLTEuMCIKICAgICAgICAgIGlkPSJzaWduZWRR
C:b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJMTiZPC92ZXJp
C:dyljYXRpb25Db2RlOmNvZGU+CiAgPFNPZ25hdHVyZSB4bWxuc20iaHR0cDovL3d3
C:dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CjAgIDxTaWduZWRRJbmZvPgogICAg
C:PENhbm9uaWNhbGl6YXRpb25NZXRob2QKIEFsZ29yaXRobT0iaHR0cDovL3d3dy53
C:My5vcmcvMjAwMS8xMC94bWwtZXhjLWMxNG4jIi8+CjAgICA8U2lnbmF0dXJlTWV0
C:aG9kCiBBBgdvcmll0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDExMDQveGlsZHNP
C:Zy1tb3JlIj4KICAgIDxSZWZlcmVuY2UgVVJJPSIjc2lnbmVk
C:Q29kZSI+CjAgICAgPFYyZW5zZm9ybXM+CjAgICAgIDxUcmFuc2ZvcmlkeiE29y
C:aXR0bT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3B1
C:ZC1zaWduYXR1cmUiLz4KICAgICA8L1RyYW5zZm9ybXM+CjAgICAgPERpZ2VzdE1l
C:dGhvZAogQWxnbn3JpdGhtPSJodHRwOi8vd3d3LnZlcnZy8yMDAxLzA0L3htbGVu
C:YyNzaGEyNTYiLz4KIDxEaWdlc3R5YWxlZT53Z3lXM25aUG9FU2lnbmF0dXJlTWV0
C:UW5iZHRVNFYtTDhJb2ZkhbZm9vZGU9CiAgICA8L1JlZmV0
C:ZW5jZT4KICAgPC9TaWduZWRRJbmZvPgogICAgU2lnbmF0dXJlVmFsdWU+CjAgTCU0
C:UGZ5UUpdSkJGMEdXU0VQRkNKAm15d0NFcVIyaDRMRCTnZTZyUSTkbm1LRkZDdUNa
C:Uy8zU0xLQXgwTDF3CiBRREZPMmUwWTY5azJHNy9MR0UzNlgzdG9mbG9iRk0xb0d3
C:amE4K0dNVnJhb3RvNXhBZDQvQUY3ZUhla2dBewlECiBvOXRveG9hMmgweVY0QTRQ
C:bVh6clU2Uzg2WHRDYlVFk1MvV003Mm55bjQ3em9VQ3p6UEtIWkJSeWVXZWWhRlEr
C:CiBqVWJSUFNEk01N0hIUUERnmVhWGVMUnZ0UEVUZ1VPNGFWSVZTdWdjNE9VQVpa
C:d2JZYlpyQzZ3T2FRcXFxQVppCiAzMGFQT0JZYkF2SE1TbVdTUytoRmtic2hvbUpm
C:SHhiOTdURDJncmxZTnJRSPxWGS3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z
C:VFV3MUN6QTdqZndBPT0KICAgPC9TaWduYXR1cmVWYXxlZT4KICAgPETleUluZm8+
C:CiAgICA8WDUwOURhdGE+CjAgICA8WDUwOUNlcnRpZmlljYXRlPgogTUljRVNUQ0NB
C:ekdnQXJkQkFnSUJBakFOQmdrcWhtaUc5dzBCQVfZrRkFEQmlNUNXN3Q1FZRFRZRUUDf
C:d0pWVXpFTaogTURFrR0EXVUVDQk1DUTBFEEZEQVNCZ05WQkfjVEMweHZeUJCym1k
C:bGJHVnpNuk13RVfZRFZRUUtd3BKUTBGTwogVG1CVVRVTklNUnN3R1FZRFRZRUURF
C:eEJPHTBTGT1RqpLVUVU5JSUZSR1UwUWRMEV3SGhjTk1UTXDNake0TURBDwogTURB
C:d1doY05NVGD3TWpBM01qTTFFPVFU1V2pCc01Rc3dDUVL3d3dy53My5vcmcvMjAw
C:MS8xMC94bWwtZXhjLWMxNG4jIi8+CjAgICA8U2lnbmF0dXJlTWV0
```

```

C:R0ExVUVDQk1DUTBFaAogRkRBU0JnTlZCQWNUQzB4dmN5QkJibWRsYkdWek1SY3dG
C:UV1EV1FRS0V3NVdZV3hwWkdGMGIzSWdWRTFEU0RFaAogTUI4R0ExVUVBeE1ZVm1G
C:c2FXUmhkRz15SUZST1EwZ2dWRVZUVkNCRFJWS1VNSU1CSWpBTkJna3Foa2lHOXcw
C:QgogQVFFRkFBT0NBUThtBTU1JQkNnS0NBuUVBby9jd3ZYaGJWWwwUkRXV3ZveWVa
C:cEVUVlpWVmNNQ292VVZOZy9zdWogV2ludU1nRVdnVlFGcnoweEEwNHBFAfhDRlZ2
C:NGV2Y1VwZWtKNWJ1cVUxZ21ReU9zQ0tRbGhPSFRkUGp2a0M1dQogcERxYTUxRmxr
C:MFRNYU1rSVFqcZdhVUtDbUE0Ukc0dFRUR0svRWpSMWl4OC9EMGdIWVZSbGR5MVlQ
C:ck1QK291NwogNWJpVm5Jb3MrSGlmckF0ck12NHFFcXdMTDRGVFPBVXBhQ2EyQm1n
C:WGZ5MkNTU1FieEQ1T3IxZ2NTYTN2dXJoNqogc1BNQ054cWFYbU1YbVFPcFMRRHVf
C:QnFNTTh0bGRhTjdSWW9qVUVLckdWc05rNwK5eTIvN3NqbJf6eXlVUGY3dgogTDRH
C:Z0RZcWhKWvdWNjFEblhneC9KZDZDV3h2c25ERjZzY3NjUXpVVEVsK2h5d0lEQVFB
C:Qm80SC9NSUg4TUF3RwogQTFVZEV3RUIvd1FDTUFBd0hRWURWUjBPQkJZRUZQWkvj
C:SVFjRC9CaJJRnovTEVSdW8yQURKdmlNSUdNQmdOVgogSFNNRWDZUXdnWUdBRk8w
C:LzdrRWgzRnVFS1MrUS9rWUhhRC9XNndpaG9XYWtaREJpTVFzd0NRWURWUWFHRXdk
C:VgogVXpFTE1Ba0dBmVVFQ0JNQ1EwRXhGREFTQmdOVkJBY1RDMHh2Y3lCQmJtZGxi
C:R1Z6TVJNd0VRWURWUWFLRXdwSgogUTBGT1RpQ1VUVU5JTVJZd0dRWURWUWFERXhK
C:SlEwRk9UaUJVVfVOSU1GUkZVMVFNUTBHQ0FRRXdEZ1lEV1IwUAogQVFILOJBUURB
C:Z2VTUM0R0ExVWRId1FuTUNvd0k2QWhvQitHSFdoMGRIQTZMeTlqY213dWFXtMhi
C:bTR1YjNkBgogTDNSdFkyZ3VZM0pzTUEwR0NTcUdTSWlZRFFFQkN3VUFBNELCQVFC
C:MnFTeTdlaSs0M2N1YktVS3dXUHJ6ejl5LwogSWtyTWVKR0tqbzQwbis5dWVrYXcz
C:REolRXFpT2YvcVo0cGpCRCSrb1I2QkpDYjZOUXVRS3dub0F6NWxFNFNzdQogeTUR
C:aTkzb1QzSGZ5VmM0Z05NSW9IbTFQUZe5bDdEQktyYndiekF1YS8waktXVnpydm1W
C:N1RCZmp4RDNBuW8xUgogY1U1ZEJyNklqYmRMRmxuTzV4MEcwbXJHN3g1T1VQdXVy
C:aWh5aVVSceZEchdIOEtBSDF3TWNDcFhHWEZSdEdLawogd3lkZ3lWUUF0eTdvdGts
C:L3ozY1prQ1ZUMzRnUHZNzBzUjYrUXhVeTh1MEX6RjVBL2JlWWFACHhTWUczMWft
C:TAogQWRYaXRUv0ZpcGFJR2VhOWxFR0ZNMew5K0JnN1h6Tm40blZMWG9reUVCm2Jn
C:UzRzY0c2UXpuWDIzRkdrCiAgIDwvWUWOUNlcnRpZmljYXRlPgogICA8L1g1MDlE
C:YXRhPgogICA8L0tleUluZm8+CiAgPC9TaWduYXR1cmU+CgkJPC92ZXJpZmljYXRp
C:b25Db2RlOnNpZ25lZENvZGU+Cg==
C:      </verificationCode:code>
C:      </verificationCode:encodedSignedCode>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>

```

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

3.2.2. EPP <delete> Command

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

3.2.3. EPP <renew> Command

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

3.2.4. EPP <transfer> Command

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

3.2.5. EPP <update> Command

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

4. Formal Syntax

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

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

4.1. Verification Code Extension Schema

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  <complexType name="codesType">
```



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

</schema>
END
```

5. IANA Considerations

5.1. XML Namespace

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

Registration request for the verificationCode namespace:

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

Registration request for the verificationCode XML schema:

URI: ietf:params:xml:ns:verificationCode-1.0
Registrant Contact: 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: "Verification Code Extension for the Extensible Provisioning Protocol (EPP)"

Document status: Standards Track

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

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

6. 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-verificationcode.

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. Net::DRI

Organization: Dot and Co

Name: Net::DRI

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

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 extension described in this document is based on the security services described by EPP [RFC5730] and protocol layers used by EPP. The security considerations described in these other specifications apply to this specification as well.

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

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

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

8. References

8.1. Normative References

- [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <<https://www.rfc-editor.org/info/rfc2045>>.
- [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>>.
- [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>>.
- [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>>.

[W3C.CR-xmlsig-core2-20120124]

Cantor, S., Roessler, T., Eastlake, D., Yiu, K., Reagle, J., Solo, D., Datta, P., and F. Hirsch, "XML Signature Syntax and Processing Version 2.0", World Wide Web Consortium CR CR-xmlsig-core2-20120124, January 2012, <<http://www.w3.org/TR/2012/CR-xmlsig-core2-20120124>>.

8.2. Informative References

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

[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. Acknowledgements

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

- o Gurshabad Grover
- o Rick Wilhelm
- o John Levine

Appendix B. Change History

B.1. Change from 00 to 01

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

B.2. Change from 01 to 02

1. Added support for the notApplicable status value.

B.3. Change from 02 to 03

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

B.4. Change from 03 to 04

1. Ping update.

B.5. Change from 04 to REGEXT 00

1. Changed to regext working group draft by changing draft-gould-eppext-verificationcode to draft-ietf-regext-verificationcode.

B.6. Change from REGEXT 00 to REGEXT 01

1. Ping update.

B.7. Change from REGEXT 01 to REGEXT 02

1. Ping update.

B.8. Change from REGEXT 02 to REGEXT 03

1. Moved RFC 7451 to an informational reference based on a check done by the Idnits Tool.
2. Replaced the IANA Registrant Contact to be "IESG".

B.9. Change from REGEXT 03 to REGEXT 04

1. Added the Implementation Status section.
2. Revised the sentence "The data verified by the VSP MUST be stored by the VSP along with the generated verification code to address any compliance issues." to "The VSP MUST store the proof of verification and the generated verification code; and MAY store the verified data.", and added text to the Security Considerations section associated with storing the verification data, based on feedback from Gurshabad Grover.

B.10. Change from REGEXT 04 to REGEXT 05

1. Removed the "The Verification Service Provider (VSP) MUST store the verification data in compliance with the applicable privacy laws and regulations." sentence from the Security Considerations, based on feedback from Rick Wilhelm and agreement from Gurshabad Grover.
2. Added the sentence "It is up to server policy what action to take if the verification code type is not set by the grace period." to section 2.2 "Verification Profile", to clarify what happens when the verification code grace period expires. This is based on an issue raised by Gurshabad Grover at the IETF-103 REGEXT meeting.

B.11. Change from REGEXT 05 to REGEXT 06

1. Removed the "The VSP MUST store the proof of verification and the generated verification code; and MAY store the verified data."

sentence from the Introduction, based on feedback from John Levine.

Author's Address

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

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

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: August 5, 2019

M. Loffredo
M. Martinelli
IIT-CNR/Registro.it
February 1, 2019

Registration Data Access Protocol (RDAP) Partial Response
draft-loffredo-regext-rdap-partial-response-03

Abstract

The Registration Data Access Protocol (RDAP) does not include capabilities to request partial responses. In fact, according to the user authorization, the server can only return full responses. Partial responses capability, especially in the case of search queries, could bring benefits to both clients and servers. This document describes a RDAP query extension that allows clients to specify their preference for obtaining a partial response.

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 August 5, 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 (<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. Approaches to Partial Response Implementation | 3 |
| 3. RDAP Path Segment Specification | 5 |
| 3.1. Brief Field Set | 6 |
| 3.2. Full Field Set | 7 |
| 3.3. Subsetting Metadata | 8 |
| 3.3.1. Representing Subsetting Links | 8 |
| 4. RDAP Conformance | 9 |
| 5. Implementation Status | 9 |
| 5.1. IIT-CNR/Registro.it | 10 |
| 6. Security Considerations | 10 |
| 7. IANA Considerations | 11 |
| 8. Acknowledgements | 11 |
| 9. References | 11 |
| 9.1. Normative References | 11 |
| 9.2. Informative References | 12 |
| Appendix A. Change Log | 13 |
| Authors' Addresses | 14 |

1. Introduction

The use of partial response in RESTful API ([REST]) design is very common. The rationale is quite simple: instead of returning objects in API responses with all data fields, only a subset is returned. The benefit is obvious: less data transferred over the network mean less bandwidth usage, faster server response, less CPU time spent both on the server and the client, as well as less memory usage on the client.

Several leading APIs providers (e.g. LinkedIn [LINKEDIN], Facebook [FACEBOOK], Google [GOOGLE]) implement the partial response feature by providing an optional query parameter by which users require the fields they wish to receive. Partial response is also considered a leading principle by many best practices guidelines in REST APIs implementation ([REST-API1], [REST-API2]) in order to improve performance, save on bandwidth and possibly accelerate the overall interaction. In other contexts, for example in digital libraries and bibliographic catalogues, servers can provide responses according to different element sets (i.e. "brief" to get back a short response and "full" to get back the complete response)

Currently, RDAP does not provide a client with any way to request a partial response: the server can only provide the client with the full response ([RFC7483]). Furthermore, servers cannot define the limits of the results according to partial responses and this causes strong inefficiencies.

The protocol described in this specification extends RDAP search capabilities to enable partial responses, by adding a new query parameter and using a RESTful web service. The service is implemented using the Hypertext Transfer Protocol (HTTP) ([RFC7230]) and the conventions described in RFC 7480 ([RFC7480]).

Impact on the current state of RDAP implementation is low.

1.1. Conventions Used in This Document

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

2. Approaches to Partial Response Implementation

Looking at the implementation experiences described above, two approaches to the implementation of partial response can be detected:

- o the client declares explicitly the data fields to get back;
- o the client declares a name identifying a server pre-defined set of data fields.

The former is more flexible than the latter, because clients can specify all the data fields they need. Anyway, it has some drawbacks:

- o Fields have to be declared according to a given syntax. This is a simple task when the data structure of the object is flat, but it is much more difficult when the object has a tree structure like the one of a JSON object. The presence of arrays and deep nested objects contribute to complicate both the syntax definition of the query and, consequently, the processing phase on the server side.
- o Clients should perfectly know the returned object to avoid cases when the required fields are not compliant with the object data structure.
- o The request of some fields cannot match the user access levels. Clients could put unauthorized fields in their requests and servers should define a strategy for providing a response: to

return always an error response or to return a response ignoring the unauthorized fields.

In addition to those listed above, RDAP responses raise some specific issues:

- o Most of the relevant information of the entity object is included in the jCard but such information cannot be easily selected because it is split into the items of a jagged array.
- o RDAP responses contain some properties providing service information (e.g. rdapConformance, links, notices, remarks, etc.) which are not normally selected but they are just as important. They could be returned anyway but, in this case, the server would provide unrequested data.

As an example compliant to the first approach, the Catnap Query Language ([CQL]) is a comprehensive expression language that can be used to customize the JSON response of a RESTful web service. The practical application of CQL to RDAP responses points out that declaring explicitly the output fields would still be acceptable when a few fields are requested but it would become very complicated if the fields should be more. In the following, two CQL expressions for a search domain query are shown (Figure 1): in the first, only objectClassName and ldhName are requested, in the second, the fields of a possible WHOIS-like response are listed.

```
https://example.com/rdap/domains?name=example*.com
    &fields=domainSearchResults(objectClassName,ldhName)

https://example.com/rdap/domains?name=example*.com
    &fields=domainSearchResults(objectClassName,ldhName,unicodeName,
        status,
        events(eventAction,eventDate),
        entities(objectClassName,handle,roles),
        nameservers(objectClassName,ldhName))
```

Figure 1: Examples of CQL expressions for a search domain query

The latter approach seems to facilitate RDAP interoperability. In fact, servers can define some basic field sets which, if known to the clients, can increase the probability to get a valid response. The usage of field sets lets the query string be less complex. In addition, the definition of pre-defined sets of fields makes easier to establish the results limits.

Finally, considering that there is not a real need for RDAP users to have the maximum flexibility in defining all the possible sets of logically connected fields (for example, users interested in domains usually need to know the status, the creation date, the expire date of each domain), the latter approach is preferred.

3. RDAP Path Segment Specification

The new query parameter is an OPTIONAL extension of search path segments defined in RFC 7482 ([RFC7482]). The query parameter is "fieldSet" whose value is a string identifying a server pre-defined set of fields (Figure 2). Values REQUIRED to be implemented are:

- o id: the server provides only the "objectClassName" field and the key field ("handle" for entities, "ldhName" for domains and nameservers). This field set can be used when the client wants to obtain a collection of object identifiers (Figure 3);
- o brief: it contains the fields that can be included in a "short" response. This field set can be used when the client is asking for a subset of the full response which gives a basic knowledge of each object. The fields are those defined in Section 3.1;
- o full: it contains all the information the server can provide for a particular object. Additional considerations are reported in Section 3.2.

Fields belonging to brief and full field sets should be provided according to users access levels. RDAP providers MAY add any property providing service information. Servers MAY implement additional field sets not included in the list above. Servers SHOULD also define a "default" field set.

https://example.com/rdap/domains?name=example*.com&fieldSet=id

Figure 2: Example of RDAP search query reporting the fieldSet parameter

```
{
  "rdapConformance": [
    "rdap_level_0",
  ],
  ...
  "domainSearchResults": [
    {
      "objectClassName": "domain",
      "ldhName": "example1.com"
    },
    {
      "objectClassName": "domain",
      "ldhName": "example2.com"
    },
    ...
  ]
}
```

Figure 3: Example of RDAP response according to the "id" field set

3.1. Brief Field Set

In order to ensure the highest degree of interoperability, the brief field set should contain the most commonly used data elements. Based on the assumption that an RDAP server will return almost the same data as those replied by the corresponding Whois service, the elements included in the brief field set could be those identified in RFC 7485 ([RFC7485]) as mostly supported (i.e. by more than one third of contacted services).

Therefore, RDAP servers are RECOMMENDED to return the following elements in the brief field set (Table 1):

| Object class | Whois property | RDAP property |
|--------------|---------------------|--|
| Domain | Domain Name | ldhName |
| | Domain Status | status |
| | Creation Date | event whose eventAction type is "registration" |
| | Expiration Date | event whose eventAction type is "expiration" |
| | Update Date | event whose eventAction type is "last update" |
| Nameserver | Name Server | ldhName |
| Entity | Entity ID | handle |
| | Entity Name | vcards fn |
| | Entity Organization | vcards org |
| | Entity Email | vcards email |
| | Entity Phone | vcards tel with type="voice" |
| | Entity Fax | vcards tel with type="fax" |
| | Entity Country | country name in vcard adr |
| | Entity City | locality in vcard adr |
| | Entity Postal Code | postal code in vcard adr |

Table 1: Elements included in brief field set

The term "Entity" refers to any kind of contact.

3.2. Full Field Set

With regards to the full field set, some additional considerations can be made about how second level objects could be represented. In fact, since the topmost objects could be returned according to different field sets, the same thing could go for their related objects. As a consequence, the full response could range from the one containing no relationship up to a response where each related object is in turn in full format.

FOR DISCUSSION: Should this specification furtherly detail the full field set according to the different representations of the related objects?

3.3. Subsetting 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 parameter described in this specification recommends servers to provide additional information in their responses about the available field sets. Such information is collected in a new data structures named "subsetting_metadata" containing the following fields:

- o "currentFieldSet": the value of fieldSet parameter as specified in the query string;
- o "availableFieldSets": an array of objects each one describing an available field set:
 - * "name": the field set name;
 - * "description": a human-readable description of the field set;
 - * "default": whether the field set is applied by default;
 - * "links": an array of links as described in RFC 8288 ([RFC8288]) containing the query string that applies the field set.

Both "currentFieldSet" and "availableFieldSets" are OPTIONAL fields of the "subsetting_metadata" structure. In particular, the "currentFieldSet" field is provided when the query string contains a valid value for fieldSet parameter, while the "availableFieldSets" field SHOULD be provided when the fieldSet parameter is missing in the query string or when it is present and the server implements more than a field set for the RDAP object. At least the "name" field is REQUIRED in each item of the "availableFieldSets" array while the other fields are RECOMMENDED.

3.3.1. Representing Subsetting Links

An RDAP server MAY use the "links" array of the "subsetting_metadata" section to provide ready-made references ([RFC8288]) to the available field set (Figure 4). Each link represents a reference to an alternate view of the results.

```
{
  "rdapConformance": [
    "rdap_level_0",
    "subsetting_level_0"
  ],
  ...
  "subsetting_metadata": {
    "currentFieldSet": "brief",
    "availableFieldSets": [
      {
        "name": "id",
        "description": "Contains \"objectClassName\" and the key field",
        "default": false,
        "links": [
          {
            "value": "https://example.com/rdap/domains?name=*nr.com
                        &fieldSet=brief",
            "rel": "alternate",
            "href": "https://example.com/rdap/domains?name=*nr.com
                        &fieldSet=id",
            "title": "Result Subset Link",
            "type": "application/rdap+json"
          },
          ...
        ]
      },
      ...
    ],
    "domainSearchResults": [
      ...
    ]
  ]
}
```

Figure 4: Example of a "subsetting_metadata" instance

4. RDAP Conformance

Servers returning the "subsetting_metadata" section in their responses MUST include "subsetting_level_0" in the rdapConformance array.

5. 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".

5.1. 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.
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. Security Considerations

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. Partial response can contribute together with other strategies (e.g. restricting search functionality, limiting the rate of search requests, truncating and paging results) to mitigate this risk.

Furthermore, partial response can help RDAP operators to regulate access control based on client identification, implemented by HTTP basic or digest authentication as described in RFC 7481 ([RFC7481]) or by a federated authentication system ([I-D.hollenbeck-regext-rdap-openid]). In fact, RDAP operators can follow different, not alternative, approaches to the building of responses according to the user access levels:

- o the list of fields for each set (except "id") can be different according to the user access levels. At present, this is already implemented for the full response, but it could be done also for the other defined field sets. In some cases, it might happen that brief and full field sets are exactly the same;
- o some field sets could be available only to some users. In this case, servers could define additional field sets to those indicated above ("id", "brief", "full"), making them available only to users with specific access levels.

Servers can also define different results limits according to the available field sets, so a more flexible truncation strategy can be realized and users can take advantage of a more efficient results paging implementation
([I-D.loffredo-regext-rdap-sorting-and-paging]).

Therefore, the new parameter presented in this document provides the RDAP operators with a way to implement a secure server without penalizing its efficiency.

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgements

The authors would like to acknowledge Scott Hollenbeck for his contribution to this document.

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>>.
- [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>>.
- [RFC7485] Zhou, L., Kong, N., Shen, S., Sheng, S., and A. Servin, "Inventory and Analysis of WHOIS Registration Objects", RFC 7485, DOI 10.17487/RFC7485, March 2015, <<https://www.rfc-editor.org/info/rfc7485>>.
- [RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <<https://www.rfc-editor.org/info/rfc8288>>.

9.2. Informative References

- [CQL] Whitaker, G., "Catnap Query Language Reference", September 2017, <<https://github.com/gregwhitaker/catnap/wiki/Catnap-Query-Language-Reference>>.
- [FACEBOOK] facebook.com, "facebook for developers - Using the Graph API", July 2017, <<https://developers.facebook.com/docs/graph-api/using-graph-api>>.
- [GOOGLE] google.com, "Making APIs Faster: Introducing Partial Response and Partial Update", March 2010, <<http://googlecode.blogspot.it/2010/03/making-apis-faster-introducing-partial.html>>.
- [HATEOAS] Jedrzejewski, B., "HATEOAS - a simple explanation", 2018, <<https://www.e4developer.com/2018/02/16/hateoas-simple-explanation/>>.

- [I-D.hollenbeck-regext-rdap-openid]
Hollenbeck, S., "Federated Authentication for the Registration Data Access Protocol (RDAP) using OpenID Connect", draft-hollenbeck-regext-rdap-openid-10 (work in progress), August 2018.
- [I-D.loffredo-regext-rdap-sorting-and-paging]
Loffredo, M., Martinelli, M., and S. Hollenbeck, "Registration Data Access Protocol (RDAP) Query Parameters for Result Sorting and Paging", draft-loffredo-regext-rdap-sorting-and-paging-05 (work in progress), September 2018.
- [LINKEDIN]
linkedin.com, "Java One 2009: Building Consistent RESTful APIs in a High Performance Environment", July 2009, <<https://blog.linkedin.com/2009/07/08/brandon-duncan-java-one-building-consistent-restful-apis-in-a-high-performance-environment>>.
- [REST]
Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000, <http://www.restapitutorial.com/media/RESTful_Best_Practices-v1_1.pdf>.
- [REST-API1]
Jobinesh, P., "RESTful Java Web Services - Second Edition", September 2015.
- [REST-API2]
Masse, M., "REST API Design Rulebook", October 2011.
- [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>>.

Appendix A. Change Log

- 00: Initial version.
- 01: Added Catnap Query Language as an example of language that can be used to declare explicitly the output fields of RDAP responses. Revised some sentences and references.
- 02: Added "Subsetting Metadata" and "RDAP Conformance" sections.
- 03: Added "Brief Field Set" and "Full Field Set" sections.

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>

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: August 4, 2019

M. Loffredo
M. Martinelli
IIT-CNR/Registro.it
January 31, 2019

Registration Data Access Protocol (RDAP) Reverse search capabilities
draft-loffredo-regext-rdap-reverse-search-04

Abstract

The Registration Data Access Protocol (RDAP) does not include query capabilities to find the list of domains related to a set of entities matching a given search pattern. Even if such capabilities, commonly referred as reverse search, respond to some needs not yet readily fulfilled by the current Whois protocol, they have raised concerns from two perspectives: server processing impact and data privacy. Anyway, the impact of the reverse queries on RDAP servers processing is the same as the standard searches and it can be reduced by implementing policies to deal with large result sets, while data privacy risks can be prevented by RDAP access control functionalities. This document describes RDAP query extensions that allow clients to request a reverse search based on the domains-entities relationship.

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 August 4, 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 (<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. RDAP Path Segment Specification | 4 |
| 3. Implementation Considerations | 5 |
| 3.1. JSON in URLs | 5 |
| 4. Implementation Status | 6 |
| 4.1. IIT-CNR/Registro.it | 7 |
| 5. Privacy Considerations | 7 |
| 6. Security Considerations | 7 |
| 7. IANA Considerations | 7 |
| 8. Acknowledgements | 7 |
| 9. References | 8 |
| 9.1. Normative References | 8 |
| 9.2. Informative References | 9 |
| Appendix A. Change Log | 10 |
| Authors' Addresses | 10 |

1. Introduction

Reverse Whois is a service provided by many web applications that allow users to find domain names owned by an individual or a company starting from the owner details, such as name and email. Even if it has been considered useful for some legal purposes (e.g. uncovering trademark infringements, detecting cybercrime cases), its availability as a standardised Whois capability has been objected for two main reasons, which now don't seem to conflict with an RDAP implementation.

The first objection has been caused by the potential risks of privacy violation. However, TLDs community is considering a new generation of Registration Directory Services ([ICANN-RDS1],[ICANN-RDS2]), which provide access to sensitive data under some permissible purposes and according to adequate policies to enforce the requestor accreditation, authentication, authorization, and terms and conditions of data use. It is well known that such security policies are not implemented in Whois ([RFC3912]), while they are in RDAP

([RFC7481]). Therefore, RDAP permits a reverse search implementation complying with privacy protection principles.

Another objection to the implementation of a reverse search capability has been connected with its impact on server processing. Since RDAP supports search queries, the impact of both standard and reverse searches is equivalent and can be mitigated by servers adopting ad hoc strategies. Furthermore, reverse search is almost always performed by specifying an entity role (e.g. registrant, technical contact) and this can contribute to restricting the result set.

Reverse searches, such as finding the list of domain names associated with contacts, nameservers or DNSSEC keys, may be useful to registrars as well. Usually, registries adopt out-of-band mechanisms to provide results to registrars asking for reverse searches on their domains. Possible reasons of such requests are:

- o the loss of synchronization between the registrar database and the registry database;
- o the need of such data to perform massive EPP ([RFC5730]) updates (e.g. changing the contacts of a set of domains, etc.).

Currently, RDAP does not provide any way for a client to search for the collection of domains associated with an entity ([RFC7482]). A query (lookup or search) on domains can return the array of entities related to a domain with different roles (registrant, registrar, administrative, technical, reseller, etc.), but the reverse operation is not allowed. Only reverse searches to find the collection of domains related to a nameserver (ldhName or ip) can be requested. Since entities can be in relation with all RDAP objects ([RFC7483]), the availability of a reverse search can be common to all RDAP query paths.

The protocol described in this specification aims to extend the RDAP query capabilities to enable reverse search based on the domains-entities relationship (the classic Reverse Whois scenario). The extension is implemented by adding new path segments (i.e. search paths) and using a RESTful web service ([REST]). The service is implemented using the Hypertext Transfer Protocol (HTTP) ([RFC7230]) and the conventions described in RFC 7480 ([RFC7480]).

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 Path Segment Specification

The new search paths are OPTIONAL extensions of path segments defined in RFC 7482 ([RFC7482]). The search paths are:

Syntax: domains?entityHandle=<reverse search pattern>

Syntax: domains?entityFn=<reverse search pattern>

Syntax: domains?entityEmail=<reverse search pattern>

Syntax: domains?entityAddr=<reverse search pattern>

The reverse search pattern is a JSON ([RFC8259]) object including two members: "value" and "role". The "value" member represents the search pattern to be applied to the corresponding entity field and can be a JSON type primitive or object. The "role" member is a string whose possible values are those detailed in Section 10.2.4 of RFC 7483 ([RFC7483]). The former is REQUIRED while the latter is OPTIONAL to allow RDAP servers to provide reverse search capabilities without specifying any role.

The search patterns corresponding to the "value" in the first two cases (Figure 1) are the same as specified in paragraph Section 3.2.3 of RFC 7482 ([RFC7482]).

```
domains?entityHandle={"value":"CID-40*","role":"registrant"}
```

```
domains?entityFn={"value":"Bobby*","role":"registrant"}
```

Figure 1: Examples of RDAP queries to find all domains related to a registrant whose handle matches "CID-40*" and whose formatted name matches "Bobby*"

The last two reverse searches are considered by gTLD stakeholders very useful to improve RDS searchability ([ICANN-RDS1], [ICANN-RA]).

Searches for domains by related entity email are specified using this form:

```
domains?entityEmail={"value":"XXXX","role":"ZZZZ"}
```

where XXXX is a search pattern representing an email address as defined in RFC 5322 ([RFC5322]).

Searches for domains by related entity postal address are specified using this form:

```
domains?entityAddr={"value":YYYY,"role":"ZZZZ"}
```

where YYYY is a JSON object containing the information described in Section 2.4 of RFC 5733 ([RFC5733]), respectively: "street", "city", "sp", "pc" and "cc" (Figure 2). All the members of the postal address object are OPTIONAL but at least one is REQUIRED. The constraints on the members are implicitly joined by AND.

```
domains?entityAddr={"value":{"cc":"CA","city":"Sydney"},"role":"registrant"}
```

Figure 2: Example of a RDAP query to find all domains related to a registrant whose postal address contains the country code equals to "CA" and the city equals to "Sydney"

3. Implementation Considerations

The implementation of the proposed extension is technically feasible. The search paths "handle" and "fn" are used as standard paths to search for entities. With regards to the last two reverse searches, both email and postal address information are usually required by the registries but, while the former is usually mapped onto a DBMS indexed field, the latter is mapped onto a combination of non-indexed fields. As a consequence while the former should not significantly decrease the performance, the latter might have an impact on server processing. Anyway, this impact is evaluated to be the same as other query capabilities already presented in RDAP (e.g. wildcard prefixed search pattern) so the risks to generate huge result sets are the same as those related to other standard searches and can be mitigated by adopting the same policies (e.g. restricting search functionalities, limiting the rate of search requests according to the user profile, truncating and paging the results, returning partial responses).

3.1. JSON in URLs

Many web services, including RDAP, rely on the HTTP GET method to take advantage from some of its features:

- o GET requests can be cached;
- o GET requests remain in the browser history;
- o GET requests can be bookmarked.

Sometimes, it happens that such advantages should be combined with the requirement to pass objects and arrays in the query string. JSON is the best candidate as data interchange format, but it contains

some characters that are forbidden from appearing in a URL. Anyway, escaping the invalid characters is not an issue because, on the client side, modern browsers automatically encode URLs and, on the server side, several URL encoding/decoding libraries for all web development programming languages are available. The downside of URL encoding is that it can make a pretty long URL, which, depending on the initial length and the number of invalid characters, might exceed the practical limit of web browsers (i.e. 2,000 characters).

Other solutions to pass a JSON expression in a URL could be:

- o converting JSON to Base64 ([RFC4648]), but binary data are unreadable;
- o using a JSON variation that complies with URL specifications and maintains readability like Rison ([RISON]), URLON ([URLON]) or JSURL ([JSURL]).

The extensions proposed in this document rely on URL encoding because it is widely supported and the risk to exceed the maximum URL length is considered to be very unlikely in RDAP.

4. 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".

4.1. 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.
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

5. Privacy Considerations

The use of the capability described in this document SHOULD be compliant with the rules about privacy protection each RDAP provider is subject to. Sensitive registration data SHOULD be protected and accessible for permissible purposes only. Therefore, it is recommended that RDAP servers provide reverse search only to those requestors who are authorized according to a lawful basis. Some potential users of this capability include registrars searching for their own domains and operators in the exercise of an official authority or performing a specific task in the public interest that is set out in law. Another scenario consists of permitting reverse searches, which take into account only those entities that have previously given the explicit consent for publishing and processing their personal data.

6. Security Considerations

Security services required to provide controlled access to the operations specified in this document are described in RFC 7481 ([RFC7481]).

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgements

The authors would like to acknowledge Scott Hollenbeck, Francisco Arias, Gustavo Lozano and Eduardo Alvarez for their contribution to this document.

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>>.
- [RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, DOI 10.17487/RFC3912, September 2004, <<https://www.rfc-editor.org/info/rfc3912>>.
- [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <<https://www.rfc-editor.org/info/rfc5322>>.
- [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>>.
- [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>>.

9.2. Informative References

- [ICANN-RA] Internet Corporation For Assigned Names and Numbers, "Registry Agreement", July 2017, <<https://newgtlds.icann.org/sites/default/files/agreements/agreement-approved-31jul17-en.pdf>>.
- [ICANN-RDS1] Internet Corporation For Assigned Names and Numbers, "Final Report from the Expert Working Group on gTLD Directory Services: A Next-Generation Registration Directory Service (RDS)", June 2014, <<https://www.icann.org/en/system/files/files/final-report-06jun14-en.pdf>>.
- [ICANN-RDS2] Internet Corporation For Assigned Names and Numbers, "Final Issue Report on a Next-Generation gTLD RDS to Replace WHOIS", October 2015, <<http://whois.icann.org/sites/default/files/files/final-issue-report-next-generation-rds-07oct15-en.pdf>>.
- [JSURL] github.com, "JSURL", 2016, <<https://github.com/Sage/jsurl>>.
- [REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000, <http://www.restapitutorial.com/media/RESTful_Best_Practices-v1_1.pdf>.
- [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>>.

- [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>>.
- [RISON] github.com, "Rison - Compact Data in URIs", 2017, <<https://github.com/Nanonid/rison>>.
- [URLON] github.com, "URL Object Notation", 2017, <<https://github.com/cerebral/urlon>>.

Appendix A. Change Log

- 00: Initial version.
- 01: Revised some sentences and references.
- 02: Added "entityEmail" and "entityAddr" path segments. Removed "entityRole" path segment. Revised "Acknowledgements" section.
- 03: Added "JSON in URLs" section.
- 04: Revised some sentences in "Introduction" section. Added "Privacy Considerations" section.

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>

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=wJlCDLil6KTWypN7T6vc6nWEemEYe99HjflXYlxmQV-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/>