

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: December 3, 2020

G. Lozano
ICANN
Jun 1, 2020

Registry Data Escrow Specification
draft-ietf-regext-data-escrow-10

Abstract

This document specifies the format and contents of data escrow deposits targeted primarily for domain name registries. The specification is designed to be independent of the underlying objects that are being escrowed and therefore it could also be used for purposes other than domain name registries.

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 December 3, 2020.

Copyright Notice

Copyright (c) 2020 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	3
3. Problem Scope	5
4. Conventions Used in This Document	6
4.1. Date and Time	6
5. Protocol Description	6
5.1. Root element <deposit>	7
5.2. Rebuilding the registry from data escrow deposits	8
6. Formal Syntax	9
6.1. RDE Schema	9
7. Internationalization Considerations	11
8. IANA Considerations	11
9. Implementation Status	12
9.1. Implementation in the gTLD space	12
10. Security Considerations	13
11. Privacy Considerations	13
12. Acknowledgments	14
13. Change History	14
13.1. Changes from 00 to 01	14
13.2. Changes from 01 to 02	15
13.3. Changes from 02 to 03	16
13.4. Changes from 03 to 04	16
13.5. Changes from 04 to 05	16
13.6. Changes from 05 to 06	16
13.7. Changes from 06 to 07	16
13.8. Changes from 07 to 08	16
13.9. Changes from 08 to 09	17
13.10. Changes from 09 to 10	17
13.11. Changes from 10 to 11	17
13.12. Changes from 11 to REGEXT 00	17
13.13. Changes from version REGEXT 00 to REGEXT 01	17
13.14. Changes from version REGEXT 01 to REGEXT 02	17
13.15. Changes from version REGEXT 02 to REGEXT 03	17
13.16. Changes from version REGEXT 03 to REGEXT 04	17
13.17. Changes from version REGEXT 04 to REGEXT 05	18
13.18. Changes from version REGEXT 05 to REGEXT 06	18
13.19. Changes from version REGEXT 06 to REGEXT 07	18
13.20. Changes from version REGEXT 07 to REGEXT 08	18
13.21. Changes from version REGEXT 08 to REGEXT 09	19
13.22. Changes from version REGEXT 09 to REGEXT 10	19
14. Example of a Full Deposit	19
15. Example of a Differential Deposit	20
16. Example of an Incremental Deposit	20
17. References	21
17.1. Normative References	21
17.2. Informative References	22

Author's Address 22

1. Introduction

Registry Data Escrow is the process by which a registry periodically submits data deposits to a third-party called an escrow agent. These deposits comprise the minimum data needed by a third-party to resume operations if the registry cannot function and is unable or unwilling to facilitate an orderly transfer of service. For example, for a domain name registry or registrar, the data to be deposited would include all the objects related to registered domain names, e.g., names, contacts, name servers, etc.

The goal of data escrow is higher resiliency of registration services, for the benefit of Internet users. The beneficiaries of a registry are not just those registering information there, but also the users of services relying on the registry data.

In the context of domain name registries, registration data escrow is a requirement for generic top-level domains (e.g., Specification 2 of the ICANN Base Registry Agreement, see [ICANN-GTLD-RA-20170731]) and some country code top-level domain managers are also currently escrowing data. There is also a similar requirement for ICANN-accredited domain registrars.

This document specifies a format for data escrow deposits independent of the objects being escrowed. An independent specification is required for each type of registry/set of objects that is expected to be escrowed.

The format for data escrow deposits is specified using the Extensible Markup Language (XML) 1.0 as described in [W3C.REC-xml-20081126] and XML Schema notation as described in [W3C.REC-xmlschema-1-20041028] and [W3C.REC-xmlschema-2-20041028].

Readers are advised to read the terminology section carefully to understand the precise meanings of Differential and Incremental Deposits as the definitions used in this document are different from the definitions typically used in the domain of data backups.

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.

Deposit. Deposits can be of three kinds: Full, Differential or Incremental. For all kinds of deposits, the universe of registry objects to be considered for data escrow are those objects necessary in order to offer the registry services.

Differential Deposit. Contains data that reflects all transactions involving the database that were not reflected in the last previous Full, Incremental or Differential Deposit, as the case may be. Differential Deposit files will contain information from all database objects that were added, modified or deleted since the previous deposit was completed as of its defined Timeline Watermark.

Domain Name. See definition of Domain name in [RFC8499].

Escrow Agent. The organization designated by the registry or the third-party beneficiary to receive and guard data escrow deposits from the registry.

Full Deposit. Contains the registry data that reflects the current and complete registry database and will consist of data that reflects the state of the registry as of a defined Timeline Watermark for the deposit.

Incremental Deposit. Contains data that reflects all transactions involving the database that were not reflected in the last previous Full Deposit. Incremental Deposit files will contain information from all database objects that were added, modified or deleted since the previous Full Deposit was completed as of its defined Timeline Watermark. If the Timeline Watermark of an Incremental Deposit were to cover (i.e., one or more Incremental or Differential Deposits exist for the period between the Timeline Watermark of a Full and an Incremental or Differential Deposit) the Timeline Watermark of another Incremental or Differential Deposit since the last Full Deposit, the more recent deposit MUST contain all the transactions of the earlier deposit.

Registrar. See definition of Registrar in [RFC8499].

Registry. See definition of Registry in [RFC8499].

Third-Party Beneficiary. Is the organization that, under extraordinary circumstances, would receive the escrow deposits the registry transferred to the escrow agent. This organization could be a backup registry, registry regulator, contracting party of the registry, etc.

Timeline Watermark. Point in time on which to base the collecting of database objects for a deposit. Deposits are expected to be consistent to that point in time.

Top-Level Domain. See definition of Top-Level Domain (TLD) in [RFC8499].

3. Problem Scope

In the past few years, the issue of registry continuity has been carefully considered in the gTLD and ccTLD space. Various organizations have carried out risk analyses and developed business continuity plans to deal with those risks, should they materialize.

One of the solutions considered and used, especially in the gTLD space, is Registry Data Escrow as a way to ensure the continuity of registry services in the extreme case of registry failure.

So far, almost every registry that uses Registry Data Escrow has its own specification. It is anticipated that more registries will be implementing escrow especially with an increasing number of domain registries coming into service, adding complexity to this issue.

It would seem beneficial to have a standardized specification for Registry Data Escrow that can be used by any registry to submit its deposits.

While the domain name industry has been the main target for this specification, it has been designed to be as general as possible.

Specifications covering the objects used by registration organizations shall identify the format and contents of the deposits a registry has to make, such that a different registry would be able to rebuild the registration services of the former, without its help, in a timely manner, with minimum disruption to its users.

Since the details of the registration services provided vary from registry to registry, specifications covering the objects used by registration organizations shall provide mechanisms that allow its extensibility to accommodate variations and extensions of the registration services.

Given the requirement for confidentiality and the importance of accuracy of the information that is handled in order to offer registration services, parties using this specification shall define confidentiality and integrity mechanisms for handling the registration data.

Specifications covering the objects used by registration organizations shall not include in the specification transient objects that can be recreated by the new registry, particularly those of delicate confidentiality, e.g., DNSSEC KSK/ZSK private keys.

Details that are a matter of policy should be identified as such for the benefit of the implementers.

Non-technical issues concerning data escrow, such as whether to escrow data and under which purposes the data may be used, are outside of scope of this document.

Parties using this specification shall use a signaling mechanism to control the transmission, reception and validation of data escrow deposits. The definition of such a signaling mechanism is out of the scope of this document.

4. Conventions Used in This Document

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

The XML namespace prefix "rdeObj1" and "rdeObj2" with the corresponding namespaces "urn:example:params:xml:ns:rdeObj1-1.0" and "urn:example:params:xml:ns:rdeObj2-1.0" are used as example data escrow objects.

4.1. Date and Time

Numerous fields indicate "dates", such as the creation and expiry dates for objects. These fields SHALL contain timestamps indicating the date and time in UTC, specified in Internet Date/Time Format (see [RFC3339], Section 5.6) with the time-offset specified as "Z".

5. Protocol Description

The following is a format for data escrow deposits as produced by a registry. The deposits are represented in XML. Only the format of the objects deposited is defined. Nothing is prescribed about the method used to transfer such deposits between the registry and the escrow agent or vice versa.

The protocol intends to be object agnostic allowing the "overload" of abstract elements using the "substitutionGroup" attribute of the XML Schema element to define the actual elements of an object to be escrowed.

The specification for each object to be escrowed MUST declare the identifier to be used to reference the object to be deleted or added/modified.

5.1. Root element <deposit>

The container or root element for a Registry Data Escrow deposit is <deposit>.

The <deposit> element contains the following attributes:

- o A REQUIRED "type" attribute that is used to identify the kind of deposit:
 - * FULL: Full.
 - * INCR: Incremental.
 - * DIFF: Differential.
- o A REQUIRED "id" attribute that is used to uniquely identify the escrow deposit. Each registry is responsible for maintaining its own escrow deposits' identifier space to ensure uniqueness.
- o A "prevId" attribute that can be used to identify the previous Incremental, Differential or Full Deposit. This attribute is REQUIRED in Differential Deposits ("DIFF" type), is OPTIONAL in Incremental Deposits ("INCR" type), and is not used in Full Deposits ("FULL" type).
- o An OPTIONAL "resend" attribute that is incremented each time the escrow deposit failed the verification procedure at the receiving party and a new escrow deposit needs to be generated by the registry for that specific date. The first time a deposit is generated the attribute is either omitted or MUST be "0". If a deposit needs to be generated again, the attribute MUST be set to "1", and so on.

The <deposit> element contains the following the child elements:

5.1.1. Child <watermark> element

A REQUIRED <watermark> element contains the date-time corresponding to the Timeline Watermark of the deposit.

5.1.2. Child <rdeMenu> element

This element contains auxiliary information of the data escrow deposit.

A REQUIRED <rdeMenu> element contains the following child elements:

- o A REQUIRED <version> element that identifies the RDE protocol version, this value MUST be 1.0.
- o One or more <objURI> elements that contain namespace URIs representing the <contents> and <deletes> element objects.

5.1.3. Child <deletes> element

For Differential Deposits, this element contains the list of objects that have been deleted since the previous deposit of any type. For Incremental Deposits, this element contains the list of objects that have been deleted since the previous Full Deposit.

This section of the deposit MUST NOT be present in Full Deposits.

5.1.4. Child <contents> element

For Full Deposits this element contains all objects. For Differential Deposits, this element contains the list of objects that have been added or modified since the previous deposit of any type. For Incremental Deposits, this element contains the list of objects that have been added or modified since the previous Full Deposit.

5.2. Rebuilding the registry from data escrow deposits

When applying Incremental or Differential Deposits (when rebuilding the registry from data escrow deposits), the relative order of the <deletes> and <contents> elements is important because dependencies may exist between the objects. All the <deletes> elements MUST be applied first, in the order that they appear. All the <contents> elements MUST be applied next, in the order that they appear.

If an object is present in the <contents> or <deletes> section of several deposits (e.g. Full and Differential) the registry data from the latest deposit (as defined by the Timeline Watermark) SHOULD be used when rebuilding the registry. An object SHOULD NOT exist multiple times either in the <contents> or <deletes> elements in a single deposit.

When rebuilding a registry, the <deletes> section MUST be ignored if present in a Full Deposit.

6. Formal Syntax

RDE is specified in XML Schema notation. The formal syntax presented here is a complete schema representation of RDE suitable for automated validation of RDE 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. RDE Schema

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

  <annotation>
    <documentation>
      Registry Data Escrow schema
    </documentation>
  </annotation>

  <!-- Root element -->
  <element name="deposit" type="rde:escrowDepositType"/>

  <!-- RDE types -->
  <complexType name="escrowDepositType">
    <sequence>
      <element name="watermark" type="dateTime"/>
      <element name="rdeMenu" type="rde:rdeMenuType"/>
      <element name="deletes" type="rde:deletesType" minOccurs="0"/>
      <element name="contents" type="rde:contentsType" minOccurs="0"/>
    </sequence>
    <attribute name="type" type="rde:depositTypeType" use="required"/>
    <attribute name="id" type="rde:depositIdType" use="required"/>
    <attribute name="prevId" type="rde:depositIdType"/>
    <attribute name="resend" type="unsignedShort" default="0"/>
  </complexType>

  <!-- Menu type -->
  <complexType name="rdeMenuType">
    <sequence>
      <element name="version" type="rde:versionType"/>
      <element name="objURI" type="anyURI" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</schema>
END
```

```
</complexType>

<!-- Deletes Type -->
<complexType name="deletesType">
  <sequence minOccurs="0" maxOccurs="unbounded">
    <element ref="rde:delete"/>
  </sequence>
</complexType>

<element name="delete" type="rde:deleteType" abstract="true" />
<complexType name="deleteType">
  <complexContent>
    <restriction base="anyType"/>
  </complexContent>
</complexType>

<!-- Contents Type -->
<complexType name="contentsType">
  <sequence minOccurs="0" maxOccurs="unbounded">
    <element ref="rde:content"/>
  </sequence>
</complexType>

<element name="content" type="rde:contentType" abstract="true" />
<complexType name="contentType">
  <complexContent>
    <restriction base="anyType"/>
  </complexContent>
</complexType>

<!-- Type of deposit -->
<simpleType name="depositTypeType">
  <restriction base="token">
    <enumeration value="FULL"/>
    <enumeration value="INCR"/>
    <enumeration value="DIFF"/>
  </restriction>
</simpleType>

<!-- Deposit identifier type -->
<simpleType name="depositIdType">
  <restriction base="token">
    <pattern value="\w{1,13}"/>
  </restriction>
</simpleType>

<!-- A RDE version number is a dotted pair of decimal numbers -->
<simpleType name="versionType">
```

```
<restriction base="token">
  <pattern value="[1-9]+\.[0-9]+"/>
  <enumeration value="1.0"/>
</restriction>
</simpleType>

</schema>
END
```

7. Internationalization Considerations

Data escrow deposits are 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.

8. IANA Considerations

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

Registration request for the RDE namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the RDE XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See the "Formal Syntax" section of this document.

9. 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".

9.1. Implementation in the gTLD space

Organization: ICANN

Name: ICANN Registry Agreement

Description: the ICANN Base Registry Agreement requires Registries, Data Escrow Agents, and ICANN to implement this specification. ICANN receives daily notifications from Data Escrow Agents confirming that more than 1,200 gTLDs are sending deposits that comply with this specification. ICANN receives on a weekly basis per gTLD, from more than 1,200 gTLD registries, a Bulk Registration Data Access file that also complies with this specification. In addition, ICANN is aware of Registry Service Provider transitions using data files that conform to this specification.

Level of maturity: production.

Coverage: all aspects of this specification are implemented.

Version compatibility: versions 03 - 08 are known to be implemented.

Contact: gustavo.lozano@icann.org

URL: <https://www.icann.org/resources/pages/registries/registries-agreements-en>

10. Security Considerations

This specification does not define the security mechanisms to be used in the transmission of the data escrow deposits, since it only specifies the minimum necessary to enable the rebuilding of a registry from deposits without intervention from the original registry.

Depending on local policies, some elements, or, most likely, the whole deposit will be considered confidential. As such, the parties SHOULD take all the necessary precautions such as encrypting the data at rest and in transit to avoid inadvertent disclosure of private data. Regardless of the precautions taken by the parties regarding data at rest and in transit, authentication credentials MUST NOT be escrowed.

Authentication of the parties passing data escrow deposit files is also of the utmost importance. The escrow agent MUST properly authenticate the identity of the registry before accepting data escrow deposits. In a similar manner, the registry MUST authenticate the identity of the escrow agent before submitting any data.

Additionally, the registry and the escrow agent MUST use integrity checking mechanisms to ensure the data transmitted is what the source intended. Validation of the contents by the escrow agent is RECOMMENDED to ensure not only that the file was transmitted correctly from the registry, but also that the contents are "meaningful".

Note: if Transport Layer Security (TLS) is used when providing an escrow services, the recommendations in [RFC7525] MUST be implemented.

11. Privacy Considerations

This specification defines a format that may be used to escrow personal data. The process of data escrow is governed by a legal document agreed by the parties, and such legal document must ensure that privacy-sensitive and/or personal data receives the required protection.

12. Acknowledgments

Special suggestions that have been incorporated into this document were provided by James Gould, Edward Lewis, Jaap Akkerhuis, Lawrence Conroy, Marc Groeneweg, Michael Young, Chris Wright, Patrick Mevzek, Stephen Morris, Scott Hollenbeck, Stephane Bortzmeyer, Warren Kumari, Paul Hoffman, Vika Mpisane, Bernie Hoeneisen, Jim Galvin, Andrew Sullivan, Hiro Hotta, Christopher Browne, Daniel Kalchev, David Conrad, James Mitchell, Francisco Obispo, Bhadresh Modi and Alexander Mayrhofer.

Shoji Noguchi and Francisco Arias participated as co-authors until version 07 providing invaluable support for this document.

13. Change History

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

13.1. Changes from 00 to 01

1. Included DNSSEC elements as part of the basic <domain> element as defined in RFC 5910.
2. Included RGP elements as part of the basic <domain> element as defined in RFC 3915.
3. Added support for IDNs and IDN variants.
4. Eliminated the <summary> element and all its subordinate objects, except <watermarkDate>.
5. Renamed <watermarkDate> to <watermark> and included it directly under root element.
6. Renamed root element to <deposit>.
7. Added <authinfo> element under <registrar> element.
8. Added <roid> element under <registrar> element.
9. Reversed the order of the <deletes> and <contents> elements.
10. Removed <rdeDomain:status> minOccurs="0".
11. Added <extension> element under root element.
12. Added <extension> element under <contact> element.

13. Removed <period> element from <domain> element.
 14. Populated the "Security Considerations" section.
 15. Populated the "Internationalization Considerations" section.
 16. Populated the "Extension Example" section.
 17. Added <deDate> element under <domain> element.
 18. Added <icannID> element under <registrar> element.
 19. Added <eppParams> element under root element.
 20. Fixed some typographical errors and omissions.
- 13.2. Changes from 01 to 02
1. Added definition for "canonical" in the "IDN variants Handling" section.
 2. Clarified that "blocked" and "reserved" IDN variants are optional.
 3. Made <rdeRegistrar:authInfo> optional.
 4. Introduced substitutionGroup as the mechanism for extending the protocol.
 5. Moved <eppParams> element to be child of <contents>.
 6. Text improvements in the Introduction, Terminology, and Problem Scope per Jay's suggestion.
 7. Removed <trDate> from <rdeDomain> and added <trnData> instead, which include all the data from the last (pending/processed) transfer request.
 8. Removed <trDate> from <rdeContact> and added <trnData> instead, which include all the data from the last (pending/processed) transfer request.
 9. Fixed some typographical errors and omissions.

13.3. Changes from 02 to 03

1. Separated domain name objects from protocol.
2. Moved <extension> elements to be child of <deletes> and <contents>, additionally removed <extension> element from <rdeDomain>, <rdeHost>, <rdeContact>, <rdeRegistrar> and <rdeIDN> elements.
3. Modified the definition of <rde:id> and <rde:prevId>.
4. Added <rdeMenu> element under <deposit> element.
5. Fixed some typographical errors and omissions.

13.4. Changes from 03 to 04

1. Removed <eppParams> objects.
2. Populated the "Extension Guidelines" section.
3. Fixed some typographical errors and omissions.

13.5. Changes from 04 to 05

1. Fixes to the XSD.
2. Extension Guidelines moved to dnrd-mappings draft.
3. Fixed some typographical errors and omissions.

13.6. Changes from 05 to 06

1. Fix resend definition.

13.7. Changes from 06 to 07

1. Editorial updates.
2. schemaLocation removed from RDE Schema.

13.8. Changes from 07 to 08

1. Ping update.

13.9. Changes from 08 to 09

1. Ping update.

13.10. Changes from 09 to 10

1. Implementation Status section was added.

13.11. Changes from 10 to 11

1. Ping update.

13.12. Changes from 11 to REGEXT 00

1. Internet Draft (I-D) adopted by the REGEXT WG.

13.13. Changes from version REGEXT 00 to REGEXT 01

1. Privacy consideration section was added.

13.14. Changes from version REGEXT 01 to REGEXT 02

1. Updated the Security Considerations section to make the language normative.
2. Updated the rde XML schema to remove the dependency with the eppcom namespace reference.
3. Editorial updates.
4. Remove the reference to RFC 5730.
5. Added complete examples of deposits.

13.15. Changes from version REGEXT 02 to REGEXT 03

1. The <contents> section changed from MUST to SHOULD, in order to accommodate an Incremental or Differential Deposit that only includes deletes.
2. Editorial updates.

13.16. Changes from version REGEXT 03 to REGEXT 04

1. Moved [RFC8499] to the Normative References section.

13.17. Changes from version REGEXT 04 to REGEXT 05

1. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regext/UNo6YxapgjyerAYv0223zEuzjFk>
2. The examples of deposits were moved to their own sections.
3. <deposit> elements definition moved to section 5.1.
4. The DIFF example was modified to make it more representative of a differential deposit.

13.18. Changes from version REGEXT 05 to REGEXT 06

1. Normative references for XLM, XML Schema added.
2. Text added to define that version MUST be 1.0.
3. Normative SHOULD replaced should in the second paragraph in the security section.

13.19. Changes from version REGEXT 06 to REGEXT 07

1. Registration contact changed in section 8.

13.20. Changes from version REGEXT 07 to REGEXT 08

1. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regext/hDLz2ym4oR-ukA4Fm-QJ8FzaxxE>
2. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regext/780Xw-z1RMRb79nmZ6ABmRTolFU>
3. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regext/YnPnrSedrCcgQ2AXbjBTuQzqMds>
4. Changes based on the feedback provided here:
https://mailarchive.ietf.org/arch/msg/regext/BiV0NHi_k7cYwTiLdLwVgqEcFuo

13.21. Changes from version REGEXT 08 to REGEXT 09

1. Changes based on the feedback provided here:
https://mailarchive.ietf.org/arch/msg/regext/x_8twvi-MS4dDDRfAZfNJH92UaQ
2. Changes based on the feedback provided here:
https://mailarchive.ietf.org/arch/msg/regext/B3QTxUCWUE4R_QharAQ1A3041j0

13.22. Changes from version REGEXT 09 to REGEXT 10

1. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regext/UaMNV11xh60ldjppqHHYc3TNsfhg>

14. Example of a Full Deposit

Example of a Full Deposit with the two example objects rdeObj1 and rdeObj2:

```
<?xml version="1.0" encoding="UTF-8"?>
<rde:deposit
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns:rdeObj1="urn:example:params:xml:ns:rdeObj1-1.0"
  xmlns:rdeObj2="urn:example:params:xml:ns:rdeObj2-1.0"
  type="FULL"
  id="20191018001">
  <rde:watermark>2019-10-17T23:59:59Z</rde:watermark>
  <rde:rdeMenu>
    <rde:version>1.0</rde:version>
    <rde:objURI>urn:example:params:xml:ns:rdeObj1-1.0</rde:objURI>
    <rde:objURI>urn:example:params:xml:ns:rdeObj2-1.0</rde:objURI>
  </rde:rdeMenu>
  <rde:contents>
    <rdeObj1:rdeObj1>
      <rdeObj1:name>EXAMPLE</rdeObj1:name>
    </rdeObj1:rdeObj1>
    <rdeObj2:rdeObj2>
      <rdeObj2:id>fsh8013-EXAMPLE</rdeObj2:id>
    </rdeObj2:rdeObj2>
  </rde:contents>
</rde:deposit>
```

15. Example of a Differential Deposit

Example of a Differential Deposit with the two example objects rdeObj1 and rdeObj2:

```
<?xml version="1.0" encoding="UTF-8"?>
<rde:deposit
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns:rdeObj1="urn:example:params:xml:ns:rdeObj1-1.0"
  xmlns:rdeObj2="urn:example:params:xml:ns:rdeObj2-1.0"
  type="DIFF"
  id="20191019001" prevId="20191018001">
  <rde:watermark>2019-10-18T23:59:59Z</rde:watermark>
  <rde:rdeMenu>
    <rde:version>1.0</rde:version>
    <rde:objURI>urn:example:params:xml:ns:rdeObj1-1.0</rde:objURI>
    <rde:objURI>urn:example:params:xml:ns:rdeObj2-1.0</rde:objURI>
  </rde:rdeMenu>
  <rde:contents>
    <rdeObj1:rdeObj1>
      <rdeObj1:name>EXAMPLE2</rdeObj1:name>
    </rdeObj1:rdeObj1>
    <rdeObj2:rdeObj2>
      <rdeObj2:id>sh8014-EXAMPLE</rdeObj2:id>
    </rdeObj2:rdeObj2>
  </rde:contents>
</rde:deposit>
```

16. Example of an Incremental Deposit

Example of an Incremental Deposit with the two example objects rdeObj1 and rdeObj2:

```
<?xml version="1.0" encoding="UTF-8"?>
<rde:deposit
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns:rdeObj1="urn:example:params:xml:ns:rdeObj1-1.0"
  xmlns:rdeObj2="urn:example:params:xml:ns:rdeObj2-1.0"
  type="INCR"
  id="20200317001" prevId="20200314001">
  <rde:watermark>2020-03-16T23:59:59Z</rde:watermark>
  <rde:rdeMenu>
    <rde:version>1.0</rde:version>
    <rde:objURI>urn:example:params:xml:ns:rdeObj1-1.0</rde:objURI>
    <rde:objURI>urn:example:params:xml:ns:rdeObj2-1.0</rde:objURI>
  </rde:rdeMenu>
  <rde:deletes>
    <rdeObj1:delete>
      <rdeObj1:name>EXAMPLE1</rdeObj1:name>
    </rdeObj1:delete>
    <rdeObj2:delete>
      <rdeObj2:id>fsh8013-EXAMPLE</rdeObj2:id>
    </rdeObj2:delete>
  </rde:deletes>
  <rde:contents>
    <rdeObj1:rdeObj1>
      <rdeObj1:name>EXAMPLE2</rdeObj1:name>
    </rdeObj1:rdeObj1>
    <rdeObj2:rdeObj2>
      <rdeObj2:id>sh8014-EXAMPLE</rdeObj2:id>
    </rdeObj2:rdeObj2>
  </rde:contents>
</rde:deposit>
```

17. References

17.1. 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>>.
- [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>>.

- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [W3C.REC-xml-20081126]
Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition) REC-xml-20081126", November 2008, <<https://www.w3.org/TR/2008/REC-xml-20081126/>>.
- [W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition REC-xmlschema-1-20041028", October 2004, <<https://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>>.
- [W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition REC-xmlschema-2-20041028", October 2004, <<https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>>.

17.2. Informative References

- [ICANN-GTLD-RA-20170731]
ICANN, "Base Registry Agreement 2017-07-31", July 2017, <<https://newgtlds.icann.org/sites/default/files/agreements/agreement-approved-31jul17-en.pdf>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [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>>.

Author's Address

Gustavo Lozano
Internet Corporation for Assigned Names and Numbers
12025 Waterfront Drive, Suite 300
Los Angeles 90292
United States of America

Phone: +1.310.823.9358
Email: gustavo.lozano@icann.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 19, 2021

G. Lozano
ICANN
J. Gould
C. Thippeswamy
VeriSign
Dec 16, 2020

Domain Name Registration Data (DNRD) Objects Mapping
draft-ietf-regext-dnrd-objects-mapping-11

Abstract

This document specifies the format, contents and semantics of Domain Name Registration Data (DNRD) Escrow deposits for a Domain Name Registry.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 19, 2021.

Copyright Notice

Copyright (c) 2020 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. Models	5
2.1. XML Model	5
2.2. CSV Model	6
3. Terminology	6
3.1. Glossary	6
4. Conventions Used in This Document	7
4.1. Date and Time	7
4.2. Country names	8
4.3. Telephone numbers	8
4.4. CSV Integrity Check	8
4.5. IP addresses	8
4.6. Conventions applicable to the CSV Model	8
5. Object Description	17
5.1. Domain Name Object	17
5.2. Host Object	36
5.3. Contact Object	46
5.4. Registrar Object	64
5.5. IDN Table Reference Object	72
5.6. NNDN Object	75
5.7. EPP Parameters Object	80
5.8. Policy Object	82
5.9. Header Object	82
5.10. DNRD Common Objects Collection	85
6. RDE IDN Variants handling	85
7. Profile	86
8. Data escrow agent extended verification process	86
9. Formal Syntax	87
9.1. RDE CSV Schema	87
9.2. RDE Domain Object	97
9.3. CSV Domain Object	100
9.4. RDE Host Object	103
9.5. CSV Host Object	105
9.6. RDE Contact Object	107
9.7. CSV Contact Object	110
9.8. RDE Registrar Object	116
9.9. CSV Registrar Object	119
9.10. RDE IDN Table Reference Objects	122
9.11. CSV IDN Language Object	123
9.12. EPP Parameters Object	124
9.13. NNDN Object	125
9.14. CSV NNDN Object	127
9.15. Policy Object	129
9.16. Header Object	130
9.17. DNRD Common Objects	132
10. Internationalization Considerations	132

11. IANA Considerations	132
12. Implementation Status	140
12.1. Implementation in the gTLD space	141
13. Security Considerations	141
14. Privacy Considerations	142
15. Acknowledgments	142
16. Change History	143
16.1. Changes from draft-arias-noguchi-registry-data-escrow-02 to -dnrd-objects-mapping-00	143
16.2. Changes from 00 to 01	143
16.3. Changes from 01 to 02	144
16.4. Changes from 02 to 03	144
16.5. Changes from 03 to 04	144
16.6. Changes from 04 to 05	145
16.7. Changes from 05 to 06	146
16.8. Changes from 06 to 07	146
16.9. Changes from 07 to 08	147
16.10. Changes from 08 to 09	147
16.11. Changes from 09 to 10	147
16.12. Changes from 10 to REGEXT 00	147
16.13. Changes REGEXT 00 to REGEXT 01	147
16.14. Changes REGEXT 01 to REGEXT 02	147
16.15. Changes REGEXT 02 to REGEXT 03	149
16.16. Changes REGEXT 03 to REGEXT 04	149
16.17. Changes REGEXT 04 to REGEXT 05	150
16.18. Changes REGEXT 05 to REGEXT 06	150
16.19. Changes REGEXT 06 to REGEXT 07	150
16.20. Changes REGEXT 07 to REGEXT 08	150
16.21. Changes REGEXT 08 to REGEXT 09	151
16.22. Changes REGEXT 09 to REGEXT 10	151
16.23. Changes REGEXT 10 to REGEXT 11	151
17. Example of a Full Deposit using the XML model	151
18. Example of Differential Deposit using the XML model	157
19. Example of a Full Deposit using the CSV model	158
20. Example of Differential Deposit using the CSV model	168
21. References	178
21.1. Normative References	178
21.2. Informative References	181
Authors' Addresses	182

1. Introduction

Registry Data Escrow (RDE) is the process by which a registry periodically submits data deposits to a third-party called an escrow agent. These deposits comprise the minimum data needed by a third-party to resume operations if the registry cannot function and is unable or unwilling to facilitate an orderly transfer of service. For example, for a domain name registry or registrar, the data to be

deposited would include all the objects related to registered domain names, e.g., names, contacts, name servers, etc.

The goal of data escrow is higher resiliency of registration services, for the benefit of Internet users. The beneficiaries of a registry are not just those registering information there, but also the users of services relying on the registry data.

In the context of domain name registries, registration data escrow is a requirement for generic top-level domains (e.g., Specification 2 of the ICANN Base Registry Agreement, see [ICANN-GTLD-RA-20170731]) and some country code top-level domain managers are also currently escrowing data. There is also a similar requirement for ICANN-accredited domain registrars.

This document defines the standard set of objects for a Domain Name Registry that uses the Registry Data Escrow Specification described in [I-D.ietf-regext-data-escrow] for escrow. The set of objects include:

- o Domain: Internet domain names that are typically provisioned in a Domain Name Registry using the EPP domain name mapping [RFC5731]. The attributes defined in the EPP domain name mapping [RFC5731] are fully supported by this document.
- o Host: Internet host names that are typically provisioned in a Domain Name Registry using the EPP host mapping [RFC5732]. The attributes defined in the EPP host mapping [RFC5732] are fully supported by this document.
- o Contact: Individual or organization social information provisioned in a Domain Name Registry using the EPP contact mapping [RFC5733]. The attributes defined in the EPP contact mapping [RFC5733] are fully supported by this document.
- o Registrar: The organization that sponsors objects like domains, hosts, and contacts in a Domain Name Registry.
- o NNDN (NNDN's not domain name): Domain Name Registries may maintain domain names without being persisted as domain objects in the registry system, for example, a list of reserved names not available for registration. The NNDN is a lightweight domain-like object that is used to escrow domain names not maintained as domain name objects.

This document defines the following pseudo-objects:

- o IDN Table Reference: Internationalized Domain Names (IDN) included in the Domain Object Data Escrow include references to the IDN Table and Policy used in IDN registration.
- o EPP parameters: Contains the EPP parameters supported by the Registry Operator.
- o Header: Used to specify counters of objects in the database at a certain point in time (watermark).
- o Policy: Used to specify OPTIONAL elements from this specification that are REQUIRED based on the business model of the registry.

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

2. Models

This document defines two different models that can be used to deposit data escrow objects: XML and CSV.

The data escrow deposit MAY contain a mix of both models but an object MUST be escrowed only in one model.

This document does not suggest the use of a particular model, and both are equivalent. A Domain Name Registry may choose the model that is more appropriate for the peculiarities of its systems. For example, a registry may use the CSV-export functionality of the Relational Database Management System (RDBMS) for escrow; therefore, the CSV model may be more appropriate. Another registry may use the code developed for EPP to implement escrow.

2.1. XML Model

XML: The XML model includes all the deposit information (meta-data and data) in an XML document. The definition of the XML format is fully defined in the XML schemas. As a convention, the objects represented using the XML model are referenced using RDE and an XML namespace that is prefixed with "rde". For example, the Domain Name object represented using the XML model can be referred to as the RDE Domain Name with the XML namespace including rdeDomain (urn:ietf:params:xml:ns:rdeDomain-1.0).

2.2. CSV Model

CSV: The CSV model uses XML to define the data escrow format of the data contained in referenced Comma-Separated Values (CSV) files. As a convention, the objects represented using the CSV model is referenced using CSV and an XML namespace that is prefixed with "csv". For example, the Domain Name object represented using the CSV model can be referred to as the CSV Domain Name with the XML namespace including csvDomain (urn:ietf:params:xml:ns:csvDomain-1.0).

3. 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.

3.1. Glossary

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

- o Allocated: a status of some label with respect to a zone, whereby the label is associated administratively to some entity that has requested the label. This term (and its cognates "allocation" and "to allocate") may represent the first step on the way to delegation in the DNS.
- o Comma-Separated Values (CSV), see [RFC4180].
- o Domain name: see definition of Domain name in [RFC8499].
- o Extensible Provisioning Protocol (EPP), see definition of the Extensible Provisioning Protocol in [RFC8499].
- o Fully-Qualified Domain Name (FQDN), see definition of FQDN in [RFC8499].
- o Internationalized Domain Name (IDN), see definition of Internationalized Domain Name in [RFC8499].
- o Label: see definition of Label in [RFC8499].
- o Registrant: see definition of Registrant in [RFC8499].
- o Registrar: see definition of Registrar in [RFC8499].

- o Registry: see definition of Registry in [RFC8499].
- o Registry-class domain name (RCDN): refers to a top-level domain (TLD) or any other domain name at any level in the DNS tree for which a Registry (either directly or through an affiliate company) provides Registry Services for other organizations or individuals. For example: .COM, .ORG, .BIZ, .CO.JP, .B.BR.
- o Registry Data Escrow (RDE): registry data escrow is the process by which a registry periodically submits data deposits to a third-party called an escrow agent. These deposits comprise the minimum data needed by a third-party to resume operations if the registry cannot function and is unable or unwilling to facilitate an orderly transfer of service.
- o Registry services: services offered by the Registry critical to the following tasks: the provisioning of domain names on receipt of requests and data from registrars; responding to registrar queries for status information relating to the DNS servers for the RCDN; dissemination of RCDN zone files; operation of the Registry DNS servers; responding to queries for contact and other information concerning DNS registrations in the RCDN; and any other products or services that only a Registry is capable of providing, by reason of its designation as the Registry. Typical examples of Registry Services are DNS resolution for the RCDN, WHOIS and EPP.
- o SRS: Shared Registration System, see also [ICANN-GTLD-AGB-20120604].
- o Top-Level Domain Name (TLD), see definition of Top-Level Domain in [RFC8499].
- o UTC: Coordinated Universal Time, as maintained by the Bureau International des Poids et Mesures (BIPM); see also [RFC3339].

4. Conventions Used in This Document

4.1. Date and Time

Numerous fields indicate "dates", such as the creation and expiry dates for domain names. These fields SHALL contain timestamps indicating the date and time in UTC as specified in [RFC3339], with no offset from the zero meridian.

4.2. Country names

Country identifiers SHALL be represented using two character identifiers as specified in [ISO-3166-1].

4.3. Telephone numbers

Telephone numbers (both voice and facsimile) SHALL be formatted based on structures defined in [ITU-E164]. Telephone numbers described in this specification are character strings that MUST begin with a plus sign ("+", ASCII value 0x2B), followed by a country code defined in [ITU-E164], followed by a dot (".", ASCII value 0x2E), followed by a sequence of digits representing the telephone number.

4.4. CSV Integrity Check

A checksum MAY be used to verify the integrity of the CSV files, for example, if another layer (i.e., encryption of an archive containing the deposit files) does not provide integrity. By default the CRC32 algorithm (see, 8.1.1.6.2 of [V42]) is used. A stronger algorithm, such as SHA-256 (see, [RFC6234]) MAY be used for enhanced security if required.

4.5. IP addresses

The syntax of IP addresses MUST conform to the text representation of either Internet Protocol Version 4 [RFC0791] or Internet Protocol Version 6 [RFC5952].

4.6. Conventions applicable to the CSV Model

4.6.1. CSV Parent Child Relationship

The CSV model represents a relational model, where the CSV files represent relational tables, the fields of the CSV files represent columns of the tables, and each line of the CSV file represents a record. As in a relational model, the CSV files can have relationships utilizing primary keys in the parent CSV file definitions and foreign keys in the child CSV file definitions for a 1-to-many relationship. The primary keys are not explicitly defined, but the foreign keys are using the boolean "parent" field attribute in the child CSV file. The relationships between the CSV files are used to support a cascade replace or cascade delete of records starting from the parent record in Differential and Incremental Deposits (see [I-D.ietf-regext-data-escrow]).

The following is an example of the CSV file definitions, using the element <rdeCsv:csv> (see Section 4.6.2.1), for a Sample object

consisting of a parent "sample" CSV File Definition and a child "sampleStatuses" CSV File Definition. The primary key for the Sample object is the field <csvSample:fName> that is used as the foreign key in the "sampleStatuses" CSV File Definition by specifying the "parent=true" attribute. If a Sample record is updated or deleted in a Differential or Incremental Deposit, it should cascade replace the data using the records included in the child "sampleStatuses" CSV File Definition or cascade delete the existing records in the child "sampleStatuses" CSV File Definition, respectively.

```
<csvSample:contents>
...
  <rdeCsv:csv name="sample" sep=",">
    <rdeCsv:fields>
      <csvSample:fName/>
      <rdeCsv:fClID/>
      <rdeCsv:fCrRr/>
      <rdeCsv:fCrID/>
      <rdeCsv:fCrDate/>
      <rdeCsv:fUpRr/>
      <rdeCsv:fUpID/>
      <rdeCsv:fUpDate/>
      <rdeCsv:fExDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="75E2D22F">
        sample-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
  <rdeCsv:csv name="sampleStatuses" sep=",">
    <rdeCsv:fields>
      <csvSample:fName parent="true"/>
      <csvSample:fStatus/>
      <rdeCsv:fStatusDescription/>
      <rdeCsv:fLang/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="EB9C558E">
        sampleStatuses-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvSample:contents>
```

4.6.2. CSV elements

4.6.2.1. <rdeCsv:csv> element

To support the CSV model, an element is defined for each object that substitutes for the <rde:content> element and for the <rde:delete> element, that contains one or more <rdeCsv:csv> elements. For example, the Domain Name Object (Section 5.1) defines the <csvDomain:contents> element, that substitutes for the <rde:content> element, and the <csvDomain:deletes> element, that substitutes for the <rde:delete> element. Both the <csvDomain:contents> element and the <csvDomain:deletes> elements contain one or more <rdeCsv:csv> elements. The <rdeCsv:csv> element has the following child elements:

<rdeCsv:fields> Ordered list of CSV fields used in the CSV files. There are one or more child elements that substitute for the <rdeCsv:field> abstract element. Each element defines the format of the CSV field contained in the CSV files. The <rdeCsv:field> elements support the "type" attribute that defines the XML simple data type of the field element. The <rdeCsv:field> elements support the "isRequired" attribute, with a default value of "false", when set to "true" indicates that the field must be non-empty in the CSV files and when set to "false" indicates that the field MAY be empty in the CSV files. The "isRequired" attribute MAY be specifically set for the field elements within the XML schema and MAY be overridden when specifying the fields under the <rdeCsv:fields> element. The <rdeCsv:field> element supports an OPTIONAL "parent" attribute that identifies the field as a reference to a parent object, as defined in CSV Parent Child Relationship (Section 4.6.1). For example, the <rdeCsv:csv name="domainStatuses"> <csvDomain:fName> field SHOULD set the "parent" attribute to "true" to identify it as the parent domain name of the domain status.

<rdeCsv:files> A list of one or more CSV files using the <rdeCsv:file> child element. The <rdeCsv:file> child element defines a reference to the CSV file name and has the following optional attributes:

compression If the CSV file is compressed, the "compression" attribute defines the compression format. For example, setting this attribute to "gzip" signals that the CSV file is compressed using the GZIP file format (see, [RFC1952]). The supported compression formats are negotiated out-of-band.

encoding Defines the encoding of the CSV file with the default encoding of "UTF-8".

cksum Defines the checksum of the CSV file, as described in Section 4.4, using the algorithm defined by the "cksumAlg" attribute. If the "cksumAlg" attribute is not present, the checksum is calculated using "CRC32".

cksumAlg Defines the checksum algorithm used to calculate the "cksum" attribute, with the default value of "CRC32". If the value "SHA256" is specified, the SHA-256 algorithm (see, [RFC6234]) MUST be used to calculate the "cksum" attribute. Parties receiving and processing data escrow deposits MUST support CRC32 and SHA-256. If this attribute is present, the "cksum" attribute MUST also be present. Additional checksum algorithms are negotiated out-of-band.

The <rdeCsv:csv> element requires a "name" attribute that defines the purpose of the CSV file with values like "domain", "host", "contact". The supported "name" attribute values are defined for each object type. The OPTIONAL "sep" attribute defines the CSV separator character with the default separator character of ",". The need for quoting/escaping of the CSV data could be avoided by choosing a separator character that is not in the data set of the CSV files.

The following is an example of the <csvDomain:contents> <rdeCsv:csv> element for domain name records where the <rdeCsv:fRegistrant> is set as required with isRequired="true".

```
<csvDomain:contents>
...
  <rdeCsv:csv name="domain" sep=",">
    <rdeCsv:fields>
      <csvDomain:fName/>
      <rdeCsv:fRoid/>
      <rdeCsv:fIdnTableId/>
      <csvDomain:fOriginalName/>
      <rdeCsv:fRegistrant isRequired="true"/>
      <rdeCsv:fCLID/>
      <rdeCsv:fCrRr/>
      <rdeCsv:fCrID/>
      <rdeCsv:fCrDate/>
      <rdeCsv:fUpRr/>
      <rdeCsv:fUpID/>
      <rdeCsv:fUpDate/>
      <rdeCsv:fExDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="75E2D01F">
        domain-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvDomain:contents>
```

The following is example of the "domain-YYYYMMDD.csv" file with one record matching the <rdeCsv:fields> definition.

```
domain1.example,Ddomain2-TEST,,,registrantid,registrarX,registrarX,
clientY,2009-04-03T22:00:00.0Z,registrarX,clientY,
2009-12-03T09:05:00.0Z,2025-04-03T22:00:00.0Z
```

The following is an example of the `<csvDomain:deletes>` `<rdeCsv:csv>` element for domain name records.

```
<csvDomain:deletes>
...
  <rdeCsv:csv name="domain">
    <rdeCsv:fields>
      <csvDomain:fName/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="6F2B988F">
        domain-delete-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvDomain:deletes>
```

The following is example of the "domain-delete-YYYYMMDD.csv" file with three records that matches the single `<csvDomain:fName>` field.

```
domain1.example
domain2.example
domainN.example
```

4.6.2.2. CSV common field elements

The `<rdeCsv:fields>` element defined in the `<rdeCsv:csv>` element (Section 4.6.2.1) section has child elements that substitute for the abstract `<rdeCsv:field>` element. By convention `<rdeCsv:field>` elements include an 'f' prefix to identify them as field definition elements. There are a set of common field elements that are used across multiple data escrow objects. The common field elements are defined using the "urn:ietf:params:xml:ns:rdeCsv-1.0" namespace and using the "rdeCsv" sample namespace prefix. The CSV common field elements include:

```
<rdeCsv:fUName> UTF-8 encoded name field with
  type="eppcom:labelType".
```

```
<rdeCsv:fRoid> Repository Object Identifier (ROID) field with
  type="eppcom:roidType" and isRequired="true".
```

```
<rdeCsv:fRegistrant> Registrant contact identifier with
  type="eppcom:clIDType".
```

<rdeCsv:fStatusDescription> The object status description, which is free form text describing the rationale for the status, with type="normalizedString".

<rdeCsv:fClID> Identifier of the client (registrar) that sponsors the object with type="eppcom:clIDType" and isRequired="true".

<rdeCsv:fCrRr> Identifier of the registrar, defined in Section 5.4, of the client that created the object with type="eppcom:clIDType".

<rdeCsv:fCrID> Identifier of the client that created the object with type="eppcom:clIDType".

<rdeCsv:fUpRr> Identifier of the registrar, defined in Section 5.4, of the client that last updated the object with type="eppcom:clIDType".

<rdeCsv:fUpID> Identifier of the client that last updated the object with type="eppcom:clIDType".

<rdeCsv:fReRr> Identifier of the registrar, defined in Section 5.4, of the client that requested the transfer with type="eppcom:clIDType" and isRequired="true".

<rdeCsv:fReID> Identifier of the client that requested the transfer with type="eppcom:clIDType".

<rdeCsv:fAcRr> Identifier of the registrar, defined in Section 5.4, of the client that should take or took action with type="eppcom:clIDType" and isRequired="true".

<rdeCsv:fAcID> Identifier of the client that should take or took action for transfer with type="eppcom:clIDType".

<rdeCsv:fCrDate> Created date of object with type="dateTime".

<rdeCsv:fUpDate> Updated date of object with type="dateTime".

<rdeCsv:fExDate> Expiration date of object with type="dateTime".

<rdeCsv:fReDate> Date that transfer was requested with type="dateTime" and isRequired="true".

<rdeCsv:fAcDate> Date that transfer action should be taken or has been taken with type="dateTime" and isRequired="true".

<rdeCsv:fTrDate> Date of last transfer with type="dateTime".

`<rdeCsv:fTrStatus>` State of the most recent transfer request with `type="eppcom:trStatusType"` and `isRequired="true"`.

`<rdeCsv:fTokenType>` General token field with `type="token"`.

`<rdeCsv:fLang>` General language field with `type="language"`.

`<rdeCsv:fIdnTableId>` IDN Table Identifier used for IDN domain names with `type="token"`.

`<rdeCsv:fPositiveIntegerType>` General positive integer field with `type="positiveInteger"`.

`<rdeCsv:fUrl>` Contains the URL of an object like a registrar object with `type="anyURI"`.

`<rdeCsv:fCustom>` Custom field with name attribute that defines the custom field name" with `type="token"`.

4.6.3. Internationalized and Localized Elements

Some elements MAY be provided in either internationalized form ("int") or localized form ("loc"). Those elements use a field value or "isLoc" attribute to specify the form used. If an "isLoc" attribute is used, a value of "true" indicates the use of the localized form and a value of "false" indicates the use of the internationalized form. This MAY override the form specified for a parent element. A value of "int" is used to indicate the internationalized form and a value of "loc" is used to indicate the localized form. When the internationalized form ("int") is provided, the field value MUST be represented in a subset of UTF-8 that can be represented in the 7-bit US-ASCII character set. When the localized form ("loc") is provided, the field value MAY be represented in unrestricted UTF-8.

The field elements below of the "registrar" <rdeCsv:csv">
<rdeCsv:fields> element specify the internationalized form with the
isLoc="false" attribute.

```
...
<csvRegistrar:contents>
...
  <rdeCsv:csv name="registrar" sep=",">
    <rdeCsv:fields>
      <csvRegistrar:fId/>
      <rdeCsv:fRoid/>
      <csvRegistrar:fName isLoc="false"/>
      <csvRegistrar:fGurid/>
      <csvRegistrar:fStatus/>
      <csvContact:fStreet isLoc="false" index="0"/>
      <csvContact:fStreet isLoc="false" index="1"/>
      <csvContact:fStreet isLoc="false" index="2"/>
      <csvContact:fCity isLoc="false" />
      <csvContact:fSp isLoc="false" />
      <csvContact:fPc isLoc="false" />
      <csvContact:fCc isLoc="false" />
      <csvContact:fVoice/>
      <csvContact:fVoiceExt/>
      <csvContact:fFax/>
      <csvContact:fFaxExt/>
      <csvContact:fEmail isRequired="false"/>
      <rdeCsv:fUrl/>
      <csvRegistrar:fWhoisUrl/>
      <rdeCsv:fCrRr/>
      <rdeCsv:fCrID/>
      <rdeCsv:fCrDate/>
      <rdeCsv:fUpRr/>
      <rdeCsv:fUpID/>
      <rdeCsv:fUpDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="306178BB">
        registrar-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvRegistrar:contents>
...
```


The following is an example of using the <csvContact:fPostalType> field value to define the internationalized or localized form of the remainder of the "contactPostal" field values.

```
...
<csvContact:contents>
...
  <rdeCsv:csv name="contactPostal">
    <rdeCsv:fields>
      <csvContact:fId parent="true"/>
      <csvContact:fPostalType/>
      <csvContact:fName/>
      <csvContact:fOrg/>
      <csvContact:fStreet index="0"/>
      <csvContact:fStreet index="1"/>
      <csvContact:fStreet index="2"/>
      <csvContact:fCity/>
      <csvContact:fSp/>
      <csvContact:fPc/>
      <csvContact:fCc/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="02CC2504">
        contactPostal-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvContact:contents>
...
```

5. Object Description

This section describes the base objects supported by this specification:

5.1. Domain Name Object

The domain name object is based on the EPP domain name mapping specified in [RFC5731]. The domain name object supports both the XML Model and the CSV Model, defined in the Models (Section 2) section. The elements used for both models are defined in the following sections.

5.1.1. XML Model

There are two elements used in the data escrow of the domain name objects for the XML model including the `<rdeDomain:domain>`, under the `<rde:contents>` element, and the `<rdeDomain:delete>` element, under the `<rde:deletes>` element.

5.1.1.1. `<rdeDomain:domain>` object

The domain element is based on the EPP domain `<info>` response for an authorized client (see Section 3.1.2. of [RFC5731]) with additional data from an EPP `<transfer>` Query Response, see Section 3.1.3. of [RFC5731], Registry Grace Period (RGP) status from [RFC3915], and data from the EPP `<secDns:create>` command, see Section 5.2.1. of [RFC5910].

A `<domain>` element substitutes for the `<abstractDomain>` abstract element to define a concrete definition of a domain. The `<abstractDomain>` element can be replaced by other domain definitions using the XML schema substitution groups feature.

The `<domain>` element contains the following child elements:

- o A `<name>` element that contains the fully-qualified name of the domain name object. For IDNs the A-Label is used (see [RFC5891], Section 4.4).
- o A `<roid>` element that contains the repository object identifier assigned to the domain name object when it was created.
- o An OPTIONAL `<uName>` element that contains the fully-qualified domain name in Unicode character set. It MUST be provided if available.
- o An OPTIONAL `<idnTableId>` element that references the IDN Table used for the IDN. This corresponds to the "id" attribute of the `<idnTableRef>` element. This element MUST be present if the domain name is an IDN.
- o An OPTIONAL `<originalName>` element is used to indicate that the domain name is an IDN variant. This element contains the domain name used to generate the IDN variant.
- o One or more `<status>` elements that contain the current status descriptors associated with the domain name.
- o Zero or more OPTIONAL `<rgpStatus>` elements to represent "pendingDelete" sub-statuses, including "redemptionPeriod",

"pendingRestore", and "pendingDelete", that a domain name can be in as a result of grace period processing as specified in [RFC3915].

- o An OPTIONAL <registrant> element that contains the identifier for the human or organizational social information object associated as the holder of the domain name object.
- o Zero or more OPTIONAL <contact> elements that contain identifiers for the human or organizational social information objects associated with the domain name object.
- o An OPTIONAL <ns> element that contains the fully-qualified names of the delegated host objects or host attributes (name servers) associated with the domain name object. See Section 1.1 of [RFC5731] for a description of the elements used to specify host objects or host attributes.
- o A <clID> element that contains the identifier of the sponsoring registrar.
- o An OPTIONAL <crRr> element that contains the identifier of the registrar that created the domain name object. An OPTIONAL client attribute is used to specify the client that performed the operation.
- o An OPTIONAL <crDate> element that contains the date and time of the domain name object creation. This element MUST be present if the domain name has been allocated.
- o An OPTIONAL <exDate> element that contains the date and time identifying the end (expiration) of the domain name object's registration period. This element MUST be present if the domain name has been allocated.
- o An OPTIONAL <upRr> element that contains the identifier of the registrar that last updated the domain name object. This element MUST NOT be present if the domain has never been modified. An OPTIONAL client attribute is used to specify the client that performed the operation.
- o An OPTIONAL <upDate> element that contains the date and time of the most recent domain-name-object modification. This element MUST NOT be present if the domain name object has never been modified.

- o An OPTIONAL <secDNS> element that contains the public key information associated with Domain Name System security (DNSSEC) extensions for the domain name as specified in [RFC5910].
- o An OPTIONAL <trDate> element that contains the date and time of the most recent domain name object successful transfer. This element MUST NOT be present if the domain name object has never been transferred.
- o An OPTIONAL <trnData> element that contains the following child elements related to the last transfer request of the domain name object. This element MUST NOT be present if a transfer request for the domain name has never been created.
- * A <trStatus> element that contains the state of the most recent transfer request.
- * A <reRr> element that contains the identifier of the registrar that requested the domain name object transfer. An OPTIONAL client attribute is used to specify the client that performed the operation.
- * A <reDate> element that contains the date and time that the transfer was requested.
- * An <acRr> element that contains the identifier of the registrar that should act upon a PENDING transfer request. For all other status types, the value identifies the registrar that took the indicated action. An OPTIONAL client attribute is used to specify the client that performed the operation.
- * An <acDate> element that contains the date and time of a required or completed response. For a PENDING request, the value identifies the date and time by which a response is required before an automated response action will be taken by the registry. For all other status types, the value identifies the date and time when the request was completed.
- * An OPTIONAL <exDate> element that contains the end of the domain name object's validity period (expiry date) if the transfer caused or causes a change in the validity period.

Example of a domain name object:

```
...
<rdeDomain:domain>
  <rdeDomain:name>xn--exempl-gva.example</rdeDomain:name>
  <rdeDomain:roid>Dexample1-TEST</rdeDomain:roid>
  <rdeDomain:idnTableId>pt-BR</rdeDomain:idnTableId>
  <rdeDomain:originalName>example.example</rdeDomain:originalName>
  <rdeDomain:status s="ok"/>
  <rdeDomain:registrant>jdl234</rdeDomain:registrant>
  <rdeDomain:contact type="admin">sh8013</rdeDomain:contact>
  <rdeDomain:contact type="tech">sh8013</rdeDomain:contact>
  <rdeDomain:ns>
    <domain:hostObj>ns1.example.com</domain:hostObj>
    <domain:hostObj>ns1.example1.example</domain:hostObj>
  </rdeDomain:ns>
  <rdeDomain:clID>RegistrarX</rdeDomain:clID>
  <rdeDomain:crRr client="jdoe">RegistrarX</rdeDomain:crRr>
  <rdeDomain:crDate>1999-04-03T22:00:00.0Z</rdeDomain:crDate>
  <rdeDomain:exDate>2025-04-03T22:00:00.0Z</rdeDomain:exDate>
</rdeDomain:domain>
...
```

5.1.1.2. <rdeDomain:delete> object

The <rdeDomain:delete> element contains the fully-qualified domain name that was deleted and purged.

Example of <rdeDomain:delete> object:

```
...
<rde:deletes>
  ...
  <rdeDomain:delete>
    <rdeDomain:name>foo.example</rdeDomain:name>
    <rdeDomain:name>bar.example</rdeDomain:name>
  </rdeDomain:delete>
  ...
</rde:deletes>
...
```

5.1.2. CSV Model

For the CSV Model of the domain name object, the <csvDomain:contents> child element of the <rde:contents> element is used to hold the new or updated domain name objects for the deposit. The <csvDomain:deletes> child element of the <rde:deletes> element is used to hold the deleted or purged domain name objects for the

deposit. Both the `<csvDomain:contents>` and `<csvDomain:deletes>` elements contain one or more `<rdeCsv:csv>` elements with a set of named CSV file definitions using the `<rdeCsv:csv>` "name" attribute.

Differential and Incremental Deposits are based on changes to the domain name objects. The updated domain name object data under the `<csvDomain:contents>` element is a cascade replace down all of the domain name CSV files starting with the parent "domain" CSV File Definition (Section 5.1.2.1.1). The child CSV file definitions include a `<csvDomain:fName parent="true">` field. All the child CSV file definition data for the domain name objects in the parent "domain" CSV File Definition (Section 5.1.2.1.1) MUST first be deleted and then set using the data in the child CSV files. The deleted domain name object data under the `<csvDomain:deletes>` element is a cascade delete starting from the "domain" Deletes CSV File Definition (Section 5.1.2.2.1).

5.1.2.1. `<csvDomain:contents>`

The `<csvDomain:contents>` is used to hold the new or updated domain name object information for the deposit. The `<csvDomain:contents>` is split into separate CSV file definitions using named `<rdeCsv:csv>` elements with the "name" attribute. The following sections include the supported domain name CSV file definitions:

5.1.2.1.1. "domain" CSV File Definition

The "domain" CSV File Definition defines the fields and CSV file references used for the parent domain name object records. All the other domain name CSV file definitions are child CSV files based on the inclusion of the `<csvDomain:fName parent="true">` field.

The following "csvDomain" field elements MUST be used in the "domain" `<rdeCsv:csv>` `<rdeCsv:fields>` element:

`<csvDomain:fName>` Domain name field with `type="eppcom:labelType"` and `isRequired="true"`.

The following "csvDomain" field elements MAY be used in the "domain" `<rdeCsv:csv>` `<rdeCsv:fields>` element:

`<csvDomain:fOriginalName>` Fully-qualified name of the original IDN domain name object related to the variant domain name object with `type="eppcom:labelType"`.

The following "rdeCsv" and "csvRegistrar" fields, MUST be used in the "domain" `<rdeCsv:csv>` `<rdeCsv:fields>` element:

<rdeCsv:fROID> Registry Object Identifier (ROID) for the domain name object with `isRequired="true"`.

<rdeCsv:fClID> or <csvRegistrar:fGurid> A choice of:

<rdeCsv:fClID> Identifier of the sponsoring client with `isRequired="true"`.

<csvRegistrar:fGurid> Contains the Globally Unique Registrar Identifier (GURID) assigned by ICANN with `type="positiveInteger"` and `isRequired="true"`.

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MAY be used in the "domain" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fCrRr> Identifier of the registrar, defined in Section 5.4, of the client that created the domain name object.

<rdeCsv:fCrID> Identifier of the client that created the domain name object.

<rdeCsv:fUpRr> Identifier of the registrar, defined in Section 5.4, of the client that last updated the domain name object.

<rdeCsv:fUpID> Identifier of the client that last updated the domain name object.

<rdeCsv:fUName> UTF8 encoded domain name for the <csvDomain:fName> field element.

<rdeCsv:fIdnTableId> IDN Table Identifier used for the IDN domain name object that MUST match a <rdeCsv:fIdnTableId> field element in the "idnLanguage" CSV files, as defined in Section 5.5.2.

<rdeCsv:fRegistrant> Registrant contact identifier for the domain name object.

<rdeCsv:fCrDate> Created date and time of the domain name object.

<rdeCsv:fUpDate> Date and time of the last update to the domain name object. This field MUST NOT be set if the domain name object has never been modified.

<rdeCsv:fExDate> Expiration date and time for the domain name object.

<rdeCsv:fTrDate> Date and time of the last transfer for the domain name object. This field MUST NOT be set if the domain name object has never been transferred.

Example of a "domain" <csvDomain:contents> <rdeCsv:csv> element.

```
...
<csvDomain:contents>
...
  <rdeCsv:csv name="domain">
    <rdeCsv:fields>
      <csvDomain:fName/>
      <rdeCsv:fRoid/>
      <rdeCsv:fIdnTableId/>
      <csvDomain:fOriginalName/>
      <rdeCsv:fRegistrant/>
      <rdeCsv:fClID/>
      <rdeCsv:fCrRr/>
      <rdeCsv:fCrID/>
      <rdeCsv:fCrDate/>
      <rdeCsv:fUpRr/>
      <rdeCsv:fUpID/>
      <rdeCsv:fUpDate/>
      <rdeCsv:fExDate isRequired="true"/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="5E403BD6">
        domain-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvDomain:contents>
...
```


Example of the corresponding domain-YYYYMMDD.csv file. The file contains four records (two active ASCII domains, original IDN with LANG-1 language rules, and variant IDN with LANG-1 language rules).

```
domain1.example,Ddomain1-TEST,,,registrantid,registrarX,registrarX,
clientY,2009-04-03T22:00:00.0Z,registrarX,clientY,
2009-12-03T09:05:00.0Z,2025-04-03T22:00:00.0Z
domain2.example,Ddomain2-TEST,,,registrantid,registrarX,registrarX,
clientY,1999-04-03T22:00:00.0Z,registrarX,clientY,
2009-12-03T09:05:00.0Z,2025-04-03T22:00:00.0Z
xn--bc123-3ve.example,Dxnabc123-TEST,LANG-1,,registrantid,registrarX,
registrarX,clientY,2009-04-03T22:00:00.0Z,registrarX,clientY,
2009-12-03T09:05:00.0Z,2025-04-03T22:00:00.0Z
xn--bc321-3ve.example,Dxnabc321-TEST,LANG-1,xn--bc123-3ve.example,
registrantid,registrarX,registrarX,clientY,2009-04-03T22:00:00.0Z,
registrarX,clientY,2009-12-03T09:05:00.0Z,2025-04-03T22:00:00.0Z
```

5.1.2.1.2. "domainContacts" CSV File Definition

The "domainContacts" CSV File Definition defines the fields and CSV file references used for the domain name object link records to contact objects, as described in Contact Object (Section 5.3).

The following "csvDomain" field elements, defined for the "domain" CSV File Definition (Section 5.1.2.1.1), MUST be used in the "domainContacts" <rdeCsv:csv> <rdeCsv:fields> element:

<csvDomain:fName> The name of the domain object that is linked to the contact object with isRequired="true".

<csvDomain:fContactType> The contact type for the contact object link with type="domain:contactAttrType" and isRequired="true". The supported contact type values include "admin" for the administration contact, "billing" for the billing contact, and "tech" for the technical contact.

The following "csvContact" fields, defined for the "contact" CSV File Definition (Section 5.3.2.1.1), MUST be used in the "domainContacts" <rdeCsv:csv> <rdeCsv:fields> element:

<csvContact:fId> The server-unique contact identifier with isRequired="true".

Example of a "domainContacts" <csvDomain:contents> <rdeCsv:csv> element.

```
...
<csvDomain:contents>
...
  <rdeCsv:csv name="domainContacts">
    <rdeCsv:fields>
      <csvDomain:fName parent="true"/>
      <csvContact:fId/>
      <csvDomain:fContactType/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="6B976A6C">
        domainContacts-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvDomain:contents>
...
```

Example of the corresponding domainContacts-YYYYMMDD.csv file. The file contains an admin, tech, and billing contact for the four domain names domain1.example, domain2.example, xn--bc123-3ve.example and xn--bc321-3ve.example.

```
domain1.example,domain1admin,admin
domain1.example,domain1tech,tech
domain1.example,domain1billing,billing
domain2.example,domain2admin,admin
domain2.example,domain2tech,tech
domain2.example,domain2billing,billing
xn--bc123-3ve.example,xnabc123admin,admin
xn--bc123-3ve.example,xnabc123tech,tech
xn--bc123-3ve.example,xnabc123billing,billing
xn--bc321-3ve.example,xnabc123admin,admin
xn--bc321-3ve.example,xnabc123tech,tech
xn--bc321-3ve.example,xnabc123billing,billing
```

5.1.2.1.3. "domainStatuses" CSV File Definition

The "domainStatuses" CSV File Definition defines the fields and CSV file references used for the domain name object statuses.

The following "csvDomain" fields, defined for the "domain" CSV File Definition (Section 5.1.2.1.1), MUST be used in the "domainStatuses" <rdeCsv:csv> <rdeCsv:fields> element:

<csvDomain:fName> Domain name of status with isRequired="true".

<csvDomain:fStatus> The status of the domain name with type="domain:statusValueType" and isRequired="true".

<csvDomain:fRgpStatus> The RGP status, as a sub-status of the <csvDomain:fStatus> "pendingDelete" status value, with type="rgp:statusValueType" as defined in [RFC3915].

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MAY be used in the "domainStatuses" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fStatusDescription> Domain name object status description which is free form text describing the rationale for the status.

<rdeCsv:fLang> Language of the <rdeCsv:fStatusDescription> field.

Example of a "domainStatuses" <csvDomain:contents> <rdeCsv:csv> element.

```
...
<csvDomain:contents>
...
  <rdeCsv:csv name="domainStatuses">
    <rdeCsv:fields>
      <csvDomain:fName parent="true"/>
      <csvDomain:fStatus/>
      <rdeCsv:fStatusDescription/>
      <rdeCsv:fLang/>
      <csvDomain:fRgpStatus/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="98D139A3">
        domainStatuses-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvDomain:contents>
...
```

Example of the corresponding domainStatuses-YYYYMMDD.csv file. The file contains the statuses for the four domain names domain1.example, domain2.example, xn--bc123-3ve.example and xn--bc321-3ve.example.

```
domain1.example,clientUpdateProhibited,"Disallow update",
en,
domain1.example,clientDeleteProhibited,"Disallow delete",
en,
domain2.example,ok,,,
xn--bc123-3ve.example,ok,,,
xn--bc321-3ve.example,ok,,,
```

5.1.2.1.4. "domainNameServers" CSV File Definition

The "domainNameServers" CSV File Definition defines the fields and CSV file references used for the domain name delegated hosts (name servers). The "domainNameServers" CSV files define the relationship between a domain name object and a delegated host. The "domainNameServers" CSV File is used to support the <domain:hostObj> model, defined in [RFC5731].

The following "csvDomain" fields, defined for the "domain" CSV File Definition (Section 5.1.2.1.1), MUST be used in the "domainNameServers" <rdeCsv:csv> <rdeCsv:fields> element:

<csvDomain:fName> Domain name using the delegated host with isRequired="true".

The following "csvHost" and "rdeCsv" field elements MUST be used in the "domainNameServers" <rdeCsv:csv> <rdeCsv:fields> element:

<csvHost:fName> or <rdeCsv:fROID> A choice of:

<csvHost:fName> Host name field with type="eppcom:labelType" and isRequired="true".

<rdeCsv:fROID> Host object Registry Object Identifier (ROID) assigned to the host object with isRequired="true".

Example of a "domainNameServers" <csvDomain:contents> <rdeCsv:csv> element.

```
...
<csvDomain:contents>
...
  <rdeCsv:csv name="domainNameServers">
    <rdeCsv:fields>
      <csvDomain:fName parent="true"/>
      <rdeCsv:fRoid/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="8FE6E9E1">
        domainNameServers-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvDomain:contents>
...
```

Example of the corresponding domainNameServers-YYYYMMDD.csv file. The file contains the delegated hosts (name servers) for the four domain names domain1.example, domain2.example, xn--bc123-3ve.example and xn--bc321-3ve.example referenced via the <rdeCsv:fRoid> field element.

```
domain1.example,Hns1_domain1_test-TEST
domain1.example,Hns2_domain1_test-TEST
domain2.example,Hns1_domain2_test-TEST
domain2.example,Hns2_domain2_test-TEST
xn--bc123-3ve.example,Hns1_example_test-TEST
xn--bc123-3ve.example,Hns2_example_test-TEST
xn--bc321-3ve.example,Hns1_example_test-TEST
xn--bc321-3ve.example,Hns2_example_test-TEST
```

5.1.2.1.5. "domainNameServersAddresses" CSV File Definition

The "domainNameServersAddresses" CSV File Definition defines the fields and CSV file references used for supporting the domain host attributes model.

The following "csvDomain" fields, defined for the "domain" CSV File Definition (Section 5.1.2.1.1), MUST be used in the "domainNameServersAddresses" <rdeCsv:csv> <rdeCsv:fields> element:

<csvDomain:fName> Domain name using the delegated host with host
<csvHost:fName> and isRequired="true".

The following "rdeCsv" fields, defined in section Host CSV model
elements (Section 5.2.2), MUST be used in the
"domainNameServersAddresses" <rdeCsv:csv> <rdeCsv:fields> element:

<csvHost:fName> Host name field with type="eppcom:labelType" and
isRequired="true".

The following "csvHost" fields, defined in section Host CSV model
elements (Section 5.2.2), MAY be used in the
"domainNameServersAddresses" <rdeCsv:csv> <rdeCsv:fields> element:

<csvHost:fAddr> IP addresses associated with the host object with
type="host:addrStringType".

<csvHost:fAddrVersion> IP addresses version associated with the host
object with type="host:ipType". "host:ipType" has the enumerated
values of "v4" or "v6".

Example of a "domainNameServersAddresses" <csvDomain:contents>
<rdeCsv:csv> element.

```
...
<csvDomain:contents>
...
  <rdeCsv:csv name="domainNameServersAddresses">
    <rdeCsv:fields>
      <csvDomain:fName parent="true"/>
      <csvHost:fName/>
      <csvHost:fAddr/>
      <csvHost:fAddrVersion/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="D3B77438">
        domainNameServersAddresses-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvDomain:contents>
...
```

Example of the corresponding domainNameServersAddresses-YYYYMMDD.csv file. The file contains the delegated hosts (name servers) for the four domain names domain1.example, domain2.example, xn--bc123-3ve.example and xn--bc321-3ve.example.

```
domain1.example,ns1.domain1.example,192.0.2.1,v4
domain1.example,ns2.domain1.example,2001:DB8::1,v6
domain2.example,ns1.example.net,,
domain2.example,ns2.example.net,,
xn--bc123-3ve.example,ns1.example.net,,
xn--bc123-3ve.example,ns2.example.net,,
xn--bc321-3ve.example,ns1.example.net,,
xn--bc321-3ve.example,ns2.example.net,,
```

5.1.2.1.6. "dnssec" CSV File Definition

The "dnssec" CSV File Definition defines the fields and CSV file references used for the domain name object DNSSEC records (DS or Key Data).

The following "csvDomain" field elements MUST be used in the "dnssec" <rdeCsv:csv> <rdeCsv:fields> element when the DS Data Interface per [RFC5910] is used:

<csvDomain:fKeyTag> Contains the DS key tag value per [RFC5910] with type="unsignedShort" and isRequired="true".

<csvDomain:fDsAlg> Contains the DS algorithm value per [RFC5910] with type="unsignedByte" and isRequired="true".

<csvDomain:fDigestType> Contains the DS digest type value per [RFC5910] with type="unsignedByte" and isRequired="true".

<csvDomain:fDigest> Contains the DS digest value per [RFC5910] with type="hexBinary" and isRequired="true".

The following "csvDomain" field elements MUST be used in the "dnssec" <rdeCsv:csv> <rdeCsv:fields> element when the Key Data Interface per [RFC5910] is used and MAY be used in the "dnssec" <rdeCsv:csv> <rdeCsv:fields> element when the DS Data Interface per [RFC5910] is used:

<csvDomain:fFlags> Contains the flags field value per [RFC5910] with type="unsignedShort" and isRequired="true".

<csvDomain:fProtocol> Contains the Key protocol value per [RFC5910] with type="unsignedByte" and isRequired="true".

<csvDomain:fKeyAlg> Contains the Key algorithm value per [RFC5910] with type="unsignedByte" and isRequired="true".

<csvDomain:fPubKey> Contains the public key value per [RFC5910] with type="secDNS:keyType" and isRequired="true".

The following "csvDomain" field elements MAY be used in the "dnssec" <rdeCsv:csv> <rdeCsv:fields> element:

<csvDomain:fMaxSigLife> Indicates a child's preference for the number of seconds after signature generation when the parent's signature on the DS information provided by the child will expire with type="secDNS:maxSigLifeType" defined in [RFC5910].

The following "domain" fields, defined for the "domain" CSV File Definition (Section 5.1.2.1.1), MUST be used in the "dnssec" <rdeCsv:csv> <rdeCsv:fields> element:

<csvDomain:fName> Domain name of the domain name object associated with the DNSSEC record and isRequired="true".

Example of a "dnssec" <csvDomain:contents> <rdeCsv:csv> element with the DS Data Interface of [RFC5910]:

```
<csvDomain:contents>
...
  <rdeCsv:csv name="dnssec">
    <rdeCsv:fields>
      <csvDomain:fName parent="true"/>
      <csvDomain:fMaxSigLife/>
      <csvDomain:fKeyTag/>
      <csvDomain:fDsAlg/>
      <csvDomain:fDigestType/>
      <csvDomain:fDigest/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="10ED6C42">
        dnssec-ds-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvDomain:contents>
...
```


Example of the corresponding dnssec-ds-YYYYMMDD.csv file. The file contains two DS records for domain1.example.

```
domain1.example,604800,30730,8,2,91C9B176EB/////F1C46F6A55
domain1.example,604800,61882,8,2,9F8FEAC94B/////1272AF09F3
```

Example of a "dnssec" <csvDomain:contents> <rdeCsv:csv> element with the Key Data Interface of [RFC5910]:

```
<csvDomain:contents>
...
  <rdeCsv:csv name="dnssec">
    <rdeCsv:fields>
      <csvDomain:fName parent="true"/>
      <csvDomain:fMaxSigLife/>
      <csvDomain:fFlags/>
      <csvDomain:fProtocol/>
      <csvDomain:fKeyAlg/>
      <csvDomain:fPubKey/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="183C3F79">
        dnssec-key-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvDomain:contents>
...
```

Example of the corresponding dnssec-key-YYYYMMDD.csv file. The file contains two key records for domain1.example.

```
domain1.example,604800,257,3,8,AwEAAZD1+z////G1jqviK8c=
domain1.example,604800,257,3,8,AwEAAbntWP////vwDitt940=
```

5.1.2.1.7. "domainTransfer" CSV File Definition

The "domainTransfer" CSV File Definition defines the fields and CSV file references used for the domain name object pending and completed transfer records. No additional field elements were added for use in the "domainTransfer" <rdeCsv:csv> <rdeCsv:fields> element.

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MUST be used in the "domainTransfer" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fTrStatus> State of the most recent transfer request with isRequired="true".

<rdeCsv:fReRr> Identifier of the registrar, defined in Section 5.4, of the client that requested the transfer with isRequired="true".

<rdeCsv:fReDate> Date and time that the transfer was requested with isRequired="true".

<rdeCsv:fAcRr> Identifier of the registrar, defined in Section 5.4, of the client that should take or took action with isRequired="true".

<rdeCsv:fAcDate> Date and time that the transfer action should be taken or has been taken with isRequired="true".

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MAY be used in the "domainTransfer" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fExDate> Expiration date if the transfer command caused or causes a change in the validity period.

<rdeCsv:fReID> Identifier of the client that requested the transfer.

<rdeCsv:fAcID> Identifier of the client that should take or took action for transfer.

The following "csvDomain" fields, defined for the "domain" CSV File Definition (Section 5.1.2.1.1), MUST be used in the "domainTransfer" <rdeCsv:csv> <rdeCsv:fields> element:

<csvDomain:fName> Domain name of the domain name object involved in the transfer with isRequired="true".

Example of a "domainTransfer" <csvDomain:contents> <rdeCsv:csv> element.

```
...
<csvDomain:contents>
...
  <rdeCsv:csv name="domainTransfer">
    <rdeCsv:fields>
      <csvDomain:fName parent="true"/>
      <rdeCsv:fTrStatus/>
      <rdeCsv:fReRr/>
      <rdeCsv:fReID/>
      <rdeCsv:fReDate/>
      <rdeCsv:fAcRr/>
      <rdeCsv:fAcID/>
      <rdeCsv:fAcDate/>
      <rdeCsv:fExDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="2E5A9ACD">
        domainTransfer-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvDomain:contents>
...
```

Example of the corresponding domainTransfer-YYYYMMDD.csv file. The file contains one domain transfer record with a pending status.

```
domain1.example,pending,registrarX,clientY,
2011-03-08T19:38:00.0Z,registrarY,,2011-03-13T23:59:59.0Z,
2025-04-03T22:00:00.0Z
```

5.1.2.2. <csvDomain:deletes>

The <csvDomain:deletes> is used to hold the deleted domain name objects in a Differential or Incremental Deposit. All the domain name object data is deleted as part of a cascade delete. The <csvDomain:deletes> is split into separate CSV file definitions using named <rdeCsv:csv> elements with the "name" attribute. The following section defines the supported domain name deletes CSV file definition.

5.1.2.2.1. "domain" Deletes CSV File Definition

The following "csvDomain" field elements MUST be used in the deletes "domain" <rdeCsv:csv> <rdeCsv:fields> element:

<csvDomain:fName> Domain name field with type="eppcom:labelType" and isRequired="true".

Example of a "domain" <csvDomain:deletes> <rdeCsv:csv> element:

```
...
<csvDomain:deletes>
...
  <rdeCsv:csv name="domain">
    <rdeCsv:fields>
      <csvDomain:fName/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="A06D8194">
        domain-delete-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvDomain:deletes>
...
```

Example of the corresponding domain-delete-YYYYMMDD.csv file. The file contains two domain name records.

```
domain1.example
domain2.example
```

5.2. Host Object

The host object is based on the EPP host name mapping in [RFC5732]. The host object supports both the XML Model and the CSV Model, defined in Models (Section 2) section. The elements used for both models are defined in the following sections. Both the <csvHost:contents> and <csvHost:deletes> elements contain one or more <rdeCsv:csv> elements with a set of named CSV file definitions using the <rdeCsv:csv> "name" attribute.

5.2.1. XML Model

There are two elements used in the data escrow of the host objects for the XML model including the `<rdeHost:host>`, under the `<rdeHost:contents>` element, and the `<rdeHost:delete>` element, under the `<rde:deletes>` element.

A `<rdeHost:host>` element substitutes for the `<rdeHost:abstractHost>` abstract element to define a concrete definition of a host. The `<rdeHost:abstractHost>` element can be replaced by other host definitions using the XML schema substitution groups feature.

5.2.1.1. `<rdeHost:host>` element

The RDE host object is based on the EPP host `<info>` response for an authorized client (Section 3.1.2. of [RFC5732]).

The OPTIONAL `<host>` element contains the following child elements:

- o A `<name>` element that contains the fully-qualified name of the host object.
- o A `<roid>` element that contains the repository object identifier assigned to the host object when the object was created.
- o One or more `<status>` elements that describe the status of the host object.
- o Zero or more `<addr>` elements that contain the IP addresses associated with the host object.
- o A `<clID>` element that contains the identifier of the sponsoring registrar.
- o An OPTIONAL `<crRr>` element that contains the identifier of the registrar that created the host object. An OPTIONAL client attribute is used to specify the client that performed the operation.
- o An OPTIONAL `<crDate>` element that contains the date and time of host-object creation.
- o An OPTIONAL `<upRr>` element that contains the identifier of the registrar that last updated the host object. This element MUST NOT be present if the host object has never been modified. An OPTIONAL client attribute is used to specify the client that performed the operation.

- o An OPTIONAL <upDate> element that contains the date and time of the most recent host-object modification. This element MUST NOT be present if the host object has never been modified.
- o An OPTIONAL <trDate> element that contains the date and time of the most recent host object successful transfer. This element MUST NOT be present if the domain name object has never been transferred.

Example of <host> object:

```
...
<rdeHost:host>
  <rdeHost:name>ns1.example1.example</rdeHost:name>
  <rdeHost:roid>Hns1_example_test-TEST</rdeHost:roid>
  <rdeHost:status s="ok"/>
  <rdeHost:status s="linked"/>
  <rdeHost:addr ip="v4">192.0.2.2</rdeHost:addr>
  <rdeHost:addr ip="v4">192.0.2.29</rdeHost:addr>
  <rdeHost:addr ip="v6">2001:DB8:1::1</rdeHost:addr>
  <rdeHost:clID>RegistrarX</rdeHost:clID>
  <rdeHost:crRr>RegistrarX</rdeHost:crRr>
  <rdeHost:crDate>1999-05-08T12:10:00.0Z</rdeHost:crDate>
  <rdeHost:upRr>RegistrarX</rdeHost:upRr>
  <rdeHost:upDate>2009-10-03T09:34:00.0Z</rdeHost:upDate>
</rdeHost:host>
...
```

5.2.1.2. <rdeHost:delete> object

The <rdeHost:delete> element contains the fully-qualified domain name of a host that was deleted. The <rdeHost:delete> element also supports host removal based on roid to support SRS systems in which different hosts with the same fully-qualified domain name are active at the same time.

Example of <rdeHost:delete> object:

```
...
<rde:deletes>
  ...
  <rdeHost:delete>
    <rdeHost:name>ns1.example.example</rdeHost:name>
  </rdeHost:delete>
  ...
</rde:deletes>
...
```

5.2.2. CSV Model

For the CSV Model of the host object, the `<csvHost:contents>` child element of the `<rde:contents>` element is used to hold the new or updated host objects for the deposit. The `<csvHost:deletes>` child element of the `<rde:deletes>` element is used to hold the deleted or purged host objects for the deposit.

Differential and Incremental Deposits are based on changes to the host objects. The updated host object data under the `<csvHost:contents>` element is a cascade replace down all of the host CSV files starting with the parent "host" CSV File Definition (Section 5.2.2.1.1). The child CSV file definitions include a `<rdeCsv:fRoid parent="true">` field. All the child CSV file definition data for the host objects in the parent "host" CSV File Definition (Section 5.2.2.1.1) MUST first be deleted and then set using the data in the child CSV files. The deleted host object data under the `<csvHost:deletes>` element is a cascade delete starting from the "host" Deletes CSV File Definition (Section 5.2.2.2.1).

5.2.2.1. `<csvHost:contents>`

The `<csvHost:contents>` is used to hold the new or updated host object information for the deposit. The `<csvHost:contents>` is split into separate CSV file definitions using named `<rdeCsv:csv>` elements with the "name" attribute. The following sections include the supported host CSV file definitions.

5.2.2.1.1. "host" CSV File Definition

The "host" CSV File Definition defines the fields and CSV file references used for the host object records.

The following "csvHost" field elements MUST be used in the "host" `<rdeCsv:csv>` `<rdeCsv:fields>` element:

`<csvHost:fName>` Host name field with type="eppcom:labelType" and `isRequired="true"`.

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MUST be used in the "host" `<rdeCsv:csv>` `<rdeCsv:fields>` element:

`<rdeCsv:fRoid>` Repository Object Identifier (ROID) assigned to the host object with `isRequired="true"`.

The following "rdeCsv" and "csvRegistrar" fields, MAY be used in the "host" `<rdeCsv:csv>` `<rdeCsv:fields>` element:

<rdeCsv:fClID> or <csvRegistrar:fGurid> A choice of:

<rdeCsv:fClID> Identifier of the sponsoring client with
isRequired="true".

<csvRegistrar:fGurid> Contains the Globally Unique Registrar
Identifier (GURID) assigned by ICANN with
type="positiveInteger" and isRequired="true".

<rdeCsv:fCrRr> Identifier of the registrar, defined in Section 5.4,
of the client that created the host object.

<rdeCsv:fCrID> Identifier of the client that created the host
object.

<rdeCsv:fUpRr> Identifier of the registrar, defined in Section 5.4,
of the client that last updated the host object.

<rdeCsv:fUpID> Identifier of the client that last updated the host
object.

<rdeCsv:fCrDate> Date and time that the host object was created.

<rdeCsv:fUpDate> Date and time that the host object was last
updated. This field MUST NOT be set if the domain name object has
never been modified.

<rdeCsv:fTrDate> Date and time that the host object was last
transferred. This field MUST NOT be set if the domain name object
has never been transferred.

Example of a "host" <csvHost:contents> <rdeCsv:csv> element.

```
...
<csvHost:contents>
...
  <rdeCsv:csv name="host">
    <rdeCsv:fields>
      <csvHost:fName/>
      <rdeCsv:fRoid/>
      <rdeCsv:fClID/>
      <rdeCsv:fCrRr/>
      <rdeCsv:fCrID/>
      <rdeCsv:fCrDate/>
      <rdeCsv:fUpRr/>
      <rdeCsv:fUpID/>
      <rdeCsv:fUpDate/>
      <rdeCsv:fTrDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="6F1E58E5">
        host-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvHost:contents>
...
```

Example of the corresponding host-YYYYMMDD.csv file. The file contains six host records with four being internal hosts and two being external hosts.

```
ns1.domain1.example,Hns1_example_test-TEST,registrarX,registrarX,
clientY,1999-05-08T12:10:00.0Z,registrarX,
clientY,2009-10-03T09:34:00.0Z,2007-01-08T09:19:00.0Z
ns2.domain1.example,Hns2_domain1_test-TEST,registrarX,registrarX,
clientY,1999-05-08T12:10:00.0Z,registrarX,
clientY,2009-10-03T09:34:00.0Z,2007-01-08T09:19:00.0Z
ns1.domain2.example,Hns1_domain2_test-TEST,registrarX,registrarX,
clientY,1999-05-08T12:10:00.0Z,registrarX,
clientY,2009-10-03T09:34:00.0Z,2007-01-08T09:19:00.0Z
ns2.domain2.example,Hns2_domain2_test-TEST,registrarX,registrarX,
clientY,1999-05-08T12:10:00.0Z,registrarX,
clientY,2009-10-03T09:34:00.0Z,2007-01-08T09:19:00.0Z
ns1.example.net,Hns1_example_test-TEST,registrarX,registrarX,
clientY,1999-05-08T12:10:00.0Z,registrarX,
clientY,2009-10-03T09:34:00.0Z,2007-01-08T09:19:00.0Z
ns2.example.net,Hns2_example_test-TEST,registrarX,registrarX,
clientY,1999-05-08T12:10:00.0Z,registrarX,
clientY,2009-10-03T09:34:00.0Z,2007-01-08T09:19:00.0Z
```

5.2.2.1.2. "hostStatuses" CSV File Definition

The "hostStatuses" CSV File Definition defines the fields and CSV file references used for the host object statuses.

The following "csvHost" fields, defined for the "host" CSV File Definition (Section 5.2.2.1.1), MUST be used in the "hostStatuses" <rdeCsv:csv> <rdeCsv:fields> element:

<csvHost:fStatus> The status of the host with
type="host:statusValueType" and isRequired="true".

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MUST be used in the "hostStatuses" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fRoid> Host object Registry Object Identifier (ROID)
assigned to the host object with isRequired="true".

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MAY be used in the "hostStatuses" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fStatusDescription> Host object status description which is
free form text describing the rationale for the status.

<rdeCsv:fLang> Language of the <rdeCsv:fStatusDescription> field.

Example of a "hostStatuses" <csvHost:contents> <rdeCsv:csv> element.

```
...
<csvHost:contents>
...
  <rdeCsv:csv name="hostStatuses">
    <rdeCsv:fields>
      <rdeCsv:fRoid parent="true"/>
      <csvHost:fStatus/>
      <rdeCsv:fStatusDescription/>
      <rdeCsv:fLang/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="0DAE0583">
        hostStatuses-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvHost:contents>
...
```

Example of the corresponding hostStatuses-YYYYMMDD.csv file. The file contains the statuses for the six host names ns1.domain1.example, ns2.domain1.example, ns1.domain2.example, ns2.domain2.example, ns1.example.net and ns2.example.net.

```
Hns1_domain1_test-TEST,ok,,
Hns2_domain1_test-TEST,ok,,
Hns1_domain2_test-TEST,ok,,
Hns2_domain2_test-TEST,ok,,
Hns1_example_test-TEST,ok,,
Hns2_example_test-TEST,ok,,
```

5.2.2.1.3. "hostAddresses" CSV File Definition

The "hostAddresses" CSV File Definition defines the fields and CSV file references used for the host object IP addresses.

The following "csvHost" field elements MUST be used in the "hostAddresses" <rdeCsv:csv> <rdeCsv:fields> element:

```
<csvHost:fAddr> IP addresses associated with the host object with
  type="host:addrStringType". The attribute "isRequired" MUST equal
  "true".
```

<csvHost:fAddrVersion> IP addresses version associated with the host object with type="host:ipType". "host:ipType" has the enumerated values of "v4" or "v6". The attribute "isRequired" MUST equal "true".

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MUST be used in the "hostAddresses" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fRoid> Host object Registry Object Identifier (ROID) assigned to the host object with isRequired="true".

Example of a "hostAddresses" <csvHost:contents> <rdeCsv:csv> element.

```
...
<csvHost:contents>
...
  <rdeCsv:csv name="hostAddresses">
    <rdeCsv:fields>
      <rdeCsv:fRoid parent="true"/>
      <csvHost:fAddr isRequired="true"/>
      <csvHost:fAddrVersion isRequired="true"/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="28B194B0">
        hostAddresses-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvHost:contents>
...
```

Example of the corresponding hostAddresses-YYYYMMDD.csv file. The file contains the IP addresses for the host names ns1.domain1.example, ns2.domain1.example, ns1.domain2.example and ns2.domain2.example.

```
Hns1_domain1_test-TEST,192.0.2.1,v4
Hns2_domain1_test-TEST,2001:DB8::1,v6
Hns1_domain2_test-TEST,192.0.2.2,v4
Hns2_domain2_test-TEST,2001:DB8::2,v6
```

5.2.2.2. <csvHost:deletes>

The <csvHost:deletes> is used to hold the deleted host objects in a Differential or Incremental Deposit. All the host object data is deleted as part of a cascade delete. The <csvHost:deletes> is split into separate CSV file definitions using named <rdeCsv:csv> elements with the "name" attribute. The following section defines the supported host deletes CSV file definition.

5.2.2.2.1. "host" Deletes CSV File Definition

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MUST be used in the "host" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fRoid> Repository Object Identifier (ROID) assigned to the host object with isRequired="true".

Example of a "host" <csvHost:deletes> <rdeCsv:csv> element.

```
...
<csvHost:deletes>
...
  <rdeCsv:csv name="host">
    <rdeCsv:fields>
      <rdeCsv:fRoid/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="777F5F0E">
        host-delete-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvHost:deletes>
...
```

Example of the host-delete-YYYYMMDD.csv file. The file contains four host records.

```
Hns1_domain1_test-TEST
Hns2_domain1_test-TEST
Hns1_domain2_test-TEST
Hns2_domain2_test-TEST
```

5.3. Contact Object

The contact object is based on the EPP contact name mapping in [RFC5733]. The contact object supports both the XML Model and the CSV Model, defined in Models (Section 2) section. The elements used for both models are defined in the following sections.

5.3.1. XML Model

There are two elements used in the data escrow of the contact objects for the XML model including the `<rdeContact:contact>`, under the `<rdeContact:contents>` element, and the `<rdeContact:delete>` element, under the `<rde:deletes>` element.

A `<contact>` element substitutes for the `<abstractContact>` abstract element to define a concrete definition of a contact. The `<abstractContact>` element can be replaced by other contact definitions using the XML schema substitution groups feature.

5.3.1.1. `<rdeContact:contact>` object

The contact object is based on the EPP contact `<info>` response for an authorized client (Section 3.1.2. of [RFC5733]) with some additions including the data from an EPP `<transfer>` Query Response, see Section 3.1.3. of [RFC5733].

The OPTIONAL `<contact>` element contains the following child elements:

- o A `<id>` element that contains the server-unique identifier of the contact object
- o A `<roid>` element that contains the Repository Object Identifier assigned to the contact object when the object was created.
- o One or more `<status>` elements that describe the status of the contact object.
- o One or two `<postalInfo>` elements that contain postal-address information. Two elements are provided so that address information can be provided in both internationalized and localized forms; a "type" attribute is used to identify the two forms. If an internationalized form (type="int") is provided, element content MUST be represented in a subset of UTF-8 that can be represented in the 7-bit US-ASCII character set. If a localized form (type="loc") is provided, element content MAY be represented in unrestricted UTF-8. The `<postalInfo>` element contains the following child elements:

- * A <name> element that contains the name of the individual or role represented by the contact.
- * An OPTIONAL <org> element that contains the name of the organization with which the contact is affiliated.
- * An <addr> element that contains address information associated with the contact. An <addr> element contains the following child elements:
 - + One, two, or three OPTIONAL <street> elements that contain the contact's street address.
 - + A <city> element that contains the contact's city.
 - + An OPTIONAL <sp> element that contains the contact's state or province.
 - + An OPTIONAL <pc> element that contains the contact's postal code.
 - + A <cc> element that contains the contact's two-letter country code.
- o An OPTIONAL <voice> element that contains the contact's voice telephone number.
- o An OPTIONAL <fax> element that contains the contact's facsimile telephone number.
- o An <email> element that contains the contact's email address.
- o A <clID> element that contains the identifier of the sponsoring registrar.
- o An OPTIONAL <crRr> element that contains the identifier of the registrar that created the contact object. An OPTIONAL client attribute is used to specify the client that performed the operation.
- o An OPTIONAL <crDate> element that contains the date and time of contact-object creation.
- o An OPTIONAL <upRr> element that contains the identifier of the registrar that last updated the contact object. This element MUST NOT be present if the contact has never been modified. An OPTIONAL client attribute is used to specify the client that performed the operation.

- o An OPTIONAL <upDate> element that contains the date and time of the most recent contact-object modification. This element MUST NOT be present if the contact object has never been modified.
- o An OPTIONAL <trDate> element that contains the date and time of the most recent contact object successful transfer. This element MUST NOT be present if the contact object has never been transferred.
- o An OPTIONAL <trnData> element that contains the following child elements related to the last transfer request of the contact object:
 - * A <trStatus> element that contains the state of the most recent transfer request.
 - * A <reRr> element that contains the identifier of the registrar that requested the domain name object transfer. An OPTIONAL client attribute is used to specify the client that performed the operation.
 - * An <acRr> element that contains the identifier of the registrar that should act upon a PENDING transfer request. For all other status types, the value identifies the registrar that took the indicated action. An OPTIONAL client attribute is used to specify the client that performed the operation.
 - * A <reDate> element that contains the date and time that the transfer was requested.
 - * An <acDate> element that contains the date and time of a required or completed response. For a PENDING request, the value identifies the date and time by which a response is required before an automated response action will be taken by the registry. For all other status types, the value identifies the date and time when the request was completed.
- o An OPTIONAL <disclose> element that identifies elements that requiring exceptional server-operator handling to allow or restrict disclosure to third parties. See Section 2.9 of [RFC5733] for a description of the child elements contained within the <disclose> element.

Example <contact> object:

```
...
<rdeContact:contact>
  <rdeContact:id>sh8013</rdeContact:id>
  <rdeContact:roid>Csh8013-TEST</rdeContact:roid>
  <rdeContact:status s="linked"/>
  <rdeContact:status s="clientDeleteProhibited"/>
  <rdeContact:postalInfo type="int">
    <contact:name>John Doe</contact:name>
    <contact:org>Example Inc.</contact:org>
    <contact:addr>
      <contact:street>123 Example Dr.</contact:street>
      <contact:street>Suite 100</contact:street>
      <contact:city>Dulles</contact:city>
      <contact:sp>VA</contact:sp>
      <contact:pc>20166-6503</contact:pc>
      <contact:cc>US</contact:cc>
    </contact:addr>
  </rdeContact:postalInfo>
  <rdeContact:voice x="1234">+1.7035555555</rdeContact:voice>
  <rdeContact:fax>+1.7035555556</rdeContact:fax>
  <rdeContact:email>jdoe@example.example</rdeContact:email>
  <rdeContact:clID>RegistrarX</rdeContact:clID>
  <rdeContact:crRr client="jdoe">RegistrarX</rdeContact:crRr>
  <rdeContact:crDate>2009-09-13T08:01:00.0Z</rdeContact:crDate>
  <rdeContact:upRr client="jdoe">RegistrarX</rdeContact:upRr>
  <rdeContact:upDate>2009-11-26T09:10:00.0Z</rdeContact:upDate>
  <rdeContact:trDate>2009-12-03T09:05:00.0Z</rdeContact:trDate>
  <rdeContact:trnData>
    <rdeContact:trStatus>pending</rdeContact:trStatus>
    <rdeContact:reRr client="jstiles">clientW</rdeContact:reRr>
    <rdeContact:reDate>2011-03-08T19:38:00.0Z</rdeContact:reDate>
    <rdeContact:acRr client="rmiles">RegistrarX</rdeContact:acRr>
    <rdeContact:acDate>2011-03-13T23:59:59.0Z</rdeContact:acDate>
  </rdeContact:trnData>
  <rdeContact:disclose flag="0">
    <contact:voice/>
    <contact:email/>
  </rdeContact:disclose>
</rdeContact:contact>
...
```

5.3.1.2. <rdeContact:delete> object

The <rdeContact:delete> element contains the id of a contact that was deleted.

Example of <rdeContact:delete> object:

```
...
<rde:deletes>
  ...
  <rdeContact:delete>
    <rdeContact:id>sh8013-TEST</rdeContact:id>
    <rdeContact:id>co8013-TEST</rdeContact:id>
  </rdeContact:delete>
  ...
</rde:deletes>
...
```

5.3.2. CSV Model

For the CSV Model of the contact object, the <csvContact:contents> child element of the <rde:contents> element is used to hold the new or updated contacts objects for the deposit. The <csvContact:deletes> child element of the <rde:deletes> element is used to hold the deleted or purged contact objects for the deposit. Both the <csvContact:contents> and <csvContact:deletes> elements contain one or more <rdeCsv:csv> elements with a set of named CSV file definitions using the <rdeCsv:csv> "name" attribute.

Differential and Incremental Deposits are based on changes to the contact objects. The updated contact object data under the <csvContact:contents> element is a cascade replace down all of the contact CSV files starting with the parent "contact" CSV File Definition (Section 5.3.2.1.1). The child CSV file definitions include a <csvContact:fId parent="true"> field. All the child CSV file definition data for the contact objects in the parent "contact" CSV File Definition (Section 5.3.2.1.1) MUST first be deleted and then set using the data in the child CSV files. The deleted contact object data under the <csvContact:deletes> element is a cascade delete starting from the "contact" Deletes CSV File Definition (Section 5.3.2.2.1).

5.3.2.1. <csvContact:contents>

The <csvContact:contents> is used to hold the new or updated contact object information for the deposit. The <csvContact:contents> is split into separate CSV file definitions using named <rdeCsv:csv> elements with the "name" attribute. The following sections include the supported contact CSV file definitions.

5.3.2.1.1. "contact" CSV File Definition

The "contact" CSV File Definition defines the fields and CSV file references used for the contact object records.

The following "csvContact" field elements MUST be used in the "contact" <rdeCsv:csv> <rdeCsv:fields> element:

<csvContact:fId> Contains the server-unique contact identifier with type="eppcom:clIDType" and isRequired="true".

<csvContact:fEmail> Contains the contact's email address with type="eppcom:minTokenType" and isRequired="true".

The following field elements MAY be used in the "contact" <rdeCsv:csv> <rdeCsv:fields> element:

<csvContact:fVoice> Contains the contact's voice telephone number with type="contact:e164StringType".

<csvContact:fVoiceExt> Contains the contact's voice telephone number extension with type="token".

<csvContact:fFax> Contains the contact's facsimile telephone number with type="contact:e164StringType".

<csvContact:fFaxExt> Contains the contact's facsimile telephone number extension with type="token".

The following "rdeCsv" and "csvRegistrar" fields, MUST be used in the "contact" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fRoid> The Registry Object Identifier (ROID) for the contact object with isRequired="true".

<rdeCsv:fClID> or <csvRegistrar:fGurid> A choice of:

<rdeCsv:fClID> Identifier of the sponsoring client with isRequired="true".

<csvRegistrar:fGurid> Contains the Globally Unique Registrar Identifier (GURID) assigned by ICANN with type="positiveInteger" and isRequired="true".

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MAY be used in the "contact" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fCrRr> Identifier of the registrar, defined in Section 5.4, of the client that created the contact object.

<rdeCsv:fCrID> Identifier of the client that created the contact object.

<rdeCsv:fUpRr> Identifier of the registrar, defined in Section 5.4, of the client that last updated the contact object.

<rdeCsv:fUpID> Identifier of the client that last updated the contact object.

<rdeCsv:fCrDate> Created date and time of the contact object.

<rdeCsv:fUpDate> Date and time of the last update to the contact object. This field MUST NOT be set if the domain name object has never been modified.

<rdeCsv:fTrDate> Date and time of the last transfer for the contact object. This field MUST NOT be set if the domain name object has never been transferred.

Example of a "contact" <csvContact:contacts> <rdeCsv:csv> element.

```
...
<csvContact:contents>
...
  <rdeCsv:csv name="contact">
    <rdeCsv:fields>
      <csvContact:fId/>
      <rdeCsv:fRoid/>
      <csvContact:fVoice/>
      <csvContact:fVoiceExt/>
      <csvContact:fFax/>
      <csvContact:fFaxExt/>
      <csvContact:fEmail/>
      <rdeCsv:fClID/>
      <rdeCsv:fCrRr/>
      <rdeCsv:fCrID/>
      <rdeCsv:fCrDate/>
      <rdeCsv:fUpRr/>
      <rdeCsv:fUpID/>
      <rdeCsv:fUpDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="8587AA49">
        contact-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvContact:contents>
...
```

Example of the contact-YYYYMMDD.csv file. The file contains nine object contact records.

```
domainladmin,Cdomainladmin-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registrarX,
clientY,2009-09-13T08:01:00.0Z,registrarX,clientY,
2009-11-26T09:10:00.0Z
domain1tech,Cdomain1tech-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registrarX,
clientY,2009-09-13T08:01:00.0Z,registrarX,clientY,
2009-11-26T09:10:00.0Z
domainlbilling,Cdomainlbilling-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registrarX,
clientY,2009-09-13T08:01:00.0Z,registrarX,clientY,
2009-11-26T09:10:00.0Z
domain2admin,Cdomain2admin-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registrarX,
clientY,2009-09-13T08:01:00.0Z,registrarX,clientY,
2009-11-26T09:10:00.0Z
domain2tech,Cdomain2tech-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registrarX,
clientY,2009-09-13T08:01:00.0Z,registrarX,clientY,
2009-11-26T09:10:00.0Z
domain2billing,Cdomain2billing-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registrarX,
clientY,2009-09-13T08:01:00.0Z,registrarX,clientY,
2009-11-26T09:10:00.0Z
xnabc123admin,Cxnabc123admin-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registrarX,
clientY,2009-09-13T08:01:00.0Z,registrarX,clientY,
2009-11-26T09:10:00.0Z
xnabc123tech,Cxnabc123tech-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registrarX,
clientY,2009-09-13T08:01:00.0Z,registrarX,clientY,
2009-11-26T09:10:00.0Z
xnabc123billing,Cxnabc123billing-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registrarX,
clientY,2009-09-13T08:01:00.0Z,registrarX,clientY,
2009-11-26T09:10:00.0Z
```

5.3.2.1.2. "contactStatuses" CSV File Definition

The "contactStatuses" CSV File Definition defines the fields and CSV file references used for the contact object statuses.

The following "csvContact" field elements, defined for the "contact" CSV File Definition (Section 5.3.2.1.1), MUST be used in the "contactStatuses" <rdeCsv:csv> <rdeCsv:fields> element:

<csvContact:fId> Server-unique contact identifier of status with
isRequired="true" and parent="true".

<csvContact:fStatus> The status of the contact with
type="contact:statusValueType" and isRequired="true".

The following "rdeCsv" fields, defined in section CSV common field
elements (Section 4.6.2.2), MAY be used in the "contactStatuses"
<rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fStatusDescription> The contact object status description
which is free form text describing the rationale for the status.

<rdeCsv:fLang> Language of the <rdeCsv:fStatusDescription> field.

Example of a "contactStatuses" <csvContact:contents> <rdeCsv:csv>
element.

```
...
<csvContact:contents>
...
  <rdeCsv:csv name="contactStatuses">
    <rdeCsv:fields>
      <csvContact:fId parent="true"/>
      <csvContact:fStatus/>
      <rdeCsv:fStatusDescription/>
      <rdeCsv:fLang/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="137E13EC">
        contactStatuses-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvContact:contents>
...
```

Example of the corresponding contactStatuses-YYYYMMDD.csv file. The file contains the statuses for the nine contact identifiers.

```
domain1admin,ok,,
domain1tech,ok,,
domain1billing,ok,,
domain2admin,ok,,
domain2tech,ok,,
domain2billing,ok,,
xnabc123admin,ok,,
xnabc123tech,ok,,
xnabc123billing,ok,,
```

5.3.2.1.3. "contactPostal" CSV File Definition

The "contactPostal" CSV File Definition defines the fields and CSV file references used for the contact postal info object records.

The following "csvContact" field elements MUST be used in the "contactPostal" <rdeCsv:csv> <rdeCsv:fields> element:

<csvContact:fPostalType> Contains the form of the postal-address information with type="contact:postalLineType" and isRequired="true". This field specifies the form ("int" or "loc"), as defined in Section 4.6.3, of the <csvContact:fName>, <csvContact:fOrg>, <csvContact:fStreet>, <csvContact:fCity>, <csvContact:fSp>, <csvContact:fPc>, <csvContact:fCc> fields.

<csvContact:fName> Contains the contact's name of the individual or role represented by the contact with type="contact:postalLineType" and isRequired="true". An OPTIONAL "isLoc" attribute is used to indicate the localized or internationalized form as defined in Section 4.6.3.

<csvContact:fStreet> Contains the contact's street address line with type="contact:fPostalLineType". An index attribute is required to indicate which street address line the field represents with index "0" for the first line and incrementing for each line up to index "2" for the third line. An OPTIONAL "isLoc" attribute is used to indicate the localized or internationalized form as defined in Section 4.6.3.

<csvContact:fCity> Contains the contact's city with type="contact:postalLineType" and isRequired="true". An OPTIONAL "isLoc" attribute is used to indicate the localized or internationalized form as defined in Section 4.6.3.

<csvContact:fCc> Contains the contact's country code with type="contact:ccType" and isRequired="true". An OPTIONAL "isLoc" attribute is used to indicate the localized or internationalized form as defined in Section 4.6.3.

The following "csvContact" field elements MAY be used in the "contactPostal" <rdeCsv:csv> <rdeCsv:fields> element:

<csvContact:fOrg> Contains the name of the organization with which the contact is affiliated with type="contact:optPostalLineType". An OPTIONAL "isLoc" attribute is used to indicate the localized or internationalized form as defined in Section 4.6.3.

<csvContact:fSp> Contains the contact's state or province with type="contact:optPostalLineType". An OPTIONAL "isLoc" attribute is used to indicate the localized or internationalized form as defined in Section 4.6.3.

<csvContact:fPc> Contains the contact's postal code with type="contact:pcType". An OPTIONAL "isLoc" attribute is used to indicate the localized or internationalized form as defined in Section 4.6.3.

The following "csvContact" fields, defined for the "contact" CSV File Definition (Section 5.3.2.1.1), MUST be used in the "contactPostal" <rdeCsv:csv> <rdeCsv:fields> element:

<csvContact:fId> Server-unique contact identifier for the contact object with isRequired="true" and parent="true".

Example of a "contactPostal" <csvContact:contents> <rdeCsv:csv> element.

```
...
<csvContact:contents>
...
  <rdeCsv:csv name="contactPostal">
    <rdeCsv:fields>
      <csvContact:fId parent="true"/>
      <csvContact:fPostalType/>
      <csvContact:fName/>
      <csvContact:fOrg/>
      <csvContact:fStreet index="0"/>
      <csvContact:fStreet index="1"/>
      <csvContact:fStreet index="2"/>
      <csvContact:fCity/>
      <csvContact:fSp/>
      <csvContact:fPc/>
      <csvContact:fCc/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="1456A89C">
        contactPostal-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvContact:contents>
...
```

Example of the contactPostal-YYYYMMDD.csv file. The file contains nine contact postal records.

```
domainladmin,int,"John Doe","Example Inc.",
"123 Example Dr.,"Suite 100",,Reston,VA,20190,US
domainltech,int,"John Doe","Example Inc.",
"123 Example Dr.,"Suite 100",,Reston,VA,20190,US
domainlbilling,int,"John Doe","Example Inc.",
"123 Example Dr.,"Suite 100",,Reston,VA,20190,US
domain2admin,int,"John Doe","Example Inc.",
"123 Example Dr.,"Suite 100",,Reston,VA,20190,US
domain2tech,int,"John Doe","Example Inc.",
"123 Example Dr.,"Suite 100",,Reston,VA,20190,US
domain2billing,int,"John Doe","Example Inc.",
"123 Example Dr.,"Suite 100",,Reston,VA,20190,US
xnabc123admin,int,"John Doe","Example Inc.",
"123 Example Dr.,"Suite 100",,Reston,VA,20190,US
xnabc123tech,int,"John Doe","Example Inc.",
"123 Example Dr.,"Suite 100",,Reston,VA,20190,US
xnabc123billing,int,"John Doe","Example Inc.",
"123 Example Dr.,"Suite 100",,Reston,VA,20190,US
```

5.3.2.1.4. "contactTransfer" CSV File Definition

The "contactTransfer" CSV File Definition defines the fields and CSV file references used for the contact object pending and completed transfer records. No additional field elements were added for use in the "contactTransfer" <rdeCsv:csv> <rdeCsv:fields> element. The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MUST be used in the "contactTransfer" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fTrStatus> State of the most recent transfer request with isRequired="true".

<rdeCsv:fReRr> Identifier of the registrar, defined in Section 5.4, of the client that requested the transfer with isRequired="true".

<rdeCsv:fReDate> Date and time that the transfer was requested with isRequired="true".

<rdeCsv:fAcRr> Identifier of the registrar, defined in Section 5.4, of the client that should take or took action with isRequired="true".

<rdeCsv:fAcDate> Date and time that the transfer action should be taken or has been taken with isRequired="true".

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MAY be used in the "contactTransfer" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fReID> Identifier of the client that requested the transfer.

<rdeCsv:fAcID> Identifier of the client that should take or took action for transfer.

The following "csvContact" fields, defined for the "contact" CSV File Definition (Section 5.3.2.1.1), MUST be used in the "contactTransfer" <rdeCsv:csv> <rdeCsv:fields> element:

<csvContact:fId> Server-unique contact identifier for the contact object with isRequired="true".

Example of a "contactTransfer" <csvContact:contents> <rdeCsv:csv> element.

```
...
<csvContact:contents>
...
  <rdeCsv:csv name="contactTransfer">
    <rdeCsv:fields>
      <csvContact:fId parent="true"/>
      <rdeCsv:fTrStatus/>
      <rdeCsv:fReRr/>
      <rdeCsv:fReID/>
      <rdeCsv:fReDate/>
      <rdeCsv:fAcRr/>
      <rdeCsv:fAcID/>
      <rdeCsv:fAcDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="788D308E">
        contactTransfer-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvContact:contents>
...
```

Example of the contactTransfer-YYYYMMDD.csv file. The file contains one contact transfer record in pending status.

```
xnabc123admin,clientApproved,registrarX,clientX,  
2011-04-08T19:38:00.0Z,registrarY,clientY,2011-04-09T20:38:00.0Z
```

5.3.2.1.5. "contactDisclose" CSV File Definition

The "contactDisclose" CSV File Definition defines the fields and CSV file references used for the contact disclose object records.

The following "csvContact" field elements MAY be used in the "contactDisclose" <rdeCsv:csv> <rdeCsv:fields> element:

<csvContact:fDiscloseFlag> Contains flag with a value of "true" or "1" (one) notes the preference to allow disclosure of the specified elements as an exception to the stated data-collection policy. A value of "false" or "0" (zero) notes a client preference to not allow disclosure of the specified elements as an exception to the stated data-collection policy with type="boolean". The additional fields define specific exceptional disclosure preferences based on the <csvContact:fDiscloseFlag> field.

<csvContact:fDiscloseNameLoc> Exceptional disclosure preference flag for the localized form of the contact name with type="boolean".

<csvContact:fDiscloseNameInt> Exceptional disclosure preference flag for the internationalized form of the contact name with type="boolean".

<csvContact:fDiscloseOrgLoc> Exceptional disclosure preference flag for the localized form of the contact organization with type="boolean".

<csvContact:fDiscloseOrgInt> Exceptional disclosure preference flag for the internationalized form of the contact organization with type="boolean".

<csvContact:fDiscloseAddrLoc> Exceptional disclosure preference flag for the localized form of the contact address with type="boolean".

<csvContact:fDiscloseAddrInt> Exceptional disclosure preference flag for the internationalized form of the contact address with type="boolean".

<csvContact:fDiscloseVoice> Exceptional disclosure preference flag of the contact voice telephone number with type="boolean".

<csvContact:fDiscloseFax> Exceptional disclosure preference flag of the contact facsimile telephone number with type="boolean".

<csvContact:fDiscloseEmail> Exceptional disclosure preference flag of the contact email address with type="boolean".

The following "csvContact" fields, defined for the "contact" CSV File Definition (Section 5.3.2.1.1), MUST be used in the "contactDisclose" <rdeCsv:csv> <rdeCsv:fields> element:

<csvContact:fId> Server-unique contact identifier for the contact object with isRequired="true".

Example of a "contactDisclose" <csvContact:contents> <rdeCsv:csv> element.

```
...
<csvContact:contents>
...
  <rdeCsv:csv name="contactDisclose">
    <rdeCsv:fields>
      <csvContact:fId parent="true"/>
      <csvContact:fDiscloseFlag/>
      <csvContact:fDiscloseNameLoc/>
      <csvContact:fDiscloseNameInt/>
      <csvContact:fDiscloseOrgLoc/>
      <csvContact:fDiscloseOrgInt/>
      <csvContact:fDiscloseAddrLoc/>
      <csvContact:fDiscloseAddrInt/>
      <csvContact:fDiscloseVoice/>
      <csvContact:fDiscloseFax/>
      <csvContact:fDiscloseEmail/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="1141EFD4">
        contactDisclose-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvContact:contents>
...
```

Example of the contactDisclose-YYYYMMDD.csv file. The file contains one disclosure records, disabling disclosure of voice, fax, and email.

```
xnabc123admin,0,0,0,0,0,0,0,0,1,1,1
```

5.3.2.2. <csvContact:deletes>

The <csvContact:deletes> is used to hold the deleted contact objects in a Differential or Incremental Deposit. All the contact object data is deleted as part of a cascade delete. The <csvContact:deletes> is split into separate CSV file definitions using named <rdeCsv:csv> elements with the "name" attribute. The following section defines the supported contact deletes CSV file definition.

5.3.2.2.1. "contact" Deletes CSV File Definition

The following "csvContact" field elements MUST be used in the deletes "contact" <rdeCsv:csv> <rdeCsv:fields> element:

<csvContact:fId> Contains the server-unique contact identifier with type="eppcom:clIDType" and isRequired="true".

Example of a "contact" <csvContact:deletes> <rdeCsv:csv> element.

```
...
<csvContact:deletes>
...
  <rdeCsv:csv name="contact">
    <rdeCsv:fields>
      <csvContact:fId/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="0C4B70DC">
        contact-delete-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvContact:deletes>
...
```

Example of the contact-delete-YYYYMMDD.csv file. The file contains six contact records.

```
domain1admin  
domain1tech  
domain1billing  
domain2admin  
domain2tech  
domain2billing
```

5.4. Registrar Object

The registrar object represents the sponsoring client for other objects, and is typically referred to as the sponsoring registrar. The registrar object supports both the XML Model and the CSV Model, defined in Section 2. The elements used for both models are defined in the following sections.

5.4.1. XML Model

There are two elements used in the data escrow of the registrar objects for the XML model including the `<rdeRegistrar:registrar>`, under the `<rdeRegistrar:contents>` element, and the `<rdeRegistrar:delete>` element, under the `<rde:deletes>` element.

A `<rdeRegistrar:registrar>` element substitutes for the `<rdeRegistrar:abstractRegistrar>` abstract element to define a concrete definition of a registrar. The `<rdeRegistrar:abstractRegistrar>` element can be replaced by other domain definitions using the XML schema substitution groups feature.

5.4.1.1. `<rdeRegistrar:registrar>` element

The `<registrar>` element contains the following child elements:

- o An `<id>` element that contains the Registry-unique identifier of the registrar object. This `<id>` has a superordinate relationship to a subordinate `<clID>`, `<crRr>` or `<upRr>` of domain, contact and host objects.
- o An `<name>` element that contains the name of the registrar.
- o An OPTIONAL `<gurid>` element that contains the Globally Unique Registrar Identifier (GURID) assigned by ICANN.
- o An OPTIONAL `<status>` element that contains the operational status of the registrar. Possible values are: ok, readonly and terminated.

- o One or two OPTIONAL <postalInfo> elements that contain postal-address information. Two elements are provided so that address information can be provided in both internationalized and localized forms; a "type" attribute is used to identify the two forms. If an internationalized form (type="int") is provided, element content MUST be represented in a subset of UTF-8 that can be represented in the 7-bit US-ASCII character set. If a localized form (type="loc") is provided, element content MAY be represented in unrestricted UTF-8. The <postalInfo> element contains the following child elements:
 - * A <addr> element that contains address information associated with the registrar. The <addr> element contains the following child elements:
 - + One, two, or three OPTIONAL <street> elements that contain the registrar's street address.
 - + A <city> element that contains the registrar's city.
 - + An OPTIONAL <sp> element that contains the registrar's state or province.
 - + An OPTIONAL <pc> element that contains the registrar's postal code.
 - + A <cc> element that contains the registrar's country code.
- o An OPTIONAL <voice> element that contains the registrar's voice telephone number.
- o An OPTIONAL <fax> element that contains the registrar's facsimile telephone number.
- o An OPTIONAL <email> element that contains the registrar's email address.
- o An OPTIONAL <url> element that contains the registrar's URL.
- o An OPTIONAL <whoisInfo> elements that contains whois information. The <whoisInfo> element contains the following child elements:
 - * An OPTIONAL <name> element that contains the name of the registrar WHOIS server listening on TCP port 43 as specified in [RFC3912].
 - * An OPTIONAL <url> element that contains the name of the registrar WHOIS server listening on TCP port 80/443.

- o An OPTIONAL <crDate> element that contains the date and time of registrar-object creation.
- o An OPTIONAL <upDate> element that contains the date and time of the most recent registrar-object modification. This element MUST NOT be present if the registrar-object has never been modified.

Example of a <registrar> object:

```
...
<rdeRegistrar:registrar>
  <rdeRegistrar:id>RegistrarX</rdeRegistrar:id>
  <rdeRegistrar:name>Registrar X</rdeRegistrar:name>
  <rdeRegistrar:gurid>8</rdeRegistrar:gurid>
  <rdeRegistrar:status>ok</rdeRegistrar:status>
  <rdeRegistrar:postalInfo type="int">
    <rdeRegistrar:addr>
      <rdeRegistrar:street>123 Example Dr.</rdeRegistrar:street>
      <rdeRegistrar:street>Suite 100</rdeRegistrar:street>
      <rdeRegistrar:city>Dulles</rdeRegistrar:city>
      <rdeRegistrar:sp>VA</rdeRegistrar:sp>
      <rdeRegistrar:pc>20166-6503</rdeRegistrar:pc>
      <rdeRegistrar:cc>US</rdeRegistrar:cc>
    </rdeRegistrar:addr>
  </rdeRegistrar:postalInfo>
  <rdeRegistrar:voice x="1234">+1.7035555555</rdeRegistrar:voice>
  <rdeRegistrar:fax>+1.7035555556</rdeRegistrar:fax>
  <rdeRegistrar:email>jdoe@example.example</rdeRegistrar:email>
  <rdeRegistrar:url>http://www.example.example</rdeRegistrar:url>
  <rdeRegistrar:whoisInfo>
    <rdeRegistrar:name>whois.example.example</rdeRegistrar:name>
    <rdeRegistrar:url>http://whois.example.example</rdeRegistrar:url>
  </rdeRegistrar:whoisInfo>
  <rdeRegistrar:crDate>2005-04-23T11:49:00.0Z</rdeRegistrar:crDate>
  <rdeRegistrar:upDate>2009-02-17T17:51:00.0Z</rdeRegistrar:upDate>
</rdeRegistrar:registrar>
...
```

5.4.1.2. <rdeRegistrar:delete> object

The <rdeRegistrar:delete> element contains the id of a registrar that was deleted.

Example of <rdeRegistrar:delete> object:

```
...
<rde:deletes>
  ...
  <rdeRegistrar:delete>
    <rdeRegistrar:id>agnt0001-TEST</rdeRegistrar:id>
  </rdeRegistrar:delete>
  ...
</rde:deletes>
...
```

5.4.2. CSV Model

For the CSV Model of the registrar object, the <csvRegistrar:contents> child element of the <rde:contents> element is used to hold the new or updated registrar objects for the deposit. The <csvRegistrar:deletes> child element of the <rde:deletes> element is used to hold the deleted or purged registrar objects for the deposit. Both the <csvRegistrar:contents> and <csvRegistrar:deletes> elements contain one or more <rdeCsv:csv> elements with a set of named CSV file definitions using the <rdeCsv:csv> "name" attribute.

Differential and Incremental Deposits are based on changes to the registrar objects. The updated registrar object data under the <csvContact:contents> element is a cascade replace down all of the registrar CSV files starting with the parent "registrar" CSV File Definition (Section 5.4.2.1.1). The child CSV file definitions include a <csvRegistrar:fId parent="true"> field. All the child CSV file definition data for the registrar objects in the parent "registrar" CSV File Definition (Section 5.4.2.1.1) MUST first be deleted and then set using the data in the child CSV files. The deleted registrar object data under the <csvRegistrar:deletes> element is a cascade delete starting from the "registrar" Deletes CSV File Definition (Section 5.4.2.2.1).

5.4.2.1. <csvRegistrar:contents>

The <csvRegistrar:contents> is used to hold the new or updated registrar object information for the deposit. The <csvRegistrar:contents> is split into separate CSV file definitions using named <rdeCsv:csv> elements with the "name" attribute. The following sections include the supported contact CSV file definitions.

5.4.2.1.1. "registrar" CSV File Definition

The "registrar" CSV File Definition defines the fields and CSV file references used for the registrar object records.

The following "csvRegistrar" field elements MUST be used in the "registrar" <rdeCsv:csv> <rdeCsv:fields> element:

<csvRegistrar:fId> or <csvRegistrar:fGurid> A choice of:

<csvRegistrar:fId> Contains the server-unique registrar identifier with type="eppcom:clIDType" and isRequired="true".

<csvRegistrar:fGurid> Contains the Globally Unique Registrar Identifier (GURID) assigned by ICANN with type="positiveInteger" and isRequired="true".

<csvRegistrar:fName> Contains the name of the registrar with type="normalizedString" and isRequired="true".

The following field elements MAY be used in the "registrar" <rdeCsv:csv> <rdeCsv:fields> element:

<csvRegistrar:fStatus> Contains the status of the registrar with type="csvRegistrar:statusValueType".

<csvRegistrar:fGurid> Contains the ID assigned by ICANN with type="positiveInteger". This field is included in this section in addition to the section above to support optionally providing the <csvRegistrar:fGurid> field when the <csvRegistrar:fId> field is used.

<csvRegistrar:fWhoisUrl> Contains the Whois URL of the registrar with type="anyURI".

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MAY be used in the "registrar" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fCrDate> Created date and time of the registrar object.

<rdeCsv:fUpDate> Date and time of the last update to the registrar object. This field MUST NOT be set if the domain name object has never been modified.

<rdeCsv:fUrl> URL for the registrar web home page.

The following "csvContact" fields, defined in section Contact Object (Section 5.3), MAY be used in the "registrar" <rdeCsv:csv> <rdeCsv:fields> element:

<csvContact:fStreet> Registrar street address line with an "index" attribute that represents the order of the street address line from "0" to "2". An OPTIONAL "isLoc" attribute that is used to indicate the localized or internationalized form, as defined in Section 4.6.3.

<csvContact:fCity> Registrar city with an OPTIONAL "isLoc" attribute that is used to indicate the localized or internationalized form, as defined in Section 4.6.3.

<csvContact:fCc> Registrar country code with an OPTIONAL "isLoc" attribute that is used to indicate the localized or internationalized form, as defined in Section 4.6.3.

<csvContact:fEmail> Registrar email address. The attribute "isRequired" MUST equal "false".

<csvContact:fSp> Registrar state or province with an OPTIONAL "isLoc" attribute that is used to indicate the localized or internationalized form, as defined in Section 4.6.3.

<csvContact:fPc> Registrar postal code with an OPTIONAL "isLoc" attribute that is used to indicate the localized or internationalized form, as defined in Section 4.6.3.

<csvContact:fVoice> Registrar voice telephone number.

<csvContact:fVoiceExt> Registrar voice telephone number extension.

<csvContact:fFax> Registrar facsimile telephone number.

<csvContact:fFaxExt> Registrar facsimile telephone number extension.

Example of a "registrar" <csvRegistrar:contents> <rdeCsv:csv> element.

```
...
<csvRegistrar:contents>
...
  <rdeCsv:csv name="registrar">
    <rdeCsv:fields>
      <csvRegistrar:fId/>
      <csvRegistrar:fName isLoc="false"/>
      <csvRegistrar:fGurid/>
      <csvRegistrar:fStatus/>
      <csvContact:fStreet isLoc="false" index="0"/>
      <csvContact:fStreet isLoc="false" index="1"/>
      <csvContact:fStreet isLoc="false" index="2"/>
      <csvContact:fCity isLoc="false"/>
      <csvContact:fSp isLoc="false" />
      <csvContact:fPc isLoc="false" />
      <csvContact:fCc isLoc="false"/>
      <csvContact:fVoice/>
      <csvContact:fVoiceExt/>
      <csvContact:fFax/>
      <csvContact:fFaxExt/>
      <csvContact:fEmail isRequired="false"/>
      <rdeCsv:fUrl/>
      <csvRegistrar:fWhoisUrl/>
      <rdeCsv:fCrDate/>
      <rdeCsv:fUpDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="57F6856F">
        registrar-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvRegistrar:contents>
...
```

Example of the registrar-YYYYMMDD.csv file. The file contains one registrar record.

```
registrarX,"Example Inc.",8,ok,"123 Example Dr.",
"Suite 100",,Dulles,VA,20166-6503,US,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,http://www.example.example,
http://whois.example.example,2005-04-23T11:49:00.0Z,
2009-02-17T17:51:00.0Z
```

5.4.2.2. <csvRegistrar:deletes>

The <csvRegistrar:deletes> is used to hold the deleted registrar objects in a Differential or Incremental Deposit. All the registrar object data is deleted as part of a cascade delete. The <csvRegistrar:deletes> is split into separate CSV file definitions using named <rdeCsv:csv> elements with the "name" attribute. The following section defines the supported registrar deletes CSV file definition.

5.4.2.2.1. "registrar" Deletes CSV File Definition

The following "csvRegistrar" field elements MUST be used in the deletes "registrar" <rdeCsv:csv> <rdeCsv:fields> element:

<csvRegistrar:fId> or <csvRegistrar:fGurid> A choice of:

<csvRegistrar:fId> Contains the server-unique registrar identifier with type="eppcom:clIDType" and isRequired="true".

<csvRegistrar:fGurid> Contains the Globally Unique Registrar Identifier (GURID) assigned by ICANN with type="positiveInteger". The attribute "isRequired" MUST equal "true".

Example of a "registrar" <csvRegistrar:deletes> <rdeCsv:csv> element.

```
...
<csvRegistrar:deletes>
...
  <rdeCsv:csv name="registrar">
    <rdeCsv:fields>
      <csvRegistrar:fId/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="5CB20A52">
        registrar-delete-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvRegistrar:deletes>
...
```

Example of the registrar-delete-YYYYMMDD.csv file. The file contains one registrar record.

registrarZ

5.5. IDN Table Reference Object

The Internationalized Domain Names (IDN) table reference object is a pseudo-object that is used to provide a short reference to the IDN Table and Policy used in IDN registrations. The IDN reference object supports both the XML and the CSV Model, defined in the Models (Section 2) section. The elements used for both models are defined in the following sections.

5.5.1. XML Model

There is one element used in the data escrow of the IDN table reference objects for the XML model that is the `<rdeIDN:idnTableRef>`, under the `<rde:contents>` element.

5.5.1.1. `<rdeIDN:idnTableRef>` object

The `<rdeIDN:idnTableRef>` contains the following elements. An "id" attribute is used to specify an identifier for the IDN table.

- o An `<url>` element that contains the URL of the IDN table that is being referenced.
- o A `<urlPolicy>` element that contains the URL of the IDN policy document. If IDN variants are generated algorithmically, the policy document MUST define the algorithm and the state of the implicit generated IDN variants. For a list of suggested states for implicit IDN variants, please see [variantTLDsReport].

Example of `<idnTableRef>` object:

```
...
<rdeIDN:idnTableRef id="pt-BR">
  <rdeIDN:url>
    http://www.iana.org/domains/idn-tables/tables/br_pt-br_1.0.html
  </rdeIDN:url>
  <rdeIDN:urlPolicy>
    http://registro.br/dominio/regras.html
  </rdeIDN:urlPolicy>
</rdeIDN:idnTableRef>
...
```


5.5.2. CSV Model

The IDN domain names, defined in Section 5.1, MAY have references to the IDN language identifier using the `<rdeCsv:fIdnTableId>` field element. The IDN table reference object defines the mapping of a language identifier to a language table URL. The language table URL defines the character code points that can be used for the language identifier. The elements used for the IDN table reference object is defined in this section. The `<csvIDN:contents>` child element of the `<rde:contents>` element is used to hold the new or updated IDN table reference objects for the deposit. The `<csvIDN:deletes>` child element of the `<rde:deletes>` element is used to hold the deleted or purged IDN table reference objects for the deposit. Both the `<csvIDN:contents>` and `<csvIDN:deletes>` elements contain one or more `<rdeCsv:csv>` elements with a set of named CSV file definitions using the `<rdeCsv:csv>` "name" attribute.

5.5.2.1. `<csvIDN:contents>`

The `<csvIDN:contents>` is used to hold the new or updated IDN table reference object information for the deposit. The `<csvIDN:contents>` is split into separate CSV file definitions using named `<rdeCsv:csv>` elements with the "name" attribute. The following sections include the supported IDN table reference CSV file definitions.

5.5.2.1.1. "idnLanguage" CSV File Definition

The "idnLanguage" CSV File Definition defines the fields and CSV file references used for the IDN table reference object records.

The following "rdeCsv" fields, defined in Section 4.6.2.2, MUST be used in the "idnLanguage" `<rdeCsv:csv>` `<rdeCsv:fields>` element:

`<rdeCsv:fIdnTableId>` The language identifier that matches the values for the `<rdeCsv:fIdnTableId>` field element in the "domain" CSV File Definition (Section 5.1.2.1.1) files. The attribute "isRequired" MUST equal "true".

`<rdeCsv:fUrl>` URL that defines the character code points that can be used for `<csvDomain:fName>` field in the "domain" CSV File Definition Section 5.1.2.1.1 files. The attribute "isRequired" MUST equal "true".

Example of a "idnLanguage" <csvIDN:contents> <rdeCsv:csv> element.

```
...
<csvIDN:contents>
...
  <rdeCsv:csv name="idnLanguage" sep=",">
    <rdeCsv:fields>
      <rdeCsv:fIdnTableId isRequired="true"/>
      <rdeCsv:fUrl isRequired="true"/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="D6B0424F">
        idnLanguage-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvIDN:contents>
...
```

Example of the corresponding idnLanguage-YYYYMMDD.csv file. The file contains two IDN language records.

```
LANG-1,
http://www.iana.org/domains/idn-tables/tables/test_tab1_1.1.txt
LANG-2,
http://www.iana.org/domains/idn-tables/tables/test_tab2_1.1.txt
```

5.5.2.2. <csvIDN:deletes>

The <csvIDN:deletes> is used to hold the deleted IDN table reference objects in a Differential or Incremental Deposit. The <csvIDN:deletes> is split into separate CSV file definitions using named <rdeCsv:csv> elements with the "name" attribute. The following section defines the supported IDN table reference deletes CSV file definition.

5.5.2.2.1. "idnLanguage" Deletes CSV File Definition

The following "idnLanguage" field elements MUST be used in the deletes "idnLanguage" <rdeCsv:csv> <rdeCsv:fields> element:

```
<rdeCsv:fIdnTableId> The language identifier that matches the values
  for the <rdeCsv:fIdnTableId> field element in the "domain" CSV
  File Definition (Section 5.1.2.1.1) files. The attribute
  "isRequired" MUST equal "true".
```

Example of a "idnLanguage" <csvIDN:deletes> <rdeCsv:csv> element.

```
...
<csvIDN:deletes>
...
  <rdeCsv:csv name="idnLanguage">
    <rdeCsv:fields>
      <rdeCsv:fIdnTableId isRequired="true"/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="4A28A569">
        idnLanguage-delete-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvIDN:deletes>
...
```

Example of the idnLanguage-delete-YYYYMMDD.csv file. The file contains one IDN language record.

LANG-2

5.6. NNDN Object

An NNDN (NNDN's not domain name) can be used to store registry reserved names or (blocked, withheld or mirrored) IDN variants.

Domain Name Registries may maintain domain names without their being persisted as domain objects in the registry system, for example, a list of reserved names not available for registration. The NNDN is a lightweight domain-like object that is used to escrow domain names not maintained as domain name objects.

A domain name can only exist as a domain name object or an NNDN object, but not both.

The NNDN object supports both the XML and the CSV Model, defined in the Models (Section 2) section. The elements used for both models are defined in the following sections.

5.6.1. XML Model

There are two elements used in the data escrow of the NNDN objects for the XML model including the <rdeNNDN:NNDN>, under the

<rde:contents> element, and the <rdeNNDN:delete> element, under the <rde:deletes> element.

A <rdeNNDN:NNDN> element substitutes for the <rdeNNDN:abstractNNDN> abstract element to define a concrete definition of an NNDN. The <rdeNNDN:abstractDomain> element can be replaced by other NNDN definitions using the XML schema substitution groups feature.

5.6.1.1. <rdeNNDN:NNDN> object

The <rdeNNDN:NNDN> element contains the following child elements:

- o An <aName> element that contains the fully-qualified qualified name of the NNDN. For IDNs the A-Label is used (see [RFC5891], Section 4.4).
- o An OPTIONAL <uName> element that contains the fully-qualified name of the NNDN in Unicode character set. It MUST be provided if available.
- o An OPTIONAL <idnTableId> element that references the IDN Table used for the NNDN. This corresponds to the "id" attribute of the <idnTableRef> element. This element MUST be present if the NNDN is an IDN.
- o An OPTIONAL <originalName> element is used to indicate that the NNDN is used for an IDN variant. This element contains the domain name used to generate the IDN variant.
- o A <nameState> element that indicates the state of the NNDN: blocked, withheld or mirrored.
 - * If an NNDN is considered undesirable for registration (i.e., unavailable for allocation to anyone), then the NNDN will be tagged as "blocked".
 - * If an NNDN is considered a potential registration of a domain name object for a registrant, then the NNDN will be tagged as "withheld". This status is only used when the NNDN is used for an IDN variant.
 - * If an NNDN is considered a mirrored IDN variant of a domain name object, then the NNDN will be tagged as "mirrored". A mirroringNS attribute is used to specify if the mirrored IDN variant uses the NS mirror mechanism, meaning that the activated variant domain name (i.e., NNDN) is delegated in the DNS using the same NS records as in the <originalName>. The default value of mirroringNS is true. If another mechanism

such as DNAME is used, the value of mirroringNS attribute MUST be false.

- o An OPTIONAL <crDate> element that contains the date and time of the NNDN object creation.

Example of an <rdeNNDN:NNDN> object:

```
...
<rdeNNDN:NNDN>
  <rdeNNDN:aName>xn--exempl-gva.example</rdeNNDN:aName>
  <rdeNNDN:idnTableId>pt-BR</rdeNNDN:idnTableId>
  <rdeNNDN:originalName>example.example</rdeNNDN:originalName>
  <rdeNNDN:nameState>withheld</rdeNNDN:nameState>
  <rdeNNDN:crDate>2005-04-23T11:49:00.0Z</rdeNNDN:crDate>
</rdeNNDN:NNDN>
...
```

5.6.1.2. <rdeNNDN:delete> object

The <rdeNNDN:delete> element contains the NNDN that was deleted, i.e., the <aName>.

Example of an <rdeNNDN::delete> object:

```
...
<rde:deletes>
  ...
  <rdeNNDN:delete>
    <rdeNNDN:aName>xn--pingino-q2a.example</rdeNNDN:aName>
  </rdeNNDN:delete>
  ...
</rde:deletes>
...
```

5.6.2. CSV Model

For the CSV Model of the NNDN object, the <csvNNDN:contents> child element of the <rde:contents> element is used to hold the new or updated NNDN objects for the deposit. The <csvNNDN:deletes> child element of the <rde:deletes> element is used to hold the deleted or purged NNDN objects for the deposit. Both the <csvNNDN:contents> and <csvNNDN:deletes> elements contain one or more <rdeCsv:csv> elements with a set of named CSV file definitions using the <rdeCsv:csv> "name" attribute.

5.6.2.1. <csvNNDN:contents>

The <csvNNDN:contents> is used to hold the new or updated NNDN object information for the deposit. The <csvNNDN:contents> is split into separate CSV file definitions using named <rdeCsv:csv> elements with the "name" attribute. The following sections include the supported NNDN CSV file definitions.

5.6.2.1.1. "NNDN" CSV File Definition

The "NNDN" CSV File Definition defines the fields and CSV file references used for the NNDN object records.

The following "csvNNDN" field elements MUST be used in the "NNDN" <rdeCsv:csv> <rdeCsv:fields> element:

<csvNNDN:fAName> Fully-qualified name of the NNDN with type="eppcom:labelType" and isRequired="true". For IDNs the A-Label is used (see [RFC5891], Section 4.4).

<csvNNDN:fNameState> State of the NNDN: blocked or withheld with type="rdeNNDN:nameState" and isRequired="true". See Section 5.6.1.1 for a description of the possible values for the <rdeNNDN:nameState> element.

The following field elements MAY be used in the "NNDN" <rdeCsv:csv> <rdeCsv:fields> element:

<csvNNDN:fOriginalName> Domain name used to generate the IDN variant with type="eppcom:labelType".

<csvNNDN:fMirroringNS> Defines whether the "mirroring" <csvNNDN:fNameState> uses the NS mirror mechanism, as described for the <rdeNNDN:nameState> "mirroringNS" attribute in Section 5.6.1.1, with type="boolean". If the field element is not defined the default value is "true".

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.6.2.2), MAY be used in the "NNDN" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fCrDate> Created date and time of the NNDN object.

<rdeCsv:fUName> Name of the NNDN in Unicode character set for the <csvNNDN:fAName> field element.

<rdeCsv:fIdnTableId> IDN Table Identifier for the NNDN that matches an IDN Table Reference Object record, as defined in Section 5.5.2.

Example of an "NNDN" <csvNNDN:contents> <rdeCsv:csv> element:

```
...
<csvNNDN:contents>
...
  <rdeCsv:csv name="NNDN" sep=",">
    <rdeCsv:fields>
      <csvNNDN:fAName/>
      <rdeCsv:fIdnTableId/>
      <csvNNDN:fOriginalName/>
      <csvNNDN:fNameState/>
      <csvNNDN:fMirroringNS/>
      <rdeCsv:fCrDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="085A7CE4">
        NNDN-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvNNDN:contents>
...
```

Example of the corresponding NNDN-YYYYMMDD.csv file. The file contains two NNDN records for an IDN with one blocked variant and one mirrored variant.

```
xn--bc456-3ve.example,LANG-1,xn--bc123-3ve.example,
blocked,,2005-04-23T11:49:00.0Z
xn--bc789-3ve.example,LANG-1,xn--bc123-3ve.example,
mirrored,1,2005-04-23T11:49:00.0Z
```

5.6.2.2. <csvNNDN:deletes>

The <csvNNDN:deletes> is used to hold the deleted NNDN objects in a Differential or Incremental Deposit. The <csvNNDN:deletes> is split into separate CSV file definitions using named <rdeCsv:csv> elements with the "name" attribute. The following section defines the supported NNDN deletes CSV file definition.

5.6.2.2.1. "NNDN" Deletes CSV File Definition

The following "NNDN" field elements MUST be used in the deletes "NNDN" <rdeCsv:csv> <rdeCsv:fields> element:

`<csvNNDN:fAName>` Fully-qualified name of the NNDN with `type="eppcom:labelType"` and `isRequired="true"`.

Example of an "NNDN" `<csvNNDN:deletes>` `<rdeCsv:csv>` element.

```
...
<csvNNDN:deletes>
...
  <rdeCsv:csv name="NNDN">
    <rdeCsv:fields>
      <csvNNDN:fAName/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="A41F1D9B">
        NNDN-delete-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
...
</csvNNDN:deletes>
...
```

Example of the corresponding NNDN-delete-YYYYMMDD.csv file. The file contains one NNDN records.

```
xn--bc456-3ve.example
```

5.7. EPP Parameters Object

The EPP Parameters Object is a pseudo-object that defines the set of object and object extension services supported by the registry, as defined in [RFC5730]. The EPP Parameters Object is only defined as XML but could be used in the XML model or CSV model. The EPP Parameters Object is defined using the `<rdeEppParams:eppParams>` element. The EPP Parameters Object SHOULD be included if the registry supports EPP. A maximum of one EPP Parameters Object MUST exist at a certain point in time (watermark).

The syntax and content of the `<rdeEppParams:eppParams>` children elements is as explained in section 2.4 of [RFC5730]. The children of the `<eppParams>` are as follows:

- o One or more `<version>` elements that indicate the EPP versions supported by the registry.
- o One or more `<lang>` elements that indicate the identifiers of the text response languages supported by the registry's EPP server.

- o One or more <objURI> elements that contain namespace URIs representing the objects that the registry's EPP server is capable of managing.
- o An OPTIONAL <svcExtension> element that contains one or more <extURI> elements that contain namespace URIs representing object extensions supported by the registry's EPP server.
- o A <dcP> element that contains child elements used to describe the server's privacy policy for data collection and management. See section 2.4 of [RFC5730] for more details.

Example of <eppParams> element object:

```
...
<rdeEppParams:eppParams>
  <rdeEppParams:version>1.0</rdeEppParams:version>
  <rdeEppParams:lang>en</rdeEppParams:lang>
  <rdeEppParams:objURI>urn:ietf:params:xml:ns:domain-1.0
    </rdeEppParams:objURI>
  <rdeEppParams:objURI>urn:ietf:params:xml:ns:contact-1.0
    </rdeEppParams:objURI>
  <rdeEppParams:objURI>urn:ietf:params:xml:ns:host-1.0
    </rdeEppParams:objURI>
  <rdeEppParams:svcExtension>
    <epp:extURI>urn:ietf:params:xml:ns:rgp-1.0</epp:extURI>
    <epp:extURI>urn:ietf:params:xml:ns:secDNS-1.1</epp:extURI>
  </rdeEppParams:svcExtension>
  <rdeEppParams:dcP>
  <epp:access><epp:all/></epp:access>
    <epp:statement>
      <epp:purpose>
        <epp:admin/>
        <epp:prov/>
      </epp:purpose>
      <epp:recipient>
        <epp:ours/>
        <epp:public/>
      </epp:recipient>
      <epp:retention>
        <epp:stated/>
      </epp:retention>
    </epp:statement>
  </rdeEppParams:dcP>
</rdeEppParams:eppParams>
...
```

5.8. Policy Object

The Policy object is a pseudo-object that is used to specify which OPTIONAL elements from the XML Model are REQUIRED based on the business model of the registry. For the CSV Model, the OPTIONAL "isRequired" attribute of the <rdeCsv:field> elements, defined in Section 4.6.2.1, is used to specify which OPTIONAL fields are REQUIRED based on the business model of the registry.

5.8.1. <rdePolicy:policy> object

The OPTIONAL <policy> contains the following attributes:

- o An <element> that defines that the referenced <element> is REQUIRED.
- o <scope> that defines the XPath (see, [W3C.REC-xpath-31-20170321]) of the element referenced by <element>.

Example of <rdePolicy:policy> object:

```
...  
<rdePolicy:policy scope="//rde:deposit/rde:contents/rdeDomain:domain"  
  element="rdeDomain:registrant" />  
...
```

5.9. Header Object

The Header Object is a pseudo-object that is used to specify the number of objects in the repository at a specific point in time (watermark) regardless of the type of deposit: Differential, Full or Incremental Deposit. The Header Object may also be used to provide additional information on the contents of the deposit. The Header Object is only defined as XML but one header object MUST always be present per escrow deposit regardless of using XML Model or CSV Model. The Header Object is defined using the <rdeHeader:header> element.

5.9.1. <rdeHeader:header> object

The <rdeHeader:header> contains the following elements:

- o A choice of one of the elements defined in the "repositoryTypeGroup" group element that indicates the unique identifier for the repository being escrowed. Possible elements are:

- * A `<rdeHeader:tld>` element that defines TLD or the RCDN being escrowed in the case of a Registry data escrow deposit. For IDNs the A-Label is used (see [RFC5891], Section 4.4).
- * A `<rdeHeader:registrar>` element that defines the Registrar ID corresponding to a Registrar data escrow deposit. In the case of an ICANN-accredited Registrar, the `<rdeHeader:registrar>` element MUST be the IANA Registrar ID assigned by ICANN.
- * A `<rdeHeader:ppsp>` element that defines the provider ID corresponding to a Privacy and Proxy Services Provider data escrow deposit. In the case of an ICANN-accredited Privacy and Proxy Services Provider, the `<rdeHeader:ppsp>` element MUST be the unique ID assigned by ICANN.
- * A `<rdeHeader:reseller>` element that defines the provider ID corresponding to a Reseller data escrow deposit.
- o A `<count>` element that contains the number of objects in the SRS at a specific point in time (watermark) regardless of the type of deposit: Differential, Full or Incremental. The `<count>` element supports the following attributes:
 - * A "uri" attribute reflects the XML namespace URI of the primary objects for the XML Model and CSV Model. For example, the "uri" is set to "urn:ietf:params:xml:ns:rdeDomain-1.0" for domain name objects using the XML Model, and the "uri" is set to "urn:ietf:params:xml:ns:csvDomain-1.0" for domain name objects using the CSV Model.
 - * An OPTIONAL "rcdn" attribute indicates the RCDN of the objects included in the `<count>` element. For IDNs the A-Label is used [RFC5891], Section 4.4. If the "rcdn" attribute is present, the value of the `<count>` element must include only objects related to registrations in the same and lower levels. For example in a data escrow deposit for the .EXAMPLE TLD, a value of "example" in the "rcdn" attribute within the `<count>` element indicates the number of objects in the TLD including objects in other RCDNs within the TLD, whereas a value of "com.example" indicates the number of elements for objects under "com.example" and lower levels. Omitting the "rcdn" attribute indicates that the total includes all objects of the specified "uri" in the repository (e.g. the TLD, Registrar, or PPSP).
 - * An OPTIONAL "registrarId" attribute indicates the identifier of the sponsoring Registrar of the objects included in the `<count>` element. In the case of an ICANN-accredited Registrar, the value MUST be the IANA Registrar ID assigned by ICANN.

- o An OPTIONAL <contentTag> element that contains a tag that defines the expected content in the deposit. The producer and consumer of the deposits will coordinate the set of possible <contentTag> element values.

Example of <rdeHeader:header> object referencing only the XML Model objects:

```
...
<rdeHeader:header>
  <rdeHeader:tld>test</rdeHeader:tld>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:rdeDomain-1.0">2</rdeHeader:count>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:rdeHost-1.0">1</rdeHeader:count>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:rdeContact-1.0">1</rdeHeader:count>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:rdeRegistrar-1.0">1
  </rdeHeader:count>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:rdeIDN-1.0">1</rdeHeader:count>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:rdeNNDN-1.0">1</rdeHeader:count>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:rdeEppParams-1.0">1
  </rdeHeader:count>
</rdeHeader:header>
...
```

Example of <rdeHeader:header> object referencing the CSV and XML Model objects:

```
...
<rdeHeader:header>
  <rdeHeader:tld>test</rdeHeader:tld>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:csvDomain-1.0">2</rdeHeader:count>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:csvHost-1.0">1</rdeHeader:count>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:csvContact-1.0">1</rdeHeader:count>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:csvRegistrar-1.0">1
  </rdeHeader:count>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:csvIDN-1.0">1</rdeHeader:count>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:csvNNDN-1.0">1</rdeHeader:count>
  <rdeHeader:count
    uri="urn:ietf:params:xml:ns:rdeEppParams-1.0">1
  </rdeHeader:count>
</rdeHeader:header>
...
```

5.10. DNRD Common Objects Collection

The DNRD Common Objects Collection contains data structures referenced by two or more of the main objects in the XML model.

6. RDE IDN Variants handling

Depending on the Registration Policy of the Registry, for a domain name there may be multiple variant names. See [variantTLDsReport] for further detail on IDN variants.

A registry could choose to escrow IDN variants as domains or NNDN objects. A specific IDN variant can be represented in the escrow deposit, as a domain or as an NNDN object, but not both.

If using domain objects to represent IDN variants, the normal behavior during restoration of an SRS based on an escrow deposit is to restore the IDN variants as a mirrored variant. If the registration data of the IDN variant is different from the original name, the details of this specific implementation MUST be described in the IDN policy document.

An NNDN or a domain name are explicit representations of an IDN variant while an IDN variant computed based on an algorithm is an implicit representation. Explicit representation of an IDN variant takes precedence over an implicit representation.

7. Profile

Different business models of registries exist, therefore the registry is responsible for defining a profile that matches its particular business model. The profile mechanism allows a registry to extend this specification.

A profile is the process of:

1. Extending base objects with the mechanisms defined for XML and CSV models.
 - * In the case of the XML model, abstract elements could be use to extend the following objects: <domain>, <host>, <contact>, <NNDN> and <registrar> using XML schema substitution groups feature.
2. Defining a <policy> object to specify which OPTIONAL elements of this base specification is required based on the business model of the registry. An example is the <registrant> element that is usually REQUIRED but it is specified as OPTIONAL in this specification to support some existing business models.
3. Adding new escrowed objects using the <rde:contents> and <rde:deletes> elements.
4. Providing the XML schemas to third parties that require them to validate the escrow deposits.

8. Data escrow agent extended verification process

A Data Escrow Agent SHOULD perform an extended verification process that starts by creating a dataset to be tested by following section 5.2 in [I-D.ietf-regext-data-escrow].

The following are the minimum suggested tests on the dataset:

- o Validate the escrow deposits using the definition agreed with the registry.
 - * In the case of the XML model, the contents of the escrow deposits MUST be validated using the XML schemas of the profile.

- o Count the objects and validate that the number of objects is equal to the number objects reported in the <header> element of the escrow deposit of that point in time (watermark).
- o All contact objects linked to domain names MUST be present.
- o All registrars objects linked to other objects MUST be present.
- o No domain name exists as both a domain name and an NNDN.
- o The elements listed as required in the <policy> element MUST be present.
- o All idnTableRef definitions linked from other objects MUST be present.
- o If an EPP Parameters Object was escrowed in the past, one and only one EPP Parameters Object MUST be present.
- o The watermark is not in the future.

9. Formal Syntax

This standard is specified in XML Schema notation. The formal syntax presented here is a complete schema representation suitable for automated validation.

The <CODE BEGINS> and <CODE ENDS> tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.

9.1. RDE CSV Schema

```
<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:rdeCsv-1.0"
  xmlns:rdeCsv="urn:ietf:params:xml:ns:rdeCsv-1.0"
  xmlns:rde="urn:ietf:params:xml:ns:rde-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:rde-1.0" />
  <annotation>
    <documentation>
```

```

    Registry Data Escrow Comma-Separated Values (CSV)
</documentation>
</annotation>
<!-- csv content element -->
<element name="csv"
      type="rdeCsv:csvType" />
<!-- Definition of CSV file -->
<complexType name="csvType">
  <sequence>
    <element name="fields"
      type="rdeCsv:fieldsType" />
    <element name="files"
      type="rdeCsv:filesType" />
  </sequence>
  <attribute name="name"
    type="token"
    use="required" />
  <attribute name="sep"
    type="rdeCsv:sepType"
    default="," />
</complexType>
<!-- field separator must be a single character -->
<simpleType name="sepType">
  <restriction base="string">
    <minLength value="1" />
    <maxLength value="1" />
  </restriction>
</simpleType>
<!-- Abstract field type -->
<element name="field"
  type="rdeCsv:fieldType"
  abstract="true" />
<complexType name="fieldType">
  <sequence />
</complexType>
<!-- fieldType with optional value (isRequired=false) -->
<complexType name="fieldOptionalType">
  <complexContent>
    <extension base="rdeCsv:fieldType">
      <sequence />
      <attribute name="isRequired"
        type="boolean"
        default="false" />
      <attribute name="parent"
        type="boolean"
        default="false" />
    </extension>
  </complexContent>
</complexType>

```



```
</complexType>
<!-- fieldType with required value (isRequired=false) -->
<complexType name="fieldRequiredType">
  <complexContent>
    <extension base="rdeCsv:fieldType">
      <sequence />
      <attribute name="isRequired"
        type="boolean"
        default="true" />
      <attribute name="parent"
        type="boolean"
        default="false" />
    </extension>
  </complexContent>
</complexType>
<!-- Concrete field types -->
<!-- UTF-8 Name field (e.g. domain name) -->
<element name="fUName"
  type="rdeCsv:fNameType"
  substitutionGroup="rdeCsv:field" />
<complexType name="fNameType">
  <complexContent>
    <extension base="rdeCsv:fieldOptionalType">
      <sequence />
      <attribute name="type"
        type="token"
        default="eppcom\:labelType" />
    </extension>
  </complexContent>
</complexType>
<complexType name="fNameRequiredType">
  <complexContent>
    <extension base="rdeCsv:fieldRequiredType">
      <sequence />
      <attribute name="type"
        type="token"
        default="eppcom\:labelType" />
    </extension>
  </complexContent>
</complexType>
<!-- Registry Object Identifier (roid) field -->
<element name="fRoid"
  type="rdeCsv:fRoidType"
  substitutionGroup="rdeCsv:field" />
<complexType name="fRoidType">
  <complexContent>
    <extension base="rdeCsv:fieldRequiredType">
      <sequence />
```

```

        <attribute name="type"
                  type="token"
                  default="eppcom\:roidType" />
    </extension>
</complexContent>
</complexType>
<!-- Registrant field -->
<element name="fRegistrant"
          type="rdeCsv:fRegistrantType"
          substitutionGroup="rdeCsv:field" />
<complexType name="fRegistrantType">
  <complexContent>
    <extension base="rdeCsv:fieldOptionalType">
      <sequence />
      <attribute name="type"
                  type="token"
                  default="eppcom\:clIDType" />
    </extension>
  </complexContent>
</complexType>
<!-- Object Status Description -->
<element name="fStatusDescription"
          type="rdeCsv:fNormalizedStringType"
          substitutionGroup="rdeCsv:field" />
<!-- clID fields (fClID, fCrID, fUpID) -->
<!-- Identifier of the client that sponsors the object -->
<element name="fClID"
          type="rdeCsv:fClIDRequiredType"
          substitutionGroup="rdeCsv:field" />
<!-- Identifier of registrar of client
that created the object -->
<element name="fCrRr"
          type="rdeCsv:fClIDType"
          substitutionGroup="rdeCsv:field" />
<!-- Identifier of the client that created the object -->
<element name="fCrID"
          type="rdeCsv:fClIDType"
          substitutionGroup="rdeCsv:field" />
<!-- Identifier of registrar of client that
updated the object -->
<element name="fUpRr"
          type="rdeCsv:fClIDType"
          substitutionGroup="rdeCsv:field" />
<!-- Identifier of the client that updated the object -->
<element name="fUpID"
          type="rdeCsv:fClIDType"
          substitutionGroup="rdeCsv:field" />
<!-- Identifier of registrar of client that

```

```

requested the transfer -->
  <element name="fReRr"
    type="rdeCsv:fClIDRequiredType"
    substitutionGroup="rdeCsv:field" />
  <!-- Identifier of the client that requested
the transfer -->
  <element name="fReID"
    type="rdeCsv:fClIDType"
    substitutionGroup="rdeCsv:field" />
  <!-- Identifier of registrar client that
should take or took action -->
  <element name="fAcRr"
    type="rdeCsv:fClIDRequiredType"
    substitutionGroup="rdeCsv:field" />
  <!-- Identifier of the client that should take or
took action -->
  <element name="fAcID"
    type="rdeCsv:fClIDType"
    substitutionGroup="rdeCsv:field" />
  <complexType name="fClIDType">
    <complexContent>
      <extension base="rdeCsv:fieldOptionalType">
        <sequence />
        <attribute name="type"
          type="token"
          default="eppcom\:clIDType" />
      </extension>
    </complexContent>
  </complexType>
  <complexType name="fClIDRequiredType">
    <complexContent>
      <extension base="rdeCsv:fieldRequiredType">
        <sequence />
        <attribute name="type"
          type="token"
          default="eppcom\:clIDType" />
      </extension>
    </complexContent>
  </complexType>
  <!-- dateTime fields (fCrDate, fUpDate, fExDate) -->
  <element name="fCrDate"
    type="rdeCsv:fDateTimeType"
    substitutionGroup="rdeCsv:field" />
  <element name="fUpDate"
    type="rdeCsv:fDateTimeType"
    substitutionGroup="rdeCsv:field" />
  <element name="fExDate"
    type="rdeCsv:fDateTimeType"

```

```

        substitutionGroup="rdeCsv:field" />
<!-- Date and time that transfer was requested -->
<element name="fReDate"
    type="rdeCsv:fRequiredDateTimeType"
    substitutionGroup="rdeCsv:field" />
<!-- Date and time of a required or completed response -->
<element name="fAcDate"
    type="rdeCsv:fRequiredDateTimeType"
    substitutionGroup="rdeCsv:field" />
<element name="fTrDate"
    type="rdeCsv:fDateTimeType"
    substitutionGroup="rdeCsv:field" />
<complexType name="fDateTimeType">
    <complexContent>
        <extension base="rdeCsv:fieldOptionalType">
            <sequence />
            <attribute name="type"
                type="token"
                default="dateTime" />
        </extension>
    </complexContent>
</complexType>
<complexType name="fRequiredDateTimeType">
    <complexContent>
        <extension base="rdeCsv:fieldRequiredType">
            <sequence />
            <attribute name="type"
                type="token"
                default="dateTime" />
        </extension>
    </complexContent>
</complexType>
<!-- boolean type -->
<complexType name="fBooleanType">
    <complexContent>
        <extension base="rdeCsv:fieldOptionalType">
            <sequence />
            <attribute name="type"
                type="token"
                default="boolean" />
        </extension>
    </complexContent>
</complexType>
<complexType name="fRequiredBooleanType">
    <complexContent>
        <extension base="rdeCsv:fieldRequiredType">
            <sequence />
            <attribute name="type"

```

```
                type="token"
                default="boolean" />
        </extension>
    </complexContent>
</complexType>
<!-- unsignedByte type -->
<complexType name="fUnsignedByteType">
    <complexContent>
        <extension base="rdeCsv:fieldOptionalType">
            <sequence />
            <attribute name="type"
                type="token"
                default="unsignedByte" />
        </extension>
    </complexContent>
</complexType>
<complexType name="fRequiredUnsignedByteType">
    <complexContent>
        <extension base="rdeCsv:fieldRequiredType">
            <sequence />
            <attribute name="type"
                type="token"
                default="unsignedByte" />
        </extension>
    </complexContent>
</complexType>
<!-- unsignedShort type -->
<complexType name="fUnsignedShortType">
    <complexContent>
        <extension base="rdeCsv:fieldOptionalType">
            <sequence />
            <attribute name="type"
                type="token"
                default="unsignedShort" />
        </extension>
    </complexContent>
</complexType>
<complexType name="fRequiredUnsignedShortType">
    <complexContent>
        <extension base="rdeCsv:fieldRequiredType">
            <sequence />
            <attribute name="type"
                type="token"
                default="unsignedShort" />
        </extension>
    </complexContent>
</complexType>
<!-- hexBinary type -->
```

```
<complexType name="fHexBinaryType">
  <complexContent>
    <extension base="rdeCsv:fieldOptionalType">
      <sequence />
      <attribute name="type"
        type="token"
        default="hexBinary" />
    </extension>
  </complexContent>
</complexType>
<complexType name="fRequiredHexBinaryType">
  <complexContent>
    <extension base="rdeCsv:fieldRequiredType">
      <sequence />
      <attribute name="type"
        type="token"
        default="hexBinary" />
    </extension>
  </complexContent>
</complexType>
<!-- language type -->
<element name="fLang"
  type="rdeCsv:fLangType"
  substitutionGroup="rdeCsv:field" />
<complexType name="fLangType">
  <complexContent>
    <extension base="rdeCsv:fieldOptionalType">
      <sequence />
      <attribute name="type"
        type="token"
        default="language" />
    </extension>
  </complexContent>
</complexType>
<!-- IDN Table Identifier -->
<element name="fIdnTableId"
  type="rdeCsv:fTokenType"
  substitutionGroup="rdeCsv:field" />
<!-- State of the most recent transfer request -->
<element name="fTrStatus"
  type="rdeCsv:fTrStatusType"
  substitutionGroup="rdeCsv:field" />
<complexType name="fTrStatusType">
  <complexContent>
    <extension base="rdeCsv:fieldRequiredType">
      <sequence />
      <attribute name="type"
        type="token"
```

```

        default="eppcom\:trStatusType" />
    </extension>
</complexContent>
</complexType>
<!-- General token type -->
<complexType name="fTokenType">
    <complexContent>
        <extension base="rdeCsv:fieldOptionalType">
            <sequence />
            <attribute name="type"
                type="token"
                default="token" />
        </extension>
    </complexContent>
</complexType>
<!-- General normalizedString type -->
<complexType name="fNormalizedStringType">
    <complexContent>
        <extension base="rdeCsv:fieldOptionalType">
            <sequence />
            <attribute name="type"
                type="token"
                default="normalizedString" />
        </extension>
    </complexContent>
</complexType>
<!-- positive integer type -->
<complexType name="fPositiveIntegerType">
    <complexContent>
        <extension base="rdeCsv:fieldOptionalType">
            <sequence />
            <attribute name="type"
                type="token"
                default="positiveInteger" />
        </extension>
    </complexContent>
</complexType>
<!-- Custom / extension field type -->
<element name="fCustom"
    type="rdeCsv:fCustomType"
    substitutionGroup="rdeCsv:field" />
<complexType name="fCustomType">
    <complexContent>
        <extension base="rdeCsv:fieldOptionalType">
            <sequence />
            <attribute name="name"
                type="token" />
            <attribute name="type"

```

```

        type="token"
        default="token" />
    </extension>
</complexContent>
</complexType>
<!-- Ordered list of field definitions for the csv -->
<complexType name="fieldsType">
    <sequence maxOccurs="unbounded">
        <element ref="rdeCsv:field" />
    </sequence>
</complexType>
<!-- List of files -->
<complexType name="filesType">
    <sequence>
        <element name="file"
            type="rdeCsv:fileType"
            maxOccurs="unbounded" />
    </sequence>
</complexType>
<!-- File definition -->
<complexType name="fileType">
    <complexContent>
        <extension base="token">
            <attribute name="compression"
                type="token" />
            <attribute name="encoding"
                type="token"
                default="UTF-8" />
            <attribute name="cksum"
                type="token" />
            <attribute name="cksumAlg"
                type="token"
                default="CRC32" />
        </extension>
    </complexContent>
</complexType>
<!-- URL fields -->
<element name="fUrl"
    type="rdeCsv:anyURIType"
    substitutionGroup="rdeCsv:field" />
<complexType name="anyURIType">
    <complexContent>
        <extension base="rdeCsv:fieldOptionalType">
            <sequence />
            <attribute name="type"
                type="token"
                default="anyURI" />
        </extension>
    </complexContent>

```



```

        </complexContent>
    </complexType>
    <!--
    End of schema.
    -->
</schema>
<CODE ENDS>

```

9.2. RDE Domain Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:rdeDomain-1.0"
  xmlns:rdeDomain="urn:ietf:params:xml:ns:rdeDomain-1.0"
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns:rdeIDN="urn:ietf:params:xml:ns:rdeIDN-1.0"
  xmlns:rgp="urn:ietf:params:xml:ns:rgp-1.0"
  xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1"
  xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
  xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
  xmlns:rdeDnrdCommon="urn:ietf:params:xml:ns:rdeDnrdCommon-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0" />
  <import namespace="urn:ietf:params:xml:ns:domain-1.0" />
  <import namespace="urn:ietf:params:xml:ns:secDNS-1.1" />
  <import namespace="urn:ietf:params:xml:ns:rgp-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rde-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rdeIDN-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rdeDnrdCommon-1.0" />
  <annotation>
    <documentation>
      Registry Data Escrow Domain provisioning schema
    </documentation>
  </annotation>
  <element name="abstractDomain"
    type="rdeDomain:abstractContentType"
    substitutionGroup="rde:content"
    abstract="true" />
  <element name="domain"
    substitutionGroup="rdeDomain:abstractDomain" />
  <element name="delete"
    type="rdeDomain:deleteType"
    substitutionGroup="rde:delete" />
  <!-- Content Type -->
  <complexType name="abstractContentType">
    <complexContent>
      <extension base="rde:contentType">

```

```
<sequence>
  <element name="name"
    type="eppcom:labelType" />
  <element name="roid"
    type="eppcom:roidType" />
  <element name="uName"
    type="eppcom:labelType"
    minOccurs="0" />
  <element name="idnTableId"
    type="rdeIDN:idType"
    minOccurs="0" />
  <element name="originalName"
    type="eppcom:labelType"
    minOccurs="0" />
  <element name="status"
    type="domain:statusType"
    maxOccurs="11" />
  <element name="rgpStatus"
    type="rgp:statusType"
    minOccurs="0"
    maxOccurs="unbounded" />
  <element name="registrant"
    type="eppcom:clIDType"
    minOccurs="0" />
  <element name="contact"
    type="domain:contactType"
    minOccurs="0"
    maxOccurs="unbounded" />
  <element name="ns"
    type="domain:nsType"
    minOccurs="0" />
  <element name="clID"
    type="eppcom:clIDType" />
  <element name="crRr"
    type="rdeDnrdCommon:rrType"
    minOccurs="0" />
  <element name="crDate"
    type="dateTime"
    minOccurs="0" />
  <element name="exDate"
    type="dateTime"
    minOccurs="0" />
  <element name="upRr"
    type="rdeDnrdCommon:rrType"
    minOccurs="0" />
  <element name="upDate"
    type="dateTime"
    minOccurs="0" />
```

```

        <element name="secDNS"
            type="secDNS:dsOrKeyType"
            minOccurs="0" />
        <element name="trDate"
            type="dateTime"
            minOccurs="0" />
        <element name="trnData"
            type="rdeDomain:transferDataType"
            minOccurs="0" />
    </sequence>
</extension>
</complexContent>
</complexType>
<complexType name="transferDataType">
    <sequence>
        <element name="trStatus"
            type="eppcom:trStatusType" />
        <element name="reRr"
            type="rdeDnrdCommon:rrType" />
        <element name="reDate"
            type="dateTime" />
        <element name="acRr"
            type="rdeDnrdCommon:rrType" />
        <element name="acDate"
            type="dateTime" />
        <element name="exDate"
            type="dateTime"
            minOccurs="0" />
    </sequence>
</complexType>
<!-- Delete Type -->
<complexType name="deleteType">
    <complexContent>
        <extension base="rde:deleteType">
            <sequence>
                <element name="name"
                    type="eppcom:labelType"
                    minOccurs="0"
                    maxOccurs="unbounded" />
            </sequence>
        </extension>
    </complexContent>
</complexType>
</schema>
<CODE ENDS>
```

9.3. CSV Domain Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:csvDomain-1.0"
  xmlns:csvDomain="urn:ietf:params:xml:ns:csvDomain-1.0"
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns:rdeCsv="urn:ietf:params:xml:ns:rdeCsv-1.0"
  xmlns:rgp="urn:ietf:params:xml:ns:rgp-1.0"
  xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1"
  xmlns:domain="urn:ietf:params:xml:ns:domain-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:domain-1.0" />
  <import namespace="urn:ietf:params:xml:ns:secDNS-1.1" />
  <import namespace="urn:ietf:params:xml:ns:rgp-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rde-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rdeCsv-1.0" />
  <annotation>
    <documentation>
      Domain Name Comma-Separated Values (CSV) Object
    </documentation>
  </annotation>
  <!--
  Child elements of the <rde:contents> object
  -->
  <element name="contents"
    type="csvDomain:contentType"
    substitutionGroup="rde:content" />
  <complexType name="contentType">
    <complexContent>
      <extension base="rde:contentType">
        <sequence>
          <element ref="rdeCsv:csv"
            maxOccurs="unbounded" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!--
  Child elements of the <rde:deletes> object
  -->
  <element name="deletes"

```

```

        type="csvDomain:deleteType"
        substitutionGroup="rde:delete" />
<complexType name="deleteType">
  <complexContent>
    <extension base="rde:deleteType">
      <sequence>
        <element ref="rdeCsv:csv"
          maxOccurs="unbounded" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- Domain name field -->
<element name="fName"
  type="rdeCsv:fNameRequiredType"
  substitutionGroup="rdeCsv:field" />
<!-- RGP status field -->
<element name="fRgpStatus"
  type="csvDomain:fRgpStatusType"
  substitutionGroup="rdeCsv:field" />
<complexType name="fRgpStatusType">
  <complexContent>
    <extension base="rdeCsv:fieldOptionalType">
      <sequence />
      <attribute name="type"
        type="token"
        default="rgp\:statusValueType" />
    </extension>
  </complexContent>
</complexType>
<!-- Contact type field -->
<element name="fContactType"
  type="csvDomain:fContactsTypeType"
  substitutionGroup="rdeCsv:field" />
<complexType name="fContactsTypeType">
  <complexContent>
    <extension base="rdeCsv:fieldRequiredType">
      <sequence />
      <attribute name="type"
        type="token"
        default="domain\:contactAttrType" />
    </extension>
  </complexContent>
</complexType>
<!-- DNSSEC field types -->
<!-- Maximum signature lifetime field -->
<element name="fMaxSigLife"
  type="csvDomain:fMaxSigLifeType"

```

```
        substitutionGroup="rdeCsv:field" />
<complexType name="fMaxSigLifeType">
  <complexContent>
    <extension base="rdeCsv:fieldOptionalType">
      <sequence />
      <attribute name="type"
        type="token"
        default="secDNS\:maxSigLifeType" />
    </extension>
  </complexContent>
</complexType>
<!-- Key tag field -->
<element name="fKeyTag"
  type="rdeCsv:fRequiredUnsignedShortType"
  substitutionGroup="rdeCsv:field" />
<!-- DS Algorithm field -->
<element name="fDsAlg"
  type="rdeCsv:fRequiredUnsignedByteType"
  substitutionGroup="rdeCsv:field" />
<!-- Digest type field -->
<element name="fDigestType"
  type="rdeCsv:fRequiredUnsignedByteType"
  substitutionGroup="rdeCsv:field" />
<!-- Digest field -->
<element name="fDigest"
  type="rdeCsv:fRequiredHexBinaryType"
  substitutionGroup="rdeCsv:field" />
<!-- Flags field -->
<element name="fFlags"
  type="rdeCsv:fRequiredUnsignedShortType"
  substitutionGroup="rdeCsv:field" />
<!-- Protocol field -->
<element name="fProtocol"
  type="rdeCsv:fRequiredUnsignedByteType"
  substitutionGroup="rdeCsv:field" />
<!-- Key Algorithm field -->
<element name="fKeyAlg"
  type="rdeCsv:fRequiredUnsignedByteType"
  substitutionGroup="rdeCsv:field" />
<!-- Public Key field -->
<element name="fPubKey"
  type="csvDomain:fPubKeyType"
  substitutionGroup="rdeCsv:field" />
<complexType name="fPubKeyType">
  <complexContent>
    <extension base="rdeCsv:fieldRequiredType">
      <sequence />
      <attribute name="type" />
    </extension>
  </complexContent>
</complexType>
```

```

        type="token"
        default="secDNS\:keyType" />
    </extension>
</complexContent>
</complexType>
<!-- Original Domain Name for Variant field -->
<element name="fOriginalName"
    type="rdeCsv:fNameType"
    substitutionGroup="rdeCsv:field" />
<!-- Domain status field -->
<element name="fStatus"
    type="csvDomain:fStatusType"
    substitutionGroup="rdeCsv:field" />
<!-- Domain status based on domain-1.0.xsd -->
<complexType name="fStatusType">
    <complexContent>
        <extension base="rdeCsv:fieldRequiredType">
            <sequence />
            <attribute name="type"
                type="token"
                default="domain\:statusValueType" />
        </extension>
    </complexContent>
</complexType>
<!--
End of schema.
-->
</schema>
<CODE ENDS>

```

9.4. RDE Host Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:rdeHost-1.0"
    xmlns:rdeHost="urn:ietf:params:xml:ns:rdeHost-1.0"
    xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
    xmlns:host="urn:ietf:params:xml:ns:host-1.0"
    xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
    xmlns:rdeDnrdCommon="urn:ietf:params:xml:ns:rdeDnrdCommon-1.0"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
    <import namespace="urn:ietf:params:xml:ns:eppcom-1.0" />
    <import namespace="urn:ietf:params:xml:ns:host-1.0" />
    <import namespace="urn:ietf:params:xml:ns:rde-1.0" />
    <import namespace="urn:ietf:params:xml:ns:rdeDnrdCommon-1.0" />
    <annotation>
        <documentation>

```

```
Registry Data Escrow Host provisioning schema
</documentation>
</annotation>
<element name="abstractHost"
  type="rdeHost:abstractContentType"
  substitutionGroup="rde:content"
  abstract="true" />
<element name="host"
  substitutionGroup="rdeHost:abstractHost" />
<element name="delete"
  type="rdeHost:deleteType"
  substitutionGroup="rde:delete" />
<!-- Content Type -->
<complexType name="abstractContentType">
  <complexContent>
    <extension base="rde:contentType">
      <sequence>
        <element name="name"
          type="eppcom:labelType" />
        <element name="roid"
          type="eppcom:roidType" />
        <element name="status"
          type="host:statusType"
          maxOccurs="7" />
        <element name="addr"
          type="host:addrType"
          minOccurs="0"
          maxOccurs="unbounded" />
        <element name="clID"
          type="eppcom:clIDType" />
        <element name="crRr"
          type="rdeDnrdCommon:rrType"
          minOccurs="0" />
        <element name="crDate"
          type="dateTime"
          minOccurs="0" />
        <element name="upRr"
          type="rdeDnrdCommon:rrType"
          minOccurs="0" />
        <element name="upDate"
          type="dateTime"
          minOccurs="0" />
        <element name="trDate"
          type="dateTime"
          minOccurs="0" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
```



```

</complexType>
<!-- Delete Type -->
<complexType name="deleteType">
  <complexContent>
    <extension base="rde:deleteType">
      <choice minOccurs="0"
        maxOccurs="unbounded">
        <element name="name"
          type="eppcom:labelType" />
        <element name="roid"
          type="eppcom:roidType" />
      </choice>
    </extension>
  </complexContent>
</complexType>
</schema>
<CODE ENDS>

```

9.5. CSV Host Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:csvHost-1.0"
  xmlns:csvHost="urn:ietf:params:xml:ns:csvHost-1.0"
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns:rdeCsv="urn:ietf:params:xml:ns:rdeCsv-1.0"
  xmlns:host="urn:ietf:params:xml:ns:host-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:host-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rde-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rdeCsv-1.0" />
  <annotation>
    <documentation>
      Host Comma-Separated Values (CSV) Object
    </documentation>
  </annotation>
  <!--
  Child elements of the <rde:contents> object
  -->
  <element name="contents"
    type="csvHost:contentType"
    substitutionGroup="rde:content" />

```

```
<complexType name="contentType">
  <complexContent>
    <extension base="rde:contentType">
      <sequence>
        <element ref="rdeCsv:csv"
          maxOccurs="unbounded" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<!--
Child elements of the <rde:deletes> object
-->
<element name="deletes"
  type="csvHost:deleteType"
  substitutionGroup="rde:delete" />
<complexType name="deleteType">
  <complexContent>
    <extension base="rde:deleteType">
      <sequence>
        <element ref="rdeCsv:csv"
          maxOccurs="unbounded" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- Host name field -->
<element name="fName"
  type="rdeCsv:fNameRequiredType"
  substitutionGroup="rdeCsv:field" />
<!-- IP address field -->
<element name="fAddr"
  type="csvHost:fAddrType"
  substitutionGroup="rdeCsv:field" />
<complexType name="fAddrType">
  <complexContent>
    <extension base="rdeCsv:fieldOptionalType">
      <sequence />
      <attribute name="type"
        type="token"
        default="host\:addrStringType" />
    </extension>
  </complexContent>
</complexType>
<!-- IP address version field -->
<element name="fAddrVersion"
  type="csvHost:fAddrVersionType"
  substitutionGroup="rdeCsv:field" />
```

```

<complexType name="fAddrVersionType">
  <complexContent>
    <extension base="rdeCsv:fieldOptionalType">
      <sequence />
      <attribute name="type"
        type="token"
        default="host\:ipType" />
    </extension>
  </complexContent>
</complexType>
<!-- Host status field -->
<element name="fStatus"
  type="csvHost:fStatusType"
  substitutionGroup="rdeCsv:field" />
<!-- Host status based on host-1.0.xsd -->
<complexType name="fStatusType">
  <complexContent>
    <extension base="rdeCsv:fieldRequiredType">
      <sequence />
      <attribute name="type"
        type="token"
        default="host\:statusValueType" />
    </extension>
  </complexContent>
</complexType>
<!--
End of schema.
-->
</schema>
<CODE ENDS>

```

9.6. RDE Contact Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:rdeContact-1.0"
  xmlns:rdeContact="urn:ietf:params:xml:ns:rdeContact-1.0"
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
  xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
  xmlns:rdeDnrdCommon="urn:ietf:params:xml:ns:rdeDnrdCommon-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:contact-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rde-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rdeDnrdCommon-1.0" />

```

```
<annotation>
  <documentation>
    Registry Data Escrow contact provisioning schema
  </documentation>
</annotation>
<element name="abstractContact"
  type="rdeContact:abstractContentType"
  substitutionGroup="rde:content"
  abstract="true" />
<element name="contact"
  substitutionGroup="rdeContact:abstractContact" />
<element name="delete"
  type="rdeContact:deleteType"
  substitutionGroup="rde:delete" />
<!-- Contact Type -->
<complexType name="abstractContentType">
  <complexContent>
    <extension base="rde:contentType">
      <sequence>
        <element name="id"
          type="eppcom:clIDType" />
        <element name="roid"
          type="eppcom:roidType" />
        <element name="status"
          type="contact:statusType"
          maxOccurs="7" />
        <element name="postalInfo"
          type="contact:postalInfoType"
          maxOccurs="2" />
        <element name="voice"
          type="contact:e164Type"
          minOccurs="0" />
        <element name="fax"
          type="contact:e164Type"
          minOccurs="0" />
        <element name="email"
          type="eppcom:minTokenType" />
        <element name="clID"
          type="eppcom:clIDType" />
        <element name="crRr"
          type="rdeDnrdCommon:rrType"
          minOccurs="0" />
        <element name="crDate"
          type="dateTime"
          minOccurs="0" />
        <element name="upRr"
          type="rdeDnrdCommon:rrType"
          minOccurs="0" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```
        <element name="upDate"
            type="dateTime"
            minOccurs="0" />
        <element name="trDate"
            type="dateTime"
            minOccurs="0" />
        <element name="trnData"
            type="rdeContact:transferDataType"
            minOccurs="0" />
        <element name="disclose"
            type="contact:discloseType"
            minOccurs="0" />
    </sequence>
</extension>
</complexContent>
</complexType>
<complexType name="transferDataType">
    <sequence>
        <element name="trStatus"
            type="eppcom:trStatusType" />
        <element name="reRr"
            type="rdeDnrdCommon:rrType" />
        <element name="reDate"
            type="dateTime" />
        <element name="acRr"
            type="rdeDnrdCommon:rrType" />
        <element name="acDate"
            type="dateTime" />
    </sequence>
</complexType>
<!-- Delete Type -->
<complexType name="deleteType">
    <complexContent>
        <extension base="rde:deleteType">
            <sequence>
                <element name="id"
                    type="eppcom:clIDType"
                    minOccurs="0"
                    maxOccurs="unbounded" />
            </sequence>
        </extension>
    </complexContent>
</complexType>
</schema>
<CODE ENDS>
```

9.7. CSV Contact Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:csvContact-1.0"
  xmlns:csvContact="urn:ietf:params:xml:ns:csvContact-1.0"
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns:rdeCsv="urn:ietf:params:xml:ns:rdeCsv-1.0"
  xmlns:contact="urn:ietf:params:xml:ns:contact-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:contact-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rde-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rdeCsv-1.0" />
  <annotation>
    <documentation>
      Contact Comma-Separated Values (CSV) Object
    </documentation>
  </annotation>
  <!--
  Child elements of the <rde:contents> object
  -->
  <element name="contents"
    type="csvContact:contentType"
    substitutionGroup="rde:content" />
  <complexType name="contentType">
    <complexContent>
      <extension base="rde:contentType">
        <sequence>
          <element ref="rdeCsv:csv"
            maxOccurs="unbounded" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!--
  Child elements of the <rde:deletes> object
  -->
  <element name="deletes"
    type="csvContact:deleteType"
    substitutionGroup="rde:delete" />
  <complexType name="deleteType">
    <complexContent>

```

```

    <extension base="rde:deleteType">
      <sequence>
        <element ref="rdeCsv:csv"
          maxOccurs="unbounded" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- Server-unique contact identifier field -->
<element name="fId"
  type="csvContact:fIdType"
  substitutionGroup="rdeCsv:field" />
<complexType name="fIdType">
  <complexContent>
    <extension base="rdeCsv:fieldRequiredType">
      <sequence />
      <attribute name="type"
        type="token"
        default="eppcom\:clIDType" />
    </extension>
  </complexContent>
</complexType>
<!-- Is Registrar Contact field -->
<element name="fIsRegistrarContact"
  type="rdeCsv:fBooleanType"
  substitutionGroup="rdeCsv:field" />
<!-- voice and fax telephone number fields -->
<element name="fVoice"
  type="csvContact:fE164StringType"
  substitutionGroup="rdeCsv:field" />
<element name="fFax"
  type="csvContact:fE164StringType"
  substitutionGroup="rdeCsv:field" />
<complexType name="fE164StringType">
  <complexContent>
    <extension base="rdeCsv:fieldOptionalType">
      <sequence />
      <attribute name="type"
        type="token"
        default="contact\:e164StringType" />
    </extension>
  </complexContent>
</complexType>
<!-- voice and fax telephone extension fields -->
<element name="fVoiceExt"
  type="rdeCsv:fTokenType"
  substitutionGroup="rdeCsv:field" />
<element name="fFaxExt"

```

```

        type="rdeCsv:fTokenType"
        substitutionGroup="rdeCsv:field" />
<!-- contact email address field -->
<element name="fEmail"
    type="csvContact:fEmailType"
    substitutionGroup="rdeCsv:field" />
<complexType name="fEmailType">
    <complexContent>
        <extension base="rdeCsv:fieldRequiredType">
            <sequence />
            <attribute name="type"
                type="token"
                default="eppcom:minTokenType" />
        </extension>
    </complexContent>
</complexType>
<!--
    Postal type field
    ("loc" = localized, "int" = internationalized)
-->
<element name="fPostalType"
    type="csvContact:fPostalTypeType"
    substitutionGroup="rdeCsv:field" />
<complexType name="fPostalTypeType">
    <complexContent>
        <extension base="rdeCsv:fieldRequiredType">
            <sequence />
            <attribute name="type"
                type="token"
                default="contact\:postalInfoEnumType" />
        </extension>
    </complexContent>
</complexType>
<!-- Standard postal line field -->
<complexType name="fPostalLineType">
    <complexContent>
        <extension base="rdeCsv:fieldRequiredType">
            <sequence />
            <attribute name="type"
                type="token"
                default="contact\:postalLineType" />
            <attribute name="isLoc"
                type="boolean" />
        </extension>
    </complexContent>
</complexType>
<!-- Standard optional postal line field -->
<complexType name="fOptPostalLineType">

```



```

    <complexContent>
      <extension base="rdeCsv:fieldOptionalType">
        <sequence />
        <attribute name="type"
          type="token"
          default="contact\:optPostalLineType" />
        <attribute name="isLoc"
          type="boolean" />
      </extension>
    </complexContent>
  </complexType>
  <!-- Name of the individual or role field -->
  <element name="fName"
    type="csvContact:fPostalLineType"
    substitutionGroup="rdeCsv:field" />
  <!-- Name organization field -->
  <element name="fOrg"
    type="csvContact:fOptPostalLineType"
    substitutionGroup="rdeCsv:field" />
  <!-- Street address line field with required index attribute -->
  <!-- starting with index 0. -->
  <element name="fStreet"
    type="csvContact:fStreetType"
    substitutionGroup="rdeCsv:field" />
  <complexType name="fStreetType">
    <complexContent>
      <extension base="csvContact:fOptPostalLineType">
        <sequence />
        <attribute name="index"
          type="int"
          use="required" />
      </extension>
    </complexContent>
  </complexType>
  <!-- Contact's city field -->
  <element name="fCity"
    type="csvContact:fPostalLineType"
    substitutionGroup="rdeCsv:field" />
  <!-- Contact's state or province field -->
  <element name="fSp"
    type="csvContact:fOptPostalLineType"
    substitutionGroup="rdeCsv:field" />
  <!-- Contact's postal code field -->
  <element name="fPc"
    type="csvContact:fPcType"
    substitutionGroup="rdeCsv:field" />
  <complexType name="fPcType">
    <complexContent>

```

```
<extension base="rdeCsv:fieldOptionalType">
  <sequence />
  <attribute name="type"
    type="token"
    default="contact\:pcType" />
  <attribute name="isLoc"
    type="boolean" />
</extension>
</complexContent>
</complexType>
<!-- Contact's country code field -->
<element name="fCc"
  type="csvContact:fCcType"
  substitutionGroup="rdeCsv:field" />
<complexType name="fCcType">
  <complexContent>
    <extension base="rdeCsv:fieldRequiredType">
      <sequence />
      <attribute name="type"
        type="token"
        default="contact\:ccType" />
      <attribute name="isLoc"
        type="boolean" />
    </extension>
  </complexContent>
</complexType>
<!-- Disclosure element fields -->
<!-- Flag of "1" to allow disclosure
and "0" to disallow disclosure -->
<element name="fDiscloseFlag"
  type="csvContact:fBoolean"
  substitutionGroup="rdeCsv:field" />
<!-- Disclosure of localized name
based on fDiscloseFlag? -->
<element name="fDiscloseNameLoc"
  type="csvContact:fBoolean"
  substitutionGroup="rdeCsv:field" />
<!-- Disclosure of internationalized name
based on fDiscloseFlag? -->
<element name="fDiscloseNameInt"
  type="csvContact:fBoolean"
  substitutionGroup="rdeCsv:field" />
<!-- Disclosure of localized org
based on fDiscloseFlag? -->
<element name="fDiscloseOrgLoc"
  type="csvContact:fBoolean"
  substitutionGroup="rdeCsv:field" />
<!-- Disclosure of internationalized org
```

```
        based on fDiscloseFlag? -->
<element name="fDiscloseOrgInt"
    type="csvContact:fBoolean"
    substitutionGroup="rdeCsv:field" />
<!-- Disclosure of localized address
    based on fDiscloseFlag? -->
<element name="fDiscloseAddrLoc"
    type="csvContact:fBoolean"
    substitutionGroup="rdeCsv:field" />
<!-- Disclosure of internationalized address
    based on fDiscloseFlag? -->
<element name="fDiscloseAddrInt"
    type="csvContact:fBoolean"
    substitutionGroup="rdeCsv:field" />
<!-- Disclosure voice telephone number
    based on fDiscloseFlag? -->
<element name="fDiscloseVoice"
    type="csvContact:fBoolean"
    substitutionGroup="rdeCsv:field" />
<!-- Disclosure facsimile telephone number
    based on fDiscloseFlag? -->
<element name="fDiscloseFax"
    type="csvContact:fBoolean"
    substitutionGroup="rdeCsv:field" />
<!-- Disclosure email address
    based on fDiscloseFlag? -->
<element name="fDiscloseEmail"
    type="csvContact:fBoolean"
    substitutionGroup="rdeCsv:field" />
<complexType name="fBoolean">
    <complexContent>
        <extension base="rdeCsv:fieldOptionalType">
            <sequence />
            <attribute name="type"
                type="token"
                default="boolean" />
        </extension>
    </complexContent>
</complexType>
<!-- Contact status field -->
<element name="fStatus"
    type="csvContact:fStatusType"
    substitutionGroup="rdeCsv:field" />
<!-- Host status based on contact-1.0.xsd -->
<complexType name="fStatusType">
    <complexContent>
        <extension base="rdeCsv:fieldRequiredType">
            <sequence />
        </extension>
    </complexContent>
</complexType>
```

```

        <attribute name="type"
                    type="token"
                    default="contact\:statusValueType" />
    </extension>
</complexContent>
</complexType>
<!--
End of schema.
-->
</schema>
<CODE ENDS>

```

9.8. RDE Registrar Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:rdeRegistrar-1.0"
        xmlns:rdeRegistrar="urn:ietf:params:xml:ns:rdeRegistrar-1.0"
        xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
        xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
        xmlns:domain="urn:ietf:params:xml:ns:domain-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:domain-1.0" />
  <import namespace="urn:ietf:params:xml:ns:contact-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rde-1.0" />
  <annotation>
    <documentation>
      Registry Data Escrow registrar provisioning schema
    </documentation>
  </annotation>
  <element name="abstractRegistrar"
            type="rdeRegistrar:abstractContentType"
            substitutionGroup="rde:content"
            abstract="true" />
  <element name="registrar"
            substitutionGroup="rdeRegistrar:abstractRegistrar" />
  <element name="delete"
            type="rdeRegistrar:deleteType"
            substitutionGroup="rde:delete" />
  <!-- Content Type -->
  <complexType name="abstractContentType">
    <complexContent>
      <extension base="rde:contentType">
        <sequence>

```

```
<element name="id"
  type="eppcom:clIDType" />
<element name="name"
  type="rdeRegistrar:nameType" />
<element name="gurid"
  type="positiveInteger"
  minOccurs="0" />
<element name="status"
  type="rdeRegistrar:statusType"
  minOccurs="0" />
<element name="postalInfo"
  type="rdeRegistrar:postalInfoType"
  minOccurs="0"
  maxOccurs="2" />
<element name="voice"
  type="contact:e164Type"
  minOccurs="0" />
<element name="fax"
  type="contact:e164Type"
  minOccurs="0" />
<element name="email"
  type="eppcom:minTokenType"
  minOccurs="0" />
<element name="url"
  type="anyURI"
  minOccurs="0" />
<element name="whoisInfo"
  type="rdeRegistrar:whoisInfoType"
  minOccurs="0" />
<element name="crDate"
  type="dateTime"
  minOccurs="0" />
<element name="upDate"
  type="dateTime"
  minOccurs="0" />
</sequence>
</extension>
</complexContent>
</complexType>
<simpleType name="nameType">
  <restriction base="normalizedString">
    <minLength value="1" />
    <maxLength value="255" />
  </restriction>
</simpleType>
<simpleType name="statusType">
  <restriction base="token">
    <enumeration value="ok" />
  </restriction>
</simpleType>
```

```
        <enumeration value="readonly" />
        <enumeration value="terminated" />
    </restriction>
</simpleType>
<complexType name="postalInfoType">
    <sequence>
        <element name="addr"
            type="rdeRegistrar:addrType" />
    </sequence>
    <attribute name="type"
        type="rdeRegistrar:postalInfoEnumType"
        use="required" />
</complexType>
<simpleType name="postalInfoEnumType">
    <restriction base="token">
        <enumeration value="loc" />
        <enumeration value="int" />
    </restriction>
</simpleType>
<complexType name="addrType">
    <sequence>
        <element name="street"
            type="rdeRegistrar:optPostalLineType"
            minOccurs="0"
            maxOccurs="3" />
        <element name="city"
            type="rdeRegistrar:postalLineType" />
        <element name="sp"
            type="rdeRegistrar:optPostalLineType"
            minOccurs="0" />
        <element name="pc"
            type="rdeRegistrar:pcType"
            minOccurs="0" />
        <element name="cc"
            type="rdeRegistrar:ccType" />
    </sequence>
</complexType>
<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="whoisInfoType">
  <sequence>
    <element name="name"
      type="eppcom:labelType"
      minOccurs="0" />
    <element name="url"
      type="anyURI"
      minOccurs="0" />
  </sequence>
</complexType>
<!-- Delete Type -->
<complexType name="deleteType">
  <complexContent>
    <extension base="rde:deleteType">
      <sequence>
        <element name="id"
          type="eppcom:clIDType"
          minOccurs="0"
          maxOccurs="unbounded" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
</schema>
<CODE ENDS>

```

9.9. CSV Registrar Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:csvRegistrar-1.0"
  xmlns:csvRegistrar="urn:ietf:params:xml:ns:csvRegistrar-1.0"
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns:rdeCsv="urn:ietf:params:xml:ns:rdeCsv-1.0"
  xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
  xmlns:domain="urn:ietf:params:xml:ns:domain-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:domain-1.0" />
<import namespace="urn:ietf:params:xml:ns:contact-1.0" />
<import namespace="urn:ietf:params:xml:ns:rde-1.0" />
<import namespace="urn:ietf:params:xml:ns:rdeCsv-1.0" />
<annotation>
  <documentation>
    Registrar Comma-Separated Values (CSV) Object
  </documentation>
</annotation>
<!--
Child elements of the <rde:contents> object
-->
<element name="contents"
        type="csvRegistrar:contentType"
        substitutionGroup="rde:content" />
<complexType name="contentType">
  <complexContent>
    <extension base="rde:contentType">
      <sequence>
        <element ref="rdeCsv:csv"
                maxOccurs="unbounded" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<!--
Child elements of the <rde:deletes> object
-->
<element name="deletes"
        type="csvRegistrar:deleteType"
        substitutionGroup="rde:delete" />
<complexType name="deleteType">
  <complexContent>
    <extension base="rde:deleteType">
      <sequence>
        <element ref="rdeCsv:csv"
                maxOccurs="unbounded" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- Registrar unique identifier (short name / id) -->
<element name="fId"
```



```

        type="rdeCsv:fClIDRequiredType"
        substitutionGroup="rdeCsv:field" />
<!-- Registrar name (full name) -->
<element name="fName"
    type="csvRegistrar:fNameType"
    substitutionGroup="rdeCsv:field" />
<!-- Registrar name field -->
<complexType name="fNameType">
    <complexContent>
        <extension base="rdeCsv:fieldRequiredType">
            <sequence />
            <attribute name="type"
                type="token"
                default="normalizedString" />
            <attribute name="isLoc"
                type="boolean"
                default="false" />
        </extension>
    </complexContent>
</complexType>
<!-- Registrar GURID field -->
<element name="fGurid"
    type="rdeCsv:fPositiveIntegerType"
    substitutionGroup="rdeCsv:field" />
<!-- Registrar status field -->
<element name="fStatus"
    type="csvRegistrar:fStatusType"
    substitutionGroup="rdeCsv:field" />
<element name="fStatusName"
    type="rdeCsv:fTokenType"
    substitutionGroup="rdeCsv:field" />
<complexType name="fStatusType">
    <complexContent>
        <extension base="rdeCsv:fieldOptionalType">
            <sequence />
            <attribute name="type"
                type="token"
                default="csvRegistrar\:statusType" />
        </extension>
    </complexContent>
</complexType>
<!-- Registrar status type with optional name attr -->
<complexType name="statusType">
    <simpleContent>
        <extension base="csvRegistrar:statusValueType">
            <attribute name="name"
                type="token" />
        </extension>
    </simpleContent>
</complexType>

```

```

        </simpleContent>
    </complexType>
    <!-- Registrar status enumerated values -->
    <simpleType name="statusValueType">
        <restriction base="token">
            <enumeration value="ok" />
            <enumeration value="readonly" />
            <enumeration value="terminated" />
        </restriction>
    </simpleType>
    <!-- Whois URL field -->
    <element name="fWhoisUrl"
        type="rdeCsv:anyURIType"
        substitutionGroup="rdeCsv:field" />

    <!--
    End of schema.
    -->
</schema>
<CODE ENDS>

```

9.10. RDE IDN Table Reference Objects

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:rdeIDN-1.0"
    xmlns:rdeIDN="urn:ietf:params:xml:ns:rdeIDN-1.0"
    xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
    <import namespace="urn:ietf:params:xml:ns:rde-1.0" />
    <annotation>
        <documentation>
            Registry Data Escrow IDN provisioning schema
        </documentation>
    </annotation>
    <element name="idnTableRef"
        type="rdeIDN:contentType"
        substitutionGroup="rde:content" />
    <element name="delete"
        type="rdeIDN:deleteType"
        substitutionGroup="rde:delete" />
    <!-- Content Types -->
    <complexType name="contentType">
        <complexContent>
            <extension base="rde:contentType">
                <sequence>
                    <element name="url"
                        type="anyURI" />

```

```

        <element name="urlPolicy"
            type="anyURI" />
    </sequence>
    <attribute name="id"
        type="rdeIDN:idType"
        use="required" />
</extension>
</complexContent>
</complexType>
<complexType name="deleteType">
    <complexContent>
        <extension base="rde:deleteType">
            <sequence>
                <element name="id"
                    type="rdeIDN:idType" />
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- Simple Types -->
<simpleType name="idType">
    <restriction base="token">
        <minLength value="1" />
        <maxLength value="64" />
    </restriction>
</simpleType>
</schema>
<CODE ENDS>

```

9.11. CSV IDN Language Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:csvIDN-1.0"
    xmlns:csvIDN="urn:ietf:params:xml:ns:csvIDN-1.0"
    xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
    xmlns:rdeCsv="urn:ietf:params:xml:ns:rdeCsv-1.0"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
    <!--
    Import common element types
    -->
    <import namespace="urn:ietf:params:xml:ns:rde-1.0" />
    <import namespace="urn:ietf:params:xml:ns:rdeCsv-1.0" />
    <annotation>
        <documentation>
            IDN Language Comma-Separated Values (CSV) Object
        </documentation>
    </annotation>

```

```

</annotation>
<!--
Child elements of the <rde:contents> object
-->
<element name="contents"
          type="csvIDN:contentType"
          substitutionGroup="rde:content" />
<complexType name="contentType">
  <complexContent>
    <extension base="rde:contentType">
      <sequence>
        <element ref="rdeCsv:csv"
                  maxOccurs="unbounded" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<!--
Child elements of the <rde:deletes> object
-->
<element name="deletes"
          type="csvIDN:deleteType"
          substitutionGroup="rde:delete" />
<complexType name="deleteType">
  <complexContent>
    <extension base="rde:deleteType">
      <sequence>
        <element ref="rdeCsv:csv"
                  maxOccurs="unbounded" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<!--
End of schema.
-->
</schema>
<CODE ENDS>

```

9.12. EPP Parameters Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:rdeEppParams-1.0"
        xmlns:rdeEppParams="urn:ietf:params:xml:ns:rdeEppParams-1.0"
        xmlns:rde="urn:ietf:params:xml:ns:rde-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 namespace="urn:ietf:params:xml:ns:epp-1.0" />
<import namespace="urn:ietf:params:xml:ns:eppcom-1.0" />
<import namespace="urn:ietf:params:xml:ns:rde-1.0" />
<annotation>
  <documentation>
    Registry Data Escrow EPP Parameters schema
  </documentation>
</annotation>
<!-- Content Type -->
<element name="eppParams"
  substitutionGroup="rdeEppParams:abstractEppParams" />
<!-- Abstract Content Type -->
<element name="abstractEppParams"
  type="rdeEppParams:abstractContentType"
  substitutionGroup="rde:content"
  abstract="true" />
<complexType name="abstractContentType">
  <complexContent>
    <extension base="rde:contentType">
      <sequence>
        <element name="version"
          type="epp:versionType"
          maxOccurs="unbounded" />
        <element name="lang"
          type="language"
          maxOccurs="unbounded" />
        <element name="objURI"
          type="anyURI"
          maxOccurs="unbounded" />
        <element name="svcExtension"
          type="epp:extURIType"
          minOccurs="0" />
        <element name="dcp"
          type="epp:dcpType" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
</schema>
<CODE ENDS>

```

9.13. NNDN Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:rdeNNDN-1.0"

```

```
xmlns:rdeNNDN="urn:ietf:params:xml:ns:rdeNNDN-1.0"
xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
xmlns:rdeIDN="urn:ietf:params:xml:ns:rdeIDN-1.0"
xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
xmlns="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<import namespace="urn:ietf:params:xml:ns:eppcom-1.0" />
<import namespace="urn:ietf:params:xml:ns:rde-1.0" />
<import namespace="urn:ietf:params:xml:ns:rdeIDN-1.0" />
<annotation>
  <documentation>
    Registry Data Escrow NNDN provisioning schema
  </documentation>
</annotation>
<element name="abstractNNDN"
  type="rdeNNDN:abstractContentType"
  substitutionGroup="rde:content"
  abstract="true" />
<element name="NNDN"
  substitutionGroup="rdeNNDN:abstractNNDN" />
<element name="delete"
  type="rdeNNDN:deleteType"
  substitutionGroup="rde:delete" />
<!-- Content Type -->
<complexType name="abstractContentType">
  <complexContent>
    <extension base="rde:contentType">
      <sequence>
        <element name="aName"
          type="eppcom:labelType" />
        <element name="uName"
          type="eppcom:labelType"
          minOccurs="0" />
        <element name="idnTableId"
          type="rdeIDN:idType"
          minOccurs="0" />
        <element name="originalName"
          type="eppcom:labelType"
          minOccurs="0" />
        <element name="nameState"
          type="rdeNNDN:nameState" />
        <element name="crDate"
          type="dateTime"
          minOccurs="0" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```

<simpleType name="nameStateValue">
  <restriction base="token">
    <enumeration value="withheld" />
    <enumeration value="blocked" />
    <enumeration value="mirrored" />
  </restriction>
</simpleType>
<complexType name="nameState">
  <simpleContent>
    <extension base="rdeNNDN:nameStateValue">
      <attribute name="mirroringNS"
        type="boolean"
        default="true" />
    </extension>
  </simpleContent>
</complexType>
<!-- Delete Type -->
<complexType name="deleteType">
  <complexContent>
    <extension base="rde:deleteType">
      <sequence>
        <element name="aName"
          type="eppcom:labelType"
          minOccurs="0"
          maxOccurs="unbounded" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
</schema>
<CODE ENDS>

```

9.14. CSV NNDN Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:csvNNDN-1.0"
  xmlns:csvNNDN="urn:ietf:params:xml:ns:csvNNDN-1.0"
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns:rdeCsv="urn:ietf:params:xml:ns:rdeCsv-1.0"
  xmlns:rdeNNDN="urn:ietf:params:xml:ns:rdeNNDN-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <!--
  Import common element types
  -->
  <import namespace="urn:ietf:params:xml:ns:rde-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rdeCsv-1.0" />

```

```
<import namespace="urn:ietf:params:xml:ns:rdeNNDN-1.0" />
<annotation>
  <documentation>
    NNDN (NNDN's not domain name) (CSV) Object
  </documentation>
</annotation>
<!--
Child elements of the <rde:contents> object
-->
<element name="contents"
  type="csvNNDN:contentType"
  substitutionGroup="rde:content" />
<complexType name="contentType">
  <complexContent>
    <extension base="rde:contentType">
      <sequence>
        <element ref="rdeCsv:csv"
          maxOccurs="unbounded" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<!--
Child elements of the <rde:deletes> object
-->
<element name="deletes"
  type="csvNNDN:deleteType"
  substitutionGroup="rde:delete" />
<complexType name="deleteType">
  <complexContent>
    <extension base="rde:deleteType">
      <sequence>
        <element ref="rdeCsv:csv"
          maxOccurs="unbounded" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<!-- A-Label format name field -->
<element name="fAName"
  type="rdeCsv:fNameRequiredType"
  substitutionGroup="rdeCsv:field" />
<!-- domain name used to generate the IDN variant field -->
<element name="fOriginalName"
  type="rdeCsv:fNameType"
  substitutionGroup="rdeCsv:field" />
<!-- RGP status field -->
<element name="fNameState"
```



```
        type="csvNNDN:fNameStateType"
        substitutionGroup="rdeCsv:field" />
<complexType name="fNameStateType">
  <complexContent>
    <extension base="rdeCsv:fieldRequiredType">
      <sequence />
      <attribute name="type"
        type="token"
        default="rdeNNDN\:nameState" />
    </extension>
  </complexContent>
</complexType>
<!-- Mirroring uses NS mirror mechanism? -->
<element name="fMirroringNS"
  type="rdeCsv:fBooleanType"
  substitutionGroup="rdeCsv:field" />
<!--
End of schema.
-->
</schema>
<CODE ENDS>
```

9.15. Policy Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:rdePolicy-1.0"
  xmlns:rdePolicy="urn:ietf:params:xml:ns:rdePolicy-1.0"
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <annotation>
    <documentation>
      Registry Data Escrow Policy schema
    </documentation>
  </annotation>
  <import namespace="urn:ietf:params:xml:ns:rde-1.0" />
  <element name="policy"
    type="rdePolicy:policyType"
    substitutionGroup="rde:content" />
  <complexType name="policyType">
    <complexContent>
      <extension base="rde:contentType">
        <attribute name="scope"
          type="token"
          use="required" />
        <attribute name="element"
          type="anyURI"
          use="required" />
      </extension>
    </complexContent>
  </complexType>
</schema>
<CODE ENDS>

```

9.16. Header Object

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:rdeHeader-1.0"
  xmlns:rdeHeader="urn:ietf:params:xml:ns:rdeHeader-1.0"
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0" />
  <import namespace="urn:ietf:params:xml:ns:rde-1.0" />
  <annotation>
    <documentation>
      Data Escrow Deposit Header schema
    </documentation>
  </annotation>

```

```
<!-- Root Element -->
<element name="header"
  type="rdeHeader:contentType"
  substitutionGroup="rde:content" />
<!-- Content Type -->
<complexType name="contentType">
  <complexContent>
    <extension base="rde:contentType">
      <sequence>
        <group ref="rdeHeader:repositoryTypeGroup" />
        <element name="count"
          type="rdeHeader:countType"
          maxOccurs="unbounded" />
        <element name="contentTag"
          type="token"
          minOccurs="0" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<group name="repositoryTypeGroup">
  <choice>
    <element name="tld"
      type="eppcom:labelType" />
    <element name="registrar"
      type="positiveInteger" />
    <element name="ppsp"
      type="token" />
    <element name="reseller"
      type="token" />
  </choice>
</group>
<complexType name="countType">
  <simpleContent>
    <extension base="long">
      <attribute name="uri"
        type="anyURI"
        use="required" />
      <attribute name="rcdn"
        type="eppcom:labelType" />
      <attribute name="registrarId"
        type="positiveInteger" />
    </extension>
  </simpleContent>
</complexType>
</schema>
<CODE ENDS>
```

9.17. DNRD Common Objects

```
<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:rdeDnrCommon-1.0"
  xmlns:rdeDnrCommon="urn:ietf:params:xml:ns:rdeDnrCommon-1.0"
  xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0" />
  <annotation>
    <documentation>
      Data Escrow Deposit Common Objects schema
    </documentation>
  </annotation>
  <complexType name="rrType">
    <simpleContent>
      <extension base="eppcom:clIDType">
        <attribute name="client"
          type="eppcom:clIDType" />
      </extension>
    </simpleContent>
  </complexType>
</schema>
<CODE ENDS>
```

10. Internationalization Considerations

Data Escrow deposits are 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.

11. IANA Considerations

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

Registration request for the RDE CSV namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the RDE CSV XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.1 of this document.

Registration request for the RDE domain namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the RDE domain XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.2 of this document.

Registration request for the CSV domain namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the CSV domain XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.3 of this document.

Registration request for the RDE host namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the RDE host XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.4 of this document.

Registration request for the CSV host namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the CSV host XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.5 of this document.

Registration request for the RDE contact namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the RDE contact XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.6 of this document.

Registration request for the CSV contact namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the CSV contact XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.7 of this document.

Registration request for the RDE registrar namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the RDE registrar XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.8 of this document.

Registration request for the CSV registrar namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the CSV registrar XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.9 of this document.

Registration request for the RDE IDN namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the RDE IDN XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.10 of this document.

Registration request for the CSV IDN namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the CSV IDN XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.11 of this document.

Registration request for the RDE EPP parameters namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the RDE EPP parameters XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.12 of this document.

Registration request for the RDE NNDN namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the RDE NNDN XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.13 of this document.

Registration request for the CSV NNDN namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the CSV NNDN XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.14 of this document.

Registration request for the RDE Policy namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the RDE Policy XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.15 of this document.

Registration request for the RDE Header namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the RDE Header XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.16 of this document.

Registration request for the RDE Common Objects namespace:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

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

Registration request for the RDE Common Objects XML schema:

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

Registrant Contact: IESG <regext@ietf.org>

Note to RFC Editor: Please remove the email address from the RFC after IANA records it.

See Section 9.17 of this document.

12. 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".

12.1. Implementation in the gTLD space

Organization: ICANN

Name: ICANN Registry Agreement

Description: the ICANN Base Registry Agreement requires Registries, Data Escrow Agents, and ICANN to implement this specification. ICANN receives daily notifications from Data Escrow Agents confirming that more than 1,200 gTLDs are sending deposits that comply with this specification. ICANN receives on a weekly basis per gTLD, from more than 1,200 gTLD registries, a Bulk Registration Data Access file that also complies with this specification. In addition, ICANN is aware of Registry Service Provider transitions using data files that conform to this specification.

Level of maturity: production.

Coverage: all aspects of this specification are implemented.

Version compatibility: versions 03 - 09 are known to be implemented.

Contact: gustavo.lozano@icann.org

URL: <https://www.icann.org/resources/pages/registries/registries-agreements-en>

13. Security Considerations

This specification does not define the security mechanisms to be used in the transmission of the data escrow deposits, since it only specifies the minimum necessary to enable the rebuilding of a registry from deposits without intervention from the original registry.

Depending on local policies, some elements, or, most likely, the whole deposit will be considered confidential. As such, the parties SHOULD take all the necessary precautions such as encrypting the data at rest and in transit to avoid inadvertent disclosure of private data. Regardless of the precautions taken by the parties regarding

data at rest and in transit, authentication credentials MUST NOT be escrowed.

Authentication of the parties passing data escrow deposit files is also of the utmost importance. The escrow agent MUST properly authenticate the registry's identity before accepting data escrow deposits. The registry MUST authenticate the escrow agent's identity before submitting any data, and the data escrow agent MUST authenticate the identity of the party receiving the data escrow deposits for the purposes deemed appropriate.

Additionally, the registry and the escrow agent MUST use integrity checking mechanisms to ensure the data transmitted is what the source intended. Validation of the contents by the parties is RECOMMENDED to ensure that the file was transmitted correctly from the registry or escrow agent and that the contents are "meaningful".

A few elements in this specification contain URLs, the use of HTTP over TLS (Transport Layer Security), [RFC2818] is RECOMMENDED on the URLs.

The various data structures in the document include a few places that have internal redundancy, and if the values become inconsistent there can be harmful consequences, such as different entities using different fields as their reference.

Note: if Transport Layer Security (TLS) is used when providing an escrow services, the recommendations in [BCP195] MUST be implemented.

14. Privacy Considerations

This specification defines a format that may be used to escrow personal data. The process of data escrow is governed by a legal document agreed by the parties, and such legal document must ensure that privacy-sensitive and/or personal data receives the required protection.

15. Acknowledgments

Parts of this document are based on EPP [RFC5730] and related RFCs by Scott Hollenbeck.

Special suggestions that have been incorporated into this document were provided by Edward Lewis, Jaap Akkerhuis, Lawrence Conroy, Marc Groeneweg, Michael Young, Chris Wright, Patrick Mevzek, Stephen Morris, Scott Hollenbeck, Stephane Bortzmeyer, Warren Kumari, Paul Hoffman, Vika Mpisane, Bernie Hoeneisen, Jim Galvin, Andrew Sullivan, Hiro Hotta, Christopher Browne, Daniel Kalchev, David Conrad, James

Mitchell, Francisco Obispo, Bhadresh Modi, Alexander Mayrhofer and Benjamin Kaduk.

Shoji Noguchi and Francisco Arias participated as co-authors until version 05 providing invaluable support for this document.

16. Change History

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

16.1. Changes from draft-arias-noguchi-registry-data-escrow-02 to -dnrd-objects-mapping-00

1. Added definition for child elements under the <domain> element.
2. Added definition for child elements under the <host> element.
3. Added definition for child elements under the <contact> element.
4. Rewrote the IDN Variants Handling section to use the variant states as described in ICANN's Study of Issues Related to the Management of IDN Variant TLDs.
5. Renamed <icannID> to <gurid> in the <rdeRegistrar>.
6. Renamed <dnssec> to <secDNS> in the <domain> element.
7. Renamed <transfData> to <trnData> in the <domain> element.
8. Added <whoisInfo> element under <rdeRegistrar> element.
9. Fixed some typographical errors and omissions.

16.2. Changes from 00 to 01

1. Specify OPTIONAL elements in the draft.
2. Added NNDN object to support list of reserved names and different IDN variants models.
3. Removed subordinated host element from the domain object.
4. Added eppParams object.
5. Added variantGenerator element to the domain object.
6. Added lgr to the IDN table object.

16.3. Changes from 01 to 02

1. Updates to the all objects based on feedback from the list.
2. Start of XML and CSV drafts merge.
3. Added header object.
4. Added report object.
5. Added notification object.
6. Added Data Escrow Agent Extended Verification Process section.
7. Added Notifications from Registries to Third Parties.
8. Added Notifications from Data Escrow Agents to Third Parties.
9. Added FULL, DIFF deposit examples using the XML model only.

16.4. Changes from 02 to 03

1. Remove authinfo from the XML Schema.
2. Resend attribute is now an element
3. Scope attribute added to policy object.

16.5. Changes from 03 to 04

1. Merged draft-gould-thippeswamy-dnrd-csv-mapping-03 into draft-arias-noguchi-dnrd-objects-mapping-02.
2. Changed the cksum attribute of <rdeCsv:file> to use CRC32 and changed all of the sample cksum values to use CRC32, based on feedback from David Kipling.
3. Changed the optional <rdeCsv:sep> element to be an optional "sep" attribute value of the <rdeCsv:csv> element with a default value of "," based on feedback from David Kipling.
4. Added support for the optional "parent" attribute for the to the CSV fields to indicate a field as a reference to a parent object, based on feedback from David Kipling.
5. Added support for the CSV model for the NNDN.
6. Added support to delete hosts based on roid.

7. Added mirrored state to NNDN
 8. Minor fixes to XML XSDs.
 9. The Report and Notification objects were moved to draft-lozano-icann-registry-interfaces
 10. The section Data escrow notifications was moved to draft-lozano-icann-registry-interfaces
 11. Removed references to the `<rdeCsv:fCrRr>`, `<rdeCsv:fCrID>`, and `<rdeCsv:fCrDate>` from the "hostStatuses" and "hostAddresses" CSV files.
 12. Removed references to the `<rdeCsv:fCrRr>`, `<rdeCsv:fCrID>`, and `<rdeCsv:fCrDate>` from the "contactStatuses" CSV file.
 13. Removed references to the `<rdeCsv:fCrRr>`, `<rdeCsv:fCrID>`, and `<rdeCsv:fCrDate>` from the "domainContacts", "domainStatuses", and "domainNameServers" CSV files.
 14. Changed `<rdeCsv:fLanguage>` to `<rdeCsv:fLang>`.
 15. Replaced use of `<rdeCsv:fLang>` to new `<rdeCsv:fIdnTableId>` field in the "domain", "idnLanguage", and "NNDN" CSV files.
 16. Replaced use of `<csvHost:fName>` with `<rdeCsv:fRoid>` in the "host" `<csvHost:deletes>` `<rdeCsv:csv>` element.
 17. Changed the foreign key of the hosts to use `<rdeCsv:fRoid>` instead of `<csvHost:fName>` and removed use of `<csvHost:fName>` in the "domainNameServers", "hostStatuses", and "hostAddresses" CSV files.
 18. Added use of the MUST keyword for CSV fields that are required to be supported in an EPP based system.
 19. Removed use of the `<rdeCsv:fRoid>` field element for the "registrar" CSV file.
 20. Added definition of `<csvNNDN:fMirroringNS>` field element.
- 16.6. Changes from 04 to 05
1. Updated the examples of the full and differential deposits using the CSV and XML model.

2. Made `<rdeCsv:fExDate>` optional for the "domainTransfer" CSV file to match the XML definition.
 3. Made `<csvDomain:fOriginalName>` optional for the "domain" CSV file to match the XML definition.
 4. Made `<rdeCsv:fTrDate>` optional for the "domain" and "contact" CSV files to match the XML definition.
 5. Change `<idnTableId>` from IDREF to idType.
 6. Minor editorial changes.
- 16.7. Changes from 05 to 06
1. Revised the differential and incremental deposits for the CSV format to use cascade update / replace and delete from the parent object to be consistent with the XML format.
 2. Revised the structure of the CSV format sections to utilize sub-sections instead of a list for the CSV file definitions.
 3. Added the "CSV Parent Child Relationship" section to describe the concept of parent child relationships across CSV file definitions.
 4. Added the "domainNameServersAddresses" CSV File Definition section to support the domain host attributes model of [RFC5731].
 5. Made the required fields in the CSV format consistent with the XML format. The CSV fields updated to be required include:
`<rdeCsv:fCrDate>`, `<csvDomain:fContactType>`, `<csvDomain:fStatus>`,
`<csvDomain:fKeyTag>`, `<csvDomain:fDsAlg>`, `<csvDomain:fDigestType>`,
`<csvDomain:fDigest>`, `<csvDomain:fFlags>`, `<csvDomain:fProtocol>`,
`<csvDomain:fKeyAlg>`, `<csvDomain:fPubKey>`, `<rdeCsv:fTrStatus>`,
`<rdeCsv:fReRr>`, `<rdeCsv:fReDate>`, `<rdeCsv:fAcRr>`,
`<rdeCsv:fAcDate>`, `<csvHost:fStatus>`, `<csvContact:fCc>`,
`<csvContact:fStatus>`, `<csvContact:fPostalType>`,
`<csvRegistrar:fStatus>`, and `<csvNNDN:fNameState>`.
 6. Revised the CSV examples to use a more realistic set of records.
- 16.8. Changes from 06 to 07
1. Created "repositoryTypeGroup" group element in the rdeHeader including the `<rdeHeader:registrar>`, `<rdeHeader:ppsp>` and `<rdeHeader:tld>` elements.

2. Added the optional "rcdn" and "registrarId" attributes to the <rdeHeader:count> element
- 16.9. Changes from 07 to 08
1. The following registrar elements were made optional to support greater flexibility for the implementation of policies: status, postalInfo, email and crDate.
 2. The following domain name elements were made optional to support greater flexibility for the implementation of policies: crRr.
- 16.10. Changes from 08 to 09
1. Implementation Status section was added
- 16.11. Changes from 09 to 10
1. Editorial changes in section Section 5.1.2.1.6.
 2. Added MAY clause when the DS Data Interface is used in section Section 5.1.2.1.6.
- 16.12. Changes from 10 to REGEXT 00
1. Internet Draft (I-D) adopted by the REGEXT WG.
- 16.13. Changes REGEXT 00 to REGEXT 01
1. Added the <rdeHeader:reseller> element to the "repositoryTypeGroup" group element in the rdeHeader.
 2. Privacy consideration section was added
 3. Updates on section 8
- 16.14. Changes REGEXT 01 to REGEXT 02
1. Added a choice between the use of the <rdeCsv:fClID> or <csvRegistrar:fGurid> fields in the CSV "domain", "host", and "contact" definitions.
 2. Added a choice between the use of the <rdeCsv:fRoid> or <csvHost:fName> fields in the CSV "domainNameServers" definition.
 3. Changed "of client" to "of the client" throughout the document.

4. Modified all references of 'The attribute isRequired MUST equal "true".' to 'The attribute "isRequired" MUST equal "true".'
5. Combined the <csvDomain:fName> and <csvDomain:fContactType> fields in a single required list for the CSV "domainContacts" definition.
6. Combined the <csvDomain:fName>, <csvDomain:fStatus>, and <csvDomain:fRgpStatus> fields in a single required list for the CSV "domainStatuses" definition.
7. Moved the <rdeCsv:fCrRr> the <rdeCsv:fUpRr> fields to the MAY list for the CSV "domain", "host", and "contact" definitions.
8. Made the order of the <rdeCsv:fCrRr>, <rdeCsv:crID>, <rdeCsv:UpRr>, and <rdeCsv:UpID> fields more consistent in the CSV lists.
9. Fixed an error in the order of the <contact> object example.
10. Changed <rdeCsv:fCrDate> to be optional to match <crDate> being optional in the XML model, by having it use type rdeCsv:fDateTimeType instead of rdeCsv:fRequiredDateTimeType and ensuring that <rdeCsv:fCrDate> is included in the MAY field lists and not the MUST field lists.
11. Made <rdeCsv:fExDate> optional for the "domain" CSV definition to be consistent with the XML model, by removing the sentence 'The attribute "isRequired" MUST equal "true".' from the description and moving the field to the MAY field list.
12. Made <rdeCsv:fUpDate> optional for the "domain" and "contact" CSV definitions to be consistent with the XML model, by moving the field to the MAY field list.
13. Made <rdeCsv:fCrRr> optional to be consistent with the XML model, by having it use type rdeCsv:fClIDType instead of rdeCsv:fClIDRequiredType.
14. Made <rdeCsv:fReRr> required to be consistent with the XML model, by having it use type rdeCsv:fClIDRequiredType instead of rdeCsv:fClIDType.
15. Made the <csvRegistrar:fGurid> field in the "host", "contact", and "registrar" CSV definitions required explicitly by removing 'and isRequired="true"' and adding the sentence 'The attribute isRequired MUST equal "true".' when it is chosen as the primary field.

16. Removed extra `'/>.'` at the end of the `<csvHost:fStatus>` field description in the "hostStatuses" CSV definition.
 17. Made the `<csvRegistrar:fStatus>` field optional to be consistent with the XML model, by having `csvRegistrar:fStatusType` extend `rdeCsv:fieldOptionalType` instead of `rdeCsv:fRequiredType`.
 18. Made the `<csvContact:fEmail>` field for the "registrar" CSV definition explicitly optional to be consistent with the XML model, by adding the sentence `'The attribute isRequired MUST equal "false".'` to the field description and including the definition of `isRequired="false"` in the "registrar" CSV definition examples.
 19. Added the choice between the use of the `<csvRegistrar:fId>` and `<csvRegistrar:fGurid>` fields in the deletes "registrar" CSV definition to be consistent with the "registrar" CSV definition.
 20. Made the `<crRr>` and `<crDate>` elements optional for the host and contact objects in the XML model to be consistent with the domain object.
- 16.15. Changes REGEXT 02 to REGEXT 03
1. Added the optional element `contentTag` in the header object.
 2. Editorial updates.
- 16.16. Changes REGEXT 03 to REGEXT 04
1. Note: Updates from version REGEXT 03 to REGEXT 04 attend the feedback provided during the document shepherd review.
 2. Editorial updates.
 3. Examples now use domain names from the .example TLD.
 4. The introduction was enhanced by explaining the need for data escrow and the proposed solution.
 5. Explanation regarding NNDN was improved.
 6. Explanation regarding the CSV and XML model was improved.
 7. Section 4.5 updated to make the text clearer.
 8. `draft-arias-noguchi-registry-data-escrow` is now referenced from the I-D repository.

9. The XML prefix "rdeDomain" is now consistently used.
 10. The prevID attribute was removed from the examples of full deposits.
 11. The examples were updated to use present dates.
- 16.17. Changes REGEXT 04 to REGEXT 05
1. draft-ietf-regext-data-escrow (version 04) is now referenced from the I-D repository.
 2. The example in idnLanguage CSV file definition updated to use the sep attribute.
 3. The reference in the example in hostAddresses CSV file definition was updated.
 4. Moved [RFC0791] and [RFC5952] to the Normative References section.
- 16.18. Changes REGEXT 05 to REGEXT 06
1. Changes based on the feedback provided here:
https://mailarchive.ietf.org/arch/msg/regext/nA8eTYIrXJ44_6ullQlRLW6T74s
- 16.19. Changes REGEXT 06 to REGEXT 07
1. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regext/hDLz2ym4oR-ukA4Fm-QJ8FzaxxE>
 2. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regext/780Xw-zlRMRb79nmZ6ABmRTolFU>
- 16.20. Changes REGEXT 07 to REGEXT 08
1. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regext/UaMNVllxh60ldjppqHHYc3TNSfhg>
 2. Changes based on the feedback provided here:
https://mailarchive.ietf.org/arch/msg/regext/B3QTxUCWUE4R_QharAQlA3041j0

16.21. Changes REGEXT 08 to REGEXT 09

1. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regext/EmKW32exlPgLbBUIbS8OjdYUJWc>

16.22. Changes REGEXT 09 to REGEXT 10

1. Changes based on the feedback provided here:
https://mailarchive.ietf.org/arch/msg/regext/tmoKLAV6jhh2zp4JczjeWdr_jJE
2. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regext/m7gyDTjHuRqIQCuKMHF-OLSS99k>
3. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regext/3Acx5KHfeUdxZbx6A7zgoZHxIto>
4. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regext/3Acx5KHfeUdxZbx6A7zgoZHxIto>
5. Changes based on the feedback provided here:
https://mailarchive.ietf.org/arch/msg/regext/7JiP2fzOr8KCnzI2rwoP-_KlxZY
6. Changes based on the feedback provided here:
https://mailarchive.ietf.org/arch/msg/regext/dbuyW5YTYj4VcFHUQYC-D8OMv_g
7. Changes based on the feedback provided here:
https://mailarchive.ietf.org/arch/msg/regext/ExUZenwC8lZQe9x24-8IKT_FWm8

16.23. Changes REGEXT 10 to REGEXT 11

1. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regext/ghEr55r7CVdwUSvkvMGpol4aSh0>

17. Example of a Full Deposit using the XML model

Example of a Full Deposit using the XML model:

```
<?xml version="1.0" encoding="UTF-8"?>
<rde:deposit type="FULL" id="20191017001"
```

```
xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1"
xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
xmlns:rdeHeader="urn:ietf:params:xml:ns:rdeHeader-1.0"
xmlns:rdeDomain="urn:ietf:params:xml:ns:rdeDomain-1.0"
xmlns:rdeHost="urn:ietf:params:xml:ns:rdeHost-1.0"
xmlns:rdeContact="urn:ietf:params:xml:ns:rdeContact-1.0"
xmlns:rdeRegistrar="urn:ietf:params:xml:ns:rdeRegistrar-1.0"
xmlns:rdeIDN="urn:ietf:params:xml:ns:rdeIDN-1.0"
xmlns:rdeNNDN="urn:ietf:params:xml:ns:rdeNNDN-1.0"
xmlns:rdeEppParams="urn:ietf:params:xml:ns:rdeEppParams-1.0"
xmlns:rdePolicy="urn:ietf:params:xml:ns:rdePolicy-1.0"
xmlns:epp="urn:ietf:params:xml:ns:epp-1.0">

<rde:watermark>2019-10-17T00:00:00Z</rde:watermark>
<rde:rdeMenu>
  <rde:version>1.0</rde:version>
  <rde:objURI>urn:ietf:params:xml:ns:rdeHeader-1.0
</rde:objURI>
  <rde:objURI>urn:ietf:params:xml:ns:rdeContact-1.0
</rde:objURI>
  <rde:objURI>urn:ietf:params:xml:ns:rdeHost-1.0
</rde:objURI>
  <rde:objURI>urn:ietf:params:xml:ns:rdeDomain-1.0
</rde:objURI>
  <rde:objURI>urn:ietf:params:xml:ns:rdeRegistrar-1.0
</rde:objURI>
  <rde:objURI>urn:ietf:params:xml:ns:rdeIDN-1.0
</rde:objURI>
  <rde:objURI>urn:ietf:params:xml:ns:rdeNNDN-1.0
</rde:objURI>
  <rde:objURI>urn:ietf:params:xml:ns:rdeEppParams-1.0
</rde:objURI>
</rde:rdeMenu>

<!-- Contents -->
<rde:contents>
  <!-- Header -->
  <rdeHeader:header>
    <rdeHeader:tld>test</rdeHeader:tld>
    <rdeHeader:count
      uri="urn:ietf:params:xml:ns:rdeDomain-1.0">2
    </rdeHeader:count>
    <rdeHeader:count
      uri="urn:ietf:params:xml:ns:rdeHost-1.0">1
    </rdeHeader:count>
    <rdeHeader:count
```



```
        uri="urn:ietf:params:xml:ns:rdeContact-1.0">1
      </rdeHeader:count>
    <rdeHeader:count
      uri="urn:ietf:params:xml:ns:rdeRegistrar-1.0">1
    </rdeHeader:count>
    <rdeHeader:count
      uri="urn:ietf:params:xml:ns:rdeIDN-1.0">1
    </rdeHeader:count>
    <rdeHeader:count
      uri="urn:ietf:params:xml:ns:rdeNNDN-1.0">1
    </rdeHeader:count>
    <rdeHeader:count
      uri="urn:ietf:params:xml:ns:rdeEppParams-1.0">1
    </rdeHeader:count>
  </rdeHeader:header>

  <!-- Domain: example1.example -->
  <rdeDomain:domain>
    <rdeDomain:name>example1.example</rdeDomain:name>
    <rdeDomain:roid>Dexample1-TEST</rdeDomain:roid>
    <rdeDomain:status s="ok"/>
    <rdeDomain:registrant>jdl234</rdeDomain:registrant>
    <rdeDomain:contact type="admin">sh8013</rdeDomain:contact>
    <rdeDomain:contact type="tech">sh8013</rdeDomain:contact>
    <rdeDomain:ns>
      <domain:hostObj>ns1.example.com</domain:hostObj>
      <domain:hostObj>ns1.example1.example</domain:hostObj>
    </rdeDomain:ns>
    <rdeDomain:clID>RegistrarX</rdeDomain:clID>
    <rdeDomain:crRr client="jdoe">RegistrarX</rdeDomain:crRr>
    <rdeDomain:crDate>1999-04-03T22:00:00.0Z</rdeDomain:crDate>
    <rdeDomain:exDate>2025-04-03T22:00:00.0Z</rdeDomain:exDate>
  </rdeDomain:domain>

  <!-- Domain: example2.example -->
  <rdeDomain:domain>
    <rdeDomain:name>example2.example</rdeDomain:name>
    <rdeDomain:roid>Dexample2-TEST</rdeDomain:roid>
    <rdeDomain:status s="ok"/>
    <rdeDomain:status s="clientUpdateProhibited"/>
    <rdeDomain:registrant>jdl234</rdeDomain:registrant>
    <rdeDomain:contact type="admin">sh8013</rdeDomain:contact>
    <rdeDomain:contact type="tech">sh8013</rdeDomain:contact>
    <rdeDomain:clID>RegistrarX</rdeDomain:clID>
    <rdeDomain:crRr>RegistrarX</rdeDomain:crRr>
    <rdeDomain:crDate>1999-04-03T22:00:00.0Z</rdeDomain:crDate>
    <rdeDomain:exDate>2025-04-03T22:00:00.0Z</rdeDomain:exDate>
  </rdeDomain:domain>
```

```
<!-- Host: ns1.example.example -->
<rdeHost:host>
  <rdeHost:name>ns1.example1.example</rdeHost:name>
  <rdeHost:roid>Hns1_example_test-TEST</rdeHost:roid>
  <rdeHost:status s="ok"/>
  <rdeHost:status s="linked"/>
  <rdeHost:addr ip="v4">192.0.2.2</rdeHost:addr>
  <rdeHost:addr ip="v4">192.0.2.29</rdeHost:addr>
  <rdeHost:addr ip="v6">2001:DB8:1::1</rdeHost:addr>
  <rdeHost:clID>RegistrarX</rdeHost:clID>
  <rdeHost:crRr>RegistrarX</rdeHost:crRr>
  <rdeHost:crDate>1999-05-08T12:10:00.0Z</rdeHost:crDate>
  <rdeHost:upRr>RegistrarX</rdeHost:upRr>
  <rdeHost:upDate>2009-10-03T09:34:00.0Z</rdeHost:upDate>
</rdeHost:host>

<!-- Contact: sh8013 -->
<rdeContact:contact>
  <rdeContact:id>sh8013</rdeContact:id>
  <rdeContact:roid>Csh8013-TEST</rdeContact:roid>
  <rdeContact:status s="linked"/>
  <rdeContact:status s="clientDeleteProhibited"/>
  <rdeContact:postalInfo type="int">
    <contact:name>John Doe</contact:name>
    <contact:org>Example Inc.</contact:org>
    <contact:addr>
      <contact:street>123 Example Dr.</contact:street>
      <contact:street>Suite 100</contact:street>
      <contact:city>Dulles</contact:city>
      <contact:sp>VA</contact:sp>
      <contact:pc>20166-6503</contact:pc>
      <contact:cc>US</contact:cc>
    </contact:addr>
  </rdeContact:postalInfo>
  <rdeContact:voice x="1234">+1.7035555555
</rdeContact:voice>
  <rdeContact:fax>+1.7035555556
</rdeContact:fax>
  <rdeContact:email>jdoe@example.example
</rdeContact:email>
  <rdeContact:clID>RegistrarX</rdeContact:clID>
  <rdeContact:crRr client="jdoe">RegistrarX
</rdeContact:crRr>
  <rdeContact:crDate>2009-09-13T08:01:00.0Z
</rdeContact:crDate>
  <rdeContact:upRr client="jdoe">RegistrarX
</rdeContact:upRr>
  <rdeContact:upDate>2009-11-26T09:10:00.0Z
```

```
</rdeContact:update>
<rdeContact:trDate>2009-12-03T09:05:00.0Z
</rdeContact:trDate>
<rdeContact:disclose flag="0">
  <contact:voice/>
  <contact:email/>
</rdeContact:disclose>
</rdeContact:contact>

<!-- Registrar: RegistrarX -->
<rdeRegistrar:registrar>
  <rdeRegistrar:id>RegistrarX</rdeRegistrar:id>
  <rdeRegistrar:name>Registrar X</rdeRegistrar:name>
  <rdeRegistrar:gurid>8</rdeRegistrar:gurid>
  <rdeRegistrar:status>ok</rdeRegistrar:status>
  <rdeRegistrar:postalInfo type="int">
    <rdeRegistrar:addr>
      <rdeRegistrar:street>123 Example Dr.
    </rdeRegistrar:street>
      <rdeRegistrar:street>Suite 100
    </rdeRegistrar:street>
      <rdeRegistrar:city>Dulles</rdeRegistrar:city>
      <rdeRegistrar:sp>VA</rdeRegistrar:sp>
      <rdeRegistrar:pc>20166-6503</rdeRegistrar:pc>
      <rdeRegistrar:cc>US</rdeRegistrar:cc>
    </rdeRegistrar:addr>
  </rdeRegistrar:postalInfo>
  <rdeRegistrar:voice x="1234">+1.7035555555
</rdeRegistrar:voice>
  <rdeRegistrar:fax>+1.7035555556
</rdeRegistrar:fax>
  <rdeRegistrar:email>jdoe@example.example
</rdeRegistrar:email>
  <rdeRegistrar:url>http://www.example.example
</rdeRegistrar:url>
  <rdeRegistrar:whoisInfo>
    <rdeRegistrar:name>whois.example.example
    </rdeRegistrar:name>
    <rdeRegistrar:url>http://whois.example.example
    </rdeRegistrar:url>
  </rdeRegistrar:whoisInfo>
  <rdeRegistrar:crDate>2005-04-23T11:49:00.0Z
</rdeRegistrar:crDate>
  <rdeRegistrar:update>2009-02-17T17:51:00.0Z
</rdeRegistrar:update>
</rdeRegistrar:registrar>

<!-- IDN Table -->
```

```
<rdeIDN:idnTableRef id="pt-BR">
  <rdeIDN:url>
http://www.iana.org/domains/idn-tables/tables/br_pt-br_1.0.html
  </rdeIDN:url>
  <rdeIDN:urlPolicy>
    http://registro.br/dominio/regras.html
  </rdeIDN:urlPolicy>
</rdeIDN:idnTableRef>

<!-- NNDN: pinguino.example -->
<rdeNNDN:NNDN>
  <rdeNNDN:aName>xn--exempl-gva.example</rdeNNDN:aName>
  <rdeNNDN:idnTableId>pt-BR</rdeNNDN:idnTableId>
  <rdeNNDN:originalName>example1.example</rdeNNDN:originalName>
  <rdeNNDN:nameState>withheld</rdeNNDN:nameState>
  <rdeNNDN:crDate>2005-04-23T11:49:00.0Z</rdeNNDN:crDate>
</rdeNNDN:NNDN>

<!-- EppParams -->
<rdeEppParams:eppParams>
  <rdeEppParams:version>1.0</rdeEppParams:version>
  <rdeEppParams:lang>en</rdeEppParams:lang>
  <rdeEppParams:objURI>
    urn:ietf:params:xml:ns:domain-1.0
  </rdeEppParams:objURI>
  <rdeEppParams:objURI>
    urn:ietf:params:xml:ns:contact-1.0
  </rdeEppParams:objURI>
  <rdeEppParams:objURI>
    urn:ietf:params:xml:ns:host-1.0
  </rdeEppParams:objURI>
  <rdeEppParams:svcExtension>
    <epp:extURI>urn:ietf:params:xml:ns:rgp-1.0
    </epp:extURI>
    <epp:extURI>urn:ietf:params:xml:ns:secDNS-1.1
    </epp:extURI>
  </rdeEppParams:svcExtension>
  <rdeEppParams:dcp>
  <epp:access><epp:all/></epp:access>
    <epp:statement>
      <epp:purpose>
        <epp:admin/>
        <epp:prov/>
      </epp:purpose>
      <epp:recipient>
        <epp:ours/>
        <epp:public/>
      </epp:recipient>
```

```

        <epp:retention>
          <epp:stated/>
        </epp:retention>
      </epp:statement>
    </rdeEppParams:dcp>
  </rdeEppParams:eppParams>
<rdePolicy:policy
  scope="//rde:deposit/rde:contents/rdeDomain:domain"
  element="rdeDomain:registrant" />
</rde:contents>
</rde:deposit>

```

18. Example of Differential Deposit using the XML model

Example of a Differential Deposit using the XML model:

```

<?xml version="1.0" encoding="UTF-8"?>
<rde:deposit type="DIFF" id="20191017002" prevId="20191017001"
  xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
  xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
  xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1"
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns:rdeHeader="urn:ietf:params:xml:ns:rdeHeader-1.0"
  xmlns:rdeDomain="urn:ietf:params:xml:ns:rdeDomain-1.0"
  xmlns:rdeHost="urn:ietf:params:xml:ns:rdeHost-1.0"
  xmlns:rdeContact="urn:ietf:params:xml:ns:rdeContact-1.0"
  xmlns:rdeRegistrar="urn:ietf:params:xml:ns:rdeRegistrar-1.0"
  xmlns:rdeIDN="urn:ietf:params:xml:ns:rdeIDN-1.0"
  xmlns:rdeNNDN="urn:ietf:params:xml:ns:rdeNNDN-1.0"
  xmlns:rdeEppParams="urn:ietf:params:xml:ns:rdeEppParams-1.0"
  xmlns:epp="urn:ietf:params:xml:ns:epp-1.0">

  <rde:watermark>2019-10-17T00:00:00Z</rde:watermark>
  <rde:rdeMenu>
    <rde:version>1.0</rde:version>
    <rde:objURI>urn:ietf:params:xml:ns:rdeHeader-1.0
    </rde:objURI>
    <rde:objURI>urn:ietf:params:xml:ns:rdeContact-1.0
    </rde:objURI>
    <rde:objURI>urn:ietf:params:xml:ns:rdeHost-1.0
    </rde:objURI>
    <rde:objURI>urn:ietf:params:xml:ns:rdeDomain-1.0
    </rde:objURI>
    <rde:objURI>urn:ietf:params:xml:ns:rdeRegistrar-1.0
    </rde:objURI>
    <rde:objURI>urn:ietf:params:xml:ns:rdeIDN-1.0
    </rde:objURI>
    <rde:objURI>urn:ietf:params:xml:ns:rdeNNDN-1.0

```

```

    </rde:objURI>
    <rde:objURI>urn:ietf:params:xml:ns:rdeEppParams-1.0
  </rde:objURI>
</rde:rdeMenu>

<!-- Deletes -->
<rde:deletes>
  <rdeDomain:delete>
    <rdeDomain:name>example2.example</rdeDomain:name>
  </rdeDomain:delete>
</rde:deletes>

<!-- Contents -->
<rde:contents>
  <!-- Header -->
  <rdeHeader:header>
    <rdeHeader:tld>test</rdeHeader:tld>
    <rdeHeader:count
      uri="urn:ietf:params:xml:ns:rdeDomain-1.0">1
    </rdeHeader:count>
    <rdeHeader:count
      uri="urn:ietf:params:xml:ns:rdeHost-1.0">1
    </rdeHeader:count>
    <rdeHeader:count
      uri="urn:ietf:params:xml:ns:rdeContact-1.0">1
    </rdeHeader:count>
    <rdeHeader:count
      uri="urn:ietf:params:xml:ns:rdeRegistrar-1.0">1
    </rdeHeader:count>
    <rdeHeader:count
      uri="urn:ietf:params:xml:ns:rdeIDN-1.0">1
    </rdeHeader:count>
    <rdeHeader:count
      uri="urn:ietf:params:xml:ns:rdeNNDN-1.0">1
    </rdeHeader:count>
    <rdeHeader:count
      uri="urn:ietf:params:xml:ns:rdeEppParams-1.0">1
    </rdeHeader:count>
  </rdeHeader:header>
</rde:contents>
</rde:deposit>

```

19. Example of a Full Deposit using the CSV model

Example of a Full Deposit using the CSV model:

```

<?xml version="1.0" encoding="UTF-8"?>
<rde:deposit

```

```
xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
xmlns:rdeCsv="urn:ietf:params:xml:ns:rdeCsv-1.0"
xmlns:csvDomain="urn:ietf:params:xml:ns:csvDomain-1.0"
xmlns:csvHost="urn:ietf:params:xml:ns:csvHost-1.0"
xmlns:csvContact="urn:ietf:params:xml:ns:csvContact-1.0"
xmlns:csvRegistrar="urn:ietf:params:xml:ns:csvRegistrar-1.0"
xmlns:csvIDN="urn:ietf:params:xml:ns:csvIDN-1.0"
xmlns:rdeHeader="urn:ietf:params:xml:ns:rdeHeader-1.0"
xmlns:csvNNDN="urn:ietf:params:xml:ns:csvNNDN-1.0"
xmlns:rdeEppParams="urn:ietf:params:xml:ns:rdeEppParams-1.0"
  type="FULL"
id="20191017001">
<rde:watermark>2019-10-18T00:00:00Z</rde:watermark>
<rde:rdeMenu>
  <rde:version>1.0</rde:version>
  <rde:objURI>urn:ietf:params:xml:ns:csvDomain-1.0</rde:objURI>
  <rde:objURI>urn:ietf:params:xml:ns:csvHost-1.0</rde:objURI>
  <rde:objURI>urn:ietf:params:xml:ns:csvContact-1.0</rde:objURI>
  <rde:objURI>urn:ietf:params:xml:ns:csvRegistrar-1.0</rde:objURI>
  <rde:objURI>urn:ietf:params:xml:ns:csvIDN-1.0</rde:objURI>
  <rde:objURI>urn:ietf:params:xml:ns:csvNNDN-1.0</rde:objURI>
  <rde:objURI>urn:ietf:params:xml:ns:rdeEppParams-1.0</rde:objURI>
</rde:rdeMenu>
<rde:contents>
  <rdeHeader:header>
    <rdeHeader:tld>test</rdeHeader:tld>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:csvDomain-1.0">
      4
    </rdeHeader:count>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:csvHost-1.0">
      6
    </rdeHeader:count>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:csvContact-1.0">
      9
    </rdeHeader:count>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:csvRegistrar-1.0">
      3
    </rdeHeader:count>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:csvIDN-1.0">
      2
    </rdeHeader:count>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:csvNNDN-1.0">
      2
    </rdeHeader:count>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:rdeEppParams-1.0">
      1
    </rdeHeader:count>
```

```
</rdeHeader:header>
<csvDomain:contents>
  <rdeCsv:csv name="domain" sep=",">
    <rdeCsv:fields>
      <csvDomain:fName/>
      <rdeCsv:fRoid/>
      <rdeCsv:fIdnTableId/>
      <csvDomain:fOriginalName/>
      <rdeCsv:fRegistrant/>
      <rdeCsv:fClID/>
      <rdeCsv:fCrRr/>
      <rdeCsv:fCrID/>
      <rdeCsv:fCrDate/>
      <rdeCsv:fUpRr/>
      <rdeCsv:fUpID/>
      <rdeCsv:fUpDate/>
      <rdeCsv:fExDate isRequired="true"/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="75E2D01F">
        domain-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
  <rdeCsv:csv name="domainContacts" sep=",">
    <rdeCsv:fields>
      <csvDomain:fName parent="true"/>
      <csvContact:fId/>
      <csvDomain:fContactType/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="70A7C17B">
        domainContacts-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
  <rdeCsv:csv name="domainStatuses" sep=",">
    <rdeCsv:fields>
      <csvDomain:fName parent="true"/>
      <csvDomain:fStatus/>
      <rdeCsv:fStatusDescription/>
      <rdeCsv:fLang/>
      <csvDomain:fRgpStatus/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
```



```
        cksum="EB8C548E">
        domainStatuses-YYYYMMDD.csv
    </rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="domainNameServers" sep=", ">
    <rdeCsv:fields>
        <csvDomain:fName parent="true"/>
        <csvHost:fName parent="true"/>
    </rdeCsv:fields>
    <rdeCsv:files>
        <rdeCsv:file
            cksum="984C3097">
            domainNameServers-name-YYYYMMDD.csv
        </rdeCsv:file>
    </rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="domainNameServers" sep=", ">
    <rdeCsv:fields>
        <csvDomain:fName parent="true"/>
        <rdeCsv:fRoid/>
    </rdeCsv:fields>
    <rdeCsv:files>
        <rdeCsv:file
            cksum="569D4638">
            domainNameServers-roid-YYYYMMDD.csv
        </rdeCsv:file>
    </rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="dnssec" sep=", ">
    <rdeCsv:fields>
        <csvDomain:fName parent="true"/>
        <csvDomain:fMaxSigLife/>
        <csvDomain:fKeyTag/>
        <csvDomain:fDsAlg/>
        <csvDomain:fDigestType/>
        <csvDomain:fDigest/>
    </rdeCsv:fields>
    <rdeCsv:files>
        <rdeCsv:file
            cksum="AA15CB43">
            dnssec-ds-YYYYMMDD.csv
        </rdeCsv:file>
    </rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="dnssec" sep=", ">
    <rdeCsv:fields>
        <csvDomain:fName parent="true"/>
```

```
<csvDomain:fMaxSigLife/>
<csvDomain:fFlags/>
<csvDomain:fProtocol/>
<csvDomain:fKeyAlg/>
<csvDomain:fPubKey/>
</rdeCsv:fields>
<rdeCsv:files>
  <rdeCsv:file
    cksum="1B16F334">
    dnssec-key-YYYYMMDD.csv
  </rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="domainTransfer" sep=", ">
  <rdeCsv:fields>
    <csvDomain:fName parent="true"/>
    <rdeCsv:fTrStatus/>
    <rdeCsv:fReRr/>
    <rdeCsv:fReID/>
    <rdeCsv:fReDate/>
    <rdeCsv:fAcRr/>
    <rdeCsv:fAcID/>
    <rdeCsv:fAcDate/>
    <rdeCsv:fExDate/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file
      cksum="71170194">
      domainTransfer-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
</rdeCsv:csv>
</csvDomain:contents>
<csvHost:contents>
  <rdeCsv:csv name="host" sep=", ">
    <rdeCsv:fields>
      <csvHost:fName/>
      <rdeCsv:fRoid/>
      <rdeCsv:fClID/>
      <rdeCsv:fCrRr/>
      <rdeCsv:fCrID/>
      <rdeCsv:fCrDate/>
      <rdeCsv:fUpRr/>
      <rdeCsv:fUpID/>
      <rdeCsv:fUpDate/>
      <rdeCsv:fTrDate/>
    </rdeCsv:fields>
  <rdeCsv:files>
```

```
<rdeCsv:file
  cksum="120938E3">
  host-YYYYMMDD.csv
</rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="hostStatuses" sep=", ">
  <rdeCsv:fields>
    <rdeCsv:fRoid parent="true"/>
    <csvHost:fStatus/>
    <rdeCsv:fStatusDescription/>
    <rdeCsv:fLang/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file
      cksum="0BA504FC">
      hostStatuses-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="hostAddresses" sep=", ">
  <rdeCsv:fields>
    <rdeCsv:fRoid parent="true"/>
    <csvHost:fAddr isRequired="true"/>
    <csvHost:fAddrVersion isRequired="true"/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file
      cksum="17888F02">
      hostAddresses-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
</rdeCsv:csv>
</csvHost:contents>
<csvContact:contents>
  <rdeCsv:csv name="contact" sep=", ">
    <rdeCsv:fields>
      <csvContact:fId/>
      <rdeCsv:fRoid/>
      <csvContact:fVoice/>
      <csvContact:fVoiceExt/>
      <csvContact:fFax/>
      <csvContact:fFaxExt/>
      <csvContact:fEmail/>
      <rdeCsv:fClID/>
      <rdeCsv:fCrRr/>
      <rdeCsv:fCrID/>
      <rdeCsv:fCrDate/>
```

```
<rdeCsv:fUpRr/>
<rdeCsv:fUpID/>
<rdeCsv:fUpDate/>
</rdeCsv:fields>
<rdeCsv:files>
  <rdeCsv:file
    cksum="D7F106A5">
    contact-YYYYMMDD.csv
  </rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="contactStatuses" sep=",">
  <rdeCsv:fields>
    <csvContact:fId parent="true"/>
    <csvContact:fStatus/>
    <rdeCsv:fStatusDescription/>
    <rdeCsv:fLang/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file
      cksum="2AAF99D4">
      contactStatuses-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="contactPostal" sep=",">
  <rdeCsv:fields>
    <csvContact:fId parent="true"/>
    <csvContact:fPostalType/>
    <csvContact:fName/>
    <csvContact:fOrg/>
    <csvContact:fStreet index="0"/>
    <csvContact:fStreet index="1"/>
    <csvContact:fStreet index="2"/>
    <csvContact:fCity/>
    <csvContact:fSp/>
    <csvContact:fPc/>
    <csvContact:fCc/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file
      cksum="02CC2504">
      contactPostal-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="contactTransfer" sep=",">
  <rdeCsv:fields>
```

```

        <csvContact:fId parent="true"/>
        <rdeCsv:fTrStatus/>
        <rdeCsv:fReRr/>
        <rdeCsv:fReID/>
        <rdeCsv:fReDate/>
        <rdeCsv:fAcRr/>
        <rdeCsv:fAcID/>
        <rdeCsv:fAcDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
        <rdeCsv:file
            cksum="D0929632">
            contactTransfer-YYYYMMDD.csv
        </rdeCsv:file>
    </rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="contactDisclose" sep=",">
    <rdeCsv:fields>
        <csvContact:fId parent="true"/>
        <csvContact:fDiscloseFlag/>
        <csvContact:fDiscloseNameLoc/>
        <csvContact:fDiscloseNameInt/>
        <csvContact:fDiscloseOrgLoc/>
        <csvContact:fDiscloseOrgInt/>
        <csvContact:fDiscloseAddrLoc/>
        <csvContact:fDiscloseAddrInt/>
        <csvContact:fDiscloseVoice/>
        <csvContact:fDiscloseFax/>
        <csvContact:fDiscloseEmail/>
    </rdeCsv:fields>
    <rdeCsv:files>
        <rdeCsv:file
            cksum="89043A90">
            contactDisclose-YYYYMMDD.csv
        </rdeCsv:file>
    </rdeCsv:files>
</rdeCsv:csv>
</csvContact:contents>
<csvRegistrar:contents>
    <rdeCsv:csv name="registrar" sep=",">
        <rdeCsv:fields>
            <csvRegistrar:fId/>
            <csvRegistrar:fName isLoc="false"/>
            <csvRegistrar:fGurid/>
            <csvRegistrar:fStatus/>
            <csvContact:fStreet isLoc="false" index="0"/>
            <csvContact:fStreet isLoc="false" index="1"/>
            <csvContact:fStreet isLoc="false" index="2"/>

```

```
<csvContact:fCity isLoc="false" />
<csvContact:fSp isLoc="false" />
<csvContact:fPc isLoc="false" />
<csvContact:fCc isLoc="false" />
<csvContact:fVoice/>
<csvContact:fVoiceExt/>
<csvContact:fFax/>
<csvContact:fFaxExt/>
<csvContact:fEmail isRequired="false"/>
<rdeCsv:fUrl/>
<csvRegistrar:fWhoisUrl/>
<rdeCsv:fCrDate/>
<rdeCsv:fUpDate/>
</rdeCsv:fields>
<rdeCsv:files>
  <rdeCsv:file
    cksum="306178BB">
    registrar-YYYYMMDD.csv
  </rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
</csvRegistrar:contents>
<csvIDN:contents>
  <rdeCsv:csv name="idnLanguage" sep=",">
    <rdeCsv:fields>
      <rdeCsv:fIdnTableId isRequired="true"/>
      <rdeCsv:fUrl isRequired="true"/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="D462EAD0">
        idnLanguage-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
</csvIDN:contents>
<csvNNDN:contents>
  <rdeCsv:csv name="NNDN" sep=",">
    <rdeCsv:fields>
      <csvNNDN:fAName/>
      <rdeCsv:fIdnTableId/>
      <csvNNDN:fOriginalName/>
      <csvNNDN:fNameState/>
      <csvNNDN:fMirroringNS/>
      <rdeCsv:fCrDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
```

```
        cksum="11C80D60">
        NNDN-YYYYMMDD.csv
    </rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
</csvNNDN:contents>
<rdeEppParams:eppParams>
  <rdeEppParams:version>1.0</rdeEppParams:version>
  <rdeEppParams:lang>en</rdeEppParams:lang>
  <rdeEppParams:objURI>urn:ietf:params:xml:ns:domain-1.0
</rdeEppParams:objURI>
  <rdeEppParams:objURI>urn:ietf:params:xml:ns:host-1.0
</rdeEppParams:objURI>
  <rdeEppParams:objURI>urn:ietf:params:xml:ns:contact-1.0
</rdeEppParams:objURI>
  <rdeEppParams:svcExtension>
    <epp:extURI>urn:ietf:params:xml:ns:secDNS-1.1
    </epp:extURI>
    <epp:extURI>urn:ietf:params:xml:ns:rgp-1.0
    </epp:extURI>
  </rdeEppParams:svcExtension>
<rdeEppParams:dcp>
  <epp:access>
    <epp:all/>
  </epp:access>
  <epp:statement>
    <epp:purpose>
      <epp:admin/>
      <epp:other/>
      <epp:prov/>
    </epp:purpose>
    <epp:recipient>
      <epp:ours/>
      <epp:public/>
      <epp:unrelated/>
    </epp:recipient>
    <epp:retention>
      <epp:indefinite/>
    </epp:retention>
  </epp:statement>
</rdeEppParams:dcp>
</rdeEppParams:eppParams>
</rde:contents>
</rde:deposit>
```

20. Example of Differential Deposit using the CSV model

Example of a Differential Deposit using the CSV model:

```
<?xml version="1.0" encoding="UTF-8"?>
<rde:deposit
  xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
  xmlns:rdeCsv="urn:ietf:params:xml:ns:rdeCsv-1.0"
  xmlns:csvDomain="urn:ietf:params:xml:ns:csvDomain-1.0"
  xmlns:csvHost="urn:ietf:params:xml:ns:csvHost-1.0"
  xmlns:csvContact="urn:ietf:params:xml:ns:csvContact-1.0"
  xmlns:csvRegistrar="urn:ietf:params:xml:ns:csvRegistrar-1.0"
  xmlns:csvIDN="urn:ietf:params:xml:ns:csvIDN-1.0"
  xmlns:rdeHeader="urn:ietf:params:xml:ns:rdeHeader-1.0"
  xmlns:csvNNDN="urn:ietf:params:xml:ns:csvNNDN-1.0"
  xmlns:rdeEppParams="urn:ietf:params:xml:ns:rdeEppParams-1.0"
  type="DIFF"
  id="20191017001" prevId="20191010001">
  <rde:watermark>2019-10-18T00:00:00Z</rde:watermark>
  <rde:rdeMenu>
    <rde:version>1.0</rde:version>
    <rde:objURI>urn:ietf:params:xml:ns:csvDomain-1.0</rde:objURI>
    <rde:objURI>urn:ietf:params:xml:ns:csvHost-1.0</rde:objURI>
    <rde:objURI>urn:ietf:params:xml:ns:csvContact-1.0</rde:objURI>
    <rde:objURI>urn:ietf:params:xml:ns:csvRegistrar-1.0</rde:objURI>
    <rde:objURI>urn:ietf:params:xml:ns:csvIDN-1.0</rde:objURI>
  </rde:rdeMenu>
  <rde:deletes>
    <csvDomain:deletes>
      <rdeCsv:csv name="domain">
        <rdeCsv:fields>
          <csvDomain:fName/>
        </rdeCsv:fields>
        <rdeCsv:files>
          <rdeCsv:file
            cksum="6F2B988F">
            domain-delete-YYYYMMDD.csv
          </rdeCsv:file>
        </rdeCsv:files>
      </rdeCsv:csv>
    </csvDomain:deletes>
    <csvHost:deletes>
      <rdeCsv:csv name="host">
        <rdeCsv:fields>
          <rdeCsv:fRoid/>
        </rdeCsv:fields>
        <rdeCsv:files>
```



```
<rdeCsv:file
  cksum="E3408F5E">
  host-delete-YYYYMMDD.csv
</rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
</csvHost:deletes>
<csvContact:deletes>
  <rdeCsv:csv name="contact">
    <rdeCsv:fields>
      <csvContact:fId/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="6F2B988F">
        contact-delete-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
</csvContact:deletes>
<csvRegistrar:deletes>
  <rdeCsv:csv name="registrar">
    <rdeCsv:fields>
      <csvRegistrar:fId/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="307B87AE">
        registrar-delete-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
</csvRegistrar:deletes>
<csvIDN:deletes>
  <rdeCsv:csv name="idnLanguage">
    <rdeCsv:fields>
      <rdeCsv:fIdnTableId/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="757B573A">
        idnLanguage-delete-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
</csvIDN:deletes>
<csvNNDN:deletes>
  <rdeCsv:csv name="NNDN">
```

```
<rdeCsv:fields>
  <csvNNDN:fAName/>
</rdeCsv:fields>
<rdeCsv:files>
  <rdeCsv:file
    cksum="FF104E83">
    NNDN-delete-YYYYMMDD.csv
  </rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
</csvNNDN:deletes>
</rde:deletes>
<rde:contents>
  <rdeHeader:header>
    <rdeHeader:tld>test</rdeHeader:tld>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:csvDomain-1.0">
      2
    </rdeHeader:count>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:csvHost-1.0">
      2
    </rdeHeader:count>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:csvContact-1.0">
      3
    </rdeHeader:count>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:csvRegistrar-1.0">
      1
    </rdeHeader:count>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:csvIDN-1.0">
      1
    </rdeHeader:count>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:csvNNDN-1.0">
      1
    </rdeHeader:count>
    <rdeHeader:count uri="urn:ietf:params:xml:ns:rdeEppParams-1.0">
      1
    </rdeHeader:count>
  </rdeHeader:header>
  <csvDomain:contents>
    <rdeCsv:csv name="domain" sep=",">
      <rdeCsv:fields>
        <csvDomain:fName/>
        <rdeCsv:fRoid/>
        <rdeCsv:fIdnTableId/>
        <csvDomain:fOriginalName/>
        <rdeCsv:fRegistrant/>
        <rdeCsv:fClID/>
        <rdeCsv:fCrRr/>
        <rdeCsv:fCrID/>
```

```
<rdeCsv:fCrDate/>
<rdeCsv:fUpRr/>
<rdeCsv:fUpID/>
<rdeCsv:fUpDate/>
<rdeCsv:fExDate isRequired="true"/>
</rdeCsv:fields>
<rdeCsv:files>
  <rdeCsv:file
    cksum="75E2D01F">
    domain-YYYYMMDD.csv
  </rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="domainContacts" sep=", ">
  <rdeCsv:fields>
    <csvDomain:fName parent="true"/>
    <csvContact:fId/>
    <csvDomain:fContactType/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file
      cksum="70A7C17B">
      domainContacts-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="domainStatuses" sep=", ">
  <rdeCsv:fields>
    <csvDomain:fName parent="true"/>
    <csvDomain:fStatus/>
    <rdeCsv:fStatusDescription/>
    <rdeCsv:fLang/>
    <csvDomain:fRgpStatus/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file
      cksum="EB8C548E">
      domainStatuses-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="domainNameServers" sep=", ">
  <rdeCsv:fields>
    <csvDomain:fName parent="true"/>
    <csvHost:fName parent="true"/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file
```

```
        cksum="984C3097">
        domainNameServers-name-YYYYMMDD.csv
    </rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="domainNameServers" sep=", ">
    <rdeCsv:fields>
        <csvDomain:fName parent="true"/>
        <rdeCsv:fRoid/>
    </rdeCsv:fields>
    <rdeCsv:files>
        <rdeCsv:file
            cksum="569D4638">
            domainNameServers-roid-YYYYMMDD.csv
        </rdeCsv:file>
    </rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="dnssec" sep=", ">
    <rdeCsv:fields>
        <csvDomain:fName parent="true"/>
        <csvDomain:fMaxSigLife/>
        <csvDomain:fKeyTag/>
        <csvDomain:fDsAlg/>
        <csvDomain:fDigestType/>
        <csvDomain:fDigest/>
    </rdeCsv:fields>
    <rdeCsv:files>
        <rdeCsv:file
            cksum="AA15CB43">
            dnssec-ds-YYYYMMDD.csv
        </rdeCsv:file>
    </rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="dnssec" sep=", ">
    <rdeCsv:fields>
        <csvDomain:fName parent="true"/>
        <csvDomain:fMaxSigLife/>
        <csvDomain:fFlags/>
        <csvDomain:fProtocol/>
        <csvDomain:fKeyAlg/>
        <csvDomain:fPubKey/>
    </rdeCsv:fields>
    <rdeCsv:files>
        <rdeCsv:file
            cksum="1B16F334">
            dnssec-key-YYYYMMDD.csv
        </rdeCsv:file>
    </rdeCsv:files>
```

```
</rdeCsv:csv>
<rdeCsv:csv name="domainTransfer" sep=", ">
  <rdeCsv:fields>
    <csvDomain:fName parent="true"/>
    <rdeCsv:fTrStatus/>
    <rdeCsv:fReRr/>
    <rdeCsv:fReID/>
    <rdeCsv:fReDate/>
    <rdeCsv:fAcRr/>
    <rdeCsv:fAcID/>
    <rdeCsv:fAcDate/>
    <rdeCsv:fExDate/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file
      cksum="71170194">
      domainTransfer-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
</rdeCsv:csv>
</csvDomain:contents>
<csvHost:contents>
  <rdeCsv:csv name="host" sep=", ">
    <rdeCsv:fields>
      <csvHost:fName/>
      <rdeCsv:fRoid/>
      <rdeCsv:fClID/>
      <rdeCsv:fCrRr/>
      <rdeCsv:fCrID/>
      <rdeCsv:fCrDate/>
      <rdeCsv:fUpRr/>
      <rdeCsv:fUpID/>
      <rdeCsv:fUpDate/>
      <rdeCsv:fTrDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="120938E3">
        host-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
  <rdeCsv:csv name="hostStatuses" sep=", ">
    <rdeCsv:fields>
      <rdeCsv:fRoid parent="true"/>
      <csvHost:fStatus/>
      <rdeCsv:fStatusDescription/>
      <rdeCsv:fLang/>
```

```
</rdeCsv:fields>
<rdeCsv:files>
  <rdeCsv:file
    cksum="0BA504FC">
    hostStatuses-YYYYMMDD.csv
  </rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="hostAddresses" sep=", ">
  <rdeCsv:fields>
    <rdeCsv:fRoid parent="true"/>
    <csvHost:fAddr isRequired="true"/>
    <csvHost:fAddrVersion isRequired="true"/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file
      cksum="17888F02">
      hostAddresses-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
</rdeCsv:csv>
</csvHost:contents>
<csvContact:contents>
  <rdeCsv:csv name="contact" sep=", ">
    <rdeCsv:fields>
      <csvContact:fId/>
      <rdeCsv:fRoid/>
      <csvContact:fVoice/>
      <csvContact:fVoiceExt/>
      <csvContact:fFax/>
      <csvContact:fFaxExt/>
      <csvContact:fEmail/>
      <rdeCsv:fClID/>
      <rdeCsv:fCrRr/>
      <rdeCsv:fCrID/>
      <rdeCsv:fCrDate/>
      <rdeCsv:fUpRr/>
      <rdeCsv:fUpID/>
      <rdeCsv:fUpDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="D7F106A5">
        contact-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
  <rdeCsv:csv name="contactStatuses" sep=", ">
```

```
<rdeCsv:fields>
  <csvContact:fId parent="true"/>
  <csvContact:fStatus/>
  <rdeCsv:fStatusDescription/>
  <rdeCsv:fLang/>
</rdeCsv:fields>
<rdeCsv:files>
  <rdeCsv:file
    cksum="2AAF99D4">
    contactStatuses-YYYYMMDD.csv
  </rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="contactPostal" sep=",">
  <rdeCsv:fields>
    <csvContact:fId parent="true"/>
    <csvContact:fPostalType/>
    <csvContact:fName/>
    <csvContact:fOrg/>
    <csvContact:fStreet index="0"/>
    <csvContact:fStreet index="1"/>
    <csvContact:fStreet index="2"/>
    <csvContact:fCity/>
    <csvContact:fSp/>
    <csvContact:fPc/>
    <csvContact:fCc/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file
      cksum="02CC2504">
      contactPostal-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="contactTransfer" sep=",">
  <rdeCsv:fields>
    <csvContact:fId parent="true"/>
    <rdeCsv:fTrStatus/>
    <rdeCsv:fReRr/>
    <rdeCsv:fReID/>
    <rdeCsv:fReDate/>
    <rdeCsv:fAcRr/>
    <rdeCsv:fAcID/>
    <rdeCsv:fAcDate/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file
      cksum="D0929632">
```

```
        contactTransfer-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
  <rdeCsv:csv name="contactDisclose" sep=",">
    <rdeCsv:fields>
      <csvContact:fId parent="true"/>
      <csvContact:fDiscloseFlag/>
      <csvContact:fDiscloseNameLoc/>
      <csvContact:fDiscloseNameInt/>
      <csvContact:fDiscloseOrgLoc/>
      <csvContact:fDiscloseOrgInt/>
      <csvContact:fDiscloseAddrLoc/>
      <csvContact:fDiscloseAddrInt/>
      <csvContact:fDiscloseVoice/>
      <csvContact:fDiscloseFax/>
      <csvContact:fDiscloseEmail/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="89043A90">
        contactDisclose-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
</csvContact:contents>
<csvRegistrar:contents>
  <rdeCsv:csv name="registrar" sep=",">
    <rdeCsv:fields>
      <csvRegistrar:fId/>
      <csvRegistrar:fName isLoc="false"/>
      <csvRegistrar:fGurid/>
      <csvRegistrar:fStatus/>
      <csvContact:fStreet isLoc="false" index="0"/>
      <csvContact:fStreet isLoc="false" index="1"/>
      <csvContact:fStreet isLoc="false" index="2"/>
      <csvContact:fCity isLoc="false" />
      <csvContact:fSp isLoc="false" />
      <csvContact:fPc isLoc="false" />
      <csvContact:fCc isLoc="false" />
      <csvContact:fVoice/>
      <csvContact:fVoiceExt/>
      <csvContact:fFax/>
      <csvContact:fFaxExt/>
      <csvContact:fEmail isRequired="false"/>
      <rdeCsv:fUrl/>
      <csvRegistrar:fWhoisUrl/>
      <rdeCsv:fCrDate/>
    </rdeCsv:fields>
  </rdeCsv:csv>
</csvRegistrar:contents>
```



```
<rdeCsv:fUpDate/>
</rdeCsv:fields>
<rdeCsv:files>
  <rdeCsv:file
    cksum="306178BB">
    registrar-YYYYMMDD.csv
  </rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
</csvRegistrar:contents>
<csvIDN:contents>
  <rdeCsv:csv name="idnLanguage" sep=",">
    <rdeCsv:fields>
      <rdeCsv:fIdnTableId isRequired="true"/>
      <rdeCsv:fUrl isRequired="true"/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="D462EAD0">
        idnLanguage-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
</csvIDN:contents>
<csvNNDN:contents>
  <rdeCsv:csv name="NNDN" sep=",">
    <rdeCsv:fields>
      <csvNNDN:fAName/>
      <rdeCsv:fIdnTableId/>
      <csvNNDN:fOriginalName/>
      <csvNNDN:fNameState/>
      <csvNNDN:fMirroringNS/>
      <rdeCsv:fCrDate/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file
        cksum="11C80D60">
        NNDN-YYYYMMDD.csv
      </rdeCsv:file>
    </rdeCsv:files>
  </rdeCsv:csv>
</csvNNDN:contents>
<rdeEppParams:eppParams>
  <rdeEppParams:version>1.0</rdeEppParams:version>
  <rdeEppParams:lang>en</rdeEppParams:lang>
  <rdeEppParams:objURI>urn:ietf:params:xml:ns:domain-1.0
</rdeEppParams:objURI>
  <rdeEppParams:objURI>urn:ietf:params:xml:ns:host-1.0
```

```
</rdeEppParams:objURI>
<rdeEppParams:objURI>urn:ietf:params:xml:ns:contact-1.0
</rdeEppParams:objURI>
<rdeEppParams:svcExtension>
  <epp:extURI>urn:ietf:params:xml:ns:secDNS-1.1
  </epp:extURI>
  <epp:extURI>urn:ietf:params:xml:ns:rgp-1.0
  </epp:extURI>
</rdeEppParams:svcExtension>
<rdeEppParams:dcP>
  <epp:access>
    <epp:all/>
  </epp:access>
  <epp:statement>
    <epp:purpose>
      <epp:admin/>
      <epp:other/>
      <epp:prov/>
    </epp:purpose>
    <epp:recipient>
      <epp:ours/>
      <epp:public/>
      <epp:unrelated/>
    </epp:recipient>
    <epp:retention>
      <epp:indefinite/>
    </epp:retention>
  </epp:statement>
</rdeEppParams:dcP>
</rdeEppParams:eppParams>
</rde:contents>
</rde:deposit>
```

21. References

21.1. Normative References

- [BCP195] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/bcp195>>.
- [I-D.ietf-regext-data-escrow] Lozano, G., "Registry Data Escrow Specification", draft-ietf-regext-data-escrow-10 (work in progress), June 2020.

- [ISO-3166-1] 3166, I. S., "Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes", ISO Standard 3166, November 2006.
- [ITU-E164] International Telecommunication Union, "The international public telecommunication numbering plan", ITU-T Recommendation E.164, February 2005.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [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>>.
- [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>>.
- [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>>.

- [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>>.
- [RFC5910] Gould, J. and S. Hollenbeck, "Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)", RFC 5910, DOI 10.17487/RFC5910, May 2010, <<https://www.rfc-editor.org/info/rfc5910>>.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <<https://www.rfc-editor.org/info/rfc5952>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [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>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [V42] International Telecommunication Union, "V.42 : Error-correcting procedures for DCEs using asynchronous-to-synchronous conversion", March 2002, <<https://www.itu.int/rec/T-REC-V.42/en>>.
- [W3C.REC-xml-20081126]
Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition) REC-xml-20081126", November 2008, <<https://www.w3.org/TR/2008/REC-xml-20081126/>>.
- [W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition REC-xmlschema-1-20041028", October 2004, <<https://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>>.

- [W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes
Second Edition REC-xmlschema-2-20041028", October 2004,
<<https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>>.
- [W3C.REC-xpath-31-20170321]
Robie, J., Dyck, M., and J. Spiegel, "XML Path Language
(XPath) 3.1", March 2017,
<<https://www.w3.org/TR/2017/REC-xpath-31-20170321/>>.

21.2. Informative References

- [ICANN-GTLD-AGB-20120604]
ICANN, "gTLD Applicant Guidebook Version 2012-06-04", June
2012, <[http://newgtlds.icann.org/en/applicants/agb/
guidebook-full-04jun12-en.pdf](http://newgtlds.icann.org/en/applicants/agb/guidebook-full-04jun12-en.pdf)>.
- [ICANN-GTLD-RA-20170731]
ICANN, "Base Registry Agreement 2017-07-31", July 2017,
<[https://newgtlds.icann.org/sites/default/files/
agreements/agreement-approved-31jul17-en.pdf](https://newgtlds.icann.org/sites/default/files/agreements/agreement-approved-31jul17-en.pdf)>.
- [RFC1952] Deutsch, P., "GZIP file format specification version 4.3",
RFC 1952, DOI 10.17487/RFC1952, May 1996,
<<https://www.rfc-editor.org/info/rfc1952>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818,
DOI 10.17487/RFC2818, May 2000,
<<https://www.rfc-editor.org/info/rfc2818>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
DOI 10.17487/RFC3688, January 2004,
<<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912,
DOI 10.17487/RFC3912, September 2004,
<<https://www.rfc-editor.org/info/rfc3912>>.
- [RFC4180] Shafranovich, Y., "Common Format and MIME Type for Comma-
Separated Values (CSV) Files", RFC 4180,
DOI 10.17487/RFC4180, October 2005,
<<https://www.rfc-editor.org/info/rfc4180>>.
- [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>>.

[variantTLDsReport]

Internet Corporation for Assigned Names and Numbers
(ICANN), "A Study of Issues Related to the Management of
IDN Variant TLDs", February 2012,
<<http://www.icann.org/en/topics/idn/idn-vip-integrated-issues-final-clean-20febl2-en.pdf>>.

Authors' Addresses

Gustavo Lozano
Internet Corporation for Assigned Names and Numbers
12025 Waterfront Drive, Suite 300
Los Angeles 90292
United States of America

Phone: +1.310.823.9358
Email: gustavo.lozano@icann.org

James Gould
VeriSign, Inc.
12061 Bluemont Way
Reston 20190
United States of America

Email: jgould@verisign.com

Chethan Thippeswamy
VeriSign, Inc.
12061 Bluemont Way
Reston 20190
United States of America

Email: cthippeswamy@verisign.com

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

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

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 23, 2020.

Copyright Notice

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

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

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

Table of Contents

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

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

1. Introduction

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

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

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

1.1. Conventions Used in This Document

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

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

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

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

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

2. Migrating to Newer Versions of This Extension

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

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

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

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

3. Extension Elements

3.1. Client Commands

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

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

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

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

3.2. Currency Codes

The <fee:currency> element is used to indicate which currency fees are charged in. This value of this element MUST be a three-character currency code from [ISO4217:2015].

Note that ISO 4217:2015 provides the special "XXX" code, which MAY be used if the server uses a non-currency based system for assessing fees, such as a system of credits.

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

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

3.3. Validity Periods

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

attribute to be added to the registration period of objects by the <create>, <renew> and <transfer> commands. This element is derived from the <domain:period> element described in [RFC5731].

The <fee:period> element is OPTIONAL in <check> commands, if omitted, the server MUST determine the fee(s) using the server default period. The <fee:period> element MUST be present in <check> responses.

3.4. Fees and Credits

Servers which implement this extension will include elements in responses which provide information about the fees and/or credits associated with a given billable transaction. A fee will result in subtracting from the Account Balance (described in Section 3.5) and a credit will result in adding to the Account Balance (described in Section 3.5).

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

A server MAY respond with multiple <fee:fee> and <fee:credit> elements in the same response. In such cases, the net fee or credit applicable to the transaction is the arithmetic sum of the values of each of the <fee:fee> and/or <fee:credit> elements. This amount applies to the total additional validity period applied to the object (where applicable).

The following attributes are defined for the <fee:fee> element. These are described in detail below:

description: an OPTIONAL attribute which provides a human-readable description of the fee. Servers should provide documentation on the possible values of this attribute, and their meanings. An OPTIONAL "lang" attribute MAY be present, per the language structure in [RFC5646], to identify the language of the returned text and has a default value of "en" (English). If the "description" attribute is not present, the "lang" attribute can be ignored.

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

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

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

The <fee:credit> element can take a "description" attribute as described above. An OPTIONAL "lang" attribute MAY be present to identify the language of the returned text and has a default value of "en" (English).

3.4.1. Refunds

<fee:fee> elements MAY have an OPTIONAL "refundable" attribute which takes a boolean value. Fees may be refunded under certain circumstances, such as when a domain application is rejected (as described in [RFC8334]) or when an object is deleted during the relevant Grace Period (see below).

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

3.4.2. Grace Periods

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

The "grace-period" attribute MAY be used to indicate the relevant grace period for a fee. If a server implements the Registry Grace Period extension [RFC3915], it MUST specify the grace period for all relevant transactions.

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

3.4.3. Correlation between Refundability and Grace Periods

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

3.4.4. Applicability

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

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

3.5. Account Balance

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

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

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

If a server includes a <fee:balance> element in response to transform commands, the value of the element MUST reflect the client's account balance after any fees or credits associated with that command have been applied. If the "applied" attribute of the <fee:fee> element is "delayed", then the <fee:balance> MUST reflect the client's account balance without any fees or credits associated with that command.

3.6. Credit Limit

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

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

3.7. Classification of Objects

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

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

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

3.8. Phase and Subphase Attributes

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

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

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

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

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

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

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

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

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

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

3.9. Reason

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

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

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

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

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

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

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

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

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

4. Server Handling of Fee Information

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

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

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

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

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

If the total fee provided by the client is less than the server's own calculation of the fee or the server determines the currency is inappropriate for that command, then the server MUST reject the command with a 2004 "Parameter value range" error.

5. EPP Command Mapping

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

5.1. EPP Query Commands

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

5.1.1. EPP <check> Command

This extension defines a new command called the Fee Check Command that defines additional elements for the EPP <check> command to provide fee information along with the availability information of the EPP <check> command.

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

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

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

Example <check> command:

```
C: <?xml version="1.0" encoding="utf-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <check>
C:       <domain:check
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:           <domain:name>example.com</domain:name>
C:           <domain:name>example.net</domain:name>
C:           <domain:name>example.xyz</domain:name>
C:         </domain:check>
C:       </check>
C:     <extension>
C:       <fee:check xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:command name="create">
C:           <fee:period unit="y">2</fee:period>
C:         </fee:command>
C:         <fee:command name="renew"/>
C:         <fee:command name="transfer"/>
C:         <fee:command name="restore"/>
C:       </fee:check>
C:     </extension>
C:   <clTRID>ABC-12345</clTRID>
C: </command>
C: </epp>
```

When the server receives a <check> command that includes the extension elements described above, its response MUST contain an <extension> element, which MUST contain a child <fee:chkData> element. The <fee:chkData> element MUST contain a <fee:currency> element and a <fee:cd> element for each object referenced in the client <check> command.

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

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

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

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

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

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

If the "avail" attribute of the `<fee:cd>` element is true (1) and if no `<fee:fee>` elements are present in a `<fee:command>` element, this indicates that no fee will be assessed by the server for this command.

If the "avail" attribute of the `<fee:cd>` element is true (1), then the `<fee:command>` element MUST NOT contain a `<fee:reason>` element.

Example `<check>` response:

```
S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <resData>
S:       <domain:chkData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:cd>
S:           <domain:name avail="1">example.com</domain:name>
S:         </domain:cd>
S:       </domain:cd>
```

```
S:      <domain:name avail="1">example.net</domain:name>
S:      </domain:cd>
S:      <domain:cd>
S:      <domain:name avail="1">example.xyz</domain:name>
S:      </domain:cd>
S:      </domain:chkData>
S:      </resData>
S:      <extension>
S:      <fee:chkData
S:      <xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:      <fee:currency>USD</fee:currency>
S:      <fee:cd avail="1">
S:      <fee:objID>example.com</fee:objID>
S:      <fee:class>Premium</fee:class>
S:      <fee:command name="create">
S:      <fee:period unit="y">2</fee:period>
S:      <fee:fee
S:      <description="Registration Fee"
S:      <refundable="1"
S:      <grace-period="P5D">10.00</fee:fee>
S:      </fee:command>
S:      <fee:command name="renew">
S:      <fee:period unit="y">1</fee:period>
S:      <fee:fee
S:      <description="Renewal Fee"
S:      <refundable="1"
S:      <grace-period="P5D">10.00</fee:fee>
S:      </fee:command>
S:      <fee:command name="transfer">
S:      <fee:period unit="y">1</fee:period>
S:      <fee:fee
S:      <description="Transfer Fee"
S:      <refundable="1"
S:      <grace-period="P5D">10.00</fee:fee>
S:      </fee:command>
S:      <fee:command name="restore">
S:      <fee:fee
S:      <description="Redemption Fee">15.00</fee:fee>
S:      </fee:command>
S:      </fee:cd>
S:      <fee:cd avail="1">
S:      <fee:objID>example.net</fee:objID>
S:      <fee:class>standard</fee:class>
S:      <fee:command name="create" standard="1">
S:      <fee:period unit="y">2</fee:period>
S:      <fee:fee
S:      <description="Registration Fee"
S:      <refundable="1"
```

```

S:         grace-period="P5D">5.00</fee:fee>
S:     </fee:command>
S:     <fee:command name="renew" standard="1">
S:         <fee:period unit="y">1</fee:period>
S:         <fee:fee
S:             description="Renewal Fee"
S:             refundable="1"
S:             grace-period="P5D">5.00</fee:fee>
S:     </fee:command>
S:     <fee:command name="transfer" standard="1">
S:         <fee:period unit="y">1</fee:period>
S:         <fee:fee
S:             description="Transfer Fee"
S:             refundable="1"
S:             grace-period="P5D">5.00</fee:fee>
S:     </fee:command>
S:     <fee:command name="restore" standard="1">
S:         <fee:fee
S:             description="Redemption Fee">5.00</fee:fee>
S:     </fee:command>
S: </fee:cd>
S: <fee:cd avail="0">
S:     <fee:objID>example.xyz</fee:objID>
S:     <fee:command name="create">
S:         <fee:period unit="y">2</fee:period>
S:         <fee:reason>Only 1 year registration periods are
S:             valid.</fee:reason>
S:     </fee:command>
S: </fee:cd>
S: </fee:chkData>
S: </extension>
S: <trID>
S:     <clTRID>ABC-12345</clTRID>
S:     <svTRID>54322-XYZ</svTRID>
S: </trID>
S: </response>
S: </epp>

```

5.1.2. EPP Transfer Query Command

This extension does not add any elements to the EPP <transfer> query command, but does include elements in the response, when the extension is included in the <login> command service extensions.

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

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

- o A <fee:currency> element (as described in Section 3.2).
- o A <fee:period> element (as described in Section 3.3).
- o Zero or more <fee:fee> elements (as described in Section 3.4) containing the fees that will be charged to the gaining client.
- o Zero or more <fee:credit> elements (as described in Section 3.4) containing the credits that will be refunded to the losing client.

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

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

Example <transfer> query response:

```
S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1001">
S:       <msg>Command completed successfully; action pending</msg>
S:     </result>
S:     <resData>
S:       <domain:trnData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>example.com</domain:name>
S:         <domain:trStatus>pending</domain:trStatus>
S:         <domain:reID>ClientX</domain:reID>
S:         <domain:reDate>2019-06-08T22:00:00.0Z</domain:reDate>
S:         <domain:acID>ClientY</domain:acID>
S:         <domain:acDate>2019-06-13T22:00:00.0Z</domain:acDate>
S:         <domain:exDate>2021-09-08T22:00:00.0Z</domain:exDate>
S:       </domain:trnData>
S:     </resData>
S:     <extension>
S:       <fee:trnData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:period unit="y">1</fee:period>
S:         <fee:fee>5.00</fee:fee>
S:       </fee:trnData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54322-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>
```

5.2. EPP Transform Commands

5.2.1. EPP <create> Command

This extension adds elements to both the EPP <create> command and response, when the extension is included in the <login> command service extensions.

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

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

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

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

Example <create> command:

```
C: <?xml version="1.0" encoding="utf-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <create>
C:       <domain:create
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:period unit="y">2</domain:period>
C:         <domain:ns>
C:           <domain:hostObj>ns1.example.net</domain:hostObj>
C:           <domain:hostObj>ns2.example.net</domain:hostObj>
C:         </domain:ns>
C:         <domain:registrant>jd1234</domain:registrant>
C:         <domain:contact type="admin">sh8013</domain:contact>
C:         <domain:contact type="tech">sh8013</domain:contact>
C:         <domain:authInfo>
C:           <domain:pw>2fooBAR</domain:pw>
C:         </domain:authInfo>
C:       </domain:create>
C:     </create>
C:     <extension>
C:       <fee:create xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:fee>5.00</fee:fee>
C:       </fee:create>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```

Example <create> response:

```
S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <resData>
S:       <domain:creData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>example.com</domain:name>
S:         <domain:crDate>2019-04-03T22:00:00.0Z</domain:crDate>
S:         <domain:exDate>2021-04-03T22:00:00.0Z</domain:exDate>
S:       </domain:creData>
S:     </resData>
S:     <extension>
S:       <fee:creData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee
S:           description="Registration Fee"
S:           lang="en"
S:           refundable="1"
S:           grace-period="P5D">5.00</fee:fee>
S:         <fee:balance>-5.00</fee:balance>
S:         <fee:creditLimit>1000.00</fee:creditLimit>
S:       </fee:creData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>
```

5.2.2. EPP <delete> Command

This extension does not add any elements to the EPP <delete> command, but does include elements in the response, when the extension is included in the <login> command service extensions.

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

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

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

Example <delete> response:

```
S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <extension>
S:       <fee:delData
S:         xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:credit
S:           description="AGP Credit"
S:           lang="en">-5.00</fee:credit>
S:         <fee:balance>1005.00</fee:balance>
S:       </fee:delData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>
```

5.2.3. EPP <renew> Command

This extension adds elements to both the EPP <renew> command and response, when the extension is included in the <login> command service extensions.

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

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

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

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

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

Example <renew> command:

```
C: <?xml version="1.0" encoding="utf-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <renew>
C:       <domain:renew
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:           <domain:name>example.com</domain:name>
C:           <domain:curExpDate>2019-04-03</domain:curExpDate>
C:           <domain:period unit="y">5</domain:period>
C:         </domain:renew>
C:       </renew>
C:     <extension>
C:       <fee:renew xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:fee>5.00</fee:fee>
C:       </fee:renew>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```

Example <renew> response:

```

S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <resData>
S:       <domain:renData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:           <domain:name>example.com</domain:name>
S:           <domain:exDate>2024-04-03T22:00:00.0Z</domain:exDate>
S:         </domain:renData>
S:       </resData>
S:       <extension>
S:         <fee:renData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:           <fee:currency>USD</fee:currency>
S:           <fee:fee
S:             refundable="1"
S:             grace-period="P5D">5.00</fee:fee>
S:           <fee:balance>1000.00</fee:balance>
S:         </fee:renData>
S:       </extension>
S:       <trID>
S:         <clTRID>ABC-12345</clTRID>
S:         <svTRID>54322-XYZ</svTRID>
S:       </trID>
S:     </response>
S: </epp>

```

5.2.4. EPP <transfer> Command

This extension adds elements to both the EPP <transfer> command and response, when the value of the "op" attribute of the <transfer> command element is "request", and the extension is included in the <login> command service extensions.

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

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

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

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

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

Example <transfer> command:

```
C: <?xml version="1.0" encoding="utf-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <transfer op="request">
C:       <domain:transfer
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:period unit="y">1</domain:period>
C:         <domain:authInfo>
C:           <domain:pw roid="JD1234-REP">2fooBAR</domain:pw>
C:         </domain:authInfo>
C:       </domain:transfer>
C:     </transfer>
C:     <extension>
C:       <fee:transfer xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:fee>5.00</fee:fee>
C:       </fee:transfer>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```

Example <transfer> response:

```

S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1001">
S:       <msg>Command completed successfully; action pending</msg>
S:     </result>
S:     <resData>
S:       <domain:trnData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>example.com</domain:name>
S:         <domain:trStatus>pending</domain:trStatus>
S:         <domain:reID>ClientX</domain:reID>
S:         <domain:reDate>2019-06-08T22:00:00.0Z</domain:reDate>
S:         <domain:acID>ClientY</domain:acID>
S:         <domain:acDate>2019-06-13T22:00:00.0Z</domain:acDate>
S:         <domain:exDate>2021-09-08T22:00:00.0Z</domain:exDate>
S:       </domain:trnData>
S:     </resData>
S:     <extension>
S:       <fee:trnData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee
S:           refundable="1"
S:           grace-period="P5D">5.00</fee:fee>
S:       </fee:trnData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54322-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>

```

5.2.5. EPP <update> Command

This extension adds elements to both the EPP <update> command and response, when the extension is included in the <login> command service extensions.

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

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

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

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

Example <update> command:

```
C: <?xml version="1.0" encoding="utf-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <update>
C:       <domain:update
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:chg>
C:           <domain:registrant>sh8013</domain:registrant>
C:         </domain:chg>
C:       </domain:update>
C:     </update>
C:     <extension>
C:       <fee:update xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:fee>5.00</fee:fee>
C:       </fee:update>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```


Example <update> response:

```
S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <extension>
S:       <fee:updData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee>5.00</fee:fee>
S:       </fee:updData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>
```

6. Formal Syntax

One schema is presented here that is the EPP Fee Extension schema.

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

6.1. Fee Extension Schema

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

BEGIN

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

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

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

<!-- Child elements found in EPP commands and responses -->
<element name="check" type="fee:checkType" />
<element name="chkData" type="fee:chkDataType" />
<element name="create" type="fee:transformCommandType" />
<element name="creData" type="fee:transformResultType" />
<element name="renew" type="fee:transformCommandType" />
<element name="renData" type="fee:transformResultType" />
<element name="transfer" type="fee:transformCommandType" />
<element name="trnData" type="fee:transformResultType" />
<element name="update" type="fee:transformCommandType" />
<element name="updData" type="fee:transformResultType" />
<element name="delData" type="fee:transformResultType" />

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

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

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

```
</complexType>

<complexType name="objectCDType">
  <sequence>
    <element name="objID" type="fee:objectIdentifierType" />
    <element name="class" type="token" minOccurs="0" />
    <element name="command" type="fee:commandDataType"
      minOccurs="0" maxOccurs="unbounded" />
    <element name="reason" type="fee:reasonType" minOccurs="0" />
  </sequence>
  <attribute name="avail" type="boolean" default="1" />
</complexType>

<!-- general transform (create, renew, update, transfer) command -->
<complexType name="transformCommandType">
  <sequence>
    <element name="currency" type="fee:currencyType"
      minOccurs="0" />
    <element name="fee" type="fee:feeType"
      maxOccurs="unbounded" />
    <element name="credit" type="fee:creditType"
      minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>

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

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

```
</simpleType>

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

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

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

<simpleType name="commandEnum">
  <restriction base="token">
    <enumeration value="create"/>
    <enumeration value="delete"/>
    <enumeration value="renew"/>
    <enumeration value="update"/>
    <enumeration value="transfer"/>
    <enumeration value="restore"/>
    <enumeration value="custom"/>
  </restriction>
</simpleType>
```

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

<simpleType name="negativeDecimal">
  <restriction base="decimal">
    <maxInclusive value="0" />
  </restriction>
</simpleType>

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

<complexType name="creditType">
  <simpleContent>
    <extension base="fee:negativeDecimal">
      <attribute name="description"/>
      <attribute name="lang" type="language" default="en"/>
    </extension>
  </simpleContent>
</complexType>

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

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

```
</schema>  
END
```

7. Security Considerations

The mapping extensions described in this document do not provide any security services beyond those described by EPP [RFC5730], the EPP domain name mapping [RFC5731], and protocol layers used by EPP. The security considerations described in these other specifications apply to this specification as well. This extension passes financial information using the EPP protocol, so confidentiality and integrity protection must be provided by the transport mechanism. All transports compliant with [RFC5730] provide the needed level of confidentiality and integrity protections. The server will only provide information, including financial information, that is relevant to the authenticated client.

8. IANA Considerations

8.1. XML Namespace

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

Registration request for the fee namespace:

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

Registrant Contact: IESG

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

Registration request for the fee schema:

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

Registrant Contact: IESG

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

8.2. EPP Extension Registry

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

Name of Extension: Registry Fee Extension for the Extensible Provisioning Protocol (EPP)

Document status: Standards Track

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

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

9. Implementation Status

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

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

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

9.1. RegistryEngine EPP Service

Organization: CentralNic

Name: RegistryEngine EPP Service

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

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

Coverage: All aspects of the protocol are implemented.

Licensing: Proprietary In-House software

Contact: epp@centralnic.com

URL: <https://www.centralnic.com>

10. Acknowledgements

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

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

11. Change History

11.1. Change from 18 to 19

Added normative reference for XML Schema.

11.2. Change from 18 to 19

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

11.3. Change from 17 to 18

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

11.4. Change from 16 to 17

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

11.5. Change from 15 to 16

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

11.6. Change from 14 to 15

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

11.7. Change from 13 to 14

Moved RFC 7451 reference from Normative to Informative section.

11.8. Change from 12 to 13

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

11.9. Change from 11 to 12

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

11.10. Change from 10 to 11

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

11.11. Change from 09 to 10

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

11.12. Change from 08 to 09

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

11.13. Change from 07 to 08

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

11.14. Change from 06 to 07

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

11.15. Change from 05 to 06

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

11.16. Change from 04 to 05

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

11.17. Change from 03 to 04

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

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

11.18. Change from 02 to 03

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

11.19. Change from 01 to 02

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

11.20. Change from 00 to 01

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

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

Updated to be REGEXT WG document.

12. References

12.1. Normative References

[ISO4217:2015]

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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC3915] Hollenbeck, S., "Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP)", RFC 3915, DOI 10.17487/RFC3915, September 2004, <<https://www.rfc-editor.org/info/rfc3915>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<https://www.rfc-editor.org/info/rfc5730>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <<https://www.rfc-editor.org/info/rfc5731>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8334] Gould, J., Tan, W., and G. Brown, "Launch Phase Mapping for the Extensible Provisioning Protocol (EPP)", RFC 8334, DOI 10.17487/RFC8334, March 2018, <<https://www.rfc-editor.org/info/rfc8334>>.
- [W3C.REC-xmlschema-1-20041028] Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028, October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>>.

12.2. Informative References

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

Authors' Addresses

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

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

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

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

Jothan Frakes

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 29, 2020

J. Gould
M. Pozun
VeriSign, Inc.
February 26, 2020

Login Security Extension for the Extensible Provisioning Protocol (EPP)
draft-ietf-regext-login-security-10

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 <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 29, 2020.

Copyright Notice

Copyright (c) 2020 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. Object Attributes	4
3.1. Event	4
3.2. "[LOGIN-SECURITY]" Password	6
3.3. Dates and Times	7
4. EPP Command Mapping	7
4.1. EPP <login> Command	7
5. Formal Syntax	15
5.1. Login Security Extension Schema	15
6. IANA Considerations	17
6.1. XML Namespace	17
6.2. EPP Extension Registry	18
7. Implementation Status	18
7.1. Verisign EPP SDK	19
8. Security Considerations	19
9. Acknowledgements	20
10. References	20
10.1. Normative References	20
10.2. Informative References	21
10.3. URIs	21
Appendix A. Change History	21
A.1. Change from 00 to 01	21
A.2. Change from 01 to 02	21
A.3. Change from 02 to 03	22
A.4. Change from 03 to REGEXT 00	22
A.5. Change from REGEXT 00 to REGEXT 01	22
A.6. Change from REGEXT 01 to REGEXT 02	22
A.7. Change from REGEXT 02 to REGEXT 03	22
A.8. Change from REGEXT 03 to REGEXT 04	23
A.9. Change from REGEXT 04 to REGEXT 05	23
A.10. Change from REGEXT 05 to REGEXT 06	24
A.11. Change from REGEXT 06 to REGEXT 07	24
A.12. Change from REGEXT 07 to REGEXT 08	24
A.13. Change from REGEXT 08 to REGEXT 09	26
A.14. Change from REGEXT 09 to REGEXT 10	27
Authors' Addresses	27

1. Introduction

This document describes an Extensible Provisioning Protocol (EPP) extension for enhancing the security of the EPP login command in EPP [RFC5730]. EPP [RFC5730] includes a maximum password length of 16 characters that inhibits implementing stronger password security policies with higher entropy. 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 [RFC5730] 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", "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.

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

"whitespace" is defined by the XML schema whiteSpace datatype in [W3C.REC-xmlschema-2-20041028], which only includes the ASCII

whitespace characters #x9 (tab), #xA (linefeed), #xD (carriage return), and #x20 (space).

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. A newer version of the extension is expected to use an XML namespace with a higher version number than the prior versions.

Servers SHOULD (for a temporary migration period up to server policy) 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. The <loginSec:event> element is contained in a list of one or more elements in the <loginSec:loginSecData> element, so there MAY be multiple events returned that provide 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 includes:

- "password": Identifies a password expiry event, where the password expires in the future or has expired based on the "exDate" date and time. The "exDate" attribute MUST be set with the password expiry date and time.
- "certificate": Identifies a client certificate expiry event, where the client certificate will expire at the "exDate" date and time. The "exDate" attribute MUST be set with the certificate expiry date and time.
- "cipher": Identifies the use of an insecure or deprecated TLS cipher suite. The "name" attribute MUST be set with the name of the cipher suite, which is free-form and is not expected to be parsed and automatically addressed by the client. An example of cipher suite names can be found in the TLS Cipher Suites of the Transport Layer Security (TLS) Parameters IANA Registry [1].
- "tlsProtocol": Identifies the use of an insecure or deprecated TLS protocol. The "name" attribute MUST be set with the name of the TLS protocol, which is free-form and is not expected to be parsed and automatically addressed by the client.
- "newPW": The new password does not meet the server password complexity requirements.
- "stat": Provides a login security statistical warning that MUST set the "name" attribute to the name of the statistic sub-type.
- "custom": Custom event type that MUST set the "name" attribute with the custom event type name.
- "name": Used to define a sub-type when the "type" attribute is not "custom" or the full type name when the "type" attribute is "custom". The "name" attribute MUST be set when the "type" attribute is "stat" or "custom". The possible set of "name" values, by event type, can be discovered / negotiated out of band to EPP or using a separate EPP extension designed to provide server policy information to the client.
- "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 a connection failure or MAY be a login failure. An example is an expired certification that will result in a connection failure or an expired password that may result in a login failure.
- "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, ending when the login command was received. The format of the duration is defined by the duration primitive datatype in section 3.2.6 of [W3C.REC-xmlschema-2-20041028].

"lang": Identifies the negotiated language of the free-form description. The format of the language is defined by the language primitive datatype in section 3.3.3 of [W3C.REC-xmlschema-2-20041028]. The default is "en" (English).

Example login security event for password expiration, where the current date is 2020-03-25:

```
<loginSec:event
  type="password"
  level="warning"
  exDate="2020-04-01T22:00:00.0Z"
  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

When the [RFC5730] <pw> element contains the predefined value of "[LOGIN-SECURITY]", the <loginSec:pw> element overrides the <pw> element, which is a constant value for the server to use the <loginSec:pw> element for the password. Similarly, when the [RFC5730] <newPw> element contains the predefined value of "[LOGIN-SECURITY]", the <loginSec:newPw> element overrides the <newPw> element, 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. The server MUST NOT allow the client to set the password to the value "[LOGIN-SECURITY]".

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 information that identifies the client application software, technology, and operating system used by the server to identify functional or security constraints, current security issues, and potential future functional or security issues for the client. The server may use the information for real-time identification and client notification of security issues, such as keying off of the client application software for executing security rule checks. The server may capture the information to identify future security policy issues, such as deprecating or removing TLS cipher suites or TLS protocols. The <loginSec:userAgent> element MUST contain at least one of the following child elements:

<loginSec:app>: OPTIONAL name of the client application software with version if available, such as the name of the client SDK "EPP SDK 1.0.0". The <loginSec:app> element value can be created by appending the version number to the name of the application software, such as the Augmented Backus-Naur Form (ABNF) grammar [RFC5234] format:

app = name SP version

```
    name = 1*VCHAR
    version = 1*VCHAR
<loginSec:tech>:  OPTIONAL technology used for the client
                  software with version if available, such as "Vendor Java
                  11.0.6".  The <loginSec:tech> element value can be created by
                  including the technology vendor, technology name, and
                  technology version, such as the Augmented Backus-Naur Form
                  (ABNF) grammar [RFC5234] format:

    tech = vendor SP name SP version
    vendor = 1*VCHAR
    name = 1*VCHAR
    version = 1*VCHAR
<loginSec:os>:  OPTIONAL client operating system used with
                version if available, such as "x86_64 Mac OS X 10.15.2".  The
                <loginSec:os> element value can be created by including the
                operating system architecture, operating system name, and
                operating system version, such as the Augmented Backus-Naur
                Form (ABNF) grammar [RFC5234] format:

    os = arch SP name SP version
    arch = 1*VCHAR
    name = 1*VCHAR
    version = 1*VCHAR
<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 set 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 set if the [RFC5730] <newPW> element is
                  set to the "[LOGIN-SECURITY]" value.
```

It is RECOMMENDED that the plain text password in the <loginSec:pw> and <loginSec:newPW> elements use printable ASCII characters #x20 (space) - #x7E (~), with high entropy, such as 128 bits. If non-ASCII characters are supported with the plain text password, then use a standard for passwords with international characters; the OpaqueString PRECIS profile in [RFC8265] is recommended in the absence of other considerations.

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-1.0</extURI>
C:        </svcExtension>
C:      </svcs>
C:    </login>
C:    <extension>
C:      <loginSec:loginSec
C:        xmlns:loginSec=
C:          "urn:ietf:params:xml:ns:epp:loginSec-1.0">
C:        <loginSec:userAgent>
C:          <loginSec:app>EPP SDK 1.0.0</loginSec:app>
C:          <loginSec:tech>Vendor Java 11.0.6</loginSec:tech>
C:          <loginSec:os>x86_64 Mac OS X 10.15.2</loginSec:os>
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-1.0</extURI>
C:        </svcExtension>
C:      </svcs>
C:    </login>
C:    <extension>
C:      <loginSec:loginSec
C:        xmlns:loginSec=
C:          "urn:ietf:params:xml:ns:epp:loginSec-1.0">
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-1.0</extURI>
C:        </svcExtension>
C:      </svcs>
C:    </login>
C:    <extension>
C:      <loginSec:loginSec
C:        xmlns:loginSec=
C:          "urn:ietf:params:xml:ns:epp:loginSec-1.0">
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 when both of the following conditions hold:

Client supports extension: The 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 on 2020-03-25, 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-1.0">
S:        <loginSec:event
S:          type="password"
S:          level="warning"
S:          exDate="2020-04-01T22:00:00.0Z"
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-1.0">
S:        <loginSec:event
S:          type="password"
S:          level="error"
S:          exDate="2020-03-24T22:00:00.0Z">
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-1.0">
S:        <loginSec:event
```

```
S:         type="password"
S:         level="warning"
S:         exDate="2020-04-01T22:00:00.0Z"
S:         lang="en">
S:         Password expiration soon
S:     </loginSec:event>
S:     <loginSec:event
S:         type="certificate"
S:         level="warning"
S:         exDate="2020-04-02T22:00:00.0Z"/>
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

The EPP Login Security Extension schema is presented here.

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

5.1. Login Security Extension Schema

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
  xmlns:loginSec="urn:ietf:params:xml:ns:epp:loginSec-1.0"
  targetNamespace="urn:ietf:params:xml:ns:epp:loginSec-1.0"
  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="loginSec:userAgentType" minOccurs="0" />
      <element name="pw"
        type="loginSec:pwType" minOccurs="0" />
      <element name="newPW"
        type="loginSec:pwType" minOccurs="0" />
    </sequence>
  </complexType>
  <simpleType name="pwType">
    <restriction base="token">
      <minLength value="6" />
    </restriction>
  </simpleType>
  </schema>
END
```

```
</simpleType>
<complexType name="userAgentType">
  <choice>
    <sequence>
      <element name="app"
        type="token" />
      <element name="tech"
        type="token" minOccurs="0" />
      <element name="os"
        type="token" minOccurs="0" />
    </sequence>
    <sequence>
      <element name="tech"
        type="token" />
      <element name="os"
        type="token" minOccurs="0" />
    </sequence>
    <element name="os"
      type="token" />
  </choice>
</complexType>
<!-- Login response extension elements -->
<element name="loginSecData"
  type="loginSec:loginSecDataType" />
<complexType name="loginSecDataType">
  <sequence>
    <element name="event"
      type="loginSec:eventType"
      minOccurs="1" maxOccurs="unbounded" />
  </sequence>
</complexType>
<!-- Security event element -->
<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" />
    </extension>
  </simpleContent>
</complexType>
```

```
        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-1.0
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-1.0
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.

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".

7.1. 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-login-security.

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. Security Considerations

The Security Considerations of [RFC5730] apply in this document, and this document enhances these considerations.

The extension leaves the password (<pw> element) and new password (<newPW> element) minimum length greater than 6 characters and the maximum length up to server 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 [2].

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. The server returning of security events to unauthenticated users needs to

take into account the security/privacy issues of returning information to potential attackers.

The user agent information represents the client system of a system-to-system interface, so the user agent information MUST NOT provide any ability to track individual users or classes of users.

9. Acknowledgements

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

- o Martin Casanova
- o Scott Hollenbeck
- o Barry Leiba
- o Patrick Mevzek
- o Joseph Yee

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>>.
- [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-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

[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>>.

[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>>.

[RFC8265] Saint-Andre, P. and A. Melnikov, "Preparation, Enforcement, and Comparison of Internationalized Strings Representing Usernames and Passwords", RFC 8265, DOI 10.17487/RFC8265, October 2017, <<https://www.rfc-editor.org/info/rfc8265>>.

10.3. URIs

[1] <https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-4>

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

Appendix A. Change History

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

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:ns:epp:loginSec-0.3 based on a request from IANA with draft-ietf-regext-allocation-token.

A.3. Change from 02 to 03

1. 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.

A.4. Change from 03 to REGEXT 00

1. Changed to regext working group draft by changing draft-gould-regext-login-security to draft-ietf-regext-login-security.

A.5. Change from REGEXT 00 to REGEXT 01

1. Changed the <loginSec:userAgent> element to be structured with the <loginSec:app>, <loginSec:tech>, and <loginSec:os> sub-elements. This was based on the feedback from Martin Casanova. This resulted in the need to change the XML namespace from urn:ietf:params:xml:ns:epp:loginSec-0.3 to urn:ietf:params:xml:ns:epp:loginSec-0.4.

A.6. Change from REGEXT 01 to REGEXT 02

1. Updated the Implementation Status section from "TBD" to include the Verisign EPP SDK implementation.

A.7. Change from REGEXT 02 to REGEXT 03

1. Revised the description of the "duration" attribute to clarify that it ends when the login command was received and to clarify the format, based on the feedback from Martin Casanova.
2. Revised the sentence 'Upon a completed login command (success or failed), the extension MUST be included in the response based on the following conditions:' to 'Upon a completed login command (success or failed), the extension MUST be included in the

response based on both of the following conditions:' based on the feedback from Patrick Mevzek.

3. Updates based on the review by Joseph Yee, that include:
 1. Revised the description of the <loginSec:event> "name" attribute read 'Used to define a sub-type when the "type" attribute is not "custom" or the full type name when the "type" attribute is "custom"'. The definition of the "stat" type was updated to 'Provides a login security statistical warning that MUST set the "name" attribute to the name of the statistic.'
 2. Added the following sentence 'The server MUST NOT allow the client to set the password to the value "[LOGIN-SECURITY]".' to address the corner case where the constant is used as the password.
 3. Revised the description of the <loginSec:userAgent> element to read 'The <loginSec:userAgent> element MUST contain at least one of the following child elements:'.
 4. Revised the description of the <loginSec:userAgent> to match the child elements that can be passed, by changing "client software" to "client application software" and change "language" to "technology".
 5. Changed the XML namespace from urn:ietf:params:xml:ns:epp:loginSec-0.4 to urn:ietf:params:xml:ns:epp:loginSec-1.0.

A.8. Change from REGEXT 03 to REGEXT 04

Updates based on the review by Joseph Yee, that include:

1. Update the definition of the "stat" security event type to reference sub-type to match the language for the "name" attribute.
2. Added the sentence 'The "name" attribute MUST be set when the "type" attribute is "stat" or "custom".' to the definition of the "name" attribute for clarity.
3. Update the definition of the "userAgentType" in the XML schema to require at least one sub-element using a <choice> element.

A.9. Change from REGEXT 04 to REGEXT 05

Updates based on the review by Barry Leiba, that include:

1. In section 1.1, updated to use BCP 14 boilerplate and references as defined in RFC 8174.
2. In section 1.1, change "REQUIRED" to "required".
3. Keep the "Migration to Newer Versions of This Extension" section by removing the note for removal to the RFC Editor.

4. In section 3.1, change "MAY be multiple events returned that provides information" to "MAY be multiple events returned that provide information".
5. In section 3.1, change "free form" to "free-form".
6. In section 3.1, change "The enumerated list of "type" values include:" to "The enumerated list of "type" values includes:".
7. In section 3.1, change "Identifies the language of the free-form description if the negotiated language is something other than the default value of "en" (English)." to "Identifies the negotiated language of the free-form description. The default is "en" (English)."
8. In section 3.1, change example description from "Example login security event for a password expiring in a week:" to "Example login security event for password expiration, where the current date is 2018-03-25:".
9. In section 4.1, change "Example EPP response to a successful login command where the password will expire in a week:" to "Example EPP response to a successful login command on 2018-03-25, where the password will expire in a week:".

A.10. Change from REGEXT 05 to REGEXT 06

Updates based on the review by Brian Carpenter, that include:

1. In section 1, change the references to RFC 5730 to use links.
2. In section 2, change "(for a temporary migration period)" to "(for a temporary migration period up to server policy)".

A.11. Change from REGEXT 06 to REGEXT 07

1. Updates based on feedback from Barry Leiba, added recommendations on the characters used for the plain text password. Recommended the use of printable ASCII passwords and if non-ASCII characters are supported, to use a standard for passwords with international characters, such as the OpaqueString PRECIS profile in [RFC8265].
2. Based on the feedback from Carlos Pignataro, added "[[RFC Editor: Please remove this section.]]" to the "Change History" section.

A.12. Change from REGEXT 07 to REGEXT 08

1. Based on feedback from Eric Vyncke during the IESG review, changed [RFC8174] from the informative references into the normative references.
2. Based on feedback from Alissa Cooper during the IESG review, changed the sentence "One schema is presented here that is the EPP Login Security Extension schema." in section 5 to "The EPP Login Security Extension schema is presented here.".
3. Changed "sever policy" to "server policy" in section 8.

4. Updates based on feedback from Roman Danyliw during the IESG review:
 1. Changed "pasword" to "password" in section 1.
 2. In section 3.1, added a reference to section 3.3.3 of [W3C.REC-xmlschema-2-20041028] for the format of the "lang" attribute. Added the corresponding section (3.2.6) for the "duration" attribute.
 3. Added the "XML" prefix for each reference to "schema" in the introduction of section 5.
 4. Added the leading sentence "The Security Considerations of [RFC5730] apply in this document, and this document enhances these considerations." to section 8.
 5. Added the sentence 'The possible set of "name" values, by event type, can be discovered / negotiated out of band to EPP or using a separate EPP extension designed to provide server policy information to the client.' to the description of the "name" attribute.
 6. Added a description of how to create the <loginSec:app>, <loginSec:tech>, and <loginSec:os> values using ABNF.
5. Updates based on feedback from Alexey Melnikov during the IESG review:
 1. Added a description of "whitespace" to section 1.1.
 2. Added a description of the usage of the user agent information in section 4.1.
6. Updates based on feedback from Benjamin Kaduk during the IESG review:
 1. Added "A newer version of the extension is expected to use an XML namespace with a higher version number than the prior versions." to the first paragraph of section 2.
 2. In section 3.1, replace the sentence "There MAY be multiple events returned that provide information for the client to address." with "The <loginSec:event> element is contained in a list of one or more elements in the <loginSec:loginSecData> element, so there MAY be multiple events returned that provide information for the client to address."
 3. In section 3.1, for the "exDate" attribute, replace the sentence "At expiry there MAY be an error to connect or MAY be an error to login." with "At expiry there MAY be a connection failure or MAY be a login failure." and a similar change to the following sentence.
 4. In section 3.1, replace the description of the "cipher" type and the "tlsProtocol" type.

5. In section 3.1, add a sentence that the "exDate" attribute MUST be set for the "password" type and the "certificate" type.
6. Updates the dates by replacing 2018 with 2020.
7. In section 3.2, update the MUST override sentences for the <loginSec:pw> and the <loginSec:newPw> elements.
8. In section 4.1, update "OPTIONAL client user agent" with "OPTIONAL client user agent information" for the description of the <loginSec:userAgent> element.
9. In section 4.1, replace "MUST only be used" to "MUST only be set" for the <loginSec:pw> and <loginSec:newPw> elements.
10. Updated references of "x86_64 Mac OS X 10.11.6" to "x86_64 Mac OS X 10.15.2".
11. In section 4.1, replace "MUST be included in the response based on both of the following conditions" with "MUST be included in the response when both of the following conditions hold".
12. In section 4.1, update the "exDate" for the "password" security event error to be "2020-03-24T22:00:00.0Z" so that it's prior to the date 2020-03-25 reference previously.
13. In section 8, add the sentence "The server returning of security events to unauthenticated users needs to take into account the security/privacy issues of returning information to potential attackers." to the end of the last paragraph.
14. In section 8, change "minimum length beyond 6 characters" to "minimum length greater than 6 characters".
15. In section 8, add the sentence "The user agent information represents the client system of a system-to-system interface, so the user agent information MUST NOT provide any ability to track individual users or classes of users."

A.13. Change from REGEXT 08 to REGEXT 09

1. Based on feedback from Barry Leiba in responding to Benjamin Kaduk's discuss item, changed "It is recommended that the plain text..." to "It is RECOMMENDED that the plain text..." and "If non-ASCII characters are supported with the plain text password, then use a standard for passwords with international characters, such as the OpaqueString PRECIS profile in [RFC8265]." to "If non-ASCII characters are supported with the plain text password, then use a standard for passwords with international characters; the OpaqueString PRECIS profile in [RFC8265] is recommended in the absence of other considerations."

A.14. Change from REGEXT 09 to REGEXT 10

1. Based on feedback from Benjamin Kaduk, added the sentence "EPP [RFC5730] includes a maximum password length of 16 characters that inhibits implementing stronger password security policies with higher entropy." to the Introduction.

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>

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: March 27, 2021

M. Loffredo
M. Martinelli
IIT-CNR/Registro.it
September 23, 2020

Registration Data Access Protocol (RDAP) Partial Response
draft-ietf-regext-rdap-partial-response-16

Abstract

The Registration Data Access Protocol (RDAP) does not include capabilities to request partial responses. Servers will only return full responses that include all of the information that a client is authorized to receive. A partial response capability that limits the amount of information returned, especially in the case of search queries, could bring benefits to both clients and servers. This document describes an 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 March 27, 2021.

Copyright Notice

Copyright (c) 2020 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	3
2.1. Subsetting Metadata	3
2.1.1. RDAP Conformance	4
2.1.2. Representing Subsetting Links	4
3. Dealing with Relationships	5
4. Basic Field Sets	6
5. Negative Answers	7
6. IANA Considerations	8
7. Implementation Status	8
7.1. IIT-CNR/Registro.it	9
7.2. APNIC	9
8. Security Considerations	9
9. References	10
9.1. Normative References	10
9.2. Informative References	11
Appendix A. Approaches to Partial Response Implementation . . .	11
A.1. Specific Issues Raised by RDAP	12
Acknowledgements	13
Change Log	13
Authors' Addresses	15

1. Introduction

The use of partial responses 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 of the fields in each result object is returned. The benefit is obvious: less data transferred over the network means less bandwidth usage, faster server responses, less CPU time spent both on the server and the client, and less memory usage on the client.

Currently, RDAP does not provide a client with any way to request a partial response. Servers can only provide the client with a full response [RFC7483]. Servers cannot limit the amount of information returned in a response based on a client's preferences, and this creates inefficiencies.

The protocol described in this specification extends RDAP search capabilities to enable partial responses through the provisioning of pre-defined sets of fields that clients can submit to an RDAP service

by adding a new query parameter. The service is implemented using the Hypertext Transfer Protocol (HTTP) [RFC7230] and the conventions described in [RFC7480].

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.

2. RDAP Path Segment Specification

The path segment defined in this section is an OPTIONAL extension of search path segments defined in [RFC7482]. This document defines an RDAP query parameter, "fieldSet", whose value is a non-empty string identifying a server-defined set of fields returned in place of the full response. The field sets supported by a server are usually described in out-of-band documents (e.g., RDAP profile) together with other features. Moreover, this document defines in Section 2.1 an in-band mechanism by means of which servers can provide clients with a basic information about the supported field sets.

The following is an example of an RDAP query including the "fieldSet" parameter:

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

This solution can be implemented by RDAP providers with less effort than field selection and is easily requested by clients. The considerations that have led to this solution are described in more detail in Appendix A.

2.1. 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 server-provided links to dynamically discover available actions and access the resources it needs. In this way, the client is not required to have prior knowledge of the service and, consequently, to hard code the URIs of different resources. This allows the server to make URI changes as the API evolves without breaking clients. Definitively, a REST service should be as self-descriptive as possible.

Therefore, servers implementing the query parameter described in this specification SHOULD provide additional information in their responses about the available field sets. Such information is collected in a new JSON data structure named "subsetting_metadata" containing the following properties:

- o "currentFieldSet": "String" (REQUIRED) either the value of the "fieldSet" parameter as specified in the query string, or the field set applied by default;
- o "availableFieldSets": "AvailableFieldSet[]" (OPTIONAL) an array of objects, with each element describing an available field set. The AvailableFieldSet object includes the following members:
 - * "name": "String" (REQUIRED) the field set name;
 - * "default": "Boolean" (REQUIRED) whether the field set is applied by default. An RDAP server MUST define only one default field set;
 - * "description": "String" (OPTIONAL) a human-readable description of the field set;
 - * "links": "Link[]" (OPTIONAL) an array of links as described in [RFC8288] containing the query string that applies the field set (see Section 2.1.2).

2.1.1. RDAP Conformance

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

2.1.2. Representing Subsetting Links

An RDAP server MAY use the "links" array of the "subsetting_metadata" element to provide ready-made references [RFC8288] to the available field sets (Figure 1). The target URI in each link is the reference to an alternative to the current view of results identified by the context URI.

The "value", "rel" and "href" JSON values MUST be specified. All other JSON values are OPTIONAL.

```
{
  "rdapConformance": [
    "rdap_level_0",
    "subsetting"
  ],
  ...
  "subsetting_metadata": {
    "currentFieldSet": "afieldset",
    "availableFieldSets": [
      {
        "name": "anotherfieldset",
        "description": "Contains some fields",
        "default": false,
        "links": [
          {
            "value": "https://example.com/rdap/domains?name=example*.com
                      &fieldSet=afieldset",
            "rel": "alternate",
            "href": "https://example.com/rdap/domains?name=example*.com
                      &fieldSet=anotherfieldset",
            "title": "Result Subset Link",
            "type": "application/rdap+json"
          }
        ]
      },
      ...
    ]
  },
  ...
  "domainSearchResults": [
    ...
  ]
}
```

Figure 1: Example of a "subsetting_metadata" instance

3. Dealing with Relationships

Representation of second level objects within a field set produces additional considerations. Since the representation of the topmost returned objects will vary according to the field set in use, the response may contain no relationships (e.g., for an abbreviated field set) or may contain associated objects as in a normal RDAP query response. Each field set can indicate the format of the additional objects to be returned, in the same manner that the format of the topmost objects is controlled by the field set.

4. Basic Field Sets

This section defines three basic field sets which servers MAY implement to facilitate their interaction with clients:

- o "id": the server provides only the key field: "handle" for entities, "ldhName" for domains and nameservers. If a returned domain or nameserver is an Internationalized Domain Name (IDN) [RFC5890], then the "unicodeName" field MUST additionally be included in the response. This field set could be used when the client wants to obtain a collection of object identifiers (Figure 2);
- o "brief": the field set contains the fields that can be included in a "short" response. This field set could be used when the client is asking for a subset of the full response which provides only basic knowledge of each object;
- o "full": the field set contains all of the information the server can provide for a particular object.

The "objectClassName" field is implicitly included in each of the above field sets. RDAP providers SHOULD include a "links" field indicating the "self" link relationship. RDAP providers MAY also add any property providing service information.

Fields included in the "brief" and "full" field set responses MUST take into account the user's access and authorization levels.

```
{
  "rdapConformance": [
    "rdap_level_0",
    "subsetting"
  ],
  ...
  "domainSearchResults": [
    {
      "objectClassName": "domain",
      "ldhName": "example1.com",
      "links": [
        {
          "value": "https://example.com/rdap/domain/example1.com",
          "rel": "self",
          "href": "https://example.com/rdap/domain/example1.com",
          "type": "application/rdap+json"
        }
      ]
    },
    {
      "objectClassName": "domain",
      "ldhName": "example2.com",
      "links": [
        {
          "value": "https://example.com/rdap/domain/example2.com",
          "rel": "self",
          "href": "https://example.com/rdap/domain/example2.com",
          "type": "application/rdap+json"
        }
      ]
    },
    ...
  ]
}
```

Figure 2: Example of RDAP response according to the "id" field set

5. Negative Answers

Each request including an empty or unsupported "fieldSet" value MUST produce an HTTP 400 (Bad Request) response code. Optionally, the response MAY include additional information regarding the supported field sets in the HTTP entity body (Figure 3).

```
{
  "errorCode": 400,
  "title": "Field set 'unknownfieldset' is not valid",
  "description": [
    "Supported field sets are: 'afieldset', 'anotherfieldset'."
  ]
}
```

Figure 3: Example of RDAP error response due to an invalid field set included in the request

6. IANA Considerations

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

Extension identifier: subsetting
Registry operator: Any
Published specification: This document.
Contact: IETF <iesg@ietf.org>
Intended usage: This extension describes best practice for partial response provisioning.

7. 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 [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".

7.1. IIT-CNR/Registro.it

Responsible Organization: Institute of Informatics and Telematics of the National Research Council (IIT-CNR)/Registro.it
Location: <https://rdap.pubtest.nic.it/>
Description: This implementation includes support for RDAP queries using data from .it public test environment.
Level of Maturity: This is an "alpha" test implementation.
Coverage: This implementation includes all of the features described in this specification.
Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

7.2. APNIC

Responsible Organization: Asia-Pacific Network Information Centre
Location: <https://github.com/APNIC-net/rdap-rmp-demo/tree/partial-response>
Description: A proof-of-concept for RDAP mirroring.
Level of Maturity: This is a proof-of-concept implementation.
Coverage: This implementation includes all of the features described in this specification.
Contact Information: Tom Harrison, tomh@apnic.net

8. Security Considerations

A search query typically requires more server resources (such as memory, CPU cycles, and network bandwidth) when compared to a lookup query. This increases the risk of server resource exhaustion and subsequent denial of service. This risk can be mitigated by supporting the return of partial responses combined with other strategies (e.g. restricting search functionality, limiting the rate of search requests, and truncating and paging results).

Support for partial responses gives RDAP operators the ability to implement data access control policies based on the HTTP authentication mechanisms described in [RFC7481]. RDAP operators can vary the information returned in RDAP responses based on a client's access and authorization levels. For example:

- o the list of fields for each set can differ based on the client's access and authorization levels;
- o the set of available field sets could be restricted based on the client's access and authorization levels.

Servers can also define different result limits according to the available field sets, so a more flexible truncation strategy can be implemented. The new query parameter presented in this document

provides RDAP operators with a way to implement a server that reduces inefficiency risks.

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>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<https://www.rfc-editor.org/info/rfc5890>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [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>>.
- [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>>.
- [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>>.
- [HATEOAS] Jedrzejewski, B., "HATEOAS - a simple explanation", 2018, <<https://www.e4developer.com/2018/02/16/hateoas-simple-explanation/>>.
- [REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000, <http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>.

Appendix A. Approaches to Partial Response Implementation

Looking at the implementation experiences of partial response offered by data providers on the web, two approaches are observed:

- o the client explicitly describes the data fields to be returned;
- o the client describes a name identifying a server-defined set of data fields.

The former is more flexible than the latter because clients can specify all the data fields they need. However, 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 that of a JSON object. The presence of arrays and deep nested objects complicate both the syntax definition of the query and, consequently, the processing required on the server side;
- o clients need to recognize the returned data structure to avoid cases when the requested fields are invalid;

- o the request of some fields might not match the client's access and authorization levels. Clients might request unauthorized fields and servers have to define a strategy for responding, such as always returning an error response or returning a response that ignores the unauthorized fields.

A.1. Specific Issues Raised by RDAP

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

- o relevant entity object information is included in a 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.

It is possible to address these issues. For example, the Catnap Query Language [CQL] is a comprehensive expression language that can be used to customize the JSON response of a RESTful web service. Application of CQL to RDAP responses would explicitly identify the output fields that would be acceptable when a few fields are requested but it would become very complicated when processing a larger number of fields. In the following, two CQL expressions for a domain search query are shown (Figure 4). 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 4: Examples of CQL expressions for a domain search query

The field set approach seems to facilitate RDAP interoperability. Servers can define basic field sets which, if known to clients, can

increase the probability of obtaining a valid response. The usage of field sets makes the query string be less complex. Moreover, the definition of pre-defined sets of fields makes it easier to establish result limits.

Finally, considering that there is no real need for RDAP users to have the maximum flexibility in defining all the possible sets of logically connected fields (e.g. users interested in domains usually need to know the status, the creation date, and the expiry date of each domain), the field set approach is preferred.

Acknowledgements

The authors would like to acknowledge Scott Hollenbeck, Tom Harrison, Karl Heinz Wolf, Jasdip Singh, Patrick Mevzek, Benjamin Kaduk, Roman Danyliw, Murray Kucherawy, Erik Kline and Robert Wilton for their contribution to this document.

Change Log

- 00: Initial working group version ported from draft-loffredo-regext-rdap-partial-response-03
- 01: Removed "FOR DISCUSSION" items. Changed the basic field sets from REQUIRED to OPTIONAL. Removed the definition of fields included in "brief" field set. Provided a more detailed description of "subsetting_metadata" structure. Removed some references.
- 02: Added the "Negative Answers" section. Changed "IANA Considerations" section.
- 03: Added the "unicodeName" field in the id fieldSet when a returned domain or nameserver is an IDN. Added RFC5890 to "Normative References" section.
- 04: Recommended the RDAP providers to include a "self" link in any field set other than "full". Updated "Acknowledgements" section.
- 05: Moved "Approaches to Partial Response Implementation" section to the appendix.
- 06: Clarified the use of self links in "Basic Field Sets" section. Added APNIC to the implementations of the "Implementation Status" section.
- 07: Changed "only a subset is returned" to "only a subset of fields in each result object is returned" in the "Introduction" section. Moved the "RDAP Conformance" section up in the document. Updated the "Acknowledgements" section.
- 08: Changed the rdapConformance tag "subsetting_level_0" to "subsetting". Moved [RFC7942] to the "Normative References".
- 09: Corrected the "rdapConformance" content in Figure 2.

- 10: Corrected the JSON content in Figure 1. Clarified the meaning of both context and target URIs in a result subset link defined in Section 2.1.2. Updated the "Acknowledgements" section.
- 11: Minor pre-AD review edits.
- 12: Additional minor pre-AD review edits.
- 13: Edits due to Gen-ART review: in the first paragraph of Section 2 clarified how field sets are defined by a server, in the first sentence of Section 5 replaced SHOULD with MUST. Other minor edits due to AD review.
- 14: Edits due to IESG review:
 - * replaced "fewer data transferred" with "less data transferred" in the "Introduction" section;
 - * in the "Subsetting Metadata" section:
 - + replaced the phrase "collected in a new data structure" with the phrase "collected in a new JSON data structure";
 - + replaced "Members are:" with "The AvailableFieldSet object includes the following members:";
 - + clarified that an RDAP server MUST define only one default field set;
 - * clarified the required members of a Link object in the "Representing Subsetting Links" section;
 - * rewritten the "Dealing with Relationships" section;
 - * in the "Basic Field Sets" section:
 - + replaced the phrase "include a 'self' link in each field set" with the phrase "include a 'links' field indicating the 'self' link relationship";
 - + replaced the phrase "'unicodeName' field MUST be included" with the phrase "'unicodeName' field MUST additionally be included";
 - * in the "Negative Answers" section:
 - + replaced the phrase "the response MAY include additional information regarding the negative answer" with the phrase "the response MAY include additional information regarding the supported field sets";
 - + added a new example;
 - * replaced the phrase "and subsequent denial of service due to abuse" with the phrase "and subsequent denial of service" in "Security Considerations" section;
 - * corrected the [REST] reference in the "Informative References" section;
 - * in "Appendix A":

- + added the phrase " offered by data providers on the web"
after the phrase "Looking at the implementation experiences
of partial response";
 - + replaced the phrase "servers should define a strategy" with
the phrase "servers have to define a strategy";
 - + replaced the term "latter approach" with the term "field set
approach" in the "Appendix A.1" section;
 - * updated the "Acknowledgements" section.
- 15: Minor edit in the "Appendix A.1" section;
- 16: Changed a figure containing only an RDAP query into text. Made
the RDAP queries uniform. Other minor edits.

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: 3 November 2022

M. Loffredo
M. Martinelli
IIT-CNR/Registro.it
2 May 2022

Registration Data Access Protocol (RDAP) Reverse search capabilities
draft-ietf-regext-rdap-reverse-search-11

Abstract

The Registration Data Access Protocol (RDAP) does not include query capabilities for finding the list of domains related to a set of entities matching a given search pattern. In the RDAP context, an entity can be associated with any defined object class. Moreover, other relationships between object classes exist and might be used for providing a reverse search capability. Therefore, a reverse search can be applied to other use cases than the classic domain-entity scenario. This document describes an RDAP extension that allow servers to provide a reverse search feature based on the relationship defined in RDAP between an object class for search and any related object class. The reverse search based on the domain-entity relationship is treated as a particular case.

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 3 November 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	2
1.1. Conventions Used in This Document	4
2. RDAP Path Segment Specification	4
3. RDAP Response Specification	5
4. Reverse Searches Based on Entity Details	5
5. RDAP Conformance	6
6. Implementation Considerations	6
7. Implementation Status	7
7.1. IIT-CNR/Registro.it RDAP Server	7
7.2. IIT-CNR/Registro.it RDAP Client	7
8. IANA Considerations	8
9. Privacy Considerations	8
10. Security Considerations	9
11. Acknowledgements	9
12. References	9
12.1. Normative References	9
12.2. Informative References	10
Appendix A. Paradigms to Enforce Access Control on Reverse Search in RDAP	11
Appendix B. Change Log	12
Authors' Addresses	13

1. Introduction

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

The first objection concerns the potential risks of privacy violation. However, the domain name community is considering a new generation of Registration Directory Services [ICANN-RDS1] [ICANN-RDS2] [ICANN-RA], which provide access to sensitive data under

some permissible purposes and in accordance with appropriate policies for requestor accreditation, authentication and authorization. RDAP's reliance on HTTP means that it can make use of common HTTP-based approaches to authentication and authorization, making it more useful than Whois [RFC3912] in the context of such directory services. Since RDAP consequently permits a reverse search implementation complying with privacy protection principles, this objection is not well-founded.

The other objection to the implementation of a reverse search capability has been connected with its impact on server processing. However, the core RDAP specifications already define search queries, with similar processing requirements, so the distinction on which this objection is based is not clear.

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

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

Currently, RDAP does not provide any means for a client to search for the collection of domains associated with an entity [RFC9082]. 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 an entity can be in relationship with any RDAP object [RFC9083], the availability of a reverse search as largely intended can be common to all the object classes allowed for search. Through a further step of generalization, the meaning of reverse search in the RDAP context can be extended to include any query for retrieving all the objects in relationship with another matching a given search pattern.

The protocol described in this specification aims to extend the RDAP query capabilities to enable reverse search based on the relationships defined in RDAP between an object class for search and a related object class. The reverse search based on the domain-entity relationship is treated as a particular case of such a generic query model.

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.

2. RDAP Path Segment Specification

A generic reverse search path is described by the syntax:

```
{searchable-resource-type}/reverse_search_0/{related-resource-type}?<search-condition>
```

The path segments are defined as in the following:

- * `searchable-resource-type`: it MUST be one of the resource types for search defined in Section 3.2 of [RFC9082] (i.e. "domains", "nameservers" and "entities") or a resource type extension;
- * `related-resource-type`: it MUST be one of the resource types for lookup defined in Section 3.1 of [RFC9082] (i.e. "domain", "nameserver", "entity", "ip" and "autnum") or a resource type extension;
- * `search-condition`: a sequence of "property=search pattern" predicates separated by the ampersand character ('&', US-ASCII value 0x0026). Each "property" represents a JSON object property of the RDAP object class corresponding to "related-resource-type". Objects are only included in the search results if they satisfy all included predicates. This includes predicates that are for the same property: it is necessary in such a case for the related object to match against each of those predicates. Based on their policy, servers MAY restrict the usage of predicates to make a valid search condition, by returning a 400 (Bad Request) response when a problematic request is received.

While `related-resource-type` is defined as having one of a number of different values, the only searches defined in this document are for a `related-resource-type` of "entity". Searches for the other resource types specified in [RFC9082] and resource type extensions may be defined by future documents.

Partial string matching in search patterns is allowed as defined in section 4.1 of [RFC9082].

3. RDAP Response Specification

Reverse search responses use the formats defined in section 8 of [RFC9083], which correspond to the searchable resource types defined in Section 2.

4. Reverse Searches Based on Entity Details

Since in RDAP, an entity can be associated with any other object class, the most common kind of reverse search is one based on an entity's details. Such reverse searches arise from the query model by setting the related resource type to "entity".

By selecting a specific searchable resource type, the resulting reverse search aims at retrieving all the objects (e.g. all the domains) that are related to any entity object matching the search conditions.

This section defines the following reverse search properties servers SHOULD support regardless of the searchable resource type being selected:

Reverse search property: role
RDAP property: `$..entities[*].roles`
Reference: Section 10.2.4 of [RFC9083]

Reverse search property: handle
RDAP property: `$..entities[*].handle`
Reference: Section 5.1 of [RFC9083]

Reverse search property: fn
RDAP property: `$..entities[*].vcardArray[1][?(@[0]=='fn')][3]`
Reference: Section 6.2.1 of [RFC6350]

Reverse search property: email
RDAP property: `$..entities[*].vcardArray[1][?(@[0]=='email')][3]`
Reference: Section 6.4.2 of [RFC6350]

The mapping between the reverse search property and the corresponding RDAP response property is done through the use of a JSONPath expression [I-D.ietf-jsonpath-base].

The presence of a predicate on the reverse search property "role" means that the RDAP response property "roles" must contain at least the specified role.

The last two properties are related to jCard elements [RFC7095], but the field references are to vCard [RFC6350], since jCard is the JSON format for vCard.

Examples of reverse search paths based on the domain-entity relationship are presented in Figure 1.

```
/domains/reverse_search_0/entity?handle=CID-40*&role=technical  
  
/domains/reverse_search_0/entity?fn=Bobby*&role=registrant  
  
/domains/reverse_search_0/entity?handle=RegistrarX&role=registrar
```

Figure 1

Documents that deprecate or restructure RDAP responses such that one or more of the properties listed above becomes invalid MUST either note that the relevant reverse search is no longer available (in the case of deprecation) or describe how to continue supporting the relevant search by way of some new RDAP property (in the case of restructuring).

A server that includes additional fields in its objects in accordance with the extensibility provisions of section 6 of [RFC7480] MAY support the use of those fields in search conditions, in the same way as for the search conditions defined in this section. Support for such fields in the reverse search context MUST be documented in the extension specification.

5. RDAP Conformance

Servers complying with this specification MUST include the value "reverse_search_0" in the rdapConformance property of the help response [RFC9083]. The information needed to register this value in the "RDAP Extensions" registry is described in Section 8.

6. Implementation Considerations

To limit the impact of processing the search predicates, servers are RECOMMENDED to make use of indexes and similar functionality in their underlying data store. In addition, risks with respect to performance degradation or result set generation can be mitigated by adopting practices used for standard searches, e.g. restricting the search functionality, limiting the rate of search requests according to the user's authorization, truncating and paging the results, and returning partial responses.

7. 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 [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".

7.1. IIT-CNR/Registro.it RDAP Server

- * 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. Reverse search is allowed to authenticated users. Registrar users are allowed to perform reverse searches on their own domains and contacts. This is achieved by adding an implicit predicate to the search condition.
- * Level of Maturity: This is an "alpha" test implementation.
- * Coverage: This implementation includes all of the features described in this specification.
- * Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

7.2. IIT-CNR/Registro.it RDAP Client

- * Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it
- * Location: <https://web-rdap.pubtest.nic.it/>

- * Description: This is a Javascript web-based RDAP client. RDAP responses are retrieved from RDAP servers by the browser, parsed into an HTML representation, and displayed in a format improving the user experience. Reverse search is allowed to authenticated users.
- * Level of Maturity: This is an "alpha" test implementation.
- * Coverage: This implementation includes all of the features described in this specification.
- * Contact Information: Francesco Donini, francesco.donini@iit.cnr.it

8. IANA Considerations

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

- * Extension identifier: reverse_search_0
- * Registry operator: Any
- * Published specification: This document.
- * Contact: IETF <iesg@ietf.org>
- * Intended usage: This extension describes reverse search query patterns for RDAP.

9. Privacy Considerations

The search functionality defined in this document may affect the privacy of entities in the registry (and elsewhere) in various ways: see [RFC6973] for a general treatment of privacy in protocol specifications. Registry operators should be aware of the tradeoffs that result from implementation of this functionality.

Many jurisdictions have laws or regulations that restrict the use of "Personal Data", per the definition in [RFC6973]. Given that, registry operators should ascertain whether the regulatory environment in which they operate permits implementation of the functionality defined in this document.

In general, given the sensitivity of this functionality, it SHOULD be accessible to authorized users only, and for specific use cases only.

Since reverse search requests and responses could contain Personally Identifiable Information (PII), reverse search functionality SHOULD be available over HTTPS only.

Providing reverse search in RDAP carries the following threats as described in [RFC6973]:

- * Correlation
- * Disclosure

* Misuse of information

Therefore, RDAP providers are REQUIRED to mitigate the risk of those threats by implementing appropriate measures supported by security services (see Section 10).

10. Security Considerations

Security services required to provide controlled access to the operations specified in this document are described in [RFC7481]. A non-exhaustive list of access control paradigms an RDAP provider can implement is presented in Appendix A.

The specification of the relationship within the reverse search path allows the RDAP servers to implement different authorization policies on a per-relationship basis.

11. Acknowledgements

The authors would like to acknowledge the following individuals for their contributions to this document: Francesco Donini, Scott Hollenbeck, Francisco Arias, Gustavo Lozano, Eduardo Alvarez, Ulrich Wisser and James Gould.

Tom Harrison and Jasdip Singh provided relevant feedback and constant support to the implementation of this proposal. Their contributions have been greatly appreciated.

12. References

12.1. Normative References

- [OIDCC] OpenID Foundation, "OpenID Connect Core incorporating errata set 1", November 2014, <http://openid.net/specs/openid-connect-core-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>>.
- [RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, DOI 10.17487/RFC3912, September 2004, <<https://www.rfc-editor.org/info/rfc3912>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<https://www.rfc-editor.org/info/rfc5730>>.

- [RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <<https://www.rfc-editor.org/info/rfc6350>>.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/info/rfc6973>>.
- [RFC7095] Kewisch, P., "jCard: The JSON Format for vCard", RFC 7095, DOI 10.17487/RFC7095, January 2014, <<https://www.rfc-editor.org/info/rfc7095>>.
- [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>>.
- [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>>.
- [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

- [I-D.ietf-jsonpath-base]
Gössner, S., Normington, G., and C. Bormann, "JSONPath: Query expressions for JSON", Work in Progress, Internet-

Draft, draft-ietf-jsonpath-base-03, 16 January 2022,
<<https://www.ietf.org/archive/id/draft-ietf-jsonpath-base-03.txt>>.

[I-D.ietf-regext-rdap-openid]

Hollenbeck, S., "Federated Authentication for the Registration Data Access Protocol (RDAP) using OpenID Connect", Work in Progress, Internet-Draft, draft-ietf-regext-rdap-openid-08, 8 November 2021,
<<https://www.ietf.org/archive/id/draft-ietf-regext-rdap-openid-08.txt>>.

[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>>.

Appendix A. Paradigms to Enforce Access Control on Reverse Search in RDAP

Access control can be implemented according to different paradigms introducing increasingly stringent rules. The paradigms reported here in the following leverage the capabilities either supported natively or provided as extensions by the OpenID Connect [OIDCC]:

- * Role-Based Access Control: access rights are granted depending on roles. Generally, this is done by grouping users into fixed categories and assigning static grants to each category. A more dynamic approach can be implemented by using the OpenID Connect "scope" claim;
- * Purpose-Based Access Control: access rules are based on the notion of purpose, being the intended use of some data by a user. It can be implemented by tagging a request with the usage purpose and

making the RDAP server check the compliance between the given purpose and the control rules applied to the data to be returned. The purpose can be stated within an out-of-band process by setting the OpenID Connect RDAP-specific "purpose" claim as defined in [I-D.ietf-regext-rdap-openid];

- * Attribute-Based Access Control: rules to manage access rights are evaluated and applied according to specific attributes describing the context within which data are requested. It can be implemented by setting within an out-of-band process additional OpenID Connect claims describing the request context and making the RDAP server check the compliance between the given context and the control rules applied to the data to be returned;
- * Time-Based Access Control: data access is allowed for a limited time only. It can be implemented by assigning the users with temporary credentials linked to access grants whose scope is limited.

Appendix B. Change Log

- 00: Initial working group version ported from draft-loffredo-regext-rdap-reverse-search-04
- 01: Updated "Privacy Considerations" section.
- 02: Revised the text.
- 03: Refactored the query model.
- 04: Keepalive refresh.
- 05: Reorganized "Abstract". Corrected "Conventions Used in This Document" section. Added "RDAP Conformance" section. Changed "IANA Considerations" section. Added references to RFC7095 and RFC8174. Other minor edits.
- 06: Updated "Privacy Considerations", "Security Considerations" and "Acknowledgements" sections. Added some normative and informative references. Added Appendix A.
- 07: Updated normative references.
- 08: Changed "Implementation Status" section. Updated informative references.
- 09: Extended the query model to represent a reverse search based on any relationship between the RDAP object classes. Changed the path segment "role" into a query parameter.
- 10: Updated "Reverse Searches Based on Entity Details" section to consider the use of JSContact format instead of jCard. Added references to JSContact documents.
- 11: Updated the document based on Tom Harrison and James Gould feedback:
 - * Updated section "RDAP Path Segment Specification":
 - Clarified how servers must evaluate a reverse search including predicates that are for the same property.

- Specified the error response servers must return when receiving a wrong reverse search request according to their policy.
- Clarified that searches for the related-resource-type values other than "entity" may be defined in future documents.
- * Reviewed text in section "Reverse Searches Based on Entity Details" about reverse searches based on custom response extensions.
- * Removed references to JSContact documents in section "Reverse Searches Based on Entity Details". Moved the mapping between jCard properties used in the RDAP response and JSContact counterparts to draft-ietf-regext-rdap-jscontact.
- * Added section "RDAP Response Specification".
- * Changed the text to present reverse search as a single extension with multiple features.
- * Changed the definition of searchable-resource-type and related-resource-type to consider also the resource type extensions.
- * Replaced "reverse" with "reverse_search_0" in the generic reverse search path. Updated Figure 1 accordingly.
- * Removed the phrase "but with a special focus on its privacy implications" from both the "Abstract" and the "Introduction". Moved the mapping between jCard properties used in the RDAP response and JSContact counterparts to draft-ietf-regext-rdap-jscontact.
- * Reviewed the text of "Privacy Considerations" section.
- * Text cleaning.

Authors' Addresses

Mario Loffredo
IIT-CNR/Registro.it
Via Moruzzi,1
56124 Pisa
Italy
Email: mario.loffredo@iit.cnr.it
URI: <http://www.iit.cnr.it>

Maurizio Martinelli
IIT-CNR/Registro.it
Via Moruzzi,1
56124 Pisa
Italy
Email: maurizio.martinelli@iit.cnr.it
URI: <http://www.iit.cnr.it>

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: June 3, 2021

M. Loffredo
M. Martinelli
IIT-CNR/Registro.it
S. Hollenbeck
Verisign Labs
November 30, 2020

Registration Data Access Protocol (RDAP) Query Parameters for Result
Sorting and Paging
draft-ietf-regext-rdap-sorting-and-paging-20

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 large responses. 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 June 3, 2021.

Copyright Notice

Copyright (c) 2020 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
2.1.1. RDAP Conformance	6
2.2. "count" Parameter	6
2.3. "sort" Parameter	7
2.3.1. Sorting Properties Declaration	8
2.3.2. Representing Sorting Links	14
2.4. "cursor" Parameter	16
2.4.1. Representing Paging Links	16
3. Negative Answers	17
4. Implementation Considerations	18
5. IANA Considerations	18
6. Implementation Status	19
6.1. IIT-CNR/Registro.it	19
6.2. APNIC	19
7. Security Considerations	20
8. References	20
8.1. Normative References	20
8.2. Informative References	22
Appendix A. JSONPath operators	23
Appendix B. Approaches to Result Pagination	24
B.1. Specific Issues Raised by RDAP	26
Appendix C. Additional Implementation Notes	26
C.1. Sorting	27
C.2. Counting	27
C.3. Paging	27
Acknowledgements	28
Change Log	28
Authors' Addresses	31

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 relevant 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 (e.g. 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 [RFC7231].

However, there are some drawbacks associated with the 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 a cookie, while the information on the resource identification or the search type is included in the query string. Finally, providing custom information through HTTP headers assumes the client to have a prior knowledge of the server implementation which is widely considered a REST design anti-pattern. As a result, this document describes a specification based on the 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 regarding response size management, the search query can potentially generate a large result set that is often truncated according to server limits. Besides, it is not possible to obtain the total number of 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 might traverse the whole result set to find the relevant objects or, due to truncation, might not find them at all.

The specification described in this document 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 [RFC7480].

The implementation of the new parameters is technically feasible, as operators for counting, sorting and paging rows are currently supported by the major relational database management systems.

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.

2. RDAP Query Parameter Specification

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

- o "count": a boolean value that allows a client to request the return of the total number of objects found;
- o "sort": a string value that allows a client to request a specific sort order for the result set;
- o "cursor": a string value representing a pointer to a specific fixed size portion of the 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 server-provided links to dynamically discover available actions and access the resources it needs. In this way, the client is not required to have prior knowledge of the service and, consequently, to hard code the URIs of different resources. This allows the server to make URI changes as the API evolves without breaking clients. Definitively, a REST service should be as self-descriptive as possible.

Therefore, servers implementing the query parameters described in this specification SHOULD provide additional information in their

responses about both the available sorting criteria and possible pagination. Such information is collected in two OPTIONAL response elements named "sorting_metadata" and "paging_metadata".

The "sorting_metadata" element contains the following properties:

- o "currentSort": "String" (OPTIONAL) either the value of "sort" parameter as specified in the query string or the sort applied by default, if any;
- o "availableSorts": "AvailableSort[]" (OPTIONAL) an array of objects, with each element describing an available sort criterion. The AvailableSort object includes the following members:
 - * "property": "String" (REQUIRED) the name that can be used by the client to request the sort criterion;
 - * "default": "Boolean" (REQUIRED) whether the sort criterion is applied by default. An RDAP server MUST define only one default sorting property for each object class;
 - * "jsonPath": "String" (OPTIONAL) the JSONPath expression of the RDAP field corresponding to the property;
 - * "links": "Link[]" (OPTIONAL) an array of links as described in [RFC8288] containing the query string that applies the sort criterion.

At least one of the "currentSort" and "availableSorts" properties MUST be present.

The "paging_metadata" element contains the following fields:

- o "totalCount": "Numeric" (OPTIONAL) a numeric value representing the total number of objects found. It MUST be provided if and only if the query string contains the "count" parameter;
- o "pageSize": "Numeric" (OPTIONAL) a numeric value representing the number of objects that should have been returned in the current page. It MUST be provided if and only if the total number of objects exceeds the page size. This property is redundant for RDAP clients because the page size can be derived from the length of the search results array but, it can be helpful if the end user interacts with the server through a web browser;
- o "pageNumber": "Numeric" (OPTIONAL) a numeric value representing the number of the current page in the result set. It MUST be provided if and only if the total number of objects found exceeds the page size;

- o "links": "Link[]" (OPTIONAL) an array of links as described in [RFC8288] containing the reference to the next page. In this specification, only forward pagination is described because it is all that is necessary to traverse the result set.

2.1.1. RDAP Conformance

Servers returning the "paging_metadata" element in their response MUST include the string literal "paging" in the rdapConformance array. Servers returning the "sorting_metadata" element MUST include the string literal "sorting".

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 inefficient because the user cannot determine if the result set is complete.

The "count" parameter provides additional functionality that allows a client to request information from the server that specifies the total number of objects matching the search pattern.

The following is an example of an RDAP query including the "count" parameter:

```
https://example.com/rdap/domains?name=example*.com&count=true
```

The ABNF syntax is the following:

```
count = "count=" ( trueValue / falseValue )
trueValue = ("true" / "yes" / "1")
falseValue = ("false" / "no" / "0")
```

A trueValue means that the server MUST provide the total number of the objects in the "totalCount" field of the "paging_metadata" element (Figure 1). A falseValue means that the server MUST NOT provide this number.

```
{
  "rdapConformance": [
    "rdap_level_0",
    "paging"
  ],
  ...
  "paging_metadata": {
    "totalCount": 43
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 1: Example of RDAP response with "paging_metadata" element containing the "totalCount" field

2.3. "sort" Parameter

The RDAP protocol does not provide any capability to specify the result set 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 features provided by the RDAP server could help avoid truncation of relevant results.

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=" sortItem *( "," sortItem )
sortItem = property-ref [ ":" ( "a" / "d" ) ]
property-ref = ALPHA *( ALPHA / DIGIT / "_" )
```

"a" means that an ascending sort MUST be applied, "d" means that a descending sort MUST be applied. If the sort direction is absent, an ascending sort MUST be applied.

The following are examples of RDAP queries including the "sort" parameter:

```
https://example.com/rdap/domains?name=example*.com&sort=name
```

```
https://example.com/rdap/
domains?name=example*.com&sort=registrationDate:d
```

```
https://example.com/rdap/  
domains?name=example*.com&sort=lockedDate,name
```

Except for sorting IP addresses and values denoting dates and times, servers MUST implement sorting according to the JSON value type of the RDAP field the sorting property refers to. That is, JSON strings MUST be sorted lexicographically and JSON numbers MUST be sorted numerically. Values denoting dates and times MUST be sorted in chronological order. If IP addresses are represented as JSON strings, they MUST be sorted based on their numeric conversion.

The conversion of an IPv4 address to a number is possible since each dotted format IPv4 address is a representation of a number written in a 256-based manner: 192.168.0.1 means $1 \cdot 256^0 + 0 \cdot 256^1 + 168 \cdot 256^2 + 192 \cdot 256^3 = 3232235521$. Similarly, an IPv6 address can be converted into a number by applying the base 65536. Therefore, the numerical representation of the IPv6 address 2001:0db8:85a3:0:0:8a2e:0370:7334 is 42540766452641154071740215577757643572. Builtin functions and libraries for converting IP addresses into numbers are available in most known programming languages and relational database management systems.

If the "sort" parameter presents an allowed sorting property, it MUST be provided in the "currentSort" field of the "sorting_metadata" element.

2.3.1. Sorting Properties Declaration

In the "sort" parameter ABNF syntax, the element named "property-ref" represents a reference to a property of an RDAP object. Such a reference could be expressed by using a JSONPath expression (named "jsonpath" in the following).

JSONPath is a syntax, originally based on the XML XPath notation [W3C.CR-xpath-31-20161213], which represents a path to select an element (or a set of elements) in a JSON document [RFC8259]. For example, the jsonpath to select the value of the ASCII name inside an RDAP domain lookup response is "\$.ldhName", where \$ identifies the root of the document object model (DOM). Another way to select a value inside a JSON document is the JSON Pointer [RFC6901].

While JSONPath or JSON Pointer are both commonly adopted notations to select any value inside JSON data, neither is particularly concise and easy to use (e.g. "\$.domainSearchResults[*].events[?(@.eventAction='registration')].eventDate" is the jsonpath of the registration date in an RDAP domain search response).

Therefore, this specification defines the "property-ref" element in terms of names identifying 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);
- o multivalued properties (e.g. status, roles, variants);
- o properties representing relationships to other objects (e.g. entities).

On the contrary, properties expressed as values of other properties (e.g. registration date) could be used in such a context.

A list of properties an RDAP server MAY implement is defined. The properties are divided into two groups: object common properties and object specific properties.

- o Object common properties. Object common properties are derived from merging 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. Note that some of these properties are also defined as query path segments. These properties include:

- * Domain: name
- * Nameserver: name, ipv4, ipv6.
- * Entity: fn, handle, org, email, voice, country, cc, city.

The correspondence between these sorting properties and the RDAP object classes is shown in Table 1. Some of the sorting properties defined for the RDAP entity class are related to jCard elements [RFC7095] but, being jCard the JSON format for vCard [RFC6350], the corresponding definitions are included in vCard specification.

An RDAP server MUST NOT use the defined sorting properties with a meaning other than the one described in Table 1.

Object class	Sorting property	RDAP property	RFC 7483	RFC 6350	RFC 8605
Searchable objects	Common properties	eventAction values suffixed by "Date"	4.5		
Domain	name	unicodeName/ ldhName	5.3		
Nameserver	name	unicodeName/ ldhName	5.2		
	ipv4	v4 ipAddress	5.2		
	ipv6	v6 ipAddress	5.2		
Entity	handle	handle	5.1		
	fn	jCard fn	5.1	6.2.1	
	org	jCard org	5.1	6.6.4	
	voice	jCard tel with type="voice"	5.1	6.4.1	
	email	jCard email	5.1	6.4.2	
	country	country name in jCard adr	5.1	6.3.1	
	cc	country code in jCard adr	5.1		3.1
	city	locality in jCard adr	5.1	6.3.1	

Table 1: Sorting properties definition

Regarding the definitions in Table 1, some further considerations are needed to disambiguate some cases:

- o since the response to a search on either domains or nameservers might include both A-labels and U-labels [RFC5890] in general, a consistent sorting policy MUST treat the unicodeName and ldhName as two representations of the same value. The unicodeName value MUST be used while sorting if it is present; when the unicodeName is unavailable, the value of the ldhName MUST be used instead;
- o the jCard "sort-as" parameter MUST be ignored for the sorting capability described in this document;

- o even if a nameserver can have multiple IPv4 and IPv6 addresses, the most common configuration includes one address for each IP version. Therefore, this specification makes the assumption that nameservers have a single IPv4 and/or IPv6 value. When more than one address per IP version is presented, sorting MUST be applied to the first value;
- o multiple events with a given action on an object might be returned. If this occurs, sorting MUST be applied to the most recent event;
- o except for handle values, all the sorting properties defined for entity objects can be multivalued according to the definition of vCard as given in [RFC6350]. When more than one value is presented, sorting MUST be applied to the preferred value identified by the parameter pref="1". If the pref parameter is missing, sorting MUST be applied to the first value.

The "jsonPath" field in the "sorting_metadata" element is used to clarify the RDAP response field the sorting property refers to. The mapping between the sorting properties and the jsonpaths of the RDAP response fields is shown below. The JSONPath operators used herein are described in Appendix A.

- o Searchable objects

registrationDate

```
$.domainSearchResults[*].events[?(@.eventAction=="registration")].eventDate
```

reregistrationDate

```
$.domainSearchResults[*].events[?(@.eventAction=="reregistration")].eventDate
```

lastChangedDate

```
$.domainSearchResults[*].events[?(@.eventAction=="last changed")].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

o Domain

  name

    $.domainSearchResults[*].[unicodeName,ldhName]

o Nameserver

  name

    $.nameserverSearchResults[*].[unicodeName,ldhName]

  ipv4

    $.nameserverSearchResults[*].ipAddresses.v4[0]

  ipv6

    $.nameserverSearchResults[*].ipAddresses.v6[0]

o Entity

  handle

    $.entitySearchResults[*].handle
```



```
fn
    $.entitySearchResults[*].vcardArray[1][?(@[0]=="fn")][3]

org
    $.entitySearchResults[*].vcardArray[1][?(@[0]=="org")][3]

voice
    $.entitySearchResults[*].vcardArray[1][?(@[0]=="tel" &&
    @[1].type=="voice")][3]

email
    $.entitySearchResults[*].vcardArray[1][?(@[0]=="email")][3]

country
    $.entitySearchResults[*].vcardArray[1][?(@[0]=="adr")][3][6]

cc
    $.entitySearchResults[*].vcardArray[1][?(@[0]=="adr")][1].cc

city
    $.entitySearchResults[*].vcardArray[1][?(@[0]=="adr")][3][3]
```

Additional notes on the provided jsonpaths:

- o those related to the event dates are defined only for the "domain" object. To obtain the equivalent jsonpaths for "entity" and "nameserver", the path segment "domainSearchResults" must be replaced with "entitySearchResults" and "nameserverSearchResults" respectively;
- o those related to jCard elements are specified without taking into account the "pref" parameter. Servers that sort those values identified by the pref parameter SHOULD update a jsonpath by adding an appropriate filter. For example, if the email values identified by pref="1" are considered for sorting, the jsonpath of the "email" sorting property should be:
\$.entitySearchResults[*].vcardArray[1][?(@[0]=="email" &&
@[1].pref=="1")][3]

2.3.2. Representing Sorting Links

An RDAP server MAY use the "links" array of the "sorting_metadata" element to provide ready-made references [RFC8288] to the available sort criteria (Figure 2). Each link represents a reference to an alternate view of the results.

The "value", "rel" and "href" JSON values MUST be specified. All other JSON values are OPTIONAL.

```
{
  "rdapConformance": [
    "rdap_level_0",
    "sorting"
  ],
  ...
  "sorting_metadata": {
    "currentSort": "name",
    "availableSorts": [
      {
        "property": "registrationDate",
        "jsonPath": "$.domainSearchResults[*]
          .events[?(@.eventAction==\"registration\")].eventDate",
        "default": false,
        "links": [
          {
            "value": "https://example.com/rdap/domains?name=example*.com
              &sort=name",
            "rel": "alternate",
            "href": "https://example.com/rdap/domains?name=example*.com
              &sort=registrationDate",
            "title": "Result Ascending Sort Link",
            "type": "application/rdap+json"
          },
          {
            "value": "https://example.com/rdap/domains?name=example*.com
              &sort=name",
            "rel": "alternate",
            "href": "https://example.com/rdap/domains?name=example*.com
              &sort=registrationDate:d",
            "title": "Result Descending Sort Link",
            "type": "application/rdap+json"
          }
        ]
      },
      ...
    ]
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 2: Example of a "sorting_metadata" instance to implement result sorting

2.4. "cursor" Parameter

The cursor parameter defined in this specification can be used to encode information about any pagination method. For example, in the case of a simple implementation of the cursor parameter to represent offset pagination information, the cursor value "b2Zmc2V0PTEwMCxsaWlpdD01MA==" is the Base64 encoding of "offset=100,limit=50". Likewise, in a simple implementation to represent keyset pagination information, the cursor value "ZXhhbXBsZS10LmNvbQ==" represents the Base64 encoding of "key=example-N.com" whereby the key value identifies the last row of the current page.

Note that this specification uses a Base64 encoding for cursor obfuscation just for example. RDAP servers are NOT RECOMMENDED to obfuscate a cursor value through a mere Base64 encoding.

This solution lets RDAP providers implement a pagination method according to their needs, a user's access level, and the submitted query. Besides, servers can change the method over time without announcing anything to clients. The considerations that have led to this solution are described in more detail in Appendix B.

The ABNF syntax of the cursor parameter is the following:

```
cursor = "cursor=" 1*( ALPHA / DIGIT / "/" / "=" / "-" / "_" )
```

The following is an example of an RDAP query including the "cursor" parameter:

```
https://example.com/rdap/domains?name=example*.com
&cursor=wJlCDLl16KTWypN7T6vc6nWEmEYe99HjflXY1xmQV-M=
```

2.4.1. Representing Paging Links

An RDAP server SHOULD use the "links" array of the "paging_metadata" element to provide a ready-made reference [RFC8288] to the next page of the result set (Figure 3). Examples of additional "rel" values a server MAY implement are "first", "last", and "prev".

```

{
  "rdapConformance": [
    "rdap_level_0",
    "paging"
  ],
  ...
  "notices": [
    {
      "title": "Search query limits",
      "type": "result set truncated due to excessive load",
      "description": [
        "search results for domains are limited to 50"
      ]
    }
  ],
  "paging_metadata": {
    "totalCount": 73,
    "pageSize": 50,
    "pageNumber": 1,
    "links": [
      {
        "value": "https://example.com/rdap/domains?name=example*.com",
        "rel": "next",
        "href": "https://example.com/rdap/domains?name=example*.com
                &cursor=wJlCDLil6KTWypN7T6vc6nWEmEYe99HjflXYlxmQV-M=",
        "title": "Result Pagination Link",
        "type": "application/rdap+json"
      }
    ]
  },
  "domainSearchResults": [
    ...
  ]
}

```

Figure 3: Example of a "paging_metadata" instance to implement cursor pagination

3. Negative Answers

The constraints for the parameters values are defined by their ABNF syntax. Therefore, each request that includes an invalid value for a parameter SHOULD produce an HTTP 400 (Bad Request) response code. The same response SHOULD be returned in the following cases:

- o if in both single and multi sort the client provides an unsupported value for the "sort" parameter, as well as a value related to an object property not included in the response;

- o if the client submits an invalid value for the "cursor" parameter.

Optionally, the response MAY include additional information regarding either the supported sorting properties or the correct cursor values in the HTTP entity body (Figure 4).

```
{
  "errorCode": 400,
  "title": "Domain sorting property 'unknownproperty' is not valid",
  "description": [
    "Supported domain sorting properties are: 'aproperty', 'anotherproperty'."
  ]
}
```

Figure 4: Example of RDAP error response due to an invalid domain sorting property included in the request

4. Implementation Considerations

Implementation of the new parameters is technically feasible, as operators for counting, sorting and paging are currently supported by the major relational database management systems. Similar operators are completely or partially supported by the most well-known NoSQL databases (e.g. MongoDB, CouchDB, HBase, Cassandra, Hadoop). Additional implementation notes are included in Appendix C.

5. IANA Considerations

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

Extension identifier: paging
Registry operator: Any
Published specification: This document.
Contact: IETF <iesg@ietf.org>
Intended usage: This extension describes best practice for result set paging.

Extension identifier: sorting
Registry operator: Any
Published specification: This document.
Contact: IETF <iesg@ietf.org>
Intended usage: This extension describes best practice for result set sorting.

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 [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 the National Research Council (IIT-CNR)/Registro.it
Location: <https://rdap.pubtest.nic.it/>
Description: This implementation includes support for RDAP queries using data from .it public test environment.
Level of Maturity: This is an "alpha" test implementation.
Coverage: This implementation includes all of the features described in this specification.
Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

6.2. APNIC

Responsible Organization: Asia-Pacific Network Information Centre
Location: <https://github.com/APNIC-net/rdap-rmp-demo/tree/sorting-and-paging>
Description: A proof-of-concept for RDAP mirroring.
Level of Maturity: This is a proof-of-concept implementation.
Coverage: This implementation includes all of the features described in the specification except for nameserver sorting and unicodeName sorting.
Contact Information: Tom Harrison, tomh@apnic.net

7. Security Considerations

Security services for the operations specified in this document are described in [RFC7481].

A search query typically requires more server resources (such as memory, CPU cycles, and network bandwidth) when compared to a lookup query. This increases the risk of server resource exhaustion and subsequent denial of service. This risk can be mitigated by either restricting search functionality or limiting the rate of search requests. Servers can also reduce their load by truncating the results in a response. However, this last security policy can result in a higher inefficiency or risk due to acting on incomplete information if the RDAP server does not provide any functionality to return the truncated results.

The new parameters presented in this document provide RDAP operators with a way to implement a server that reduces inefficiency risks. The "count" parameter gives the client the ability to evaluate the completeness of a response. The "sort" parameter allows the client to obtain the most relevant information at the beginning of the result set. This can reduce the number of unnecessary search requests. Finally, the "cursor" parameter enables the user to scroll the result set by submitting a sequence of sustainable queries within server-acceptable limits.

8. References

8.1. 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>>.
- [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>>.
- [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>>.
- [RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <<https://www.rfc-editor.org/info/rfc6350>>.

- [RFC7095] Kewisch, P., "jCard: The JSON Format for vCard", RFC 7095, DOI 10.17487/RFC7095, January 2014, <<https://www.rfc-editor.org/info/rfc7095>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC8605] Hollenbeck, S. and R. Carney, "vCard Format Extensions: ICANN Extensions for the Registration Data Access Protocol (RDAP)", RFC 8605, DOI 10.17487/RFC8605, May 2019, <<https://www.rfc-editor.org/info/rfc8605>>.

8.2. 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/>>.
- [JSONPATH-COMPARISON] "JSONPath Comparison", 2020, <<https://cburgmer.github.io/json-path-comparison/>>.
- [JSONPATH-WG] "JSON Path (jsonpath)", 2020, <<https://datatracker.ietf.org/wg/jsonpath/documents/>>.
- [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>>.

- [REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000, <http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.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>>.
- [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. JSONPath operators

The jsonpaths used in this document are provided according to the Goessner v.0.8.0 proposal [GOESSNER-JSON-PATH].

Such specification requires that implementations support a set of "basic operators". These operators are used to access the elements of a JSON structure like objects and arrays, and their subelements, respectively, object members and array items. No operations are defined for retrieving parent or sibling elements of a given element. The root element is always referred to as \$ regardless of it being an object or array.

Additionally, the specification permits implementations to support arbitrary script expressions. These can be used to index into an object or array, or to filter elements from an array. While script expression behavior is implementation-defined, most implementations support the basic relational and logical operators, as well as both object member and array item access, sufficiently similar for the purpose of this document. Commonly-supported operators/functions divided into "top-level operators" and "filter operators" are documented in Table 2 and Table 3 respectively.

For more information on implementation interoperability issues, see [JSONPATH-COMPARISON]. As at the time of writing, work is beginning on a standardization effort, too: see [JSONPATH-WG].

Operator	Description
\$	Root element
.<name>	Object member access (dot-notation)
['<name>']	Object member access (bracket-notation)
[<number>]	Array item access
*	All elements within the specified scope
[?(<expression>)]	Filter expression

Table 2: JSONPath Top-Level Operators

Operator	Description
@	Current element being processed
.<name>	Object member access
.[<name1>,<name2>]	Union of object members
[<number>]	Array item access
==	Left is equal to right
!=	Left is not equal to right
<	Left is less than right
<=	Left is less than or equal to right
>	Left is greater than right
>=	Left is greater than or equal to right
&&	Logical conjunction
	Logical disjunction

Table 3: JSONPath Filter Operators

Appendix B. Approaches to Result Pagination

An RDAP query could return a response with hundreds, even thousands, of objects, especially when partial matching is used. For this reason, the cursor parameter addressing result pagination is defined to make responses easier to handle.

Presently, the most popular methods to implement pagination in a REST API include offset pagination and keyset pagination. Neither pagination method requires the server to handle the result set in a storage area across multiple requests since a new result set is generated each time a request is submitted. Therefore, they are preferred to any other method requiring the management of a REST session.

Using limit and offset operators represents the traditionally used method to implement result pagination. Both of them can be used individually:

- o "limit=N": means that the server returns the first N objects of the result set;
- o "offset=N": means that the server skips the first N objects and returns objects starting from position N+1.

When limit and offset are used together, they provide the ability to identify a specific portion of the result set. For example, the pair "offset=100,limit=50" returns the first 50 objects starting from position 101 of the result set.

Though easy to implement, offset pagination also includes drawbacks:

- o when offset has a very high value, scrolling the result set could take some time;
- o it always requires fetching all rows before dropping as many rows as specified by offset;
- o it may return inconsistent pages when data are frequently updated (i.e. real-time data).

Keyset pagination [SEEK] adds a query condition that enables the selection of the only data not yet returned. This method has been taken as the basis for the implementation of a "cursor" parameter [CURSOR] by some REST API providers [CURSOR-API1] [CURSOR-API2]. The cursor is an opaque to client URL-safe string representing a logical pointer to the first result of the next page.

Nevertheless, even keyset pagination can be troublesome:

- o it needs at least one key field;
- o it does not allow sorting simply by any field because the sorting criterion must contain a key;
- o it works best with full composite values support by data base management systems (i.e. $[x,y]>[a,b]$), emulation is possible but inelegant and less efficient;
- o it does not allow direct navigation to arbitrary pages because the result set must be scrolled in sequential order starting from the initial page;

- o implementing bi-directional navigation is tedious because all comparison and sort operations have to be reversed.

B.1. Specific Issues Raised by RDAP

Some additional considerations can be made in the RDAP context:

- o an RDAP object is a conceptual aggregation of information generally collected from more than one data structure (e.g. table) and this makes it even harder to implement keyset pagination, a task that is already quite difficult. For example, the entity object can include information from different data structures (registrars, registrants, contacts, resellers), each one with its key field mapping the RDAP entity handle;
- o depending on the number of 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 (i.e. arrays) and, therefore, data selection might therefore be a time consuming process. This situation occurs even though the selection is supported by indexes;
- o depending on the access levels defined by each RDAP operator, the increase in complexity and the decrease in flexibility of keyset pagination in comparison to offset pagination could be considered impractical.

Ultimately, both pagination methods have benefits and drawbacks.

Appendix C. Additional Implementation Notes

This section contains an overview of the main choices made during the implementation of the capabilities defined above in the RDAP public test server of Registro.it at the Institute of Informatics and Telematics of the National Research Council (IIT-CNR). The content of this section can represent a guidance for those implementers who plan to provide RDAP users with those capabilities. The RDAP public test server can be accessed at <https://rdap.pubtest.nic.it/>. Further documentation about the server features is available at <https://rdap.pubtest.nic.it/doc/README.html>.

C.1. Sorting

If no sort criterion is specified in the query string, the results are sorted by a default property: "name" for domains and nameservers, "handle" for entities. The server supports multiple property sorting but the "sorting_metadata" object includes only the links to alternative result set views sorted by a single property just to show the list of sorting properties allowed for each searchable object. The server supports all the object specific sorting properties described in the specification except for nameserver sorting based on unicodeName, that is, the "name" sorting property is mapped onto the "ldhName" response field. Regarding the object common properties, the sorting by registrationDate, expirationDate, lastChangedDate and transferDate is supported.

C.2. Counting

The counting operation is implemented through a separate query. Some relational database management systems support custom operators to get the total count together with the rows, but the resulting query can be considerably more expensive than that performed without the total count. Therefore, as "totalCount" is an optional response information, fetching always the total number of rows has been considered an inefficient solution. Furthermore, to avoid the processing of unnecessary queries, when the "count" parameter is included in the submitted query, it is not also repeated in the query strings of the "links" array provided in both "paging_metadata" and "sorting_metadata" objects.

C.3. Paging

The server implements the cursor pagination through the keyset pagination when sorting by a unique property is requested or the default sort is applied, through offset pagination otherwise. As most of the relational database management systems don't support the comparison of full composite values natively, the implementation of full keyset pagination seem to be troublesome so, at least initially, a selective applicability of keyset pagination is advisable. Moreover, the "cursor" value encodes not only information about pagination but also about the search pattern and the other query parameters in order to check the consistency of the entire query string. If the "cursor" value is inconsistent with the rest of the query string, the server returns an error response.

Acknowledgements

The authors would like to acknowledge Brian Mountford, Tom Harrison, Karl Heinz Wolf, Jasdip Singh, Erik Kline, Eric Vyncke, Benjamin Kaduk and Roman Danyliw for their contribution to the development of this document.

Change Log

- 00: Initial working group version ported from draft-loffredo-regext-rdap-sorting-and-paging-05
- 01: Removed both "offset" and "nextOffset" to keep "paging_metadata" consistent between the pagination methods. Renamed "Considerations about Paging Implementation" section in "cursor" Parameter". Removed "FOR DISCUSSION" items. Provided a more detailed description of both "sorting_metadata" and "paging_metadata" objects.
- 02: Removed both "offset" and "limit" parameters. Added ABNF syntax of the cursor parameter. Rearranged the layout of some sections. Removed some items from "Informative References" section. Changed "IANA Considerations" section.
- 03: Added "cc" to the list of sorting properties in "Sorting Properties Declaration" section. Added RFC8605 to the list of "Informative References".
- 04: Replaced "ldhName" with "name" in the "Sorting Properties Declaration" section. Clarified the sorting logic for the JSON value types and the sorting policy for multivalued fields.
- 05: Clarified the logic of sorting on IP addresses. Clarified the mapping between the sorting properties and the RDAP response fields. Updated "Acknowledgements" section.
- 06: Renamed "pageCount" to "pageSize" and added "pageNumber" in the "paging_metadata" object.
- 07: Added "Paging Responses to POST Requests" section.
- 08: Added "Approaches to Result Pagination" section to appendix. Added the case of requesting a sort on a property not included in the response to the errors listed in the "Negative Answers" section.
- 09: Updated the "Implementation Status" section to include APNIC implementation. Moved the "RDAP Conformance" section up in the document. Removed the "Paging Responses to POST Requests" section. Updated the "Acknowledgements" section. Removed unused references. In the "Sorting Properties Declaration" section:
 - * clarified the logic of sorting on events;
 - * corrected the jsonpath of the "lastChanged" sorting property;
 - * provided a JSONPath example taking into account the vCard "pref" parameter.

- 10: Corrected the jsonpaths of both "fn" and "org" sorting properties in Table 2. Corrected JSON content in Figure 2. Moved [W3C.CR-xpath-31-20161213] and [RFC7942] to the "Normative References". Changed the rdapConformance tags "sorting_level_0" and "paging_level_0" to "sorting" and "paging" respectively.
- 11: Added the "JSONPath operators" section to appendix.
- 12: Changed the content of "JSONPath operators" section.
- 13: Minor pre-AD review edits.
- 14: Additional minor pre-AD review edits.
- 15: In section "'sort' Parameter" added a paragraph providing conversions of IP addresses into their numerical representations. In section "Sorting Properties Declaration" rearranged Table 2 in a list to make the content more readable. Other minor edits due to AD review.
- 16: In section "Introduction" replaced "... large result set that could be truncated ..." with "... large result set that is often truncated ..." as suggested by Gen-ART reviewer. Added Appendix C.
- 17: Edits made:
 - * in the "Sorting and Paging Metadata" section:
 - + replaced "Members are:" with "The AvailableSort object includes the following members:";
 - + clarified that an RDAP server MUST define only one default sorting property for each object class;
 - * in the "Negative Answers" section:
 - + replaced the phrase "the response MAY include additional information regarding the negative answer" with the phrase "the response MAY include additional information regarding either the supported sorting properties or the correct cursor value";
 - + added a new example;
 - * clarified the required members of a Link object in the "Representing Sorting Links" section;
 - * corrected the [REST] reference in the "Informative References" section;
 - * replaced the phrase "and subsequent denial of service due to abuse" with the phrase "and subsequent denial of service" in "Security Considerations" section.
- 18: Edits made:
 - * in the "Introduction" section:
 - + revised the reasons for using query parameters instead of HTTP headers;
 - * in the "Sorting and Paging Metadata" section:

- + replaced the phrase "number of objects returned in the current page" with the phrase "number of objects that should have been returned in the current page" in the definition of the "pageSize" field;
 - * in the "'sort' Parameter" section:
 - + clarified the sorting logic for values denoting dates and times;
 - + replaced the IPv6 address "2001:0db8:85a3:0000:0000:8a2e:0370:7334" with "2001:0db8:85a3:0:0:8a2e:0370:7334";
 - * in the "Sorting Properties Declaration" section:
 - + replaced the sorting properties "ipV4" and "ipV6" with "ipv4" and "ipv6";
 - + replaced the sentence "Therefore, the assumption of having a single IPv4 and/or IPv6 value for a nameserver cannot be considered too stringent." with the sentence "Therefore, this specification makes the assumption that nameservers have a single IPv4 and/or IPv6 value."
 - + clarified that the sorting properties MUST NOT be used with a with a meaning other than the one described this document;
 - + specified that JSONPath operators used in this section are those defined in "Appendix A";
 - * in the "'cursor' Parameter" section:
 - + corrected the Base64 encoding of "offset=100,limit=50";
 - + clarified that RDAP servers are NOT RECOMMENDED to obfuscate a cursor value through a mere Base64 encoding;
 - * changed last sentence of second paragraph of the "Security Considerations" section;
 - * updated the "Acknowledgements" section;
 - * in "Appendix A":
 - + changed introductory paragraph;
 - + replaced "opaque URL-safe string" with "opaque to client URL-safe string";
 - * added JSONPath union operator in Table 2 of "Appendix B"
 - * changed the explanation of offset and limit operators in "Appendix B";
 - * converted the figures containing only RDAP queries into texts;
 - * changed the wildcard prefixed patterns into wildcard suffixed in all the RDAP queries;
 - * cleaned the text.
- 19: Replaced the words "encryption/encrypt" with "obfuscation/obfuscate" in the "'cursor' Parameter" section.

- 20: Added a final paragraph to Appendix A to reference the comparison between JSONPath operators and IETF JSONPath WG web site.

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/>

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 22, 2022

G. Lozano
ICANN
Feb 18, 2022

ICANN TMCH functional specifications
draft-ietf-regext-tmch-func-spec-14

Abstract

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

Status of This Memo

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

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

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

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

Registries	24
5.3.3. TMBD Trademark Claims Services for Registries	25
5.3.3.1. Domain Name Label (DNL) List	25
5.3.3.2. Notice of Registered Domain Names (NORN)	26
5.3.4. Trademark Claims Domain Name registration by Registrars	26
5.3.5. TMBD Trademark Claims Services for Registrars	28
5.3.5.1. Claims Notice Information Service (CNIS)	28
5.4. Qualified Launch Program (QLP) Period	28
5.4.1. Domain Registration	28
5.4.2. TMBD QLP Services for Registries	31
5.4.2.1. Sunrise List (SURL)	31
6. Data Format Descriptions	31
6.1. Domain Name Label (DNL) List	32
6.2. SMD Revocation List	33
6.3. List of Registered Domain Names (LORDN) file	35
6.3.1. LORDN Log file	39
6.3.1.1. LORDN Log Result Codes	42
6.4. Signed Mark Data (SMD) File	45
6.5. Trademark Claims Notice (TCN)	46
6.6. Sunrise List (SURL)	53
7. Formal Syntax	54
7.1. Trademark Claims Notice (TCN)	54
8. Acknowledgements	57
9. Change History	57
9.1. Version 04	57
9.2. Version 05	57
9.3. Version 06	57
9.4. Version 07	58
9.5. Version 08	58
9.6. Version 09	58
9.7. Version 10	58
9.8. Version 11	58
9.9. Version 12	58
9.10. Version 13	58
9.11. Version 14	58
10. IANA Considerations	58
11. Security Considerations	59
12. Privacy Considerations	60
13. References	60
13.1. Normative References	60
13.2. Informative References	61
Author's Address	63

1. Introduction

Domain Name Registries (DNRs) may operate in special modes for certain periods of time enabling trademark holders to protect their rights during the introduction of a Top Level Domain (TLD).

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

- o Sunrise Period, the Sunrise Period allows trademark holders an advance opportunity to register domain names corresponding to their marks before names are generally available to the public.
- o Trademark Claims Period, the Trademark Claims Period follows the Sunrise Period and runs for at least the first 90 days of an initial operating period of general registration. During the Trademark Claims Period, anyone attempting to register a domain name matching a mark that is recorded in the ICANN Trademark Clearinghouse (TMCH) will receive a notification displaying the relevant mark information.

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

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

2. Terminology

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

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

"tmNotice-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:tmNotice-1.0". The XML namespace prefix "tmNotice" is used, but implementations MUST NOT depend on it and

instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

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

3. Glossary

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

- o Backend Registry Operator: Entity that manages (a part of) the technical infrastructure for a Registry Operator. The Registry Operator may also be the Backend Registry Operator.
- o CA: Certificate Authority, see [RFC5280].
- o CNIS, Claims Notice Information Service: This service provides Trademark Claims Notices (TCN) to Registrars.
- o CRC32, Cyclic Redundancy Check: algorithm used in the ISO 3309 standard and in section 8.1.1.6.2 of ITU-T recommendation V.42.
- o CRL: Certificate Revocation List, see [RFC5280].
- o CSV: Comma-Separated Values, see [RFC4180]
- o Date and time, datetime: Date and time are specified following the standard "Date and Time on the Internet specification", see [RFC3339].
- o DN, Domain Name, domain name: see definition of Domain name in [RFC8499].
- o DNROID, DN Repository Object Identifier: an identifier assigned by the Registry to each DN object that unequivocally identifies said DN object. For example, if a new DN object is created for a name that existed in the past, the DN objects will have different DNROIDs.
- o DNL, Domain Name Label, the DNL is an A-label or NR-LDH label (see [RFC5890]).
- o DNL List: A list of DNLs that are covered by a PRM.
- o DNS: Domain Name System, see [RFC8499].

- o Effective allocation: A DN is considered effectively allocated when the DN object for the DN has been created in the SRS of the Registry and has been assigned to the effective user. A DN object in status "pendingCreate" or any other status that precedes the first time a DN is assigned to an end-user is not considered an effective allocation. A DN object created internally by the Registry for subsequent delegation to another Registrant is not considered an effective allocation.
- o EPP: The Extensible Provisioning Protocol, see definition of EPP in [RFC8499].
- o FQDN: Fully-Qualified Domain Name, see definition of FQDN in [RFC8499].
- o HTTP: Hypertext Transfer Protocol, see [RFC7230] and [RFC7231].
- o HTTPS: HTTP over TLS (Transport Layer Security), [RFC2818].
- o IDN: Internationalized Domain Name, see definition of IDN in [RFC8499].
- o Lookup Key: A random string of up to 51 chars from the set [a-zA-Z0-9/] to be used as the lookup key by Registrars to obtain the TCN using the CNIS. Lookup Keys are unique and are related to one DNL only.
- o LORDN, List of Registered Domain Names: This is the list of effectively allocated DNS matching a DNL of a PRM. Registries will upload this list to the TMDB (during the NORDN process).
- o Matching Rules: Some trademarks entitled to inclusion in the TMDB include characters that are impermissible in the domain name system (DNS) as a DNL. The TMV changes (using the ICANN TMCH Matching Rules [MatchingRules]) certain DNS-impermissible characters in a trademark into DNS-permissible equivalent characters
- o NORDN, Notification of Registered Domain Names: The process by which Registries upload their recent LORDN to the TMDB.
- o PGP: Pretty Good Privacy, see [RFC4880]
- o PKI: Public Key Infrastructure, see [RFC5280].
- o PRM, Pre-registered mark: Mark that has been pre-registered with the ICANN TMCH.

- o QLP Period, Qualified Launch Program Period: During this optional period, a special process applies to DNS matching the Sunrise List (SURL) and/or the DNL List, to ensure that TMHs are informed of a DN matching their PRM.
- o Registrant: see definition of Registrant in [RFC8499].
- o Registrar, Domain Name Registrar: see definition of Registrar in [RFC8499].
- o Registry, Domain Name Registry, Registry Operator: see definition of Registry in [RFC8499]. A Registry Operator is the contracting party with ICANN for the TLD.
- o SMD, Signed Mark Data: A cryptographically signed token issued by the TMV to the TMH to be used in the Sunrise Period to apply for a DN that matches a DNL of a PRM; see also [RFC7848]. An SMD generated by an ICANN-approved trademark validator (TMV) contains both the signed token and the TMV's PKIX certificate.
- o SMD File: A file containing the SMD (see above) and some human readable data. The latter is usually ignored in the processing of the SMD File. See also Section 6.4.
- o SMD Revocation List: The SMD Revocation List is used by Registries (and optionally by Registrars) during the Sunrise Period to ensure that an SMD is still valid (i.e. not revoked). The SMD Revocation List has a similar function as CRLs used in PKI.
- o SRS: Shared Registration System, see also [ICANN-GTLD-AGB-20120604].
- o SURL, Sunrise List: The list of DNLs that are covered by a PRM and eligible for Sunrise.
- o Sunrise Period: During this period DNS matching a DNL of a PRM can be exclusively obtained by the respective TMHs. For DNS matching a PRM, a special process applies to ensure that TMHs are informed on the effective allocation of a DN matching their PRM.
- o TLD: Top-Level Domain Name, see definition of TLD in [RFC8499].
- o ICANN TMCH: a central repository for information to be authenticated, stored, and disseminated, pertaining to the rights of TMHs. The ICANN TMCH is split into two functions TMV and TMDB (see below). There could be several entities performing the TMV function, but only one entity performing the TMDB function.

- o ICANN TMCH-CA: The Certificate Authority (CA) for the ICANN TMCH. This CA is operated by ICANN. The public key for this CA is the trust anchor used to validate the identity of each TMV.
- o TMDB, Trademark Clearinghouse Database: Serves as a database of the ICANN TMCH to provide information to the gTLD Registries and Registrars to support Sunrise or Trademark Claims services. There is only one TMDB in the ICANN TMCH that concentrates the information about the "verified" Trademark records from the TMVs.
- o TMH, Trademark Holder: The person or organization owning rights on a mark.
- o TMV, Trademark Validator, Trademark validation organization: An entity authorized by ICANN to authenticate and validate registrations in the TMDB ensuring the marks qualify as registered or are court-validated marks or marks that are protected by statute or treaty. This entity would also be asked to ensure that proof of use of marks is provided, which can be demonstrated by furnishing a signed declaration and one specimen of current use.
- o Trademark, mark: Marks are used to claim exclusive properties of products or services. A mark is typically a name, word, phrase, logo, symbol, design, image, or a combination of these elements. For the scope of this document only textual marks are relevant.
- o Trademark Claims, Claims: Provides information to enhance the understanding of the Trademark rights being claimed by the TMH.
- o TCN, Trademark Claims Notice, Claims Notice, Trademark Notice: A Trademark Claims Notice consist of one or more Trademark Claims and are provided to prospective Registrants of DNs.
- o TCNID, Trademark Claims Notice Identifier: An element of the Trademark Claims Notice (see above), identifying said TCN. The Trademark Claims Notice Identifier is specified in the element <tmNotice:id>.
- o Trademark Claims Period: During this period, a special process applies to DNs matching the DNL List, to ensure that TMHs are informed of a DN matching their PRM. For DNs matching the DNL List, Registrars show a TCN to prospective Registrants that has to be acknowledged before effective allocation of the DN.
- o UTC: Coordinated Universal Time, as maintained by the Bureau International des Poids et Mesures (BIPM); see also [RFC3339].

4. Architecture

4.1. Sunrise Period

Architecture of the Sunrise Period

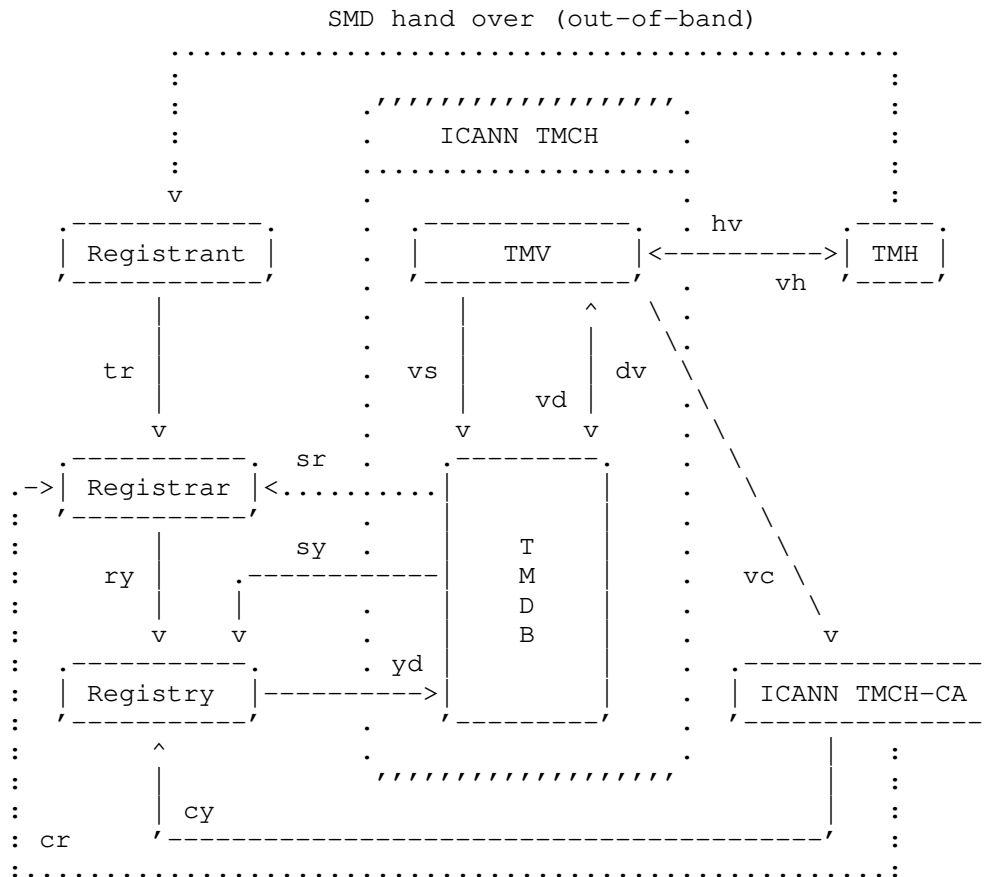


Figure 1

Figure 1 depicts the architecture of the Sunrise Period, including all the actors and interfaces.

4.2. Trademark Claims Period

Architecture of the Trademark Claims Period

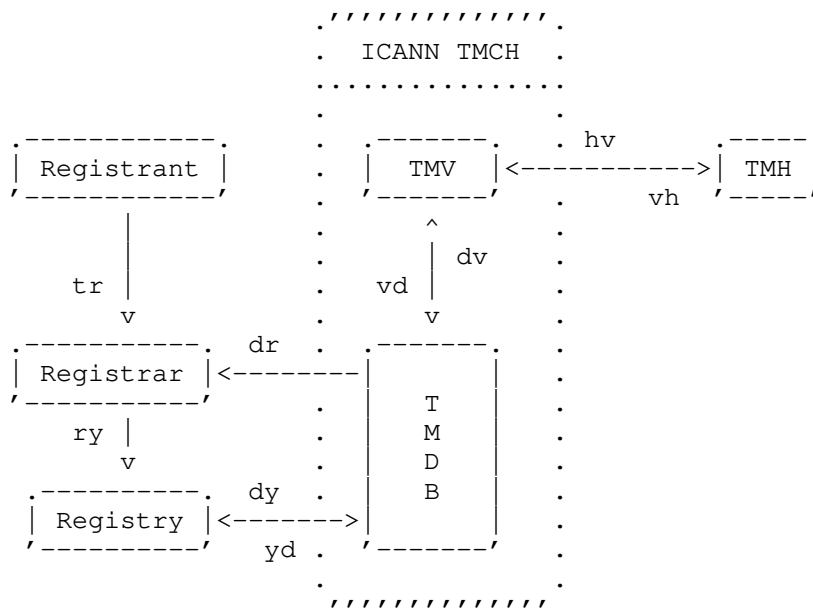


Figure 2

Figure 2 depicts the architecture of the Trademark Claims Period, including all the actors and interfaces.

4.3. Interfaces

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

4.3.1. hv

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

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

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

4.3.2. vd

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

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

4.3.3. dy

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

Not relevant during the Sunrise Period.

4.3.4. tr

The Registrant communicates with the Registrar via the tr interface.

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

4.3.5. ry

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

4.3.6. dr

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

Not relevant during the Sunrise Period.

4.3.7. yd

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

During the Trademark Claims Period, the Registry notifies the TMDB via the yd interface of all DNS effectively allocated that matched an

entry in the Registry previously downloaded DNL List during the creation of the DN.

The protocol used on the yd interface is HTTPS.

4.3.8. dv

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

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

4.3.9. vh

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

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

4.3.10. vs

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

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

Not relevant during the Trademark Claims Period.

4.3.11. sy

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

Not relevant during the Trademark Claims Period.

4.3.12. sr

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

Not relevant during the Trademark Claims Period.

4.3.13. vc

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

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

Not relevant during the Trademark Claims Period.

4.3.14. cy

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

Not relevant during the Trademark Claims Period.

4.3.15. cr

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

Not relevant during the Trademark Claims Period.

5. Process Descriptions

5.1. Bootstrapping

5.1.1. Bootstrapping for Registries

5.1.1.1. Credentials

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

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

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

5.1.1.2. IP Addresses for Access Control

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

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

This access restriction MAY be applied by the TMDB in addition to HTTP Basic access authentication (see [RFC7235]). For credentials to be used, see Section 5.1.1.1.

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

5.1.1.3. ICANN TMCH Trust Anchor

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

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

5.1.1.4. TMDB PGP Key

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

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

5.1.2. Bootstrapping for Registrars

5.1.2.1. Credentials

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

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

5.1.2.2. IP Addresses for Access Control

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

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

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

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

5.1.2.3. ICANN TMCH Trust Anchor

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

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

5.1.2.4. TMDB PGP Key

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

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

5.2. Sunrise Period

5.2.1. Domain Name registration

Domain Name registration during the Sunrise Period

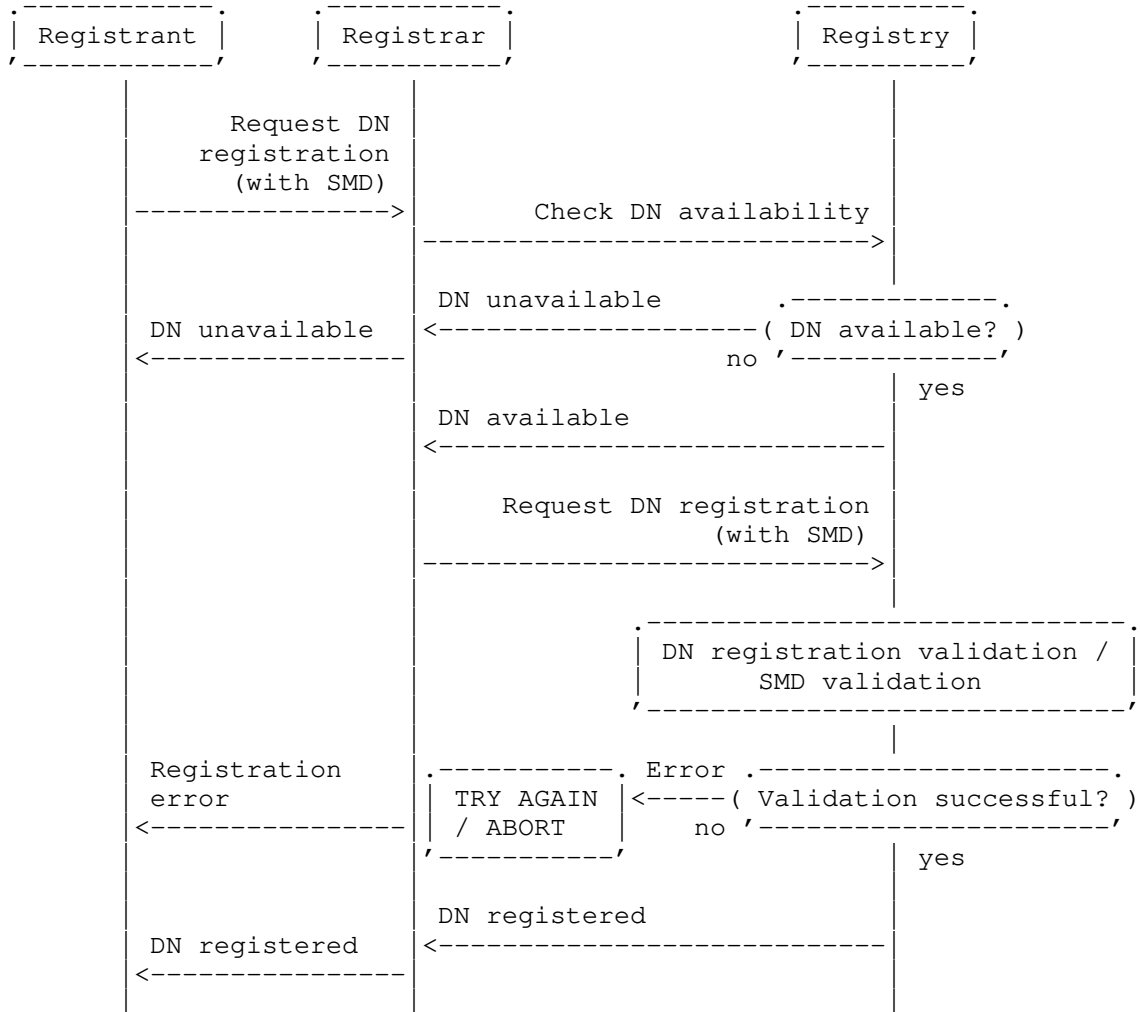


Figure 3

Figure 3 represents a synchronous DN registration workflow (usually called first come first served).

5.2.2. Sunrise Domain Name registration by Registries

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

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

Performing the minimum set of checks Registries MUST verify that:

1. An SMD has been received from the Registrar along with the DN registration.
2. The certificate of the TMV has been correctly signed by the ICANN TMCH-CA. (The certificate of the TMV is contained within the SMD.)
3. The datetime when the validation is done is within the validity period of the TMV certificate.
4. The certificate of the TMV is not listed in the TMV CRL file specified in the CRL distribution point of the TMV certificate.
5. The signature of the SMD (signed with the TMV certificate) is valid.
6. The datetime when the validation is done is within the validity period of the SMD based on <smd:notBefore> and <smd:notAfter> elements.
7. The SMD has not been revoked, i.e., is not contained in the SMD Revocation List.
8. The leftmost DNL of the DN being effectively allocated matches one of the labels (<mark:label>) elements in the SMD. For example, if the DN "xn--mgbachtv.xn--mgbh0fb" is being effectively allocated, the leftmost DNL would be "xn--mgbachtv".

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

5.2.3. TMDB Sunrise Services for Registries

5.2.3.1. SMD Revocation List

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

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

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

Update of the SMD Revocation List

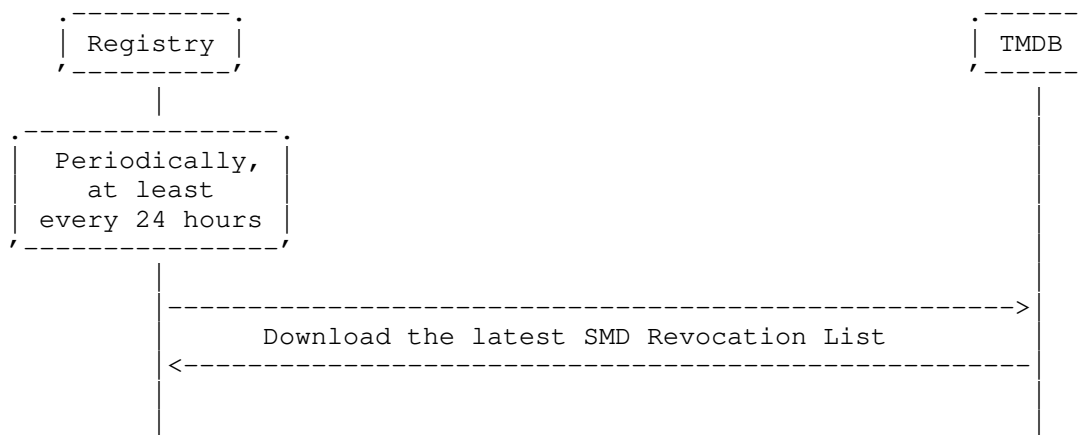


Figure 4

Figure 4 depicts the process of downloading the latest SMD Revocation List initiated by the Registry.

5.2.3.2. TMV Certificate Revocation List (CRL)

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

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

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

Update of the TMV CRL file

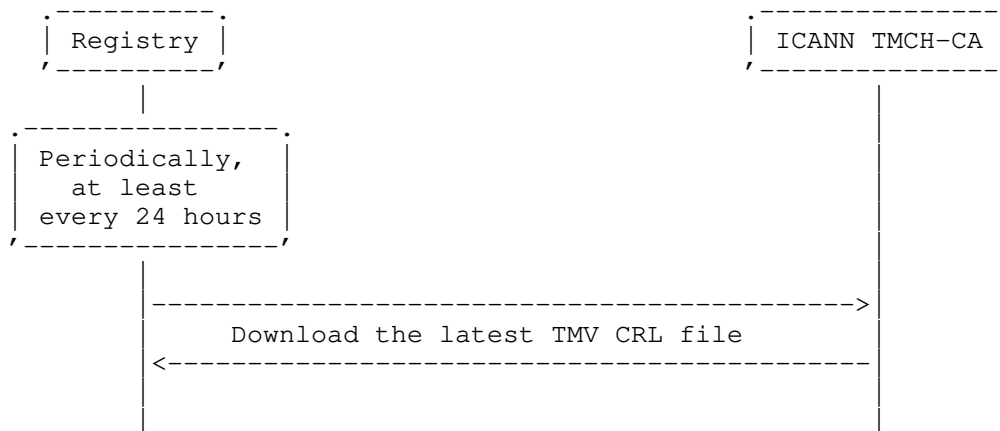


Figure 5

Figure 5 depicts the process of downloading the latest TMV CRL file initiated by the Registry.

5.2.3.3. Notice of Registered Domain Names (NORN)

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

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

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

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

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

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

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

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

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

Notification of Registered Domain Name

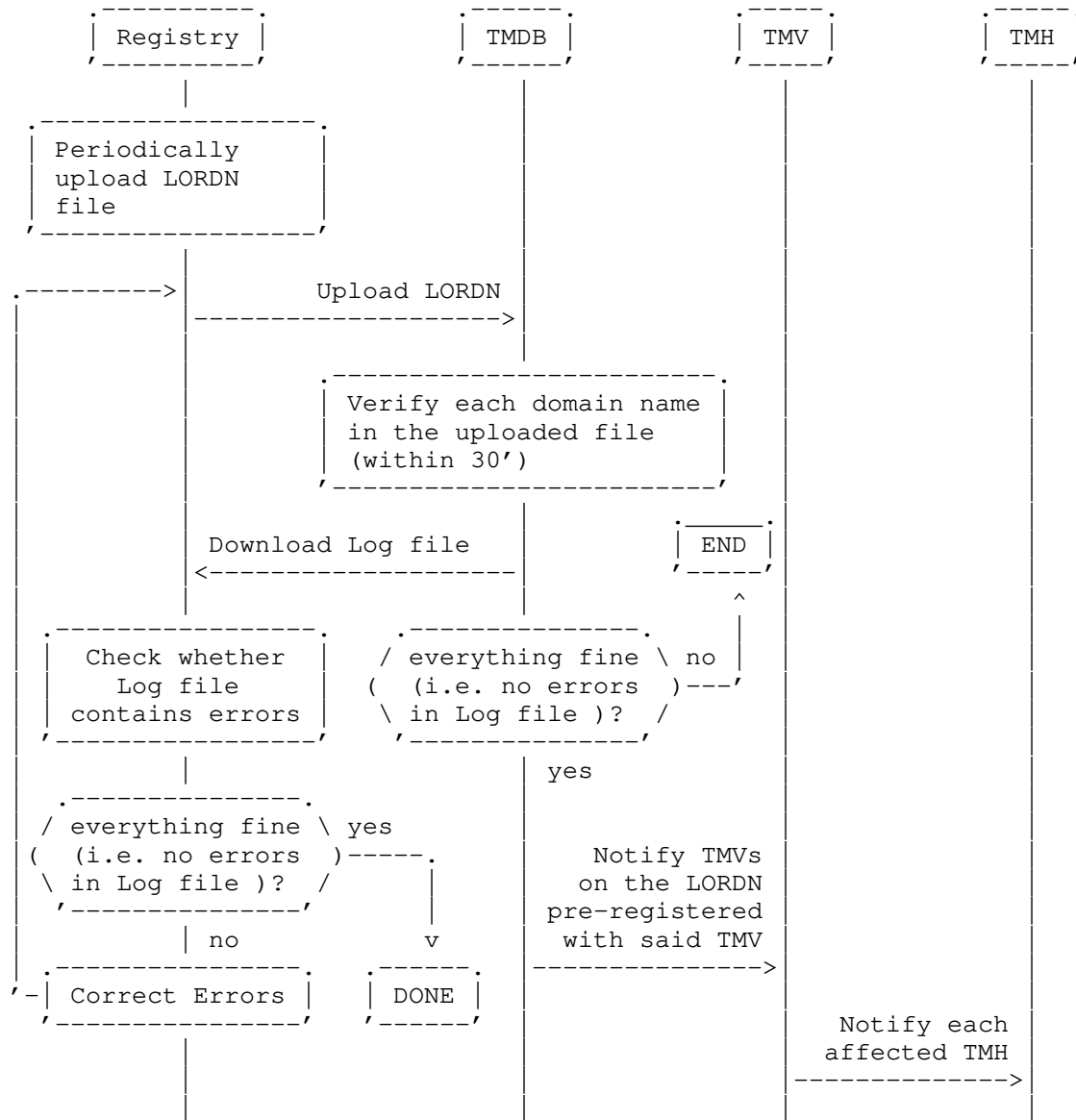


Figure 6

Figure 6 depicts the process to notify the TMH of Registered Domain Names.

The format used for the LORDN is described in Section 6.3

5.2.4. Sunrise Domain Name registration by Registrars

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

5.2.5. TMDB Sunrise Services for Registrars

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

5.3. Trademark Claims Period

5.3.1. Domain Registration

Domain Name registration during the Trademark Claims Period

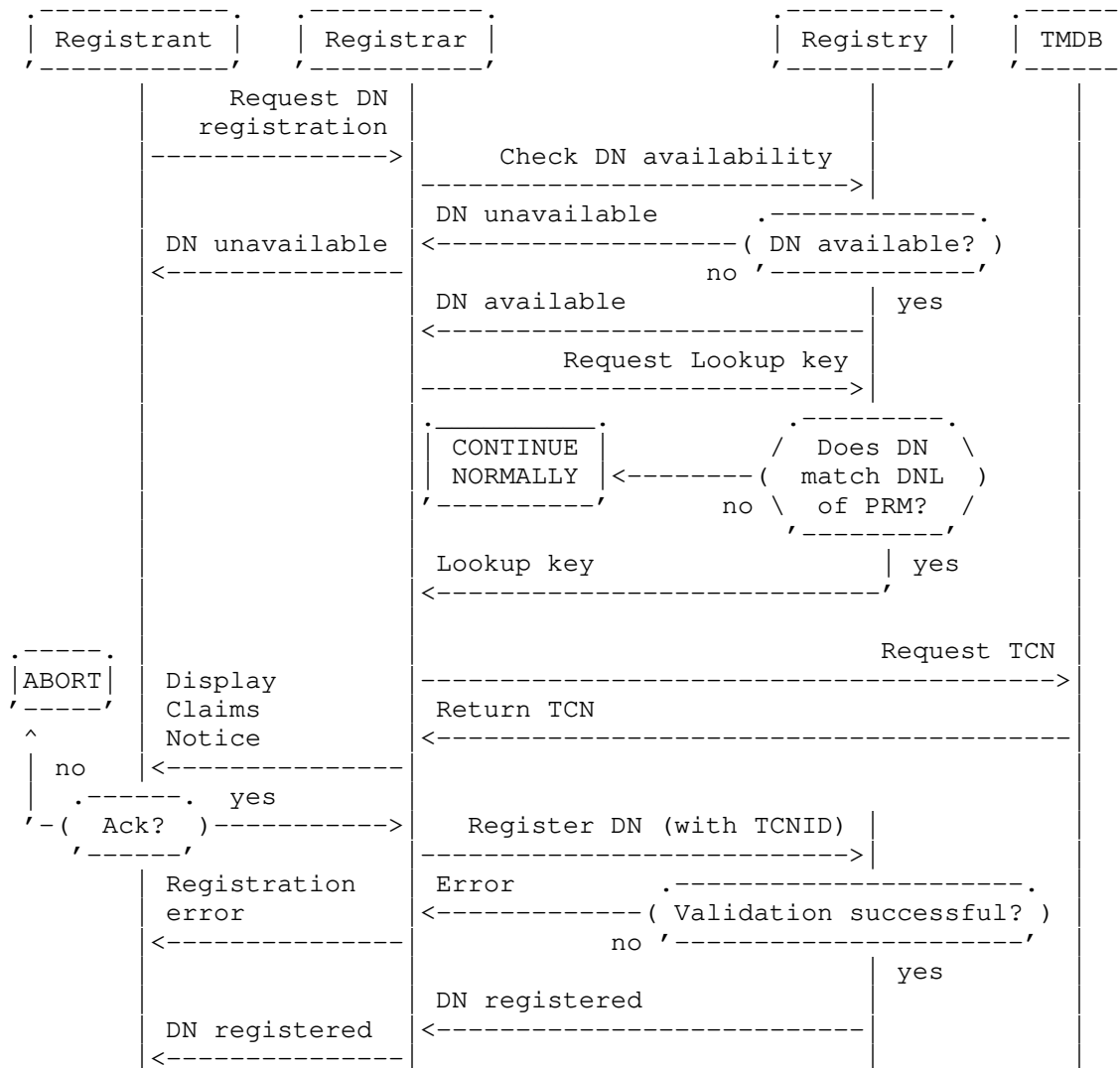


Figure 7

Figure 7 represents a synchronous DN registration workflow (usually called first come first served).

5.3.2. Trademark Claims Domain Name registration by Registries

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

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

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

3. The acceptance datetime is within the window of time defined by ICANN policy. In the gTLD round of 2012, Registrars verified that the acceptance datetime was less than or equal to 48 hours in the past, as there were no defined ICANN policies at that time. Implementers should be aware that ICANN policy may define this value in the future.
4. Using the leftmost DNL of the DN being effectively allocated, the expiration datetime provided by the Registrar, and the TMDB Notice Identifier extracted from the TCNID provided by the Registrar compute the TCN Checksum. Verify that the computed TCN Checksum match the TCN Checksum present in the TCNID. For example, if the DN "xn--mgbachtv.xn--mgbh0fb" is being effectively allocated, the leftmost DNL would be "xn--mgbachtv".

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

5.3.3. TMDB Trademark Claims Services for Registries

5.3.3.1. Domain Name Label (DNL) List

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

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

Update of the DNL List

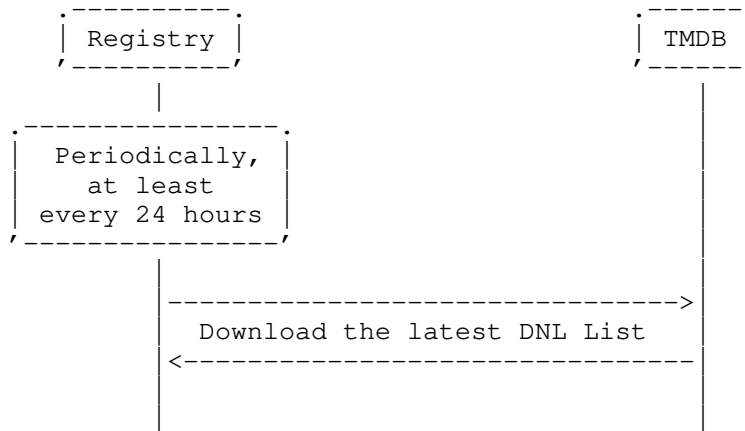


Figure 8

Figure 8 depicts the process of downloading the latest DNL list initiated by the Registry.

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

5.3.3.2. Notice of Registered Domain Names (NORN)

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

5.3.4. Trademark Claims Domain Name registration by Registrars

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

1. Use the Lookup Key received from the Registry to obtain the TCN from the TMDB using the dr interface (Section 4.3.6) Registrars MUST only query for the Lookup Key of a DN that is available for registration.

2. Present the TCN to the Registrant as described in Exhibit A, [RPM-Requirements].
3. Ask Registrant for acknowledgement, i.e. the Registrant MUST consent with the TCN, before any further processing. (The transmission of a TCNID to the Registry over the ry interface, Section 4.3.5 implies that the Registrant has expressed his/her consent with the TCN.)
4. Perform the minimum set of checks for verifying DN registrations. If any of these checks fails the Registrar MUST abort the DN registration. Each of these checks MUST be performed before the registration is sent to the Registry. Performing the minimum set of checks Registrars MUST verify that:
 1. The datetime when the validation is done is within the TCN validity based on the <tmNotice:notBefore> and <tmNotice:notAfter> elements.
 2. The leftmost DNL of the DN being effectively allocated matches the label (<tmNotice:label>) element in the TCN. For example, if the DN "xn--mgbachtv.xn--mgbh0fb" is being effectively allocated, the leftmost DNL would be "xn--mgbachtv".
 3. The Registrant has acknowledged (expressed his/her consent with) the TCN.
5. Record the date and time when the registrant acknowledged the TCN.
6. Send the registration to the Registry (ry interface, Section 4.3.5) and include the following information:
 - * TCNID (<tmNotice:id>)
 - * Expiration date of the TCN (<tmNotice:notAfter>)
 - * Acceptance datetime of the TCN.

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

Registrars SHOULD implement a cache of TCNs to minimize the number of queries sent to the TMDB. A cached TCN MUST be removed from the cache after the expiration date of the TCN as defined by <tmNotice:notAfter>.

The TMDB MAY implement rate-limiting as one of the protection mechanisms to mitigate the risk of performance degradation.

5.3.5. TMDB Trademark Claims Services for Registrars

5.3.5.1. Claims Notice Information Service (CNIS)

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

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

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

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

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

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

The TLS certificate (HTTPS) used on the dr interface (Section 4.3.6) MUST be signed by a well-know public CA. Registrars MUST perform the Certification Path Validation described in Section 6 of [RFC5280]. Registrars will be authenticated in the dr interface using HTTP Basic access authentication. The dr (Section 4.3.6) interface MUST support HTTPS keep-alive and MUST maintain the connection for up to 30 minutes.

5.4. Qualified Launch Program (QLP) Period

5.4.1. Domain Registration

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

MUST provide Sunrise and/or Trademark Claims services as described in Section 5.2 and Section 5.3.

The effective allocation scenarios are:

- o If the leftmost DNL of the DN being effectively allocated (QLP Name in this section) matches a DNL in the SURL, and an SMD is provided, then Registries MUST provide Sunrise Services (see Section 5.2) and the DN MUST be reported in a Sunrise LORDN file during the QLP Period. For example, if the DN "xn--mgbachtv.xn--mgbh0fb" is being effectively allocated, the leftmost DNL would be "xn--mgbachtv".
- o If the QLP Name matches a DNL in the SURL but does not match a DNL in the DNL List, and an SMD is NOT provided (see section 2.2 of [QLP-Addendum]), then the DN MUST be reported in a Sunrise LORDN file using the special SMD-id "99999-99999" during the QLP Period.
- o If the QLP Name matches a DNL in the SURL and also matches a DNL in the DNL List, and an SMD is NOT provided (see section 2.2 of [QLP-Addendum]), then Registries MUST provide Trademark Claims services (see Section 5.3) and the DN MUST be reported in a Trademark Claims LORDN file during the QLP Period.
- o If the QLP Name matches a DNL in the DNL List but does not match a DNL in the SURL, then Registries MUST provide Trademark Claims services (see Section 5.2) and the DN MUST be reported in a Trademark Claims LORDN file during the QLP Period.

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

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

QLP Effective Allocation Scenarios

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

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

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

- o SMD Revocation List (see Section 5.2.3.1)

- o CNIS (see Section 5.3.5.1)

5.4.2. TMDB QLP Services for Registries

5.4.2.1. Sunrise List (SURL)

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

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

Update of the SURL

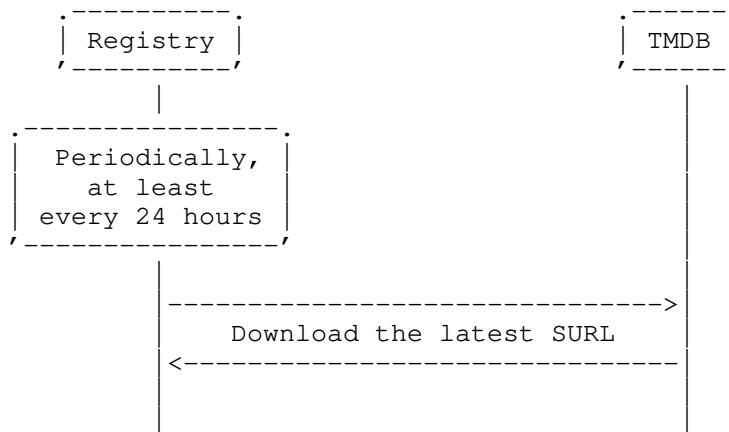


Figure 9

Figure 9 depicts the process of downloading the latest SURL initiated by the Registry.

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

6. Data Format Descriptions

6.1. Domain Name Label (DNL) List

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

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

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

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

Where:

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

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

With the header names as follows:

DNL,lookup-key,insertion-datetime

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

Where:

- + <DNL>, a Domain Name Label covered by a PRM.
- + <lookup key>, lookup key that the Registry MUST provide to the Registrar. The lookup key has the following format: <YYYY><MM><DD><vv>/<X>/<X>/<X>/<Random bits><Sequential number>, where:
 - YYYY: year that the TCN was generated.
 - MM: zero-padded month that the TCN was generated.
 - DD: zero-padded day that the TCN was generated.

- vv: version of the TCN, possible values are 00 and 01.
 - X: one hex character. This is the first, second and third hex character of encoding the <Random bits> in base16 as specified in [RFC4648].
 - Random bits: 144 random bits encoded in base64url as specified in [RFC4648].
 - Sequential number: zero-padded natural number in the range 0000000001 to 2147483647.
- + <DNL insertion datetime>, datetime in UTC that the DNL was first inserted into the DNL List. The possible two values of time for inserting a DNL to the DNL List are 00:00:00 and 12:00:00 UTC.

Example of a DNL List:

```
1,2012-08-16T00:00:00.OZ
DNL,lookup-key,insertion-datetime
example,2013041500/2/6/9/rJ1NrDO92vDsAzf7EQzgJX4R0000000001,\
  2010-07-14T00:00:00.OZ
another-example,2013041500/6/A/5/a1JAqG2vI2BmCv5PfUvuDkf40000000002,\
  2012-08-16T00:00:00.OZ
anotherexample,2013041500/A/C/7/rHdC4wnrWRvPY6nneCVtQhFj00000000003,\
  2011-08-16T12:00:00.OZ
```

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

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

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

6.2. SMD Revocation List

This section defines the format of the list of SMDs that have been revoked. The list is maintained by the TMDB and downloaded by Registries (and optionally by Registrars) in regular intervals (see Section 5.2.3.1). The SMD Revocation List is used during the Sunrise

Period to validate SMDs received. The SMD Revocation List has a similar function as CRLs used in PKI [RFC5280].

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

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

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

Where:

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

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

With the header names as follows:

smd-id,insertion-datetime

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

Where:

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

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

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

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

Example of an SMD Revocation List:

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

6.3. List of Registered Domain Names (LORDN) file

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

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

- o Sunrise LORDN file:

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

- o Trademark Claims LORDN file:

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

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

- o Sunrise LORDN file (during QLP Period):

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

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

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

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

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

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

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

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

202 Accepted

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

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

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

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

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

- o For Sunrise Period:

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

Where:

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

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

With the header names as follows:

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

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

Where:

- <roid>, DN Repository Object Identifier (DNROID) in the SRS.
 - <DN registered>, DN that was effectively allocated. For IDNs, the A-label form is used.
 - <SMD-id>, SMD ID used for registration.
 - <IANA Registrar ID>, IANA Registrar ID.
 - <datetime of registration>, date and time in UTC that the domain was effectively allocated.
 - OPTIONAL <datetime of application creation>, date and time in UTC that the application was created. The <datetime of application creation> MUST be provided in case of a DN effective allocation based on an asynchronous registration (e.g., when using auctions).

Example of a Sunrise LORDN file:

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

o For Trademark Claims Period:

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

Where:

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

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

With the header names as follows:

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

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

Where:

- <roid>, DN Repository Object Identifier (DNROID) in the SRS.
- <DN registered>, DN that was effectively allocated. For IDNs, the A-label form is used.
- <TCNID>, Trademark Claims Notice Identifier as specified in <tmNotice:id>.

- <IANA Registrar ID>, IANA Registrar ID.
- <datetime of registration>, date and time in UTC that the domain was effectively allocated.
- <datetime of acceptance of the TCN>, date and time in UTC that the TCN was acknowledged.
- OPTIONAL <datetime of application creation>, date and time in UTC that the application was created. The <datetime of application creation> MUST be provided in case of a DN effective allocation based on an asynchronous registration (e.g., when using auctions).

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

Example of a Trademark Claims LORDN file:

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

6.3.1. LORDN Log file

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

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

- o Sunrise LORDN Log file:

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

- o Trademark Claims LORDN Log file:

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

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

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

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

For example, to obtain the LORDN Log file in case of a Sunrise LORDN file with LORDN Transaction Identifier 00000000000000000001 and TLD "example" the URI would be:

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

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

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

Where:

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

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

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

With the header names as follows:

roid,result-code

- o One or more lines with: <roid>,<result code>

Where:

- + <roid>, DN Repository Object Identifier (DNROID) in the SRS.
- + <result code>, result code as described in Section 6.3.1.1.

Example of a LORDN Log file:

```
1,2012-08-16T02:15:00.0Z,2012-08-16T00:00:00.0Z,\
  00000000000000478Nzs+3VMkR8ckuUynOLmyeqTmZQSbzDuf/R50n2n5QX4=,\
  accepted,no-warnings,1
roid,result-code
SH8013-REP,2000
```

6.3.1.1. LORDN Log Result Codes

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

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

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

LORDN Log Result Code Classes:

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

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

LORDN Log Result Codes:

rc	Short Description Long Description
-----	-----
2000	OK DN Line successfully processed.
2001	OK but not processed DN Line is syntactically correct but was not processed because the LORDN file was rejected.
3601	TCN Acceptance Date after Registration Date TCN Acceptance Date in DN Line is newer than the Registration Date.
3602	Duplicate DN Line This DN Line is an exact duplicate of another DN Line in same file, DN Line ignored.
3603	DNROID Notified Earlier Same DNROID has been notified earlier, DN Line ignored.
3604	TCN Checksum invalid Based on the DN effectively allocated, the TCNID and the expiration date of the linked TCN, the TCN Checksum is invalid.
3605	TCN Expired The TCN was already expired (based on the <tmNotice:notAfter> field of the TCN) at the datetime of acknowledgement.
3606	Wrong TCNID used The TCNID used for the registration does not match the related DN.
3609	Invalid SMD used The SMD used for registration was not valid at the moment of registration based on the <smd:notBefore> and <smd:notAfter> elements. In case of an asynchronous registration, this refer to the <datetime of application creation>.

- 3610 DN reported outside of the time window
The DN was reported outside of the required 26 hours reporting window.
- 3611 DN does not match the labels in SMD
The DN does not match the labels included in the SMD.
- 3612 SMDID does not exist
The SMDID has never existed in the central repository.
- 3613 SMD was revoked when used
The SMD used for registration was revoked more than 24 hours ago of the <datetime of registration>.
In case of an asynchronous registration, the <datetime of application creation> is used when validating the DN Line.
- 3614 TCNID does not exist
The TCNID has never existed in the central repository.
- 3615 Recent-dnl-insertion outside of the time window
The DN registration is reported as a recent-dnl-insertion, but the (re) insertion into the DNL occurred more than 24 hours ago.
- 3616 Registration Date of DN in Claims before the end of Sunrise Period
The registration date of the DN is before the end of the Sunrise Period and the DN was reported in a Trademark Claims LORDN file.
- 3617 Registrar has not been approved by the TMDB
Registrar ID in DN Line has not completed Trademark Claims integration testing with the TMDB.
- 3618 Registration Date of DN in QLP LORDN file out of the QLP Period
The registration date of the DN in a QLP LORDN file is outside of the QLP Period.
- 3619 TCN was not valid
The TCN was not valid (based on the <tmNotice:notBefore> field of the TCN) at the datetime of acknowledgement.
- 4501 Syntax Error in DN Line
Syntax Error in DN Line.
- 4601 Invalid TLD used
The TLD in the DN Line does not match what is expected for this LORDN.

- 4602 Registrar ID Invalid
Registrar ID in DN Line is not a valid ICANN-Accredited Registrar.
- 4603 Registration Date in the future
The <datetime of registration> in the DN Line is in the future.
- 4606 TLD not in Sunrise or Trademark Claims Period
The <datetime of registration> was reported when the TLD was not in Sunrise or Trademark Claims Periods.
In case of an asynchronous registration, the <datetime of application creation> is used when validating the DN Line.
- 4607 Application Date in the future
The <datetime of application creation> in the DN Line is in the future.
- 4608 Application Date is later than Registration Date
The <datetime of application creation> in the DN Line is later than the <datetime of registration>.
- 4609 TCNID wrong syntax
The syntax of the TCNID is invalid.
- 4610 TCN Acceptance Date is in the future
The <datetime of acceptance of the TCN> is in the future.
- 4611 Label has never existed in the TMDB
The label in the registered DN has never existed in the TMDB.

6.4. Signed Mark Data (SMD) File

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

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

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

1. Marks: <marks>
2. smdID: <SMD-ID>
3. U-labels: <comma separated list of U-label or NR-LDH labels (see [RFC5890])>
4. notBefore: <begin validity>
5. notAfter: <end validity>
6. -----BEGIN ENCODED SMD-----
7. <encoded SMD (see [RFC7848])>
8. -----END ENCODED SMD-----

Example of an SMD File:

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

6.5. Trademark Claims Notice (TCN)

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

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

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

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

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

Example of a TCNID:

370d0b7c9223372036854775807.

Where:

+ TCN Checksum=370d0b7c

+ TMDB Notice Identifier=9223372036854775807

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

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

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

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

A Registry MUST use the leftmost DNL of the DN being effectively allocated, the expiration datetime of the TCN (provided by the Registrar) and the TMDB Notice Identifier extracted from the TCNID (provided by the Registrar) to compute the TCN Checksum. For example, if the DN "xn--mgbachtv.xn--mgbh0fb" is being effectively allocated, the leftmost DNL would be "xn--mgbachtv".

Example of computation of the TCN Checksum:

CRC32(example-one12819492009223372036854775807)=370d0b7c

- o A <tmNotice:notBefore> element that contains the start of the validity date and time of the TCN.

- o A <tmNotice:notAfter> element that contains the expiration date and time of the TCN.
- o A <tmNotice:label> element that contains the DNL covered by a PRM.
- o One or more <tmNotice:claim> elements that contain the Trademark Claim. The <tmNotice:claim> element contains the following child elements:
 - * A <tmNotice:markName> element that contains the mark text string.
 - * One or more <tmNotice:holder> elements that contains the information of the holder of the mark. An "entitlement" attribute is used to identify the entitlement of the holder, possible values are: owner, assignee or licensee. The child elements of <tmNotice:holder> include:
 - + An OPTIONAL <tmNotice:name> element that contains the name of the holder. A <tmNotice:name> MUST be specified if <tmNotice:org> is not specified.
 - + An OPTIONAL <tmNotice:org> element that contains the name of the organization holder of the mark. A <tmNotice:org> MUST be specified if <tmNotice:name> is not specified.
 - + A <tmNotice:addr> element that contains the address information of the holder of a mark. A <tmNotice:addr> contains the following child elements:
 - One, two or three OPTIONAL <tmNotice:street> elements that contains the organization's street address.
 - A <tmNotice:city> element that contains the organization's city.
 - An OPTIONAL <tmNotice:sp> element that contains the organization's state or province.
 - An OPTIONAL <tmNotice:pc> element that contains the organization's postal code.
 - A <tmNotice:cc> element that contains the organization's country code. This a two-character code from [ISO3166-2].
 - + An OPTIONAL <tmNotice:voice> element that contains the organization's voice telephone number.

- + An OPTIONAL <tmNotice:fax> element that contains the organization's facsimile telephone number.
- + An OPTIONAL <tmNotice:email> element that contains the email address of the holder.
- * Zero or more OPTIONAL <tmNotice:contact> elements that contains the information of the representative of the mark registration. A "type" attribute is used to identify the type of contact, possible values are: owner, agent or thirdparty. The child elements of <tmNotice:contact> include:
 - + A <tmNotice:name> element that contains name of the responsible person.
 - + An OPTIONAL <tmNotice:org> element that contains the name of the organization of the contact.
 - + A <tmNotice:addr> element that contains the address information of the contact. A <tmNotice:addr> contains the following child elements:
 - One, two or three OPTIONAL <tmNotice:street> elements that contains the contact's street address.
 - A <tmNotice:city> element that contains the contact's city.
 - An OPTIONAL <tmNotice:sp> element that contains the contact's state or province.
 - An OPTIONAL <tmNotice:pc> element that contains the contact's postal code.
 - A <tmNotice:cc> element that contains the contact's country code. This a two-character code from [ISO3166-2].
 - + A <tmNotice:voice> element that contains the contact's voice telephone number.
 - + An OPTIONAL <tmNotice:fax> element that contains the contact's facsimile telephone number.
 - + A <tmNotice:email> element that contains the contact's email address.

- * A `<tmNotice:jurDesc>` element that contains the name (in English) of the jurisdiction where the mark is protected. A `jurCC` attribute contains the two-character code of the jurisdiction where the mark was registered. This is a two-character code from [WIPO.ST3].
- * Zero or more OPTIONAL `<tmNotice:classDesc>` element that contains the description (in English) of the Nice Classification as defined in [WIPO-NICE-CLASSES]. A `classNum` attribute contains the class number.
- * A `<tmNotice:goodsAndServices>` element that contains the full description of the goods and services mentioned in the mark registration document.
- * An OPTIONAL `<tmNotice:notExactMatch>` element signals that the claim notice was added to the TCN based on other rule (e.g. [Claims50]) than exact match (defined in [MatchingRules]). The `<tmNotice:notExactMatch>` contains one or more:
 - + An OPTIONAL `<tmNotice:udrp>` element that signals that the claim notice was added because of a previously abused name included in an UDRP case. The `<tmNotice:udrp>` contains:
 - A `<tmNotice:caseNo>` element that contains the UDRP case number used to validate the previously abused name.
 - A `<tmNotice:udrpProvider>` element that contains the name of the UDRP provider.
 - + An OPTIONAL `<tmNotice:court>` element that signals that the claim notice was added because of a previously abused name included in a court's resolution. The `<tmNotice:court>` contains:
 - A `<tmNotice:refNum>` element that contains the reference number of the court's resolution used to validate the previously abused name.
 - A `<tmNotice:cc>` element that contains the two-character code from [ISO3166-2] of the jurisdiction of the court.
 - A `<tmNotice:courtName>` element that contains the name of the court.

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

```

<?xml version="1.0" encoding="UTF-8"?>
<tmNotice:notice xmlns:tmNotice="urn:ietf:params:xml:ns:tmNotice-1.0">
  <tmNotice:id>370d0b7c9223372036854775807</tmNotice:id>
  <tmNotice:notBefore>2010-08-14T09:00:00.0Z</tmNotice:notBefore>
  <tmNotice:notAfter>2010-08-16T09:00:00.0Z</tmNotice:notAfter>
  <tmNotice:label>example-one</tmNotice:label>
  <tmNotice:claim>
    <tmNotice:markName>Example One</tmNotice:markName>
    <tmNotice:holder entitlement="owner">
      <tmNotice:org>Example Inc.</tmNotice:org>
      <tmNotice:addr>
        <tmNotice:street>123 Example Dr.</tmNotice:street>
        <tmNotice:street>Suite 100</tmNotice:street>
        <tmNotice:city>Reston</tmNotice:city>
        <tmNotice:sp>VA</tmNotice:sp>
        <tmNotice:pc>20190</tmNotice:pc>
        <tmNotice:cc>US</tmNotice:cc>
      </tmNotice:addr>
    </tmNotice:holder>
    <tmNotice:contact type="owner">
      <tmNotice:name>Joe Doe</tmNotice:name>
      <tmNotice:org>Example Inc.</tmNotice:org>
      <tmNotice:addr>
        <tmNotice:street>123 Example Dr.</tmNotice:street>
        <tmNotice:street>Suite 100</tmNotice:street>
        <tmNotice:city>Reston</tmNotice:city>
        <tmNotice:sp>VA</tmNotice:sp>
        <tmNotice:pc>20190</tmNotice:pc>
        <tmNotice:cc>US</tmNotice:cc>
      </tmNotice:addr>
      <tmNotice:voice x="4321">+1.7035555555</tmNotice:voice>
      <tmNotice:email>jdoe@example.com</tmNotice:email>
    </tmNotice:contact>
    <tmNotice:jurDesc jurCC="US">USA</tmNotice:jurDesc>
    <tmNotice:classDesc classNum="35">
      Advertising; business management; business administration.
    </tmNotice:classDesc>
    <tmNotice:classDesc classNum="36">
      Insurance; financial affairs; monetary affairs; real estate.
    </tmNotice:classDesc>
    <tmNotice:goodsAndServices>
      Bardus populorum circumdabit se cum captiosus populum.
      Smert populorum circumdabit se cum captiosus populum.
    </tmNotice:goodsAndServices>
  </tmNotice:claim>
</tmNotice:claim>
  <tmNotice:markName>Example-One</tmNotice:markName>
  <tmNotice:holder entitlement="owner">

```

```
<tmNotice:org>Example S.A. de C.V.</tmNotice:org>
<tmNotice:addr>
  <tmNotice:street>Calle conocida #343</tmNotice:street>
  <tmNotice:city>Conocida</tmNotice:city>
  <tmNotice:sp>SP</tmNotice:sp>
  <tmNotice:pc>82140</tmNotice:pc>
  <tmNotice:cc>BR</tmNotice:cc>
</tmNotice:addr>
</tmNotice:holder>
<tmNotice:jurDesc jurCC="BR">BRAZIL</tmNotice:jurDesc>
<tmNotice:goodsAndServices>
  Bardus populorum circumdabit se cum captiosus populum.
  Smert populorum circumdabit se cum captiosus populum.
</tmNotice:goodsAndServices>
</tmNotice:claim>
<tmNotice:claim>
  <tmNotice:markName>One</tmNotice:markName>
  <tmNotice:holder entitlement="owner">
    <tmNotice:org>One Corporation</tmNotice:org>
    <tmNotice:addr>
      <tmNotice:street>Otra calle</tmNotice:street>
      <tmNotice:city>Otra ciudad</tmNotice:city>
      <tmNotice:sp>OT</tmNotice:sp>
      <tmNotice:pc>383742</tmNotice:pc>
      <tmNotice:cc>CR</tmNotice:cc>
    </tmNotice:addr>
  </tmNotice:holder>
  <tmNotice:jurDesc jurCC="CR">COSTA RICA</tmNotice:jurDesc>
  <tmNotice:goodsAndServices>
    Bardus populorum circumdabit se cum captiosus populum.
    Smert populorum circumdabit se cum captiosus populum.
  </tmNotice:goodsAndServices>
  <tmNotice:notExactMatch>
    <tmNotice:court>
      <tmNotice:refNum>234235</tmNotice:refNum>
      <tmNotice:cc>CR</tmNotice:cc>
      <tmNotice:courtName>Supreme Court of Spain</tmNotice:courtName>
    </tmNotice:court>
  </tmNotice:notExactMatch>
</tmNotice:claim>
<tmNotice:claim>
  <tmNotice:markName>One Inc</tmNotice:markName>
  <tmNotice:holder entitlement="owner">
    <tmNotice:org>One SA de CV</tmNotice:org>
    <tmNotice:addr>
      <tmNotice:street>La calle</tmNotice:street>
      <tmNotice:city>La ciudad</tmNotice:city>
      <tmNotice:sp>CD</tmNotice:sp>
```

```

      <tmNotice:pc>34323</tmNotice:pc>
      <tmNotice:cc>AR</tmNotice:cc>
    </tmNotice:addr>
  </tmNotice:holder>
  <tmNotice:jurDesc jurCC="AR">ARGENTINA</tmNotice:jurDesc>
  <tmNotice:goodsAndServices>
    Bardus populorum circumdabit se cum captiosus populum.
    Smert populorum circumdabit se cum captiosus populum.
  </tmNotice:goodsAndServices>
  <tmNotice:notExactMatch>
    <tmNotice:udrp>
      <tmNotice:caseNo>D2003-0499</tmNotice:caseNo>
      <tmNotice:udrpProvider>WIPO</tmNotice:udrpProvider>
    </tmNotice:udrp>
  </tmNotice:notExactMatch>
</tmNotice:claim>
</tmNotice:notice>

```

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

6.6. Sunrise List (SURL)

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

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

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

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

Where:

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

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

With the header names as follows:

DNL,insertion-datetime

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

Where:

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

Example of a Sunrise List:

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

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

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

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

7. Formal Syntax

7.1. Trademark Claims Notice (TCN)

The schema presented here is for a Trademark Claims Notice.

The CODE BEGINS and CODE ENDS tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.

```
<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
```

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

  <annotation>
    <documentation>
      Schema for representing a Trademark Claim Notice.
    </documentation>
  </annotation>
  <import namespace="urn:ietf:params:xml:ns:mark-1.0"/>
  <element name="notice" type="tmNotice:noticeType"/>
  <complexType name="holderType">
    <sequence>
      <element name="name" type="token" minOccurs="0"/>
      <element name="org" type="token" minOccurs="0"/>
      <element name="addr" type="tmNotice:addrType"/>
      <element name="voice" type="mark:e164Type" minOccurs="0"/>
      <element name="fax" type="mark:e164Type" minOccurs="0"/>
      <element name="email" type="mark:minTokenType" minOccurs="0"/>
    </sequence>
    <attribute name="entitlement" type="mark:entitlementType"/>
  </complexType>
  <complexType name="noticeType">
    <sequence>
      <element name="id" type="tmNotice:idType"/>
      <element name="notBefore" type="dateTime"/>
      <element name="notAfter" type="dateTime"/>
      <element name="label" type="mark:labelType"/>
      <element name="claim" type="tmNotice:claimType" minOccurs="0"
        maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <complexType name="claimType">
    <sequence>
      <element name="markName" type="token"/>
      <element name="holder" type="tmNotice:holderType"
        maxOccurs="unbounded"/>
      <element name="contact" type="tmNotice:contactType" minOccurs="0"
        maxOccurs="unbounded"/>
      <element name="jurDesc" type="tmNotice:jurDescType"/>
      <element name="classDesc" type="tmNotice:classDescType"
        minOccurs="0" maxOccurs="unbounded"/>
      <element name="goodsAndServices" type="token"/>
      <element name="notExactMatch" type="tmNotice:noExactMatchType"
        minOccurs="0"/>
    </sequence>
```

```
</complexType>
<complexType name="jurDescType">
  <simpleContent>
    <extension base="token">
      <attribute name="jurCC" type="mark:ccType" use="required"/>
    </extension>
  </simpleContent>
</complexType>
<complexType name="classDescType">
  <simpleContent>
    <extension base="token">
      <attribute name="classNum" type="integer" use="required"/>
    </extension>
  </simpleContent>
</complexType>
<complexType name="noExactMatchType">
  <choice maxOccurs="unbounded">
    <element name="udrp" type="tmNotice:udrpType"/>
    <element name="court" type="tmNotice:courtType"/>
  </choice>
</complexType>
<complexType name="udrpType">
  <sequence>
    <element name="caseNo" type="token"/>
    <element name="udrpProvider" type="token"/>
  </sequence>
</complexType>
<complexType name="courtType">
  <sequence>
    <element name="refNum" type="token"/>
    <element name="cc" type="mark:ccType"/>
    <element name="region" type="token" minOccurs="0"
      maxOccurs="unbounded"/>
    <element name="courtName" type="token"/>
  </sequence>
</complexType>
<complexType name="addrType">
  <sequence>
    <element name="street" type="token" minOccurs="1" maxOccurs="3"/>
    <element name="city" type="token"/>
    <element name="sp" type="token" minOccurs="0"/>
    <element name="pc" type="mark:pcType" minOccurs="0"/>
    <element name="cc" type="mark:ccType"/>
  </sequence>
</complexType>
<complexType name="contactType">
  <sequence>
    <element name="name" type="token"/>
```

```
<element name="org" type="token" minOccurs="0"/>
<element name="addr" type="tmNotice:addrType"/>
<element name="voice" type="mark:e164Type"/>
<element name="fax" type="mark:e164Type" minOccurs="0"/>
<element name="email" type="mark:minTokenType"/>
</sequence>
<attribute name="type" type="mark:contactTypeType"/>
</complexType>
<simpleType name="idType">
  <restriction base="token">
    <pattern value="[a-zA-F0-9]{8}\d{1,19}"/>
  </restriction>
</simpleType>
</schema>
<CODE ENDS>
```

8. Acknowledgements

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

9. Change History

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

9.1. Version 04

1. Ping update.

9.2. Version 05

1. Ping update.

9.3. Version 06

1. Updated the terminology text to reflect the text in RFC8174.
2. Updated the reference of RFC7719 to RFC8499.

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

9.4. Version 07

1. Changes based on the feedback provided here:
<https://mailarchive.ietf.org/arch/msg/regex/xZFOAajlUJzgPgZBuqlIWRcFZg/>
2. Changes based on the feedback provided here:
https://mailarchive.ietf.org/arch/msg/regex/MdOhSomd6_djLcthfW5mxWZkbWY

9.5. Version 08

1. Fixed issues detected by idnits tool.

9.6. Version 09

1. Ping update.

9.7. Version 10

1. Ping update.

9.8. Version 11

1. Editorial updates.
2. Added Privacy section.

9.9. Version 12

1. Editorial updates.

9.10. Version 13

1. Editorial updates.

9.11. Version 14

1. Editorial updates.

10. IANA Considerations

The code point assigned in support of this document is taken from the wrong point in the registration tree. Unfortunately, the code point has already been deployed in the field without following the proper

registration review process. The Designated Experts for the registry have considered the issues that correcting this action would cause for deployed implementations and have consented to the continued use of the code point.

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

Registration request for the Trademark Claims Notice namespace:

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

Registrant Contact: IETF <iesg@ietf.org> ICANN
<globalsupport@icann.org>

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

Note: Note that this assignment is made from the wrong point in the tree in order to be consistent with deployed implementations.

Registration request for the Trademark Claims Notice XML schema:

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

Registrant Contact: IETF <iesg@ietf.org> ICANN
<globalsupport@icann.org>

XML: See Section 7.1 of this document.

Note: Note that this assignment is made from the wrong point in the tree in order to be consistent with deployed implementations.

11. Security Considerations

This specification uses HTTP Basic Authentication to provide a simple application-layer authentication service. HTTPS is used in all interfaces in order to protect against most common attacks. In addition, the client identifier is tied to a set of IP addresses that are allowed to connect to the interfaces described in this document, providing an extra security measure.

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

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

The TMDB, Registries and Registrars MUST use the best practices described in [RFC7525] or its successors.

12. Privacy Considerations

This specification defines the interfaces to support the [RPM-Requirements]. Legal documents govern the interactions between the different parties, and such legal documents must ensure that privacy-sensitive and/or personal data receives the required protection.

13. References

13.1. Normative References

[Claims50]

ICANN, "Implementation Notes: Trademark Claims Protection for Previously Abused Names", July 2013, <<https://newgtlds.icann.org/en/about/trademark-clearinghouse/previously-abused-16jul13-en.pdf>>.

[MatchingRules]

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

[QLP-Addendum]

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

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

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

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.

- [RFC7848] Lozano, G., "Mark and Signed Mark Objects Mapping", RFC 7848, DOI 10.17487/RFC7848, June 2016, <<https://www.rfc-editor.org/info/rfc7848>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RPM-Requirements]
ICANN, "Rights Protection Mechanism Requirements", September 2013, <<https://newgtlds.icann.org/en/about/trademark-clearinghouse/rpm-requirements-30sep13-en.pdf>>.
- [W3C.REC-xml-20081126]
Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition) REC-xml-20081126", November 2008, <<https://www.w3.org/TR/2008/REC-xml-20081126/>>.
- [W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition REC-xmlschema-1-20041028", October 2004, <<https://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>>.
- [W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition REC-xmlschema-2-20041028", October 2004, <<https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>>.

13.2. Informative References

- [ICANN-GTLD-AGB-20120604]
ICANN, "gTLD Applicant Guidebook Version 2012-06-04", June 2012, <<http://newgtlds.icann.org/en/applicants/agb/guidebook-full-04jun12-en.pdf>>.
- [ISO3166-2]
ISO, "International Standard for country codes and codes for their subdivisions", 2006, <http://www.iso.org/iso/home/standards/country_codes.htm>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<https://www.rfc-editor.org/info/rfc2818>>.

- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC4180] Shafranovich, Y., "Common Format and MIME Type for Comma-Separated Values (CSV) Files", RFC 4180, DOI 10.17487/RFC4180, October 2005, <<https://www.rfc-editor.org/info/rfc4180>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <<https://www.rfc-editor.org/info/rfc4880>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<https://www.rfc-editor.org/info/rfc5890>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.
- [RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <<https://www.rfc-editor.org/info/rfc7235>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.

[WIPO-NICE-CLASSES]

WIPO, "WIPO Nice Classification", 2015,
<<http://www.wipo.int/classifications/nice/en>>.

[WIPO.ST3]

WIPO, "Recommended standard on two-letter codes for the
representation of states, other entities and
intergovernmental organizations", March 2007,
<http://www.iso.org/iso/home/standards/country_codes.htm>.

Author's Address

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

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

REGEXT
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2020

A. Mayrhofer
nic.at GmbH
November 01, 2019

Domain Suggestion Extension for the Extensible Provisioning Protocol
(EPP)
draft-mayrhofer-epp-domain-suggest-00

Abstract

This document specifies an EPP Extension that allows servers to suggest available domain names to clients, for example in cases where the originally desired domain name is unavailable for registration.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 4, 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	2
2. Terminology	2
3. Domain Name Suggestion Structure	3
4. Client and Server Behaviour	3
5. EPP Command Mapping	4
5.1. EPP <check> Query Command	4
6. Open Questions	5
7. Formal Syntax	6
8. Security Considerations	6
9. IANA Considerations	6
9.1. Namespace	6
9.2. XML Schema	6
10. Changelog	6
10.1. mayrhofer-epp-domain-suggestion-00	6
11. Normative References	6
Appendix A. Acknowledgments	7
Author's Address	7

1. Introduction

The Extensible Provisioning Protocol (EPP) [RFC5730] is a client-server protocol for provisioning and managing objects in shared repositories. In many cases, EPP is used to provision Domain Names between Registrars and Domain Name Registries (see [RFC5731]).

EPP provides the "check" query command to determine whether an object can be provisioned with a registry. That command is typically used to determine whether a certain domain name is available for registration at a Domain Name Registry. In case a requested domain name is not available for registration, it is desirable to suggest alternative, available names to the client. However, EPP does currently not contain data structures suitable to transport such "Domain Suggestions".

This document specifies a Command-Response level EPP extension for the EPP Domain Mapping [RFC5731], allowing servers to include such Domain Suggestions in responses to EPP "<domain:check>" commands.

2. Terminology

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

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

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.

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

3. Domain Name Suggestion Structure

In order to convey domain name suggestions, the following XML structure is defined:

- o A <ds:suggestions> element for use in responses, containing one or more <ds:name> elements
- o Each <ds:name> element contains a suggested (available) fully qualified domain name, and an OPTIONAL "for" attribute.
- o If present, the "for" attribute of the <ds:name> element MUST contain a domain name given in one of the <domain:name> elements of the corresponding command. This allows a client to correlate suggestions with originally requested names when multiple names were given in the command.

4. Client and Server Behaviour

- o A client MUST indicate support for the "urn:ietf:params:xml:ns:epp:domainSuggest-1.0" in the "<login>" command in order to receive suggestions
- o When a client indicates support for the extension, it is local server policy if and when suggestions are provided.
- o When a server attempts to provide suggestions, but fails to do so for the set of given names, it SHOULD indicate that situation with an empty <ds:suggestions> element in the response.

- o A server SHOULD NOT suggest domain names which are unavailable for registration.
- o A client hence SHOULD assume that suggested names are available for registration, without the need for an additional <check> command for those names.
- o Servers SHOULD gracefully handle situations where generation of suggestions triggers errors, and continue to process the base EPP command.
- o Servers MAY also give suggestions even if the originally requested name is available.

5. EPP Command Mapping

The only command extended is the <domain:check> command.

5.1. EPP <check> Query Command

This extension does not add any elements to the EPP <check> command described in the EPP Domain Mapping [RFC5731]. However, additional elements are defined for the <check> response:

When a <check> command has been processed successfully, the EPP <extension> element MAY contain a child <ds:suggestions> element, structured as described above.

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="0">example.net</domain:name>
S:          <domain:reason>In use</domain:reason>
S:        </domain:cd>
S:        <domain:cd>
S:          <domain:name avail="1">example.org</domain:name>
S:        </domain:cd>
S:      </domain:chkData>
S:    </resData>
S:    <extension>
S:      <ds:suggestions
S:        xmlns:ds="urn:ietf:params:xml:ns:domainSuggest-1.0">
S:        <ds:name for="example.net">my.example.net</ds:name>
S:        <ds:name for="example.com">wedosubdomains.example.com</ds:name>
S:        <ds:name>betterexample.tld</ds:name>
S:      </ds:suggestions>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

6. Open Questions

[Note to RFC Editor: Do not publish this document before that section is empty :)]

The following issues need to be solved / discussed before the extension can be deemed stable:

- o Shall there be an element in the commands to explicitly request suggestions (<ds:yesplease>).
- o Corner Case: Can error responses contain suggestions? Eg. when a domain in an unsupported TLD is given?

- o Shall suggestions be allowed in other commands?
- o More mechanics for handling keywords (back and forth?)
- o Allow conveyance of user location? Tricky, involves handling PII data...
- o Maximum number of suggestions? Order / weight of suggestions?

7. Formal Syntax

TODO: Create Schema once structure of extension is stable.

8. Security Considerations

At this stage of the document, Security Considerations of the Extension have not been discussed yet :)

9. IANA Considerations

IANA is requested to register perform registrations for the Namespace and XML schema as follows:

9.1. Namespace

TODO once stable

9.2. XML Schema

TODO once stable

10. Changelog

Note to RFC editor: Remove this entire section before publication.

10.1. mayrhofer-epp-domain-suggestion-00

Initial strawman proposal

11. 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>>.

- [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>>.
- [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. Acknowledgments

Provide in-depth review or actual text if you like your name to appear here :D

Author's Address

Alexander Mayrhofer
nic.at GmbH
Karlsplatz 1/2/9
Vienna 1010
Austria

Email: alex.mayrhofer.ietf@gmail.com