Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 22 May 2021

D. Belyavskiy

J. Gould
VeriSign, Inc.

18 November 2020

Use of Internationalized Email Addresses in EPP protocol draft-belyavskiy-epp-eai-02

Abstract

This document permits usage of Internationalized Email Addresses in the EPP protocol. The Extensible Provisioning Protocol (EPP), being developed before appearing the standards for Internationalized Email Addresses (EAI), does not support such email addressed. This document describes an EPP extension that allows EAI addresses to be used with an EPP object mapping like the EPP contact mapping.

TO BE REMOVED on turning to RFC: The document is edited in the dedicated github repo (https://github.com/beldmit/eppeai). Please send your submissions via GitHub.

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 22 May 2021.

Copyright Notice

Table of Contents

1. Introduction

RFC 6530 [RFC6530] introduced the framework for Internationalized Email Addresses. To make such addresses more widely accepted, the changes to various protocols need to be introduced.

This document describes an Extensible Provisioning Protocol (EPP) extension for using the Email Addresses Internationalized (EAI) in an extension to EPP object mappings like the EPP contact mapping [RFC5733]. The extension can be applied to any EPP object mapping that uses an email address, where the EPP contact mapping [RFC5733] is used in the examples.

The Extensible Provisioning Protocol (EPP) specified in RFC 5730 [RFC5730] is a base document for object management operations and an extensible framework that maps protocol operations to objects. The specifics of various objects managed via EPP is described in separate documents. This document is only referring to an email address as a property of a managed object, such as the <contact:email> element in the EPP contact mapping [RFC5733] or the <org:email> element in the EPP organization mapping [RFC8543].

RFC 3735 [RFC3735] provides a guideline to extend the EPP protocol for various purposes. This extension represents a Command-Response Extension.

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.

In examples, "C:" represents lines sent by a protocol client and "S:" represents lines returned by a protocol server. In examples, indentation and whitespace are provided only to illustrate element relationships and are not a required feature of this protocol.

"eai-0.1" is used as an abbreviation for "urn:ietf:params:xml:ns:epp:eai-0.1". The XML namespace prefix "eai" is used, but implementations MUST NOT depend on it. Instead, they are to employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

2. Migrating to Newer Versions of This Extension

Servers that 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.

3. Object Attributes

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

3.1. <eai:eai> Extension Element

The <eai:eai> element can be added to a command or response to override an email element using the "[EAI-ADDRESS]" value, as described in Section 3.2. The <eai:eai> element element contains the following child elements:

<eai:email>: Contains an email address matching the specification in RFC 6530 [RFC6530].

Example <eai:eai> element containing an EAI email address:

```
<eai:eai "urn:ietf:params:xml:ns:epp:eai-0.1">
  <eai:email>someaddress@example.com</eai:email>
</eai:eai>
```

3.2. [EAI-ADDRESS] Email Value

When an EPP object mapping email element contains the predefined value of "[EAI-ADDRESS]", the <eai:email> element overrides the EPP object mapping email element, which is a constant value for the server to use the <eai:email> element for the value. The "[EAI-ADDRESS]" predefined string MUST be supported by the server for the client to explicitly indicate to the server whether to use <eai:email> element in place of the EPP object email element. The server MUST NOT allow the client to set the EPP object mapping email element to the value "[EAI-ADDRESS]".

4. Email Address Specification

Email address syntax is defined in in RFC 6530 [RFC6530]. This mapping does not prescribe minimum or maximum lengths for character strings used to represent email addresses.

5. EPP commands mapping

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

5.1. EPP Query Commands

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

5.1.1. EPP <check> Command

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

5.1.2. EPP <info> command

This extension does not define additional elements to extend the EPP <info> command of an EPP object mapping, but does include additional elements to extend the EPP <info> response.

If the query was successful, the server replies with the regular EPP <resData>. If the client includes the "eai-0.1" XML namespace in the login services, the email address exists, and the email address was set using the extension in the create command (Section 5.2.1) or update command (Section 5.2.5), then the EPP object mapping email element SHOULD be set with the value "[EAI-ADDRESS]" value, as described in Section 3.2, and the <eai:eai> extension element (Section 3.1) is included in the response with the email address value.

Example <info> response for the 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:      <contact:infData
S:        xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
S:        <contact:id>sh8013</contact:id>
S:        <contact:roid>SH8013-REP</contact:roid>
S:        <contact:status s="linked"/>
S:        <contact:status s="clientDeleteProhibited"/>
S:        <contact:postalInfo type="int">
S:          <contact:name>John Doe</contact:name>
S:          <contact:org>Example Inc.</contact:org>
S:          <contact:addr>
S:            <contact:street>123 Example Dr.</contact:street>
```

```
S:            <contact:street>Suite 100</contact:street>
S:            <contact:city>Dulles</contact:city>
S:            <contact:sp>VA</contact:sp>
S:            <contact:pc>20166-6503</contact:pc>
S:            <contact:cc>US</contact:cc>
S:          </contact:addr>
S:        </contact:postalInfo>
S:        <contact:voice x="1234">+1.7035555555</contact:voice>
S:        <contact:fax>+1.7035555556</contact:fax>
S:        <contact:email>jdoe@example.com</contact:email>
S:        <contact:clID>ClientY</contact:clID>
S:        <contact:crID>ClientX</contact:crID>
S:        <contact:crDate>1999-04-03T22:00:00.0Z</contact:crDate>
S:        <contact:upID>ClientX</contact:upID>
S:        <contact:upDate>1999-12-03T09:00:00.0Z</contact:upDate>
S:        <contact:trDate>2000-04-08T09:00:00.0Z</contact:trDate>
S:        <contact:authInfo>
S:          <contact:pw>2fooBAR</contact:pw>
S:        </contact:authInfo>
S:        <contact:disclose flag="0">
S:          <contact:voice/>
S:          <contact:email/>
S:        </contact:disclose>
S:      </contact:infData>
S:    </resData>
S:    <extension>
S:      <eai:eai
S:        xmlns:eai=
S:          "urn:ietf:params:xml:ns:epp:eai-0.1">
S:        <eai:email>someaddress@example.com</eai:email>
S:      </eai:eai>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5.1.3. EPP <transfer> Command

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

5.2. EPP Transform Commands

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

5.2.1. EPP <create> command

This extension defines additional elements to extend the EPP <create> command of an object mapping like the EPP contact mapping [RFC5733]

The EPP <create> command provides a transform operation that allows a client to create an object. In addition to the EPP command elements described in an object mapping like the EPP contact mapping [RFC5733], the command MAY contain a child <eai:eai> element, as defined in Section 3.1.

Example <create> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <contact:create
C:       xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
C:        <contact:id>sh8013</contact:id>
C:        <contact:postalInfo type="int">
C:          <contact:name>John Doe</contact:name>
C:          <contact:org>Example Inc.</contact:org>
C:          <contact:addr>
C:            <contact:street>123 Example Dr.</contact:street>
C:            <contact:street>Suite 100</contact:street>
C:            <contact:city>Dulles</contact:city>
C:            <contact:sp>VA</contact:sp>
C:            <contact:pc>20166-6503</contact:pc>
C:            <contact:cc>US</contact:cc>
C:          </contact:addr>
C:        </contact:postalInfo>
C:        <contact:voice x="1234">+1.7035555555</contact:voice>
C:        <contact:fax>+1.7035555556</contact:fax>
C:        <contact:email>[EAI-ADDRESS]</contact:email>
C:        <contact:authInfo>
C:          <contact:pw>2fooBAR</contact:pw>
C:        </contact:authInfo>
C:        <contact:disclose flag="0">
C:          <contact:voice/>
C:          <contact:email/>
C:        </contact:disclose>
C:      </contact:create>
C:    </create>
C:    <extension>
C:      <eai:eai
C:        xmlns:eai=
C:          "urn:ietf:params:xml:ns:epp:eai-0.1">
C:        <eai:email>someaddress@example.com</eai:email>
C:      </eai:eai>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

5.2.2. EPP <delete> Command

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

5.2.3. EPP <renew> Command

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

5.2.4. EPP <transfer> Command

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

5.2.5. EPP <update> command

The EPP <update> command provides a transform operation that allows a client to update an object. In addition to the EPP command elements described in an object mapping like the EPP contact mapping [RFC5733], the command MAY contain a child <eai:eai> element, as defined in Section 3.1. When executing the <update> command, there are multiple possibilities of changing the email address:

* The EPP object mapping email element is not includes, which means the email address of the contact is not changed. The extension MUST NOT be present.

* The EPP object mapping email element is included with the "[EAI-ADDRESS]" value, the extension MUST be present and contain a valid email address.

* The EPP object mapping email element is included without the "[EAI-ADDRESS]" value, the extension MUST NOT be present.

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:      <contact:update
C:       xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
C:        <contact:id>sh8013</contact:id>
C:        <contact:chg>
C:          <contact:email>[EAI-ADDRESS]</contact:email>
C:        </contact:chg>
C:      </contact:update>
C:    </update>
C:    <extension>
C:      <eai:eai
C:        xmlns:eai=
C:          "urn:ietf:params:xml:ns:epp:eai-0.1">
C:        <eai:email>someaddress@example.net</eai:email>
C:      </eai:eai>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

6. Formal syntax

The Internationalized Email Addresses in EPP protocol schema is presented here.

The formal syntax shown here is a complete XML Schema representation of the object mapping suitable for automated validation of EPP XML instances. The <CODE BEGINS> and <CODE ENDS> 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.

```
<CODE BEGINS>
<?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:eai="urn:ietf:params:xml:ns:epp:eai-0.1"
   targetNamespace="urn:ietf:params:xml:ns:epp:eai-0.1"
   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>
       Use of Internationalized Email Addresses in
       Extensible Provisioning Protocol v1.0 Schema.
     </documentation>
   </annotation>

   <!-- EAI extension element -->
   <element name="eai" type="eai:eaiType" />

   <complexType name="eaiType">
     <sequence>
       <element name="email"
         type="eppcom:minTokenType"/>
     </sequence>
   </complexType>

</schema>
<CODE ENDS>
```

7. Security Considerations

Registries SHOULD validate the domain names in the provided email addresses. This can be done by validating all code points according to IDNA2008 [RFC5892].

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 RFC 3688 [RFC3688]. The following URI assignment should be made by IANA:

Registration request for the eai namespace:

URI: urn:ietf:params:xml:ns:epp:eai-0.1
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.

Registration request for the eai XML Schema:

URI: urn:ietf:params:xml:schema:epp:eai-0.1
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 IANA in the "Extensions for the Extensible Provisioning Protocol (EPP)" registry described in RFC 7451 [RFC7451]. The details of the registration are as follows:

Name of Extension: Use of Internationalized Email Addresses in EPP protocol
Document status: Standards Track
Reference: TBA
Registrant Name and Email Address: IESG, <iesg@ietf.org>
Top-Level Domains(TLDs): Any
IPR Disclosure: None
Status: Active
Notes: None

9. Implementation Considerations

For the sake of uniform syntax on the client side, it is RECOMMENDED to registries to allow any valid address, including the ASCII-only, in the <eai:email> element.

Registries MAY apply extra limitation to the email address syntax (e.g. the addresses can be limited to Left-to-Right scripts). These limitations are out of scope of this document.

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>.

[RFC3735] Hollenbeck, S., "Guidelines for Extending the Extensible Provisioning Protocol (EPP)", RFC 3735, DOI 10.17487/RFC3735, March 2004, <https://www.rfc-editor.org/info/rfc3735>.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

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

[RFC6530] Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, DOI 10.17487/RFC6530, February 2012, <https://www.rfc-editor.org/info/rfc6530>.

[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>.

10.2. Informative References

[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>.

[RFC8543] Zhou, L., Kong, N., Yao, J., Gould, J., and G. Zhou, "Extensible Provisioning Protocol (EPP) Organization Mapping", RFC 8543, DOI 10.17487/RFC8543, March 2019, <https://www.rfc-editor.org/info/rfc8543>.

Appendix A. Change History

A.1. Change from 00 to 01

1. Changed from update of RFC 5733 to use the "Placeholder Text and a New Email Element" EPP Extension approach.

A.2. Change from 01 to 02

1. Fixed the XML schema and the XML examples based on validating them.

2. Added James Gould as co-author.

3. Updated the language to apply to any EPP object mapping and to use the EPP contact mapping as an example.

4. Updated the structure of document to be consistent with the other Command-Response Extensions.

5. Replaced the use of "eppEAI" in the XML namespace and the XML namespace prefix with "eai".

6. Changed to use a pointed XML namespace with "0.1" instead of "1.0".

Authors' Addresses

Dmitry Belyavskiy
8 marta st.
Moscow
127083
Russian Federation

Phone: +7 916 262 5593
Email: beldmit@gmail.com

James Gould
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States of America

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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: May 5, 2021

T. Harrison
G. Michaelson
APNIC

November 1, 2020

RDAP Redirect with Content
draft-harrison-regext-rdap-redirect-with-content-00

Abstract

The Registration Data Access Protocol (RDAP) is used by Regional Internet Registries (RIRs) and Domain Name Registries (DNRs) to provide access to their resource registration information. When an RDAP service operator has delegated authority for a resource to a third party, the operator will configure its RDAP service to redirect requests for that resource to the third party. However, the client may be interested in the resource registration data that the first service operator has in its own right, for various reasons. This document defines a conformance code that a service operator can use to indicate that when redirecting requests, it will also include as the body of the response the RDAP object (if any) that it has in its own right for the requested resource.

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 5, 2021.

Table of Contents

1. Introduction

The Registration Data Access Protocol (RDAP) [RFC7480] is used by Regional Internet Registries (RIRs) and Domain Name Registries (DNRs) to provide access to their resource registration information. For a client, this typically involves following the bootstrap process [RFC7484] to determine the base URL for the query. When the service operator for the base URL has delegated authority for a resource to a third party, the operator will configure its RDAP service to redirect requests accordingly.

The client may also be interested in the registration data that the original service operator has, though. For example:

> the original service operator's responses may be in conformance with an externally-defined RDAP profile, such that the client finds them easier or more useful to deal with than the responses provided by the delegated service operator;
>
> the original service operator's responses may provide useful information about larger delegations of resources to delegated service operators;
>
> where both service operators record information about the same delegation (e.g. the same specific IP network), the delegated operator may omit some information recorded by the original operator which is useful to the client; or

the delegated service may be unavailable for some reason, and the client may be content to rely on the data from the original service operator in that situation.

It is already open to a service operator to return an RDAP object as the body of a redirect response: see section 6.4 of [RFC7231] and section 5.2 of [RFC7480], neither of which prohibits this behaviour. However, because this is unusual both in the RDAP context and more generally, this document defines the parameters of this behaviour, as well as a conformance code that can be used to signal that this behaviour is supported.

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. Implementation

An RDAP service operator that implements this specification MUST include an RDAP object ([RFC7483]) as the body of any non-permanent (HTTP 302, 303, or 307) redirect response that it returns, whenever it records data in its own right about the requested resource, or about a resource for which a response may be validly returned for the requested resource (for example, a less-specific IP network object).

An RDAP service operator that implements this specification MUST include the string literal "redirect_with_content" in the rdapConformance array of a "/help" response, as well as any redirect response that includes an RDAP object as its body. The service operator MAY include this string literal in other responses.

3. Security Considerations

[RFC7481] describes security requirements and considerations for RDAP generally. This specification does not introduce any new requirements/considerations past those defined in that document.

4. IANA Considerations

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

Extension identifier: redirect_with_content

Registry operator: Any

Published specification: This document.

Contact: IETF <iesg@ietf.org>

5. Acknowledgements

TBD

6. References

6.1. 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>.

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

[RFC7483] Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", RFC 7483, DOI 10.17487/RFC7483, March 2015, <https://www.rfc-editor.org/info/rfc7483>.

[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>.

6.2. Informative References

[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>.

[RFC7484] Blanchet, M., "Finding the Authoritative Registration Data (RDAP) Service", RFC 7484, DOI 10.17487/RFC7484, March 2015, <https://www.rfc-editor.org/info/rfc7484>.

Authors' Addresses

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

Email: tomh@apnic.net

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

Email: ggm@apnic.net

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

Table of Contents

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:

- A REQUIRED "type" attribute that is used to identify the kind of deposit:
  - FULL: Full.
  - INCR: Incremental.
  - DIFF: Differential.
- 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.
- 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).
- 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:

- A REQUIRED <version> element that identifies the RDE protocol version, this value MUST be 1.0.
- 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>

    <!-- 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-z1RMRb79nmZ6ABmRTo1fU

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_QharAQlA3041j0

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/UaMNvl1xh60ldjpqHHYc3TNsfhg

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 a 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: April 11, 2021

G. Lozano
ICANN
J. Gould
C. Thippeswamy
VeriSign

Oct 08, 2020

Domain Name Registration Data (DNRD) Objects Mapping
draft-ietf-regext-dnrd-objects-mapping-10

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 April 11, 2021.

Table of Contents

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:

- 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.

- 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.

- 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.

- Registrar: The organization that sponsors objects like domains, hosts, and contacts in a Domain Name Registry.

- 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:

- 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.

- EPP parameters: Contains the EPP parameters supported by the Registry Operator.

- Header: Used to specify counters of objects in the database at a certain point in time (watermark).

- 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:

- 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.

- Comma-Separated Values (CSV), see [RFC4180].

- Domain name: see definition of Domain name in [RFC8499].

- Extensible Provisioning Protocol (EPP), see definition of the Extensible Provisioning Protocol in [RFC8499].

- Fully-Qualified Domain Name (FQDN), see definition of FQDN in [RFC8499].

- Internationalized Domain Name (IDN), see definition of Internationalized Domain Name in [RFC8499].

- Label: see definition of Label in [RFC8499].

- Registrant: see definition of Registrant in [RFC8499].

- Registrar: see definition of Registrar in [RFC8499].

- Registry: see definition of Registry in [RFC8499].

- 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.

- 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.

- 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.

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

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

- 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 internalized 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:

- 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).
- A <roid> element that contains the repository object identifier assigned to the domain name object when it was created.
- An OPTIONAL <uName> element that contains the fully-qualified domain name in Unicode character set. It MUST be provided if available.
- 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.
- 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.
- One or more <status> elements that contain the current status descriptors associated with the domain name.
- 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].

- 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.

- Zero or more OPTIONAL <contact> elements that contain identifiers for the human or organizational social information objects associated with the domain name object.

- 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.

- A <clID> element that contains the identifier of the sponsoring registrar.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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].

- 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.

- 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--exampl-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>jd1234</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 parent="true"/>
    <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 parent="true"/>
  </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/>
    <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:

- A <name> element that contains the fully-qualified name of the host object.
- A <roid> element that contains the repository object identifier assigned to the host object when the object was created.
- One or more <status> elements that describe the status of the host object.
- Zero or more <addr> elements that contain the IP addresses associated with the host object.
- A <clID> element that contains the identifier of the sponsoring registrar.
- 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.
- An OPTIONAL <crDate> element that contains the date and time of host-object creation.
- 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.

- 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.

- 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 transfered.

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:

- A <id> element that contains the server-unique identifier of the contact object
- A <roid> element that contains the Repository Object IDentifier assigned to the contact object when the object was created.
- One or more <status> elements that describe the status of the contact object.
- 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.

- 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.

- 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.

- 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.

- 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.

```
domain1admin,Cdomain1admin-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registarX,
clientY,2009-09-13T08:01:00.0Z,registarX,clientY,
2009-11-26T09:10:00.0Z
domain1tech,Cdomain1tech-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registarX,
clientY,2009-09-13T08:01:00.0Z,registarX,clientY,
2009-11-26T09:10:00.0Z
domain1billing,Cdomain1billing-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registarX,
clientY,2009-09-13T08:01:00.0Z,registarX,clientY,
2009-11-26T09:10:00.0Z
domain2admin,Cdomain2admin-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registarX,
clientY,2009-09-13T08:01:00.0Z,registarX,clientY,
2009-11-26T09:10:00.0Z
domain2tech,Cdomain2tech-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registarX,
clientY,2009-09-13T08:01:00.0Z,registarX,clientY,
2009-11-26T09:10:00.0Z
domain2billing,Cdomain2billing-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registarX,
clientY,2009-09-13T08:01:00.0Z,registarX,clientY,
2009-11-26T09:10:00.0Z
xnabc123admin,Cxnabc123admin-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registarX,
clientY,2009-09-13T08:01:00.0Z,registarX,clientY,
2009-11-26T09:10:00.0Z
xnabc123tech,Cxnabc123tech-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registarX,
clientY,2009-09-13T08:01:00.0Z,registarX,clientY,
2009-11-26T09:10:00.0Z
xnabc123billing,Cxnabc123billing-TEST,+1.7035555555,1234,
+1.7035555556,,jdoe@example.example,registrarX,registarX,
clientY,2009-09-13T08:01:00.0Z,registarX,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.

```
domain1admin,int,"John Doe","Example Inc.",
"123 Example Dr.","Suite 100",,Reston,VA,20190,US
domain1tech,int,"John Doe","Example Inc.",
"123 Example Dr.","Suite 100",,Reston,VA,20190,US
domain1billing,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,0,0,0,0,0,0,0,1,1,1
```

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:

- 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.
- An <name> element that contains the name of the registrar.
- An OPTIONAL <gurid> element that contains the Globally Unique Registrar Identifier (GURID) assigned by ICANN.
- An OPTIONAL <status> element that contains the operational status of the registrar. Possible values are: ok, readonly and terminated.

- 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.

- An OPTIONAL <voice> element that contains the registrar's voice telephone number.

- An OPTIONAL <fax> element that contains the registrar's facsimile telephone number.

- An OPTIONAL <email> element that contains the registrar's email address.

- An OPTIONAL <url> element that contains the registrar's URL.

- 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.

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

- 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.

- An <url> element that contains the URL of the IDN table that is being referenced.

- 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:

- 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).

- An OPTIONAL <uName> element that contains the fully-qualified name of the NNDN in Unicode character set. It MUST be provided if available.

- 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.

- 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.

- 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.

- 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--exampl-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:

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

- One or more <objURI> elements that contain namespace URIs representing the objects that the registry's EPP server is capable of managing.

- 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.

- 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:

- An <element> that defines that the referenced <element> is REQUIRED.
- <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:

- 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.

- 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.

- 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).
- All contact objects linked to domain names MUST be present.
- All registrars objects linked to other objects MUST be present.
- No domain name exists as both a domain name and an NNDN.
- The elements listed as required in the <policy> element MUST be present.
- All idnTableRef definitions linked from other objects MUST be present.
- If an EPP Parameters Object was escrowed in the past, one and only one EPP Parameters Object MUST be present.
- 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 seperator 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>
    <!-- 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">
      <simpleContent>
        <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>
      </simpleContent>
    </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>
    </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"
```

```
                    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>
    <!-- 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" />
```

```
            <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 />
```

```
          <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" />
```

```
        <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>
      Registar 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>
    <!-- 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>
    <!--
    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:rdeDnrdCommon-1.0"
        xmlns:rdeDnrdCommon="urn:ietf:params:xml:ns:rdeDnrdCommon-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-z1RMRb79nmZ6ABmRTo1fU

16.20. Changes REGEXT 07 to REGEXT 08

1. Changes based on the feedback provided here: https://mailarchive.ietf.org/arch/msg/regext/UaMNvl1xh60ldjpqHHYc3TNsfhg

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/ExUZenwC81ZQe9x24-8IKT_FWm8

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>jd1234</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>jd1234</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--exampl-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: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>.

[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>.

[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-20feb12-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

Internet Engineering Task Force (IETF)
Internet-Draft
Intended status: Standards Track
Expires: April 22, 2021

T. Sattler
R. Carney
J. Kolker
GoDaddy Inc.

October 23, 2020

Registry Maintenance Notifications for the Extensible Provisioning Protocol (EPP) draft-ietf-regext-epp-registry-maintenance-04

Abstract

This document describes an Extensible Provision Protocol (EPP) mapping for registry's maintenance notifications.

Table of Contents

1. Introduction

Registries usually conduct maintenances and inform registrars in different ways. Given the expansion of the DNS namespace, it is now desirable to provide a method for EPP servers to notify EPP clients as well as a method for EPP clients to query EPP servers for upcoming maintenances.

This document describes an extension mapping for version 1.0 of the Extensible Provision Protocol [RFC5730]. This mapping provides a mechanism by which EPP servers may notify and EPP clients to query for upcoming maintenances.

1.1. Terminology and Definitions

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

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

"maint" is used as an abbreviation for "urn:ietf:params:xml:ns:maintenance-1.0". The XML namespace prefix "maint" 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. Object Attributes

2.1. Internationalized Domain Names

Names of affected hosts MUST be provided in Punycode according to [RFC5891].

2.2. Dates and Times

All dates and times attribute values MUST be expressed in Universal Coordinated Time (UTC) using the Gregorian calendar. The extended date-time form using upper case "T" and "Z" characters defined in ISO 8601 [RFC3339] MUST be used to represent date-time values.

2.3. Maintenance Elements

The <maint:maint> element describes a single registry maintenance event during a specific period. This element will be used at EPP <poll> messages and to extend the EPP <info> command.

For creating a new maintenance the attribute <maint:status> MUST be "active", the attribute <maint:crDate> MUST be set and the attribute <maint:upDate> SHALL NOT be present.

For updating a maintenance the attribute <maint:status> MUST be "active", the attributes <maint:crDate> and <maint:upDate> MUST be set.

For deleting a maintenance the attribute <maint:status> MUST be "inactive", and the attributes <maint:crDate> and <maint:upDate> MUST be set.

<maint:id>
MUST be present and a server unique id and SHALL NOT be changed if maintenance is updated or deleted. A human-readable description of the maintenance is identified via an OPTIONAL "msg" attribute.

<maint:systems>
MUST be present and contains one or more <maint:system> elements. The server SHOULD NOT list systems which are not affected by the maintenance.

<maint:system>
MUST be present at least once and has an element of <maint:name>, <maint:host> and <maint:impact>.

<maint:name>
MUST be present and indicates the name of the affected system, such as "EPP", "WHOIS", "DNS", "Portal", etc.

<maint:host>
MUST be present and indicates the affected maintained system contains the hostname and OPTIONAL an IP address.
Hostname SHALL be Punycode according [RFC5891].
IPv4 addresses SHALL be dotted-decimal notation.
An example of this textual representation is "192.0.2.0".
IPv6 addresses SHALL be according [RFC5952].
An example of this textual representation is "2001:db8::1:0:0:1".

```
      <maint:impact>
        MUST be present and contains the impact level; values MUST
        either be "full" or "partial".

    <maint:environment>
      MUST be present and indicates the type of the affected system;
      the attribute type is REQUIRED and SHOULD either be "production",
      "ote", "staging", "dev" or "custom". And alternatively the
      attribute name could be used to define a server specific affected
      system for example. In that case name MUST be set:
      <maint:environment type="custom" name="marketing"/>

    <maint:start>
      SHOULD be present and indicates the start of the maintenance
      according ISO 8601 [RFC3339].
      Format: YYYY-MM-DDThh:mm:ssTZ

    <maint:end>
      SHOULD be present and indicates the end of the maintenance
      according to ISO 8601 [RFC3339], and MUST be equal to or greater
      than <maint:start>.
      Format: YYYY-MM-DDThh:mm:ssTZ

    <maint:reason>
      MUST be present and contains the reason behind the maintenance;
      values SHOULD either be "planned" or "emergency".

    <maint:detail>
      MAY be present and contains URI to detailed maintenance
      description.

    <maint:description>
      MAY be present and provides a freeform description of the
      maintenance without having to create and traverse an external
      resource.

    <maint:tlds>
      SHOULD be present and contains <maint:tld> elements.

      <maint:tld>
        MUST be present and contains the affected top-level domain.
        Punycode encoded according to [RFC5891].

    <maint:intervention>
      SHOULD be present and contains <maint:connection> and
      <maint:implementation>.

      <maint:connection>
        SHOULD be present and indicates if a client needs to do
        something that is connection-related, such as a reconnect. The
        value SHALL be boolean.
```

```
      <maint:implementation>
        MUST be present and indicates if a client needs to do something
        that is implementation-related, such as a code change. The
        value SHALL be boolean.

    <maint:status>
      MUST be present and indicates the status of the maintenance.
      The value SHALL be either "active" or "inactive".

    <maint:crDate>
      MUST be present and contains the creation date of the maintenance
      according ISO 8601 [RFC3339].
      Format: YYYY-MM-DDThh:mm:ssTZ

    <maint:upDate>
      MAY be present and contains the updated date of the maintenance
      according to ISO 8601 [RFC3339], and if set MUST be equal to or
      greater than <main:crDate>.
      Format: YYYY-MM-DDThh:mm:ssTZ
```

3. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730]. The command mappings described here are specifically for the use to notify of Registry Maintenances and Registry Maintenance object mapping.

3.1. EPP Query Commands

EPP [RFC5730] provides three commands to retrieve object information: <check> to determine if an object is known to the server, <info> to retrieve detailed information associated with an object, and <transfer> to retrieve object transfer status information.

3.1.1. EPP <check> Command

Available check semantics do not apply to maintenance objects, so there is no mapping defined for the EPP <check> command.

3.1.2. EPP <transfer> Command

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

3.1.3. EPP <info> Command

EPP provides the <info> command that is used to retrieve registry maintenance information. In addition to the standard EPP command elements, the <info> command MUST contain a <maint:info> element that identifies the maintenance namespace.

The <maint:info> element MUST contain a child element. It is either <maint:id> to retrieve a specific maintenance notification or <maint:list> to query all maintenance notifications.

If a <maint:info> is send with a <maint:id> that does not exist on the server side, then the result code 2303 MUST be responded.

Please see the defintion of <maint> elements in Section 2.3.

Example <info> command with explicit <maint:id> to get one specific maintenance:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <maint:info
C:        xmlns:maint="urn:ietf:params:xml:ns:maintenance-1.0">
C:        <maint:id>2e6df9b0-4092-4491-bcc8-9fb2166dcee6</maint:id>
C:      </maint:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <info> response for one specific maintenance notification, which was requested with a explicit <maint:id>. In this case, it provides the all "required" elements and additional elements <maint:start>, <maint:end>, <maint:detail>, <maint:description>, <maint:tlds>, <maint:intervention>.

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:      <result code="1000">
S:         <msg>Command completed successfully</msg>
S:      </result>
S:      <resData>
S:        <maint:infData
S:          xmlns:maint="urn:ietf:params:xml:ns:maintenance-1.0">
S:          <maint:maint>
S:            <maint:id>2e6df9b0-4092-4491-bcc8-9fb2166dcee6
S:            </maint:id>
S:            <maint:systems>
S:              <maint:system>
S:                <maint:name>EPP</maint:name>
S:                <maint:host>epp.registry.example
S:                </maint:host>
S:                <maint:impact>blackout</maint:impact>
S:              </maint:system>
S:            </maint:systems>
S:            <maint:environment type="production"/>
S:            <maint:start>2017-09-30T06:00:00Z</maint:start>
S:            <maint:end>2017-09-30T14:25:57Z</maint:end>
```

```
S:            <maint:reason>planned</maint:reason>
S:            <maint:detail>
S:              https://www.registry.example/notice?123
S:            </maint:detail>
S:            <maint:description lang="en">free text
S:            </maint:description>
S:            <maint:tlds>
S:              <maint:tld>example</maint:tld>
S:              <maint:tld>test</maint:tld>
S:            </maint:tlds>
S:            <maint:intervention>
S:              <maint:connection>false</maint:connection>
S:              <maint:implementation>false</maint:implementation>
S:            </maint:intervention>
S:            <maint:status>active</maint:status>
S:            <maint:crDate>2017-03-08T22:10:00Z</maint:crDate>
S:          </maint:maint>
S:        </maint:infData>
S:      </resData>
S:      <trID>
S:         <clTRID>ABC-12345</clTRID>
S:         <svTRID>54321-XYZ</svTRID>
S:      </trID>
S:   </response>
S:</epp>
```

Example <info> command with <maint:list> to query all maintenances subject to server policy:

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

Example <info> response querying all maintenances subject to server policy. In this case, all the "required" elements will be returned and additional <maint:upDate>.

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:      <result code="1000">
S:         <msg>Command completed successfully</msg>
S:      </result>
S:      <resData>
```

```
S:        <maint:infData
S:          xmlns:maint="urn:ietf:params:xml:ns:maintenance-1.0">
S:          <maint:list>
S:            <maint:maint>
S:              <maint:id>2e6df9b0-4092-4491-bcc8-9fb2166dcee6
S:              </maint:id>
S:              <maint:start>2017-04-30T06:00:00Z</maint:start>
S:              <maint:end>2017-04-30T07:00:00Z</maint:end>
S:              <maint:crDate>2017-02-08T22:10:00Z</maint:crDate>
S:            </maint:maint>
S:            <maint:maint>
S:              <maint:id>91e9dabf-c4e9-4c19-a56c-78e3e89c2e2f
S:              </maint:id>
S:              <maint:start>2017-06-15T04:30:00Z</maint:start>
S:              <maint:end>2017-06-15T05:30:00Z</maint:end>
S:              <maint:crDate>2017-02-08T22:10:00Z</maint:crDate>
S:              <maint:upDate>2017-03-08T20:11:00Z</maint:upDate>
S:            </maint:maint>
S:          </maint:list>
S:        </maint:infData>
S:      </resData>
S:      <trID>
S:         <clTRID>ABC-12345</clTRID>
S:         <svTRID>54321-XYZ</svTRID>
S:      </trID>
S:   </response>
S:</epp>
```

3.1.4. EPP <poll> Command

The EPP <poll> command and response is defined in Section 2.9.2.3 of [RFC5730]. The Registry Maintenance Notification is included in the EPP <poll> response of [RFC5730].

For the Registry Maintenance Notification, there are three types of poll messages. The poll message applies whenever the domain name registry creates, updates, or deletes maintenance. In the case of a Registry Maintenance specific message, a <maint:infData> element will be included within the <resData> element of the standard <poll> response.

The <maint:infData> element will include a reference to the Registry Maintenance namespace. EPP data contained within the <maint:infData> element is formatted according to the maintenance-poll schema.

Please see the defintion of <maint> elements in Section 2.3.

Example <poll> 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:  <poll op="req"/>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <poll> response with the Registry Maintenance poll message:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:    </result>
S:  <msgQ count="1" id="12345">
S:    <qDate>2017-02-08T22:10:00Z</qDate>
S:    <msg lang="en">Registry Maintenance Notification</msg>
S:  </msgQ>
S:  <resData>
S:    <maint:infData
S:      xmlns:maint="urn:ietf:params:xml:ns:maintenance-1.0">
S:      <maint:maint>
S:        <maint:id>2e6df9b0-4092-4491-bcc8-9fb2166dcee6</maint:id>
S:        <maint:systems>
S:          <maint:system>
S:            <maint:name>EPP</maint:name>
S:            <maint:host>epp.registry.example
S:            </maint:host>
S:            <maint:impact>blackout</maint:impact>
S:          </maint:system>
S:        </maint:systems>
S:        <maint:environment type="production"/>
S:        <maint:start>2017-10-30T06:00:00Z</maint:start>
S:        <maint:end>2017-10-30T14:25:57Z</maint:end>
S:        <maint:reason>planned</maint:reason>
S:        <maint:detail>
S:          https://www.registry.example/notice?123
S:        </maint:detail>
S:        <maint:tlds>
S:          <maint:tld>example</maint:tld>
S:          <maint:tld>test</maint:tld>
S:        </maint:tlds>
S:        <maint:intervention>
S:          <maint:connection>false</maint:connection>
S:          <maint:implementation>false</maint:implementation>
S:        </maint:intervention>
```

```
S:        <maint:status>active</maint:status>
S:        <maint:crDate>2017-02-08T22:10:00Z</maint:crDate>
S:      </maint:maint>
S:    </maint:infData>
S:  </resData>
S:  <trID>
S:    <clTRID>ABC-12345</clTRID>
S:    <svTRID>54321-XYZ</svTRID>
S:  </trID>
S:  </response>
S:</epp>
```

3.2. EPP Transform Commands

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

3.2.1. EPP <create> Command

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

3.2.2. EPP <delete> Command

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

3.2.3. EPP <renew> Command

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

3.2.4. EPP <transfer> Command

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

3.2.5. EPP <update> Command

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

4. Formal Syntax

One schema is presented here that is the EPP Registry Maintenance 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 end of the schema for URI registration purposes.

4.1. Registry Maintenance EPP Mapping Schema

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

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

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

  <!--
  Child elements found in EPP commands.
  -->
  <element name="info" type="maint:infoType"/>

  <!--
   Child elements of the <info> command.
  -->
  <complexType name="infoType">
    <sequence>
      <choice>
        <element name="list"/>
        <element name="id" type="maint:idType"/>
      </choice>
    </sequence>
  </complexType>
```

```
<!--
Human-readable text may describe the maintenance
-->
<complexType name="idType">
  <simpleContent>
    <extension base="token">
      <attribute name="msg" type="token"/>
      <attribute name="lang" type="language" default="en"/>
    </extension>
  </simpleContent>
</complexType>

<!--
 Info Response element
-->
<element name="infData" type="maint:infDataType"/>

<!--
 <info> response elements.
-->
<complexType name="infDataType">
  <choice>
    <element name="list" type="maint:listDataType"/>
    <element name="maint" type="maint:maintDataType"/>
  </choice>
</complexType>

<!--
 Attributes associated with the list info response
-->
<complexType name="listDataType">
  <sequence>
    <element name="maint" type="maint:maintItemType"
     minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<!--
 Attributes associated with the list item info response
-->
<complexType name="maintItemType">
  <sequence>
    <element name="id" type="maint:idType"/>
    <element name="start" type="dateTime" minOccurs="0"/>
    <element name="end" type="dateTime" minOccurs="0"/>
    <element name="crDate" type="dateTime"/>
    <element name="upDate" type="dateTime" minOccurs="0"/>
  </sequence>
</complexType>
```

```
<!--
 Attributes associated with the maintenance info response
-->
<complexType name="maintDataType">
  <sequence>
    <element name="id" type="maint:idType"/>
    <element name="systems" type="maint:systemsType"/>
    <element name="environment" type="maint:envType"/>
    <element name="start" type="dateTime" minOccurs="0"/>
    <element name="end" type="dateTime" minOccurs="0"/>
    <element name="reason" type="maint:reasonEnum"/>
    <element name="detail" type="anyURI" minOccurs="0"/>
    <element name="description" type="maint:descriptionType"
     minOccurs="0"/>
    <element name="tlds" type="maint:tldsType" minOccurs="0"/>
    <element name="intervention" type="maint:interventionType"
     minOccurs="0"/>
    <element name="status" type="maint:statusEnum"/>
    <element name="crDate" type="dateTime"/>
    <element name="upDate" type="dateTime" minOccurs="0"/>
  </sequence>
</complexType>

<!--
 systems element
-->
<complexType name="systemsType">
  <sequence>
    <element name="system" type="maint:systemType"
     maxOccurs="unbounded"/>
  </sequence>
</complexType>

<!--
 Enumerated list of impacts
-->
<simpleType name="impactEnum">
  <restriction base="token">
    <enumeration value="partial"/>
    <enumeration value="full"/>
  </restriction>
</simpleType>

<!--
 description element
-->
<complexType name="descriptionType">
  <simpleContent>
    <extension base="string">
      <attribute name="lang" type="language" default="en"/>
    </extension>
  </simpleContent>
</complexType>
```

```
<!--
 system element
-->
<complexType name="systemType">
  <sequence>
    <element name="name" type="token"/>
    <element name="host" type="maint:addrType"/>
    <element name="impact" type="maint:impactEnum"/>
  </sequence>
</complexType>

<!--
 host element
-->
<complexType name="addrType">
  <simpleContent>
    <extension base="maint:addrStringType">
      <attribute name="ip" type="maint:ipType" default="v4"/>
    </extension>
  </simpleContent>
</complexType>

<simpleType name="addrStringType">
  <restriction base="token">
    <minLength value="3"/>
    <maxLength value="45"/>
  </restriction>
</simpleType>

<simpleType name="ipType">
  <restriction base="token">
    <enumeration value="v4"/>
    <enumeration value="v6"/>
  </restriction>
</simpleType>

<!--
 Enumerated list of environments
-->
<simpleType name="envEnum">
  <restriction base="token">
    <enumeration value="production"/>
    <enumeration value="ote"/>
    <enumeration value="staging"/>
    <enumeration value="dev"/>
    <enumeration value="custom"/>
  </restriction>
</simpleType>
```

```
    <!--
     environment element
    -->
    <complexType name="envType">
      <simpleContent>
        <extension base="token">
          <attribute name="type" type="maint:envEnum" use="required"/>
          <attribute name="name" type="token" use="optional"/>
        </extension>
      </simpleContent>
    </complexType>

    <!--
     Enumerated list of reasons
    -->
    <simpleType name="reasonEnum">
      <restriction base="token">
        <enumeration value="planned"/>
        <enumeration value="emergency"/>
      </restriction>
    </simpleType>

    <!--
     tlds element
    -->
    <complexType name="tldsType">
      <sequence>
        <element name="tld" type="eppcom:labelType"
         maxOccurs="unbounded"/>
      </sequence>
    </complexType>

    <!--
     intervention element
    -->
    <complexType name="interventionType">
      <sequence>
        <element name="connection" type="boolean"/>
        <element name="implementation" type="boolean"/>
      </sequence>
    </complexType>

    <!--
     Enumerated list of statuses
    -->
    <simpleType name="statusEnum">
      <restriction base="token">
        <enumeration value="active"/>
        <enumeration value="inactive"/>
      </restriction>
    </simpleType>
```

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

5. IANA Considerations

5.1. XML Namespace

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

Registration request for the maintenance namespace:

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

Registrant Contact: IESG

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

Registration request for the maintenance schema:

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

Registrant Contact: IESG

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

5.2. EPP Extension Registry

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

Name of Extension: "Registry Maintenance Notifications for the Extensible Provisioning Protocol (EPP)"

Document status: Standards Track

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

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

6. Security Considerations

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

7. Implementation Status

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

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Add implementation details once available.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <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, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

8.2. Informative References

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

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.

[RFC5952] Kawamura, S. and Kawashima, M., "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

[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>.

[RFC7942] Sheffer, Y. and Farrel, A., "Improving Awareness of Running Code: The Implementation Status Section", RFC 7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

Appendix A. Change History

A.1. Change from draft-sattler-epp-poll-maintenance-response to draft-sattler-epp-registry-maintenance

Updated to be EPP based instead of JSON document.

A.2. Change from draft-sattler-epp-registry-maintenance to draft-ietf-regext-epp-registry-maintenance

Adopted by the REGEXT working group.

A.3. Change from 00 to 01

Clarified maint:description and maint:environment. Changed maint:description from complexType to simpleType. Fixed typo. Added acknowledgment.

A.4. Change from 01 to 02

Update language from Domain Name Registry to Registry. Clarified XML namespace urn:ietf:params:xml:ns:maintenance-1.0. Changed host to contain hostName and hostAddr. Changed maint:tlds from MUST to SHOULD. Fixed maint:status in Schema. Changed UUID to a server unique id.

A.5. Change from 02 to 03

Changed maint:connection from MUST to SHOULD.

A.6. Change from 03 to 04

A lot of clarifications and editoral changes.

Acknowledgments

The authors wish to thank the following individuals for their feedback and suggestions (sorted alphabetically by company):

- Patrick Mevzek
- Neal McPherson, 1&1 IONOS
- Anthony Eden, DNSimple
- Christopher Martens, Donuts
- Quoc-Anh Pham, Neustar
- Raymond Zylstra, Neustar
- Andreas Huber, united-domains
- Craig Marchant, VentraIP
- James Gould, Verisign

Authors' Addresses

Tobias Sattler

Email: tobias.sattler@me.com
URI: https://tobiassattler.com

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

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

Jody Kolker
GoDaddy Inc.
14455 N. Hayden Rd. #219
Scottsdale, AZ 85260
US

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

Table of Contents

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:

- "currentFieldSet": "String" (REQUIRED) either the value of the "fieldSet" parameter as specified in the query string, or the field set applied by default;

- "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:

- "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);

- "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;

- "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:

- the list of fields for each set can differ based on the client’s access and authorization levels;
- 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:

- the client explicitly describes the data fields to be returned;
- 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:

- 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;
- clients need to recognize the returned data structure to avoid cases when the requested fields are invalid;

- 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:

- 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;

- 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: April 29, 2021

M. Loffredo
M. Martinelli
IIT-CNR/Registro.it

October 26, 2020

Registration Data Access Protocol (RDAP) Reverse search capabilities
draft-ietf-regext-rdap-reverse-search-05

Abstract

The Registration Data Access Protocol (RDAP) does not include query capabilities to find the list of domains related to a set of entities matching a given search pattern. In the RDAP context, an entity can be associated to any defined object class. Therefore, a reverse search can be applied to other use cases than the classic domain-entity scenario. This document describes RDAP query extensions that allow servers to provide a reverse search feature based on the relationship between any searchable object and the related entities.

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 29, 2021.

Copyright Notice

Table of Contents

1. Introduction

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

The first objection has been caused by the potential risks of privacy violation. However, TLDs community is considering a new generation of Registration Directory Services [ICANN-RDS1] [ICANN-RDS2] [ICANN-RA], which provide access to sensitive data under some permissible purposes and according to adequate policies to enforce the requestor accreditation, authentication, authorization, and terms and conditions of data use. It is well known that such security policies are not implemented in Whois [RFC3912], while they are in RDAP [RFC7481]. Therefore, RDAP permits a reverse search implementation complying with privacy protection principles.

Another objection to the implementation of a reverse search capability has been connected with its impact on server processing. Since RDAP supports search queries, the impact of both standard and

reverse searches is equivalent and can be mitigated by servers adopting ad hoc strategies. Furthermore, the reverse search is almost always performed by specifying an entity role (e.g. registrant, technical contact) and this can contribute to restricting the result set.

Reverse searches, such as finding the list of domain names associated with contacts 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 massive EPP [RFC5730] updates (e.g. changing the contacts of a set of domains, etc.).

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

The protocol described in this specification aims to extend the RDAP query capabilities to enable the reverse search based on the relationship between any object and the associated entities. The extension is implemented by adding new path segments (i.e. search paths) and using a RESTful web service [REST]. The service is implemented using the Hypertext Transfer Protocol (HTTP) [RFC7230] and the conventions described in [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 new search paths are OPTIONAL extensions of those defined in [RFC7482]. A generic reverse search path is described by the syntax:

{resource-type}/reverse/{role}?{property}=<search pattern>

The path segments are defined as in the following:

- resource-type: it MUST be one of resource type path segments defined in Section 3.2 of [RFC7482]: "domains", "nameservers" or "entities";
- role: it MUST be one of the roles described in Section 10.2.4 of [RFC7483]. For role independent reverse searches, the value "entity" MUST be used;
- property: it identifies the entity property to be used in matching the search pattern. A pre-defined list of properties includes: fn, handle, email, city, country, cc. The mapping between such properties and the RDAP properties is shown in Table 1. Some of the properties are related to jCard elements [RFC7095] but, being jCard the JSON format for vCard [RFC6350], the corresponding definitions are included in vCard specification. Servers MAY implement additional properties to those defined in this document.

Partial string matching is allowed as defined in section 4.1 of [RFC7482].

Reverse search property	RDAP property	RFC 7483	RFC 6350	RFC 8605
handle	handle	5.1.		
fn	jCard fn		6.2.1	
email	jCard email		6.4.2	
city	locality in jCard adr		6.3.1	
country	country name in jCard adr		6.3.1	
cc	country code in jCard adr			3.1

Table 1: Mapping between the reverse search properties and the RDAP properties

```
https://example.com/rdap/domains/reverse/technical?handle=CID-40*

https://example.com/rdap/domains/reverse/registrant?fn=Bobby*

https://example.com/rdap/domains/reverse/registrant?cc=US

https://example.com/rdap/entites/reverse/registrar?handle=RegistrarX
```

Figure 1: Examples of reverse search queries

The "country" property can be used as an alternative to "cc" when RDAP servers don't include the jCard "cc" parameter [RFC8605] in their response.

3. RDAP Conformance

Servers complying with this specification MUST include the value "reverse_search" in the rdapConformance property of the help response [RFC7483]. The information needed to register this value in the "RDAP Extensions" registry is described in Section 6.

4. Implementation Considerations

The implementation of the proposed extension is technically feasible. Both handle and fn are used as standard path segments to search for entities [RFC7482]. With regards to the other reverse search properties, namely email, city and country code, the impact of their usage on server processing is evaluated to be the same as other existing query capabilities (e.g. wildcard prefixed search pattern) so the risks to degrade the performance or to generate huge result sets can be mitigated by adopting the same policies (e.g. restricting the search functionality, limiting the rate of search requests according to the user profile, truncating and paging the results, returning partial responses).

5. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was

supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

5.1. IIT-CNR/Registro.it

Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it
Location: https://rdap.pubtest.nic.it/
Description: This implementation includes support for RDAP queries using data from the public test environment of .it ccTLD.
Level of Maturity: This is a "proof of concept" research implementation.
Coverage: This implementation includes all of the features described in this specification.
Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

6. IANA Considerations

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

Extension identifier: reverse_search
Registry operator: Any
Published specification: This document.
Contact: IETF <iesg@ietf.org>
Intended usage: This extension describes reverse search query patterns for RDAP.

7. Privacy Considerations

The use of the capability described in this document MUST be compliant with the rules about privacy protection each RDAP provider is subject to. Sensitive registration data MUST be protected and accessible for permissible purposes only. Therefore, RDAP servers MUST provide reverse search only to those requestors who are authorized according to a lawful basis. Some potential users of this capability include registrars searching for their own domains and operators in the exercise of an official authority or performing a specific task in the public interest that is set out in a law.

Another scenario consists of permitting reverse searches, which take into account only those entities that have previously given the explicit consent for publishing and processing their personal data.

8. Security Considerations

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

The specification of the entity role within the reverse search path allows the RDAP servers to implement different authorization policies on a per-role basis.

9. Acknowledgements

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

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>.

[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>.

[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>.

[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>.

[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>.

10.2. Informative References

[ICANN-RA]
Internet Corporation For Assigned Names and Numbers, "Registry Agreement", July 2017, <https://newgtlds.icann.org/sites/default/files/agreements/agreement-approved-31jul17-en.pdf>.

[ICANN-RDS1]
Internet Corporation For Assigned Names and Numbers, "Final Report from the Expert Working Group on gTLD Directory Services: A Next-Generation Registration Directory Service (RDS)", June 2014, <https://www.icann.org/en/system/files/files/final-report-06jun14-en.pdf>.

[ICANN-RDS2]
Internet Corporation For Assigned Names and Numbers, "Final Issue Report on a Next-Generation gTLD RDS to Replace WHOIS", October 2015, <http://whois.icann.org/sites/default/files/files/final-issue-report-next-generation-rds-07oct15-en.pdf>.

[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. 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.

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: 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.

Table of Contents

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:

- reducing the server response bandwidth requirements;
- improving server response time;
- improving query precision and, consequently, obtaining more relevant results;
- decreasing server query processing load;
- 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:

- lookup: the server returns only one object;
- 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:

- "count": a boolean value that allows a client to request the return of the total number of objects found;
- "sort": a string value that allows a client to request a specific sort order for the result set;
- "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:

- "currentSort": "String" (OPTIONAL) either the value of "sort" parameter as specified in the query string or the sort applied by default, if any;

- "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:

- "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;

- "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;

- "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;

- "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*256^0 + 0*256^1 + 168*256^2 + 192*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 coincise and easy to use (e.g. "$.domainSearchResults[*].events[?(@.eventActio n='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:

- properties providing service information (e.g. links, notices, remarks);
- multivalued properties (e.g. status, roles, variants);
- 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.

- 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
- 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:

- 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;

- the jCard "sort-as" parameter MUST be ignored for the sorting capability described in this document;

- 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;

- multiple events with a given action on an object might be returned. If this occurs, sorting MUST be applied to the most recent event;

- 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.

- Searchable objects

  registrationDate

  ```
  $.domainSearchResults[*].events[?(@.eventAction=="registration"
  )].eventDate
  ```

  reregistrationDate

  ```
  $.domainSearchResults[*].events[?(@.eventAction=="reregistratio
  n")].eventDate
  ```

  lastChangedDate

  ```
  $.domainSearchResults[*].events[?(@.eventAction=="last
  changed")].eventDate
  ```

  expirationDate

  ```
  $.domainSearchResults[*].events[?(@.eventAction=="expiration")]
  .eventDate
  ```

  deletionDate

```
$.domainSearchResults[*].events[?(@.eventAction=="deletion")].e
ventDate
```

reinstantiationDate

```
$.domainSearchResults[*].events[?(@.eventAction=="reinstantiati
on")].eventDate
```

transferDate

```
$.domainSearchResults[*].events[?(@.eventAction=="transfer")].e
ventDate
```

lockedDate

```
$.domainSearchResults[*].events[?(@.eventAction=="locked")].eve
ntDate
```

unlockedDate

```
$.domainSearchResults[*].events[?(@.eventAction=="unlocked")].e
ventDate
```

- Domain

name

```
$.domainSearchResults[*].[unicodeName,ldhName]
```

- Nameserver

name

```
$.nameserverSearchResults[*].[unicodeName,ldhName]
```

ipv4

```
$.nameserverSearchResults[*].ipAddresses.v4[0]
```

ipv6

```
$.nameserverSearchResults[*].ipAddresses.v6[0]
```

- Entity

handle

```
$.entitySearchResults[*].handle
```

fn

```
$.entitySearchResults[*].vcardArray[1][?(@[0]=="fn")][3]
```

org

```
$.entitySearchResults[*].vcardArray[1][?(@[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:

- 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;

- 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 "b2Zmc2V0PTEwMCxsaW1pdD01MA==" is the Base64 encoding of "offset=100,limit=50". Likewise, in a simple implementation to represent keyset pagination information, the cursor value "ZXhhbXBsZS1OLmNvbQ==" 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=wJlCDLIl6KTWypN7T6vc6nWEmEYe99Hjf1XY1xmqV-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=wJlCDLIl6KTWypN7T6vc6nWEmEYe99Hjf1XY1xmqV-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:

- 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:

- "limit=N": means that the server returns the first N objects of the result set;

- "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:

- when offset has a very high value, scrolling the result set could take some time;

- it always requires fetching all rows before dropping as many rows as specified by offset;

- 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:

- it needs at least one key field;

- it does not allow sorting simply by any field because the sorting criterion must contain a key;

- 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;

- it does not allow direct navigation to arbitrary pages because the result set must be scrolled in sequential order starting from the initial page;

- 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:

- 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;

- 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;

- 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 Counci (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: Additionl 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/

Network Working Group J. Gould
Internet-Draft R. Wilhelm
Intended status: Standards Track VeriSign, Inc.
Expires: 24 April 2021 21 October 2020

Extensible Provisioning Protocol (EPP) Secure Authorization Information for Transfer
draft-ietf-regext-secure-authinfo-transfer-04

Abstract

The Extensible Provisioning Protocol (EPP), in RFC 5730, defines the use of authorization information to authorize a transfer. The authorization information is object-specific and has been defined in the EPP Domain Name Mapping, in RFC 5731, and the EPP Contact Mapping, in RFC 5733, as password-based authorization information. Other authorization mechanisms can be used, but in practice the password-based authorization information has been used at the time of object create, managed with the object update, and used to authorize an object transfer request. What has not been fully considered is the security of the authorization information that includes the complexity of the authorization information, the time-to-live (TTL) of the authorization information, and where and how the authorization information is stored. This document defines an operational practice, using the EPP RFCs, that leverages the use of strong random authorization information values that are short-lived, that are not stored by the client, and that are stored using a cryptographic hash by the server to provide for secure authorization information used for transfers.

Status of This Memo

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

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

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

This Internet-Draft will expire on 24 April 2021.

Table of Contents

1. Introduction

The Extensible Provisioning Protocol (EPP), in [RFC5730], defines the use of authorization information to authorize a transfer. The authorization information is object-specific and has been defined in the EPP Domain Name Mapping, in [RFC5731], and the EPP Contact Mapping, in [RFC5733], as password-based authorization information. Other authorization mechanisms can be used, but in practice the password-based authorization information has been used at the time of object create, managed with the object update, and used to authorize an object transfer request. What has not been considered is the security of the authorization information that includes the complexity of the authorization information, the time-to-live (TTL) of the authorization information, and where and how the authorization information is stored. This document defines an operational practice, using the EPP RFCs, that leverages the use of strong, random authorization information values that are short-lived, that are not stored by the client, and that are stored by the server using a cryptographic hash to provide, for secure authorization information used for transfers. This operational practice can be used to support transfers of any EPP object, where the domain name object defined in [RFC5731] is used in this document for illustration purposes. Elements of the practice may be used to support the secure use of the authorization information for purposes other than transfer, but any other purposes and the applicable elements are out-of-scope for this document.

The overall goal is to have strong, random authorization information values, that are short-lived, and that are either not stored or stored as a cryptographic hash values by the non-responsible parties. In a registrant, registrar, and registry model, the registrant registers the object through the registrar to the registry. The registrant is the responsible party and the registrar and the registry are the non-responsible parties. EPP is a protocol between the registrar and the registry, where the registrar is referred to as the client and the registry is referred to as the server. The following are the elements of the operational practice and how the existing features of the EPP RFCs can be leveraged to satisfy them:

"Strong Random Authorization Information": The EPP RFCs define the

password-based authorization information value using an XML schema "normalizedString" type, so they don't restrict what can be used in any way. This operational practice defines the recommended mechanism for creating a strong random authorization value, that would be generated by the client.

"Short-Lived Authorization Information": The EPP RFCs don't explicitly support short-lived authorization information or a time-to-live (TTL) for authorization information, but there are EPP RFC features that can be leveraged to support short-lived authorization information. If authorization information is set only when there is a transfer in process, the server needs to support empty authorization information on create, support setting and unsetting authorization information, and support automatically unsetting the authorization information upon a successful transfer. All of these features can be supported by the EPP RFCs.

"Storing Authorization Information Securely": The EPP RFCs don't specify where and how the authorization information is stored in the client or the server, so there are no restrictions to define an operational practice for storing the authorization information securely. The operational practice will not require the client to store the authorization information and will require the server to store the authorization information using a cryptographic hash, with at least a 256-bit hash function such as SHA-256, and with a random salt. Returning the authorization information set in an EPP info response will not be supported.

1.1. Conventions Used in This Document

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

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

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

The examples reference XML namespace prefixes that are used for the associated XML namespaces. Implementations MUST NOT depend on the example XML namespaces and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents. The example namespace prefixes used and their associated XML namespaces include:

"domain": urn:ietf:params:xml:ns:domain-1.0
"contact": urn:ietf:params:xml:ns:contact-1.0

2. Registrant, Registrar, Registry

The EPP RFCs refer to client and server, but when it comes to transfers, there are three types of actors that are involved. This document will refer to the actors as registrant, registrar, and registry. [RFC8499] defines these terms formally for the Domain Name System (DNS). The terms are further described below to cover their roles as actors of using the authorization information in the transfer process of any object in the registry, such as a domain name or a contact:

"registrant": [RFC8499] defines the registrant as "an individual or organization on whose behalf a name in a zone is registered by the registry". The registrant can be the owner of any object in the registry, such as a domain name or a contact. The registrant interfaces with the registrar for provisioning the objects. A transfer is coordinated by the registrant to transfer the sponsorship of the object from one registrar to another. The authorization information is meant to authenticate the registrant as the owner of the object to the non-sponsoring registrar and to authorize the transfer.

"registrar": [RFC8499] defines the registrar as "a service provider that acts as a go-between for registrants and registries". The registrar interfaces with the registrant for the provisioning of objects, such as domain names and contacts, and with the registries to satisfy the registrant’s provisioning requests. A registrar may directly interface with the registrant or may indirectly interface with the registrant, typically through one or more resellers. Implementing a transfer using secure authorization information extends through the registrar’s reseller channel up to the direct interface with the registrant. The registrar’s interface with the registries uses EPP. The registrar’s interface with its reseller channel or the registrant is registrar-specific. In the EPP RFCs, the registrar is referred to as the "client", since EPP is the protocol used between the registrar and the registry. The sponsoring registrar is the authorized registrar to manage objects on behalf of the registrant. A non-sponsoring registrar is not authorized to

manage objects on behalf of the registrant. A transfer of an object's sponsorship is from one registrar, referred to as the losing registrar, to another registrar, referred to as the gaining registrar.

"registry": [RFC8499] defines the registry as "the administrative operation of a zone that allows registration of names within the zone". The registry typically interfaces with the registrars over EPP and generally does not interact directly with the registrant. In the EPP RFCs, the registry is referred to as the "server", since EPP is the protocol used between the registrar and the registry. The registry has a record of the sponsoring registrar for each object and provides the mechanism (over EPP) to coordinate a transfer of an object's sponsorship between registrars.

3. Signaling Client and Server Support

This document does not define new protocol but an operational practice using the existing EPP protocol, where the client and the server can signal support for the BCP using a namespace URI in the login and greeting extension services. The namespace URI "urn:ietf:params:xml:ns:epp:secure-authinfo-transfer-1.0" is used to signal support for the BCP. The client includes the namespace URI in an <svcExtension> <extURI> element of the [RFC5730] <login> Command. The server includes the namespace URI in an <svcExtension> <extURI> element of the [RFC5730] Greeting.

A client that receives the namespace URI in the server's Greeting extension services, can expect the following supported behavior by the server:

1. Support empty authorization information with a create command.
2. Support unsetting authorization information with an update command.
3. Support validating authorization information with an info command.
4. Support not returning an indication whether the authorization information is set or unset to the non-sponsoring registrar.
5. Support returning empty authorization information to sponsoring registrar when the authorization information is set in an info response.
6. Support allowing for the passing of a matching non-empty authorization information to authorize a transfer.
7. Support automatically unsetting the authorization information upon a successful completion of transfer.

A server that receives the namespace URI in the client's <login> Command extension services, can expect the following supported behavior by the client:

1. Support generation of authorization information using a secure random value.
2. Support only setting the authorization information when there is a transfer in process.

4. Secure Authorization Information

The authorization information in the EPP RFCs ([RFC5731] and [RFC5733]) that support transfer use password-based authorization information. Other EPP objects that support password-based authorization information for transfer can use the Secure Authorization Information defined in this document. For the authorization information to be secure it must be a strong random value and must have a short time-to-live (TTL). The security of the authorization information is defined in the following sections.

4.1. Secure Random Authorization Information

For authorization information to be secure, it MUST be generated using a secure random value. The authorization information is treated as a password, where according to [RFC4086] a high-security password must have at least 49 bits of randomness or entropy. The required length L of a password, rounded up to the largest whole number, is based on the set of characters N and the desired entropy H, in the equation L = ROUNDUP(H / log2 N). With a target entropy of 49, the required length can be calculated after deciding on the set of characters that will be randomized. The following are a set of possible character sets and the calculation of the required length.

Calculation of the required length with 49 bits of entropy and with the set of all printable ASCII characters except space (0x20), which consists of the 94 characters 0x21-0x7E.

ROUNDUP(49 / log2 94) =~ ROUNDUP(49 / 6.55) =~ ROUNDUP(7.48) = 8

Calculation of the required length with 49 bits of entropy and with the set of case-insensitive alphanumeric characters, which consists of 36 characters (a-z A-Z 0-9).

ROUNDUP(49 / log2 36) =~ ROUNDUP(49 / 5.17) =~ ROUNDUP(9.48) = 10

Considering the age of [RFC4086], the evolution of security practices, and that the authorization information is a machine-generated value, the recommendation is to use at least 128 bits of entropy. The lengths are recalculated below using 128 bits of entropy.

Calculation of the required length with 128 bits of entropy and with the set of all printable ASCII characters except space (0x20), which consists of the 94 characters 0x21-0x7E.

ROUNDUP(128 / log2 94) =~ ROUNDUP(128 / 6.55) =~ ROUNDUP(19.54) = 20

Calculation of the required length with 128 bits of entropy and with the set of case insensitive alphanumeric characters, which consists of 36 characters (a-z A-Z 0-9).

ROUNDUP(128 / log2 36) =~ ROUNDUP(128 / 5.17) =~ ROUNDUP(24.76) = 25

The strength of the random authorization information is dependent on the actual entropy of the underlying random number generator. For the random number generator, the practices defined in [RFC4086] and section 4.7.1 of the NIST Federal Information Processing Standards (FIPS) Publication 140-2 (https://csrc.nist.gov/publications/detail/fips/140/2/final) SHOULD be followed to produce random values that will be resistant to attack. A random number generator (RNG) is preferable over the use of a pseudorandom number generator (PRNG) to reduce the predictability of the authorization information. The more predictable the random number generator is, the lower the true entropy, and the longer the required length for the authorization information.

4.2. Authorization Information Time-To-Live (TTL)

The authorization information SHOULD only be set when there is a transfer in process. This implies that the authorization information has a Time-To-Live (TTL) by which the authorization information is cleared when the TTL expires. The EPP RFCs have no definition of TTL, but since the server supports the setting and unsetting of the authorization information by the sponsoring registrar, then the sponsoring registrar can apply a TTL based on client policy. The TTL client policy may be based on proprietary registrar-specific criteria which provides for a transfer-specific TTL tuned for the particular circumstances of the transaction. The sponsoring registrar will be aware of the TTL and the sponsoring registrar MUST inform the registrant of the TTL when the authorization information is provided to the registrant.

4.3. Authorization Information Storage and Transport

To protect the disclosure of the authorization information, the following requirements apply:

1. The authorization information MUST be stored by the registry using a strong one-way cryptographic hash, with at least a 256-bit hash function such as SHA-256, and with a random salt.
2. An empty authorization information MUST be stored as an undefined value that is referred to as a NULL value. The representation of an NULL (undefined) value is dependent on the type of database used.
3. The authorization information MUST NOT be stored by the losing registrar.
4. The authorization information MUST only be stored by the gaining registrar as a "transient" value in support of the transfer process.
5. The plain text version of the authorization information MUST NOT be written to any logs by the registrar or the registry.
6. All communication that includes the authorization information MUST be over an encrypted channel, such as defined in [RFC5734] for EPP.
7. The registrar's interface for communicating the authorization information with the registrant MUST be over an authenticated and encrypted channel.

4.4. Authorization Information Matching

To support the authorization information TTL, as defined in Section 4.2, the authorization information must have either a set or unset state. The unset authorization information is stored with a NULL (undefined) value. Based on the requirement to store the authorization information using a strong one-way cryptographic hash, as defined in Section 4.3, a set authorization information is stored with a non-NULL hashed value. The empty authorization information is used as input in both the create command (Section 5.1) and the update command (Section 5.2) to define the unset state. The matching of the authorization information in the info command (Section 5.3) and the transfer request command (Section 5.4) is based on the following rules:

1. Any input authorization information value MUST NOT match an unset authorization information value.
2. An empty input authorization information value MUST NOT match any authorization information value.
3. A non-empty input authorization information value MUST be hashed and matched against the set authorization information value, which is stored using the same hash algorithm.

5. Create, Transfer, and Secure Authorization Information

To make the transfer process secure using secure authorization information, as defined in Section 4, the client and server need to implement steps where the authorization information is set only when a transfer is actively in process and ensure that the authorization information is stored securely and transported only over secure channels. The steps in management of the authorization information for transfers include:

1. Registrant requests to register the object with the registrar. Registrar sends the create command, with empty authorization information, to the registry, as defined in Section 5.1.
2. Registrant requests from the losing registrar the authorization information to provide to the gaining registrar.
3. Losing registrar generates a secure random authorization information value, sends it to the registry as defined in Section 5.2, and provides it to the registrant.
4. Registrant provides the authorization information value to the gaining registrar.
5. Gaining registrar optionally verifies the authorization information with the info command to the registry, as defined in Section 5.3.
6. Gaining registrar sends the transfer request with the authorization information to the registry, as defined in Section 5.4.
7. If the transfer successfully completes, the registry automatically unsets the authorization information; otherwise the losing registrar unsets the authorization information when the TTL expires, as defined in Section 5.2.

The following sections outline the practices of the EPP commands and responses between the registrar and the registry that supports secure authorization information for transfer.

5.1. Create Command

For a create command, the registry MUST allow for the passing of an empty authorization information and MAY disallow for the passing of a non-empty authorization information. By having an empty authorization information on create, the object is initially not in the transfer process. Any EPP object extension that supports setting the authorization information with a "eppcom:pwAuthInfoType" element, can have an empty authorization information passed, such as [RFC5731] and [RFC5733].

Example of passing empty authorization information in an [RFC5731] domain name 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:authInfo>
C:          <domain:pw/>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example of passing empty authorization information in an [RFC5733] contact create command.

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <contact:create
C:       xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
C:        <contact:id>sh8013</contact:id>
C:        <contact:postalInfo type="int">
C:          <contact:name>John Doe</contact:name>
C:          <contact:addr>
C:            <contact:city>Dulles</contact:city>
C:            <contact:cc>US</contact:cc>
C:          </contact:addr>
C:        </contact:postalInfo>
C:        <contact:email>jdoe@example.com</contact:email>
C:        <contact:authInfo>
C:          <contact:pw/>
C:        </contact:authInfo>
C:      </contact:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

5.2. Update Command

For an update command, the registry MUST allow for the setting and unsetting of the authorization information. The registrar sets the authorization information by first generating a strong, random authorization information value, based on Section 4.1, and setting it in the registry in the update command. The registry SHOULD validate the randomness of the authorization information based on the length and character set required by the registry. For example, a registry that requires 20 random printable ASCII characters except space (0x20), should validate that the authorization information contains at least one upper case alpha character, one lower case alpha character, and one non-alpha numeric character. If the authorization information fails the randomness validation, the registry MUST return an EPP error result code of 2202.

Often the registrar has the "clientTransferProhibited" status set, so to start the transfer process, the "clientTransferProhibited" status needs to be removed, and the strong, random authorization information value needs to be set. The registrar MUST define a time-to-live (TTL), as defined in Section 4.2, where if the TTL expires the registrar will unset the authorization information.

Example of removing the "clientTransferProhibited" status and setting the authorization information in an [RFC5731] domain name 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:rem>
C:          <domain:status s="clientTransferProhibited"/>
C:        </domain:rem>
C:        <domain:chg>
C:          <domain:authInfo>
C:            <domain:pw>LuQ7Bu@w9?%+_HK3cayg$55$LSft3MPP
C:            </domain:pw>
C:          </domain:authInfo>
C:        </domain:chg>
C:      </domain:update>
C:    </update>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C:</epp>
```

When the registrar-defined TTL expires, the sponsoring registrar cancels the transfer process by unsetting the authorization information value and may add back statuses like the "clientTransferProbited" status. Any EPP object extension that supports setting the authorization information with a "eppcom:pwAuthInfoType" element, can have an empty authorization information passed, such as [RFC5731] and [RFC5733]. Setting an empty authorization information unsets the value. [RFC5731] supports an explicit mechanism of unsetting the authorization information, by passing the <domain:null> authorization information value. The registry MUST support unsetting the authorization information by accepting an empty authorization information value and accepting an explicit unset element if it is supported by the object extension.

Example of adding the "clientTransferProhibited" status and unsetting the authorization information explicitly in an [RFC5731] domain name 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:add>
C:          <domain:status s="clientTransferProhibited"/>
C:        </domain:add>
C:        <domain:chg>
C:          <domain:authInfo>
C:            <domain:null/>
C:          </domain:authInfo>
C:        </domain:chg>
C:      </domain:update>
C:    </update>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C:</epp>
```

Example of unsetting the authorization information with an empty authorization information in an [RFC5731] domain name 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:add>
C:          <domain:status s="clientTransferProhibited"/>
C:        </domain:add>
C:        <domain:chg>
C:          <domain:authInfo>
C:            <domain:pw/>
C:          </domain:authInfo>
C:        </domain:chg>
C:      </domain:update>
C:    </update>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C:</epp>
```

Example of unsetting the authorization information with an empty authorization information in an [RFC5733] contact 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:      <contact:update
C:        xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
C:        <contact:id>sh8013</contact:id>
C:        <contact:chg>
C:          <contact:authInfo>
C:            <contact:pw/>
C:          </contact:authInfo>
C:        </contact:chg>
C:      </contact:update>
C:    </update>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C:</epp>
```

5.3. Info Command and Response

For an info command, the registry MUST allow for the passing of a non-empty authorization information for verification. The gaining registrar can pre-verify the authorization information provided by the registrant prior to submitting the transfer request with the use of the info command. The registry compares the hash of the passed authorization information with the hashed authorization information value stored for the object. When the authorization information is not set or the passed authorization information does not match the previously set value, the registry MUST return an EPP error result code of 2202 [RFC5730].

Example of passing a non-empty authorization information in an [RFC5731] domain name info command to verify the authorization information value.

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <domain:info
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:authInfo>
C:          <domain:pw>LuQ7Bu@w9?%+_HK3cayg$55$LSft3MPP
C:          </domain:pw>
C:        </domain:authInfo>
C:      </domain:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

The info response in object extensions, such as [RFC5731] and [RFC5733], MUST NOT include the optional authorization information element with a non-empty authorization value. The authorization information is stored as a hash in the registry, so returning the plain text authorization information is not possible, unless a valid plain text authorization information is passed in the info command. The registry MUST NOT return any indication of whether the authorization information is set or unset to the non-sponsoring registrar by not returning the authorization information element in the response. The registry MAY return an indication to the sponsoring registrar that the authorization information is set by using an empty authorization information value. The registry MAY return an indication to the sponsoring registrar that the authorization information is unset by not returning the authorization information element.

Example of returning an empty authorization information in an [RFC5731] domain name info response to indicate to the sponsoring registrar that the authorization information is set.

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:authInfo>
S:          <domain:pw/>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5.4. Transfer Request Command

For a Transfer Request Command, the registry MUST allow for the passing of a non-empty authorization information to authorize a transfer. The registry compares the hash of the passed authorization information with the hashed authorization information value stored for the object. When the authorization information is not set or the passed authorization information does not match the previously set value, the registry MUST return an EPP error result code of 2202 [RFC5730]. Whether the transfer occurs immediately or is pending is up to server policy. When the transfer occurs immediately, the registry MUST return the EPP success result code of 1000 and when the transfer is pending, the registry MUST return the EPP success result code of 1001. The losing registrar MUST be informed of a successful transfer request using an EPP poll message.

Example of passing a non-empty authorization information in an [RFC5731] domain name transfer request command to authorize the transfer.

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <transfer op="request">
C:      <domain:transfer
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example1.com</domain:name>
C:        <domain:authInfo>
C:          <domain:pw>LuQ7Bu@w9?%+_HK3cayg$55$LSft3MPP
C:          </domain:pw>
C:        </domain:authInfo>
C:      </domain:transfer>
C:    </transfer>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Upon successful completion of the transfer, the registry MUST automatically unset the authorization information. If the transfer request is not submitted within the time-to-live (TTL) (Section 4.2) or the transfer is cancelled or rejected, the registrar MUST unset the authorization information as defined in Section 5.2.

6. Transition Considerations

The goal of the transition considerations to the practice defined in this document, referred to as the Secure Authorization Information Model, is to minimize the impact to the registrars by supporting incremental steps of adoption. The transtion steps are dependent on the starting point of the registry. Registries may have different starting points, since some of the elements of the Secure Authorization Information Model may have already been implemented. The considerations assume a starting point, referred to as the Classic Authorization Information Model, that have the following steps in the management of the authorization information for transfers:

1. Registrant requests to register the object with the registrar. Registrar sends the create command, with a non-empty authorization information, to the registry. The registry stores the authorization information as an encrypted value and requires a non-empty authorization information for the life of the object. The registrar may store the long-lived authorization information.
2. At the time of transfer, Registrant requests from the losing registrar the authorization information to provide to the gaining registrar.
3. Losing registrar retrieves the stored authorization information locally or queries the registry for authorization information using the info command, and provides it to the registrant. If the registry is queried, the authorization information is decrypted and the plain text authorization information is returned in the info response to the registrar.
4. Registrant provides the authorization information value to the gaining registrar.
5. Gaining registrar optionally verifies the authorization information with the info command to the registry, by passing the authorization information in the info command to the registry.
6. Gaining registrar sends the transfer request with the authorization information to the registry. The registry will decrypt the stored authorization information to compare to the passed authorization information.
7. If the transfer successfully completes, the authorization information is not touched by the registry and may be updated by the gaining registrar using the update command. If the transfer is cancelled or rejected, the losing registrar may reset the authorization information using the update command.

The gaps between the Classic Authorization Information Model and the Secure Authorization Information Model include:

1. Registry requirement for a non-empty authorization information on create and for the life of the object versus the authorization information not being set on create and only being set when a transfer is in process.
2. Registry not allowing the authorization information to be unset versus supporting the authorization to be unset in the update command.
3. Registry storing the authorization information as an encrypted value versus as a hashed value.
4. Registry support for returning the authorization information versus not returning the authorization information in the info response.
5. Registry not touching the authorization information versus the registry automatically unsetting the authorization information upon a successful transfer.
6. Registry may validate a shorter authorization information value using password complexity rules versus validating the randomness of a longer authorization information value that meets the required bits of entropy.

The transition can be handled in the three phases defined in the sub-sections Section 6.1, Section 6.2, Section 6.3.

6.1. Transition Phase 1 - Features

The goal of the "Transition Phase 1 - Features" is to implement the needed features in EPP so that the registrar can optionally implement the Secure Authorization Information Model. The features to implement are broken out by the command and responses below:

Create Command: Change the create command to make the authorization information optional, by allowing both a non-empty value and an empty value. This enables a registrar to optionally create objects without an authorization information value, as defined in Section 5.1.

Update Command: Change the update command to allow unsetting the authorization information, as defined in Section 5.2. This enables the registrar to optionally unset the authorization information when the TTL expires or when the transfer is cancelled or rejected.

Transfer Approve Command and Transfer Auto-Approve: Change the

transfer approve command and the transfer auto-approve to automatically unset the authorization information. This sets the default state of the object to not have the authorization information set. The registrar implementing the Secure Authorization Information Model will not set the authorization information for an inbound transfer and the registrar implementing the Classic Authorization Information Model will set the new authorization information upon the successful transfer.

Info Response: Change the info command to not return the authorization information in the info response, as defined in Section 5.3. This sets up the implementation of "Transition Phase 2 - Storage", since the dependency in returning the authorization information in the info response will be removed. This feature is the only one that is not an optional change to the registrar.

Info Command and Transfer Request: Change the info command and the transfer request to ensure that a registrar cannot get an indication that the authorization information is set or not set by returning the EPP error result code of 2202 when comparing a passed authorization to a non-matching set authorization information value or an unset value.

6.2. Transition Phase 2 - Storage

The goal of the "Transition Phase 2 - Storage" is to transition the registry to use hashed authorization information instead of encrypted authorization information. There is no direct impact to the registrars, since the only visible indication that the authorization information has been hashed is by not returning the set authorization information in the info response, which is addressed in Transition Phase 1 - Features (Section 6.1). There are three steps to transition the authorization information storage, which includes:

Hash New Authorization Information Values: Change the create command and the update command to hash instead of encyrpting the authorization information.

Supporting Comparing Against Encrypted and Hashed Authorization Information: Change the info command and the transfer request command to be able to compare a passed authorization information value with either a hashed or encyrpted authorization information value.

Hash Existing Encrypted Authorization Information Values: Convert the encrypted authorization information values stored in the registry database to hashed values. The update is not a visible change to the registrar. The conversion can be done over a period of time depending on registry policy.

6.3. Transition Phase 3 - Enforcement

The goal of the "Transition Phase 3 - Enforcement" is to complete the implementation of the "Secure Authorization Information Model", by enforcing the following:

Disallow Authorization Information on Create Command: Change the create command to not allow for the passing of a non-empty authorization information value.

Validate the Strong Random Authorization Information: Change the validation of the authorization information in the update command to ensure at least 128 bits of entropy.

7. IANA Considerations

7.1. XML Namespace

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

Registration request for the secure authorization information for transfer namespace:

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

7.2. EPP Extension Registry

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

Name of Extension: "Extensible Provisioning Protocol (EPP) Secure Authorization Information for Transfer"

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

8. 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".

8.1. Verisign EPP SDK

Organization: Verisign Inc.

Name: Verisign EPP SDK

Description: The Verisign EPP SDK includes both a full client implementation and a full server stub implementation of draft-ietf-regext-secure-authinfo-transfer.

Level of maturity: Development

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

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

8.2. 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: Authorization Information is "write only" in that the registrars can set the Authorization Information, but not get the Authorization Information in the Info Response.

Licensing: Proprietary In-House software

Contact: epp@centralnic.com

URL: https://www.centralnic.com

9. Security Considerations

Section 4.1 defines the use a secure random value for the generation of the authorization information. The server SHOULD define policy related to the length and set of characters that are included in the randomization to target the desired entropy level, with the recommendation of at least 49 bits for entropy. The authorization information server policy is communicated to the client using an out-of-band process. The client SHOULD choose a length and set of characters that results in entropy that meets or exceeds the server policy. A random number generator (RNG) is preferable over the use of a pseudorandom number generator (PRNG) when creating the authorization information value.

Section 4.2 defines the use of an authorization information Time-To-Live (TTL). The registrar SHOULD only set the authorization information during the transfer process by the server support for setting and unsetting the authorization information. The TTL value is up to registrar policy and the sponsoring registrar MUST inform the registrant of the TTL when providing the authorization information to the registrant.

Section 4.3 defines the storage and transport of authorization information. The losing registrar MUST NOT store the authorization information and the gaining registrar MUST only store the

authorization information as a "transient" value during the transfer process, where the authorization information MUST NOT be stored after the end of the transfer process. The registry MUST store the authorization information using a one-way cryptographic hash of at least 256 bits and with a random salt. All communication that includes the authorization information MUST be over an encrypted channel. The plain text authorization information MUST NOT be written to any logs by the registrar or the registry.

Section 4.4 defines the matching of the authorization information values. The registry stores an unset authorization information as a NULL (undefined) value to ensure that an empty input authorization information never matches it. The method used to define a NULL (undefined) value is database specific.

10. Acknowledgements

The authors wish to thank the following persons for their feedback and suggestions: Michael Bauland, Martin Casanova, Scott Hollenbeck, Jody Kolker, Patrick Mevzek, Matthew Pozun, Srikanth Veeramachaneni, and Ulrich Wisser.

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>.

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

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[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>.

[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>.

[RFC5734] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Transport over TCP", STD 69, RFC 5734, DOI 10.17487/RFC5734, August 2009, <https://www.rfc-editor.org/info/rfc5734>.

[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>.

[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>.

[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>.

Appendix A. Change History

A.1. Change from 00 to 01

1. Filled in the "Implementation Status" section with the inclusion of the "Verisign EPP SDK" and "RegistryEngine EPP Service" implementations.
2. Made small wording corrections based on private feedback.
3. Added content to the "Acknowledgements" section.

A.2. Change from 01 to 02

1. Revised the language used for the storage of the authorization information based on the feedback from Patrick Mevzek and Jody Kolker.

A.3. Change from 02 to 03

1. Updates based on the feedback from the interim REGEXT meeting held at ICANN-66:
    1. Section 3.3, include a reference to the hash algorithm to use. Broke the requirements into a list and included a the reference the text ', with at least a 256-bit hash function, such as SHA-256'.

2. Add a Transition Considerations section to cover the transition from the classic authorization information security model in the EPP RFCs to the model defined in the document.
3. Add a statement to the Introduction that elements of the practice can be used for purposes other than transfer, but with a caveat.

2. Updates based on the review by Michael Bauland, that include:
   1. In section 2, change 'there are three actors' to 'there are three types of actors' to cover the case with transfers that has two registrar actors (losing and gaining).
   2. In section 3.1, change the equations equals to be approximately equal by using '=~' instead of '=', where applicable.
   3. In section 3.3, change 'MUST be over an encrypted channel, such as [RFC5734]'' to 'MUST be over an encrypted channel, such as defined in [RFC5734]''.
   4. In section 4.1, remove the optional RFC 5733 elements from the contact create, which includes the <contact:voice>, <contact:fax>, <contact:disclose>, <contact:org>, <contact:street>, <contact:sp>, and <contact:cc> elements.
   5. In section 4.2, changed 'Example of unsetting the authorization information explicitly in an [RFC5731] domain name update command.' to 'Example of adding the "clientTransferProhibited" status and unsetting the authorization information explicitly in an [RFC5731] domain name update command.'
   6. In section 4.3, cover a corner case of the ability to return the authorization information when it's passed in the info command.
   7. In section 4.4, change 'If the transfer does not complete within the time-to-live (TTL)' to 'If the transfer is not initiated within the time-to-live (TTL)', since the TTL is the time between setting the authorization information and when it's successfully used in a transfer request. Added the case of unsetting the authorization information when the transfer is cancelled or rejected.
3. Updates based on the authorization information messages by Martin Casanova on the REGEXT mailing list, that include:
   1. Added section 3.4 'Authorization Information Matching' to clarify how the authorization information is matched, when there is set and unset authorization information in the database and empty and non-empty authorization information passed in the info and transfer commands.
   2. Added support for signaling that the authorization information is set or unset to the sponsoring registrar with the inclusion of an empty authorization information element in the response to indicate that the authorization

information is set and the exclusion of the authorization information element in the response to indicate that the authorization information is unset.

4. Made the capitalization of command and response references consistent by uppercasing section and item titles and lowercasing references elsewhere.

A.4. Change from 03 to REGEXT 00

1. Changed to regext working group draft by changing draft-gould-regext-secure-authinfo-transfer to draft-ietf-regext-secure-authinfo-transfer.

A.5. Change from REGEXT 00 to REGEXT 01

1. Added the "Signaling Client and Server Support" section to describe the mechanism to signal support for the BCP by the client and the server.
2. Added the "IANA Considerations" section with the registration of the secure authorization for transfer XML namespace and the registration of the EPP Best Current Practice (BCP) in the EPP Extension Registry.

A.6. Change from REGEXT 01 to REGEXT 02

1. Added inclusion of random salt for the hashed authorization information, based on feedback from Ulrich Wisser.
2. Added clarification that the representation of a NULL (undefined) value is dependent on the type of database, based on feedback from Patrick Mevzek.
3. Filled in the Security Considerations section.

A.7. Change from REGEXT 02 to REGEXT 03

1. Updated the XML namespace to urn:ietf:params:xml:ns:epp:secure-authinfo-transfer-1.0, which removed bcp from the namespace and bumped the version from 0.1 and 1.0. Inclusion of bcp in the XML namespace was discussed at the REGEXT interim meeting.
2. Replaced Auhtorization with Authorization based on a review by Jody Kolker.

A.8. Change from REGEXT 03 to REGEXT 04

1. Converted from xml2rfc v2 to v3.
2. Updated Acknowledgements to match the approach taken by the RFC Editor with draft-ietf-regext-login-security.
3. Changed from Best Current Practice (BCP) to Standards Track based on mailing list discussion.

Authors' Addresses

James Gould
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States of America

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

Richard Wilhelm
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States of America

Email: rwilhelm@verisign.com
URI: http://www.verisign.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: May 23, 2021

G. Lozano
ICANN

Nov 19, 2020

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on May 23, 2021.

Copyright Notice

Table of Contents

1. Introduction

Domain Name Registries (DNRs) may operate in special modes for certain periods of time enabling trademark holders to protect their rights during the introduction of a Top Level Domain (TLD).

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

- Sunrise Period, the Sunrise Period allows trademark holders an advance opportunity to register domain names corresponding to their marks before names are generally available to the public.

- Trademark Claims Period, the Trademark Claims Period follows the Sunrise Period and runs for at least the first 90 days of an initial operating period of general registration. During the Trademark Claims Period, anyone attempting to register a domain name matching a mark that is recorded in the ICANN Trademark Clearinghouse (TMCH) will receive a notification displaying the relevant mark information.

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

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

2. Terminology

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

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

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

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

3. Glossary

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

- Backend Registry Operator: Entity that manages (a part of) the technical infrastructure for a Registry Operator. The Registry Operator may also be the Backend Registry Operator.

- CA: Certificate Authority, see [RFC5280].

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

- CRC32, Cyclic Redundancy Check: algorithm used in the ISO 3309 standard and in section 8.1.1.6.2 of ITU-T recommendation V.42.

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

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

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

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

- DNROID, DN Repository Object IDentifier: an identifier assigned by the Registry to each DN object that unequivocally identifies said DN object. For example, if a new DN object is created for a name that existed in the past, the DN objects will have different DNROIDs.

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

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

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

- Effective allocation: A DN is considered effectively allocated when the DN object for the DN has been created in the SRS of the Registry and has been assigned to the effective user. A DN object in status "pendingCreate" or any other status that precedes the first time a DN is assigned to an end-user is not considered an effective allocation. A DN object created internally by the

Registry for subsequent delegation to another Registrant is not considered an effective allocation.

- EPP: The Extensible Provisioning Protocol, see definition of EPP in [RFC8499].
- FQDN: Fully-Qualified Domain Name, see definition of FQDN in [RFC8499].
- HTTP: Hypertext Transfer Protocol, see [RFC7230] and [RFC7231].
- HTTPS: HTTP over TLS (Transport Layer Security), [RFC2818].
- IDN: Internationalized Domain Name, see definition of IDN in [RFC8499].
- Lookup Key: A random string of up to 51 chars from the set [a-zA-Z0-9/] to be used as the lookup key by Registrars to obtain the TCN using the CNIS. Lookup Keys are unique and are related to one DNL only.
- LORDN, List of Registered Domain Names: This is the list of effectively allocated DNs matching a DNL of a PRM. Registries will upload this list to the TMDB (during the NORDN process).
- Matching Rules: Some trademarks entitled to inclusion in the TMDB include characters that are impermissible in the domain name system (DNS) as a DNL. The TMV changes (using the ICANN TMCH Matching Rules [MatchingRules]) certain DNS-impermissible characters in a trademark into DNS-permissible equivalent characters
- NORDN, Notification of Registered Domain Names: The process by which Registries upload their recent LORDN to the TMDB.
- PGP: Pretty Good Privacy, see [RFC4880]
- PKI: Public Key Infrastructure, see [RFC5280].
- PRM, Pre-registered mark: Mark that has been pre-registered with the ICANN TMCH.
- QLP Period, Qualified Launch Program Period: During this optional period, a special process applies to DNs matching the Sunrise List (SURL) and/or the DNL List, to ensure that TMHs are informed of a DN matching their PRM.
- Registrant: see definition of Registrant in [RFC8499].

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

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

- SMD, Signed Mark Data: A cryptographically signed token issued by the TMV to the TMH to be used in the Sunrise Period to apply for a DN that matches a DNL of a PRM; see also [RFC7848]. An SMD generated by an ICANN-approved trademark validator (TMV) contains both the signed token and the TMV's PKIX certificate.

- SMD File: A file containing the SMD (see above) and some human readable data. The latter is usually ignored in the processing of the SMD File. See also Section 6.4.

- SMD Revocation List: The SMD Revocation List is used by Registries (and optionally by Registrars) during the Sunrise Period to ensure that an SMD is still valid (i.e. not revoked). The SMD Revocation List has a similar function as CRLs used in PKI.

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

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

- Sunrise Period: During this period DNs matching a DNL of a PRM can be exclusively obtained by the respective TMHs. For DNs matching a PRM, a special process applies to ensure that TMHs are informed on the effective allocation of a DN matching their PRM.

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

- ICANN TMCH: a central repository for information to be authenticated, stored, and disseminated, pertaining to the rights of TMHs. The ICANN TMCH is split into two functions TMV and TMDB (see below). There could be several entities performing the TMV function, but only one entity performing the TMDB function.

- ICANN TMCH-CA: The Certificate Authority (CA) for the ICANN TMCH. This CA is operated by ICANN. The public key for this CA is the trust anchor used to validate the identity of each TMV.

- TMDB, Trademark Clearinghouse Database: Serves as a database of the ICANN TMCH to provide information to the gTLD Registries and Registrars to support Sunrise or Trademark Claims services. There

is only one TMDB in the ICANN TMCH that concentrates the information about the "verified" Trademark records from the TMVs.

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

- TMV, Trademark Validator, Trademark validation organization: An entity authorized by ICANN to authenticate and validate registrations in the TMDB ensuring the marks qualify as registered or are court-validated marks or marks that are protected by statute or treaty. This entity would also be asked to ensure that proof of use of marks is provided, which can be demonstrated by furnishing a signed declaration and one specimen of current use.

- Trademark, mark: Marks are used to claim exclusive properties of products or services. A mark is typically a name, word, phrase, logo, symbol, design, image, or a combination of these elements. For the scope of this document only textual marks are relevant.

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

- TCN, Trademark Claims Notice, Claims Notice, Trademark Notice: A Trademark Claims Notice consist of one or more Trademark Claims and are provided to prospective Registrants of DNs.

- TCNID, Trademark Claims Notice Identifier: An element of the Trademark Claims Notice (see above), identifying said TCN. The Trademark Claims Notice Identifier is specified in the element <tmNotice:id>.

- Trademark Claims Period: During this period, a special process applies to DNs matching the DNL List, to ensure that TMHs are informed of a DN matching their PRM. For DNs matching the DNL List, Registrars show a TCN to prospective Registrants that has to be acknowledged before effective allocation of the DN.

- UTC: Coordinated Universal Time, as maintained by the Bureau International des Poids et Mesures (BIPM); see also [RFC3339].

4. Architecture

4.1. Sunrise Period

Architecture of the Sunrise Period

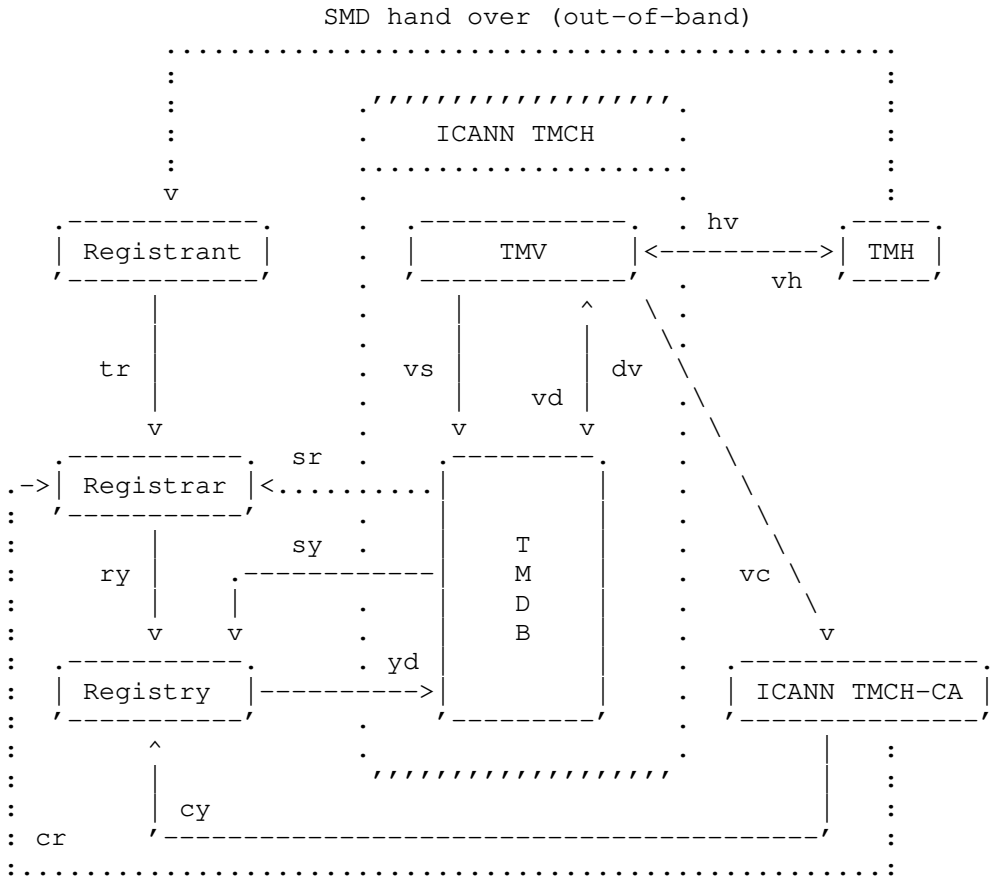

Figure 1

4.2. Trademark Claims Period

Architecture of the Trademark Claims Period

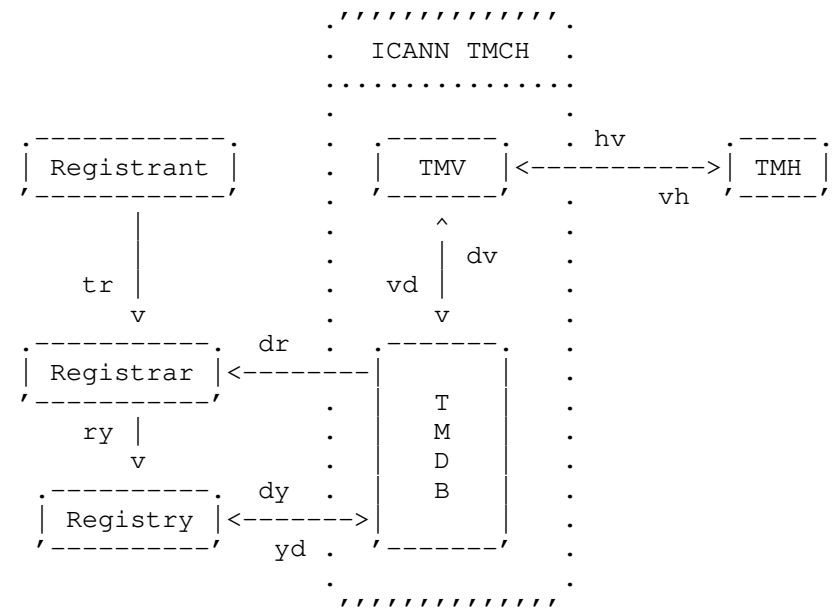

Figure 2

4.3. Interfaces

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

4.3.1. hv

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

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

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

4.3.2. vd

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

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

4.3.3. dy

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

Not relevant during the Sunrise Period.

4.3.4. tr

The Registrant communicates with the Registrar via the tr interface.

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

4.3.5. ry

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

4.3.6. dr

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

Not relevant during the Sunrise Period.

4.3.7. yd

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

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

The protocol used on the yd interface is HTTPS.

4.3.8. dv

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

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

4.3.9. vh

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

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

4.3.10. vs

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

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

Not relevant during the Trademark Claims Period.

4.3.11. sy

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

Not relevant during the Trademark Claims Period.

4.3.12. sr

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

Not relevant during the Trademark Claims Period.

4.3.13. vc

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

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

Not relevant during the Trademark Claims Period.

4.3.14. cy

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

Not relevant during the Trademark Claims Period.

4.3.15. cr

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

Not relevant during the Trademark Claims Period.

5. Process Descriptions

5.1. Bootstrapping

5.1.1. Bootstrapping for Registries

5.1.1.1. Credentials

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

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

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

5.1.1.2. IP Addresses for Access Control

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

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

This access restriction MAY be applied by the TMDB in addition to HTTP Basic access authentication (see [RFC7235]). For credentials to be used, see Section 5.1.1.1.

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

5.1.1.3. ICANN TMCH Trust Anchor

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

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

5.1.1.4. TMDB PGP Key

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

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

5.1.2. Bootstrapping for Registrars

5.1.2.1. Credentials

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

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

5.1.2.2. IP Addresses for Access Control

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

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

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

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

5.1.2.3. ICANN TMCH Trust Anchor

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

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

5.1.2.4. TMDB PGP Key

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

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

5.2. Sunrise Period

5.2.1. Domain Name registration

Domain Name registration during the Sunrise Period

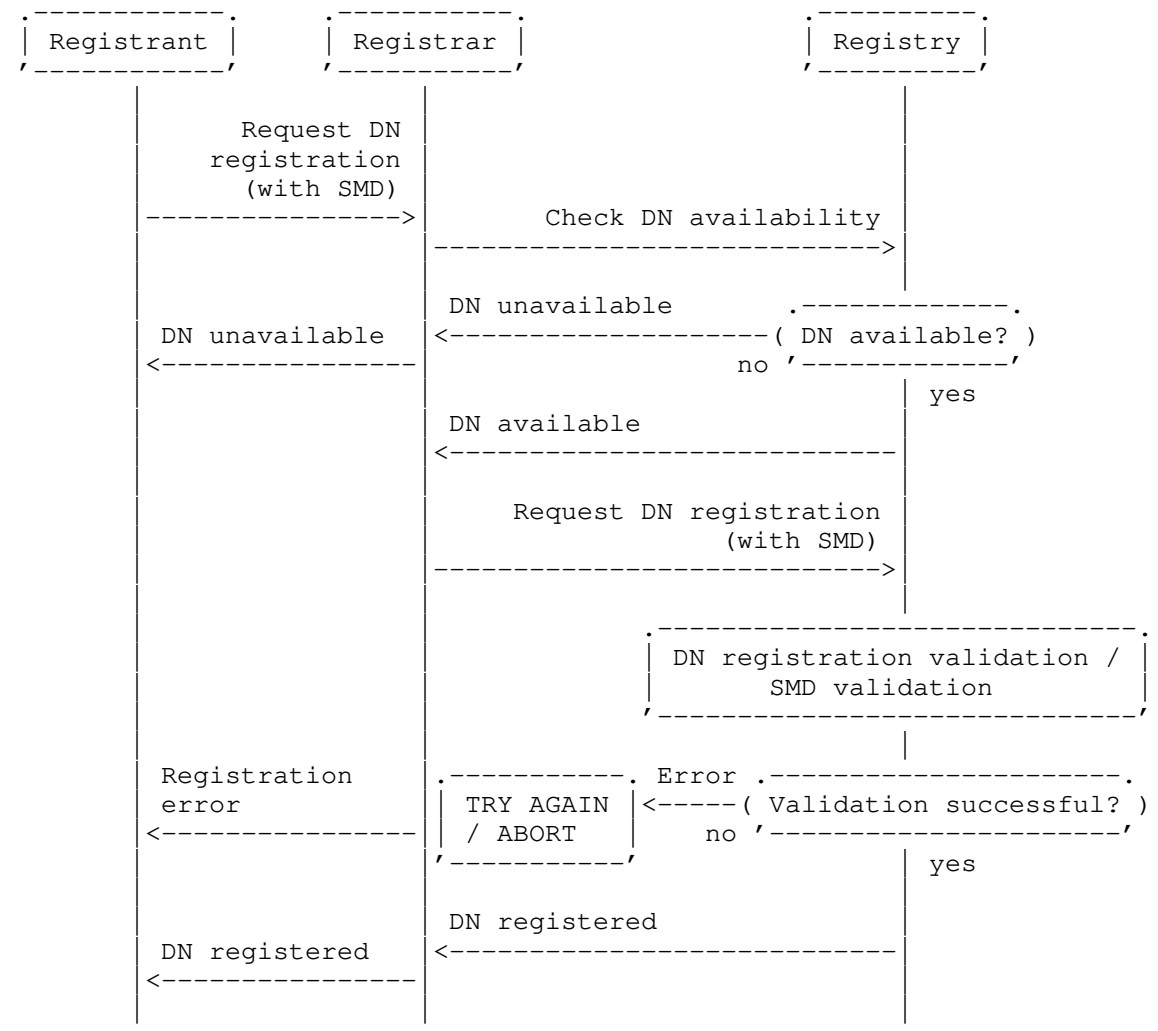

Figure 3

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

5.2.2. Sunrise Domain Name registration by Registries

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

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

Performing the minimum set of checks Registries MUST verify that:

1. An SMD has been received from the Registrar along with the DN registration.

2. The certificate of the TMV has been correctly signed by the ICANN TMCH-CA. (The certificate of the TMV is contained within the SMD.)

3. The datetime when the validation is done is within the validity period of the TMV certificate.

4. The certificate of the TMV is not listed in the TMV CRL file specified in the CRL distribution point of the TMV certificate.

5. The signature of the SMD (signed with the TMV certificate) is valid.

6. The datetime when the validation is done is within the validity period of the SMD based on <smd:notBefore> and <smd:notAfter> elements.

7. The SMD has not been revoked, i.e., is not contained in the SMD Revocation List.

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

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

5.2.3. TMDB Sunrise Services for Registries

5.2.3.1. SMD Revocation List

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

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

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

Update of the SMD Revocation List

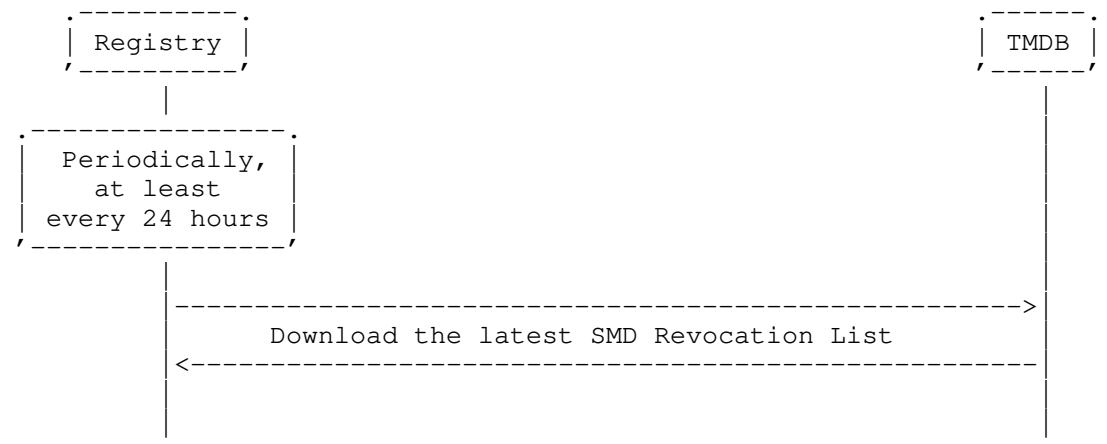

Figure 4

5.2.3.2. TMV Certificate Revocation List (CRL)

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

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

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

Update of the TMV CRL file

Figure 5

5.2.3.3. Notice of Registered Domain Names (NORN)

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

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

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

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

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

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

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

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

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

Notification of Registered Domain Name

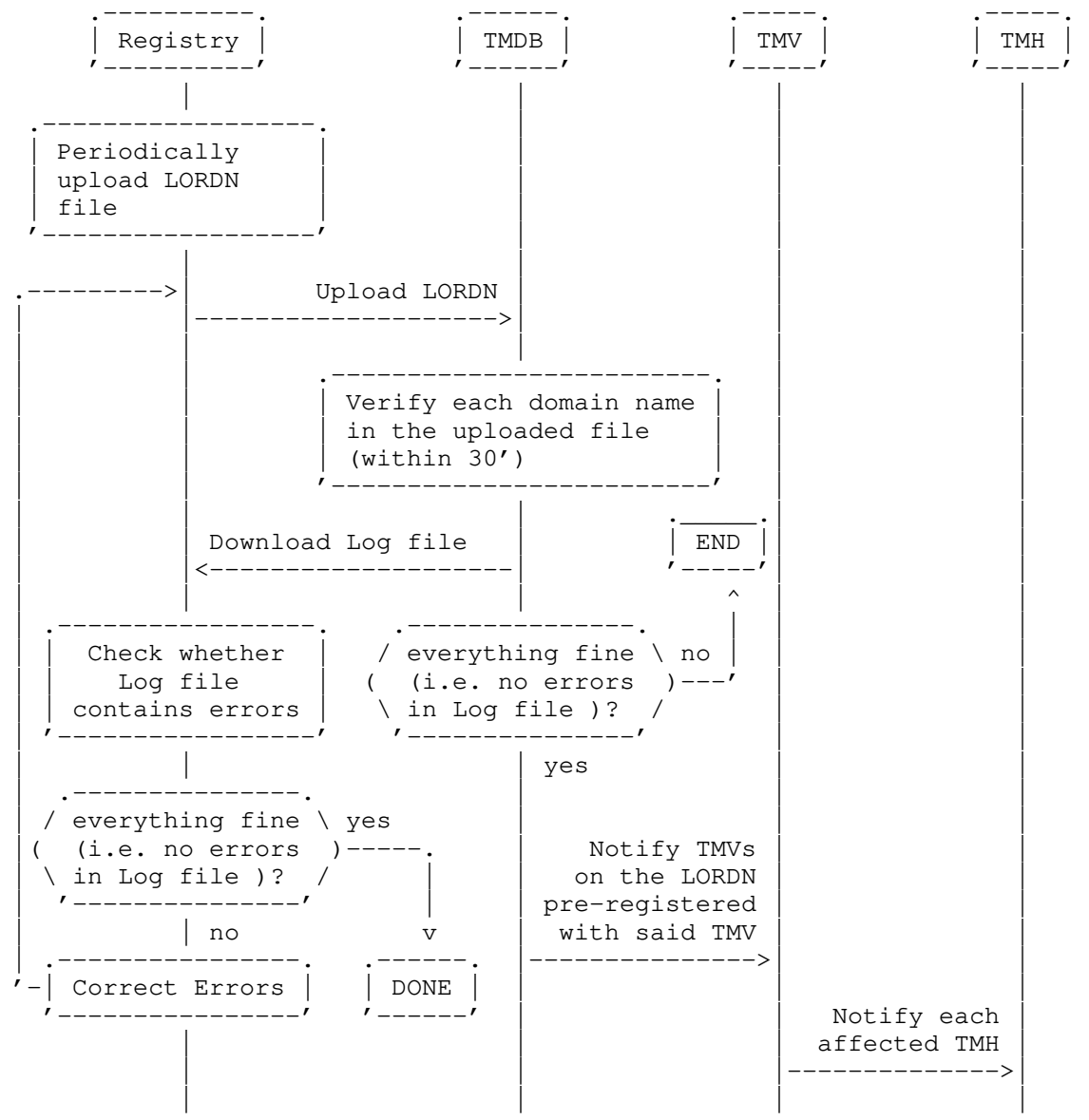

Figure 6

The format used for the LORDN is described in Section 6.3

5.2.4. Sunrise Domain Name registration by Registrars

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

5.2.5. TMDB Sunrise Services for Registrars

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

5.3. Trademark Claims Period

5.3.1. Domain Registration

Domain Name registration during the Trademark Claims Period

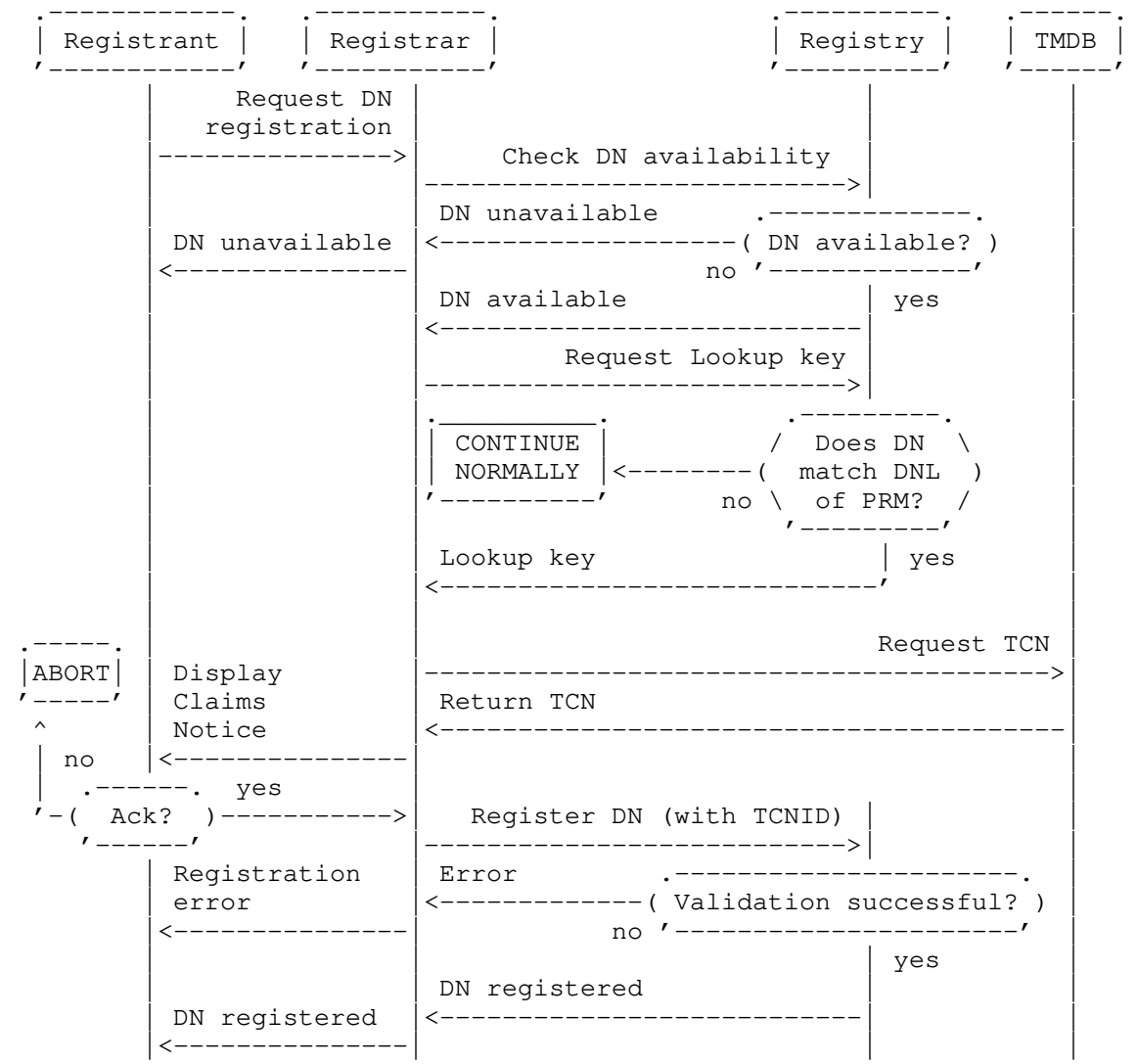

Figure 7

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

5.3.2. Trademark Claims Domain Name registration by Registries

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

- Registries MUST provide Registrars (over the ry interface, Section 4.3.5) the Lookup Key used to retrieve the TCNs for DNs that match the DNL List.

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

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

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

- Performing the minimum set of checks Registries MUST verify that:

  1. The TCNID (<tmNotice:id>), expiration datetime (<tmNotice:notAfter>) and acceptance datetime of the TCN, have been received from the Registrar along with the DN registration.

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

  2. The TCN has not expired (according to the expiration datetime sent by the Registrar).

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

4. Using the leftmost DNL of the DN being effectively allocated, the expiration datetime provided by the Registrar, and the TMDB Notice Identifier extracted from the TCNID provided by the Registrar compute the TCN Checksum. Verify that the computed TCN Checksum match the TCN Checksum present in the TCNID. For example, if the DN "xn--mgbachtv.xn--mgbh0fb" is being effectively allocated, the leftmost DNL would be "xn--mgbachtv".

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

5.3.3. TMBD Trademark Claims Services for Registries

5.3.3.1. Domain Name Label (DNL) List

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

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

Update of the DNL List

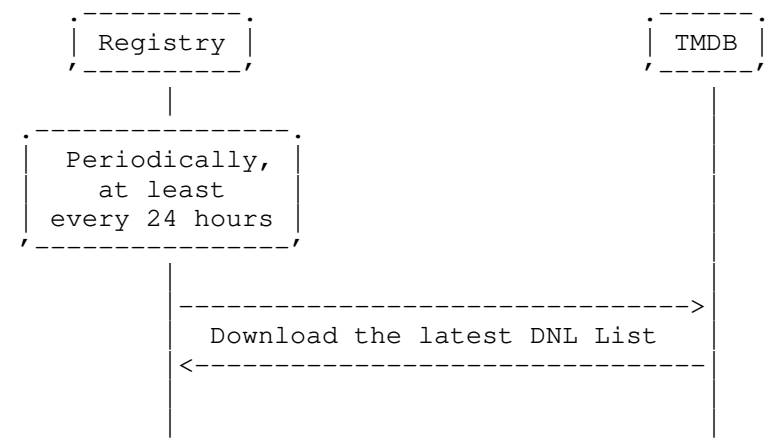

Figure 8

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

5.3.3.2. Notice of Registered Domain Names (NORN)

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

5.3.4. Trademark Claims Domain Name registration by Registrars

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

1. Use the Lookup Key received from the Registry to obtain the TCN from the TMDB using the dr interface (Section 4.3.6) Registrars MUST only query for the Lookup Key of a DN that is available for registration.

2. Present the TCN to the Registrant as described in Exhibit A, [RPM-Requirements].

3. Ask Registrant for acknowledgement, i.e. the Registrant MUST consent with the TCN, before any further processing. (The transmission of a TCNID to the Registry over the ry interface, Section 4.3.5 implies that the Registrant has expressed his/her consent with the TCN.)

4. Perform the minimum set of checks for verifying DN registrations. If any of these checks fails the Registrar MUST abort the DN registration. Each of these checks MUST be performed before the registration is sent to the Registry. Performing the minimum set of checks Registrars MUST verify that:

    1. The datetime when the validation is done is within the TCN validity based on the <tmNotice:notBefore> and <tmNotice:notAfter> elements.

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

    3. The Registrant has acknowledged (expressed his/her consent with) the TCN.

5. Record the date and time when the registrant acknowledged the TCN.

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

    * TCNID (<tmNotice:id>)

    * Expiration date of the TCN (<tmNotice:notAfter>)

    * Acceptance datetime of the TCN.

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

Registrars SHOULD implement a cache of TCNs to minimize the number of queries sent to the TMDB. A cached TCN MUST be removed from the cache after the expiration date of the TCN as defined by <tmNotice:notAfter>.

The TMDB MAY implement rate-limiting as one of the protection mechanisms to mitigate the risk of performance degradation.

5.3.5. TMBD Trademark Claims Services for Registrars

5.3.5.1. Claims Notice Information Service (CNIS)

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

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

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

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

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

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

The TLS certificate (HTTPS) used on the dr interface (Section 4.3.6) MUST be signed by a well-know public CA. Registrars MUST perform the Certification Path Validation described in Section 6 of [RFC5280]. Registrars will be authenticated in the dr interface using HTTP Basic access authentication. The dr (Section 4.3.6) interface MUST support HTTPS keep-alive and MUST maintain the connection for up to 30 minutes.

5.4. Qualified Launch Program (QLP) Period

5.4.1. Domain Registration

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

The effective allocation scenarios are:

- If the leftmost DNL of the DN being effectively allocated (QLP Name in this section) matches a DNL in the SURL, and an SMD is provided, then Registries MUST provide Sunrise Services (see Section 5.2) and the DN MUST be reported in a Sunrise LORDN file during the QLP Period. For example, if the DN "xn--mgbachtv.xn--mgbh0fb" is being effectively allocated, the leftmost DNL would be "xn--mgbachtv".

- If the QLP Name matches a DNL in the SURL but does not match a DNL in the DNL List, and an SMD is NOT provided (see section 2.2 of [QLP-Addendum]), then the DN MUST be reported in a Sunrise LORDN file using the special SMD-id "99999-99999" during the QLP Period.

- If the QLP Name matches a DNL in the SURL and also matches a DNL in the DNL List, and an SMD is NOT provided (see section 2.2 of [QLP-Addendum]), then Registries MUST provide Trademark Claims services (see Section 5.3) and the DN MUST be reported in a Trademark Claims LORDN file during the QLP Period.

- If the QLP Name matches a DNL in the DNL List but does not match a DNL in the SURL, then Registries MUST provide Trademark Claims services (see Section 5.2) and the DN MUST be reported in a Trademark Claims LORDN file during the QLP Period.

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

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

QLP Effective Allocation Scenarios

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

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

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

- SMD Revocation List (see Section 5.2.3.1)

o CNIS (see Section 5.3.5.1)

5.4.2. TMBD QLP Services for Registries

5.4.2.1. Sunrise List (SURL)

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

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

Update of the SURL

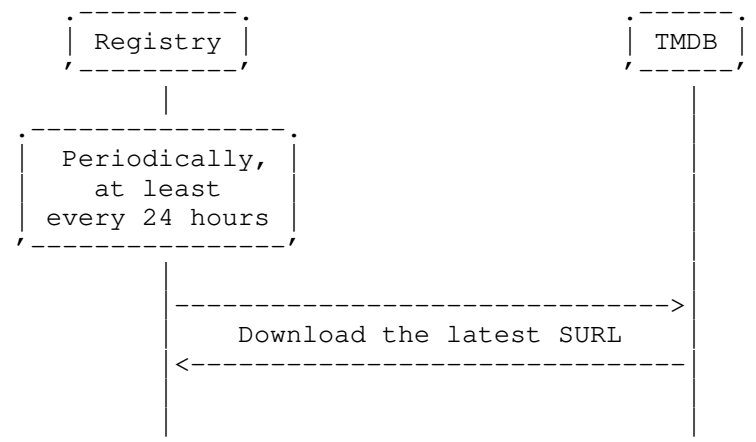

Figure 9

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

6. Data Format Descriptions

6.1. Domain Name Label (DNL) List

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

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

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

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

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

  Where:

  + <version>, version of the file, this field MUST be 1.

  + <DNL List creation datetime>, date and time in UTC that the DNL List was created.

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

  With the header names as follows:

  DNL,lookup-key,insertion-datetime

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

  Where:

  + <DNL>, a Domain Name Label covered by a PRM.

  + <lookup key>, lookup key that the Registry MUST provide to the Registrar. The lookup key has the following format: <YYYY><MM><DD><vv>/<X>/<X>/<X>/<Random bits><Sequential number>, where:

    - YYYY: year that the TCN was generated.

    - MM: zero-padded month that the TCN was generated.

    - DD: zero-padded day that the TCN was generated.

    - vv: version of the TCN, possible values are 00 and 01.

    - X: one hex character. This is the first, second and third hex character of encoding the <Random bits> in base16 as specified in [RFC4648].

- Random bits: 144 random bits encoded in base64url as specified in [RFC4648].

- Sequential number: zero-padded natural number in the range 0000000001 to 2147483647.

+ <DNL insertion datetime>, datetime in UTC that the DNL was first inserted into the DNL List. The possible two values of time for inserting a DNL to the DNL List are 00:00:00 and 12:00:00 UTC.

Example of a DNL List

```
1,2012-08-16T00:00:00.0Z
DNL,lookup-key,insertion-datetime
example,2013041500/2/6/9/rJ1NrDO92vDsAzf7EQzgjX4R0000000001,\
   2010-07-14T00:00:00.0Z
another-example,2013041500/6/A/5/alJAqG2vI2BmCv5PfUvuDkf40000000002,\
   2012-08-16T00:00:00.0Z
anotherexample,2013041500/A/C/7/rHdC4wnrWRvPY6nneCVtQhFj0000000003,\
   2011-08-16T12:00:00.0Z
```

Figure 10

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

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

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

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

6.2. SMD Revocation List

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

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

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

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

  Where:

  + <version>, version of the file, this field MUST be 1.

  + <SMD Revocation List creation datetime>, datetime in UTC that the SMD Revocation List was created.

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

  With the header names as follows:

  smd-id,insertion-datetime

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

  Where:

  + <smd-id>, identifier of the SMD that was revoked.

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

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

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

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

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

Example of an SMD Revocation List

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

Figure 11

6.3. List of Registered Domain Names (LORDN) file

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

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

- Sunrise LORDN file:

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

- Trademark Claims LORDN file:

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

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

- Sunrise LORDN file (during QLP Period):

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

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

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

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

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

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

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

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

  ```
  202 Accepted

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

- The interface provides a HTTP/400 if the request is incorrect or the syntax of the LORDN file is incorrect. The TMDB MUST return a human readable message in the HTTP Entity-body regarding the incorrect syntax of the LORDN file.

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

- The TMDB MUST return a HTTP/404 status code when trying to upload a LORDN file using the https://<tmdb-domain-name>/LORDN/<TLD>/sunrise/qlp or https://<tmdb-domain-name>/LORDN/<TLD>/claims/qlp interface outside of a QLP Period plus 26 hours.

- The interface provides a HTTP/500 status code if the system is experiencing a general failure.

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

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

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

- For Sunrise Period:
  - first line: <version>,<LORDN creation datetime>,<Number of DN Lines>

    Where:

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

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

    With the header names as follows:

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

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

    Where:

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

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

Example of a Sunrise LORDN file

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

Figure 12

- For Trademark Claims Period:
  - first line: <version>,<LORDN creation datetime>,<Number of DN Lines>

    Where:

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

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

    With the header names as follows:

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

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

    Where:

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

- <DN registered>, DN that was effectively allocated. For IDNs, the A-label form is used.

- <TCNID>, Trademark Claims Notice Identifier as specified in <tmNotice:id>.

- <IANA Registrar ID>, IANA Registrar ID.

- <datetime of registration>, date and time in UTC that the domain was effectively allocated.

- <datetime of acceptance of the TCN>, date and time in UTC that the TCN was acknowledged.

- OPTIONAL <datetime of application creation>, date and time in UTC that the application was created. The <datetime of application creation> MUST be provided in case of a DN effective allocation based on an asynchronous registration (e.g., when using auctions).

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

Example of a Trademark Claims LORDN file

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

Figure 13

6.3.1. LORDN Log file

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

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

- Sunrise LORDN Log file:

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

- Trademark Claims LORDN Log file:

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

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

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

- The interface provides a HTTP/200 status code if the interface was able to provide the LORDN Log file. The LORDN Log file is contained in the HTTP Entity-body.

- The interface provides a HTTP/204 status code if the LORDN Transaction Identifier is correct, but the server has not finalized processing the LORDN file.

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

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

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

- The interface provides a HTTP/500 status code if the system is experiencing a general failure.

For example, to obtain the LORDN Log file in case of a Sunrise LORDN file with LORDN Transaction Identifier 0000000000000000001 and TLD "example" the URI would be:

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

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

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

  Where:

  + <version>, version of the file, this field MUST be 1.

  + <LORDN Log creation datetime>, date and time in UTC that the LORDN Log was created.

  + <LORDN file creation datetime>, date and time in UTC of creation for the LORDN file that this log file is referring to.

  + <LORDN Log Identifier>, unique identifier of the LORDN Log provided by the TMDB. This identifier could be used by the Registry Operator to unequivocally identify the LORDN Log. The identified will be a string of a maximum LENGTH of 60 characters from the Base 64 alphabet.

  + <Status flag>, whether the LORDN file has been accepted for processing by the TMDB. Possible values are "accepted" or "rejected".

  + <Warning flag>, whether the LORDN Log has any warning result codes. Possible values are "no-warnings" or "warnings-present".

  + <Number of DN Lines>, number of DNs effective allocations processed in the LORDN file.

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

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

  With the header names as follows:

roid,result-code

- One or more lines with: <roid>,<result code>

  Where:

  + <roid>, DN Repository Object IDentifier (DNROID) in the SRS.

  + <result code>, result code as described in Section 6.3.1.1.

Example of a LORDN Log file

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

Figure 14

6.3.1.1. LORDN Log Result Codes

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

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

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

- err: an error is issued, LORDN file rejected by the TMDB.

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

LORDN Log Result Code Classes

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

Figure 15

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

LORDN Log result Codes

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

invalid.

3605 TCN Expired
The TCN was already expired (based on the <tmNotice:notAfter> field of the TCN) at the datetime of acknowledgement.

3606 Wrong TCNID used
The TCNID used for the registration does not match the related DN.

3609 Invalid SMD used
The SMD used for registration was not valid at the moment of registration based on the <smd:notBefore> and <smd:notAfter> elements.
In case of an asynchronous registration, this refer to the <datetime of application creation>.

3610 DN reported outside of the time window
The DN was reported outside of the required 26 hours reporting window.

3611 DN does not match the labels in SMD
The DN does not match the labels included in the SMD.

3612 SMDID does not exist
The SMDID has never existed in the central repository.

3613 SMD was revoked when used
The SMD used for registration was revoked more than 24 hours ago of the <datetime of registration>.
In case of an asynchronous registration, the <datetime of application creation> is used when validating the DN Line.

3614 TCNID does not exist
The TCNID has never existed in the central repository.

3615 Recent-dnl-insertion outside of the time window
The DN registration is reported as a recent-dnl-insertion, but the (re) insertion into the DNL occurred more than 24 hours ago.

3616 Registration Date of DN in Claims before the end of Sunrise Period
The registration date of the DN is before the end of the Sunrise Period and the DN was reported in a Trademark Claims LORDN file.

3617 Registrar has not been approved by the TMDB

Registrar ID in DN Line has not completed Trademark Claims integration testing with the TMDB.

3618 Registration Date of DN in QLP LORDN file out of the QLP Period
The registration date of the DN in a QLP LORDN file is outside of the QLP Period.

3619 TCN was not valid
The TCN was not valid (based on the <tmNotice:notBefore> field of the TCN) at the datetime of acknowledgement.

4501 Syntax Error in DN Line
Syntax Error in DN Line.

4601 Invalid TLD used
The TLD in the DN Line does not match what is expected for this LORDN.

4602 Registrar ID Invalid
Registrar ID in DN Line is not a valid ICANN-Accredited Registrar.

4603 Registration Date in the future
The <datetime of registration> in the DN Line is in the future.

4606 TLD not in Sunrise or Trademark Claims Period
The <datetime of registration> was reported when the TLD was not in Sunrise or Trademark Claims Periods.
In case of an asynchronous registration,
the <datetime of application creation> is used when validating the DN Line.

4607 Application Date in the future
The <datetime of application creation> in the DN Line is in the future.

4608 Application Date is later than Registration Date
The <datetime of application creation> in the DN Line is later than the <datetime of registration>.

4609 TCNID wrong syntax
The syntax of the TCNID is invalid.

4610 TCN Acceptance Date is in the future
The <datetime of acceptance of the TCN> is in the future.

4611 Label has never existed in the TMDB

The label in the registered DN has never existed in the TMDB.

Figure 16

6.4. Signed Mark Data (SMD) File

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

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

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

1. Marks: <marks>
2. smdID: <SMD-ID>
3. U-labels: <comma separated list of U-label or NR-LDH labels (see [RFC5890])>
4. notBefore: <begin validity>
5. notAfter: <end validity>
6. -----BEGIN ENCODED SMD-----
7. <encoded SMD (see [RFC7848])>
8. -----END ENCODED SMD-----

Example of an SMD File:

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

Figure 17

6.5. Trademark Claims Notice (TCN)

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

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

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

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

  The TCNID is a string concatenation of a TCN Checksum and the TMDB Notice Identifier. The first 8 characters of the TCNID is a TCN Checksum. The rest is the TMDB Notice Identifier, which is a zero-padded natural number in the range of 0000000000000000001 to 9223372036854775807.

  Example of a TCNID:

  370d0b7c9223372036854775807.

  Where:

  + TCN Checksum=370d0b7c

  + TMDB Notice Identifier=9223372036854775807

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

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

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

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

A Registry MUST use the leftmost DNL of the DN being effectively allocated, the expiration datetime of the TCN (provided by the Registrar) and the TMDB Notice Identifier extracted from the TCNID (provided by the Registrar) to compute the TCN Checksum. For example, if the DN "xn--mgbachtv.xn--mgbh0fb" is being effectively allocated, the leftmost DNL would be "xn--mgbachtv".

Example of computation of the TCN Checksum:

CRC32(example-one128194920092233720368547758 07)=370d0b7c

- A <tmNotice:notBefore> element that contains the start of the validity date and time of the TCN.
- A <tmNotice:notAfter> element that contains the expiration date and time of the TCN.
- A <tmNotice:label> element that contains the DNL covered by a PRM.
- One or more <tmNotice:claim> elements that contain the Trademark Claim. The <tmNotice:claim> element contains the following child elements:
  - A <tmNotice:markName> element that contains the mark text string.
  - One or more <tmNotice:holder> elements that contains the information of the holder of the mark. An "entitlement" attribute is used to identify the entitlement of the holder,

possible values are: owner, assignee or licensee. The child elements of <tmNotice:holder> include:

+ An OPTIONAL <tmNotice:name> element that contains the name of the holder. A <tmNotice:name> MUST be specified if <tmNotice:org> is not specified.

+ An OPTIONAL <tmNotice:org> element that contains the name of the organization holder of the mark. A <tmNotice:org> MUST be specified if <tmNotice:name> is not specified.

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

  - One, two or three OPTIONAL <tmNotice:street> elements that contains the organization's street address.

  - A <tmNotice:city> element that contains the organization's city.

  - An OPTIONAL <tmNotice:sp> element that contains the organization's state or province.

  - An OPTIONAL <tmNotice:pc> element that contains the organization's postal code.

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

+ An OPTIONAL <tmNotice:voice> element that contains the organization's voice telephone number.

+ An OPTIONAL <tmNotice:fax> element that contains the organization's facsimile telephone number.

+ An OPTIONAL <tmNotice:email> element that contains the email address of the holder.

* Zero or more OPTIONAL <tmNotice:contact> elements that contains the information of the representative of the mark registration. A "type" attribute is used to identify the type of contact, possible values are: owner, agent or thirdparty. The child elements of <tmNotice:contact> include:

+ A <tmNotice:name> element that contains name of the responsible person.

+ An OPTIONAL <tmNotice:org> element that contains the name of the organization of the contact.

+ A <tmNotice:addr> element that contains the address information of the contact. A <tmNotice:addr> contains the following child elements:

  - One, two or three OPTIONAL <tmNotice:street> elements that contains the contact's street address.

  - A <tmNotice:city> element that contains the contact's city.

  - An OPTIONAL <tmNotice:sp> element that contains the contact's state or province.

  - An OPTIONAL <tmNotice:pc> element that contains the contact's postal code.

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

+ A <tmNotice:voice> element that contains the contact's voice telephone number.

+ An OPTIONAL <tmNotice:fax> element that contains the contact's facsimile telephone number.

+ A <tmNotice:email> element that contains the contact's email address.

* A <tmNotice:jurDesc> element that contains the name (in English) of the jurisdiction where the mark is protected. A jurCC attribute contains the two-character code of the jurisdiction where the mark was registered. This is a two-character code from [WIPO.ST3].

* Zero or more OPTIONAL <tmNotice:classDesc> element that contains the description (in English) of the Nice Classification as defined in [WIPO-NICE-CLASSES]. A classNum attribute contains the class number.

* A <tmNotice:goodsAndServices> element that contains the full description of the goods and services mentioned in the mark registration document.

* An OPTIONAL <tmNotice:notExactMatch> element signals that the claim notice was added to the TCN based on other rule (e.g. [Claims50]) than exact match (defined in [MatchingRules]). The <tmNotice:notExactMatch> contains one or more:

  + An OPTIONAL <tmNotice:udrp> element that signals that the claim notice was added because of a previously abused name included in an UDRP case. The <tmNotice:udrp> contains:

    - A <tmNotice:caseNo> element that contains the UDRP case number used to validate the previously abused name.

    - A <tmNotice:udrpProvider> element that contains the name of the UDRP provider.

  + An OPTIONAL <tmNotice:court> element that signals that the claim notice was added because of a previously abused name included in a court's resolution. The <tmNotice:court> contains:

    - A <tmNotice:refNum> element that contains the reference number of the court's resolution used to validate the previously abused name.

    - A <tmNotice:cc> element that contains the two-character code from [ISO3166-2] of the jurisdiction of the court.

    - A <tmNotice:courtName> element that contains the name of the court.

Example of a <tmNotice:notice> object:

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

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

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

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

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

6.6. Sunrise List (SURL)

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

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

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

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

  Where:

  + <version>, version of the file, this field MUST be 1.

  + <Sunrise List creation datetime>, date and time in UTC that the Sunrise List was created.

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

  With the header names as follows:

  DNL,insertion-datetime

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

  Where:

  + <DNL>, a Domain Name Label covered by a PRM eligible for Sunrise.

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

Example of a SURL

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

Figure 18

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

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

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

7. Formal Syntax

7.1. Trademark Claims Notice (TCN)

The schema presented here is for a Trademark Claims Notice.

The BEGIN and END tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.

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

  <annotation>
    <documentation>
      Schema for representing a Trademark Claim Notice.
    </documentation>
  </annotation>
  <import namespace="urn:ietf:params:xml:ns:mark-1.0"/>
```

```
   <element name="notice" type="tmNotice:noticeType"/>
   <complexType name="holderType">
     <sequence>
       <element name="name" type="token" minOccurs="0"/>
       <element name="org" type="token" minOccurs="0"/>
       <element name="addr" type="tmNotice:addrType"/>
       <element name="voice" type="mark:e164Type" minOccurs="0"/>
       <element name="fax" type="mark:e164Type" minOccurs="0"/>
       <element name="email" type="mark:minTokenType" minOccurs="0"/>
     </sequence>
     <attribute name="entitlement" type="mark:entitlementType"/>
   </complexType>
   <complexType name="noticeType">
     <sequence>
       <element name="id" type="tmNotice:idType"/>
       <element name="notBefore" type="dateTime"/>
       <element name="notAfter" type="dateTime"/>
       <element name="label" type="mark:labelType"/>
       <element name="claim" type="tmNotice:claimType" minOccurs="0"
         maxOccurs="unbounded"/>
     </sequence>
   </complexType>
   <complexType name="claimType">
     <sequence>
       <element name="markName" type="token"/>
       <element name="holder" type="tmNotice:holderType"
         maxOccurs="unbounded"/>
       <element name="contact" type="tmNotice:contactType" minOccurs="0"
         maxOccurs="unbounded"/>
       <element name="jurDesc" type="tmNotice:jurDescType"/>
       <element name="classDesc" type="tmNotice:classDescType"
         minOccurs="0" maxOccurs="unbounded"/>
       <element name="goodsAndServices" type="token"/>
       <element name="notExactMatch" type="tmNotice:noExactMatchType"
         minOccurs="0"/>
     </sequence>
   </complexType>
   <complexType name="jurDescType">
     <simpleContent>
       <extension base="token">
         <attribute name="jurCC" type="mark:ccType" use="required"/>
       </extension>
     </simpleContent>
   </complexType>
   <complexType name="classDescType">
     <simpleContent>
       <extension base="token">
         <attribute name="classNum" type="integer" use="required"/>
```

```
      </extension>
    </simpleContent>
  </complexType>
  <complexType name="noExactMatchType">
    <choice maxOccurs="unbounded">
      <element name="udrp" type="tmNotice:udrpType"/>
      <element name="court" type="tmNotice:courtType"/>
    </choice>
  </complexType>
  <complexType name="udrpType">
    <sequence>
      <element name="caseNo" type="token"/>
      <element name="udrpProvider" type="token"/>
    </sequence>
  </complexType>
  <complexType name="courtType">
    <sequence>
      <element name="refNum" type="token"/>
      <element name="cc" type="mark:ccType"/>
      <element name="region" type="token" minOccurs="0"
        maxOccurs="unbounded"/>
      <element name="courtName" type="token"/>
    </sequence>
  </complexType>
  <complexType name="addrType">
    <sequence>
      <element name="street" type="token" minOccurs="1" maxOccurs="3"/>
      <element name="city" type="token"/>
      <element name="sp" type="token" minOccurs="0"/>
      <element name="pc" type="mark:pcType" minOccurs="0"/>
      <element name="cc" type="mark:ccType"/>
    </sequence>
  </complexType>
  <complexType name="contactType">
    <sequence>
      <element name="name" type="token"/>
      <element name="org" type="token" minOccurs="0"/>
      <element name="addr" type="tmNotice:addrType"/>
      <element name="voice" type="mark:e164Type"/>
      <element name="fax" type="mark:e164Type" minOccurs="0"/>
      <element name="email" type="mark:minTokenType"/>
    </sequence>
    <attribute name="type" type="mark:contactTypeType"/>
  </complexType>
  <simpleType name="idType">
    <restriction base="token">
      <pattern value="[a-fA-F0-9]{8}\d{1,19}"/>
    </restriction>
```

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

8. Acknowledgements

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

9. Change History

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

9.1. Version 04

1. Ping update.

9.2. Version 05

1. Ping update.

9.3. Version 06

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

2. Updated the reference of RFC7719 to RFC8499.

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

9.4. Version 07

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

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

9.5. Version 08

1. Fixed issues detected by idnits tool.

9.6. Version 09

1. Ping update.

10. IANA Considerations

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

Registration request for the Trademark Claims Notice namespace:

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

Registrant Contact: IETF <regext@ietf.org>

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

Registration request for the Trademark Claims Notice XML schema:

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

Registrant Contact: IETF <regext@ietf.org>

XML: See Section 7.1 of this document.

11. Security Considerations

This specification uses HTTP Basic Authentication to provide a simple application-layer authentication service. HTTPS is used in all interfaces in order to protect against most common attacks. In addition, the client identifier is tied to a set of IP addresses that are allowed to connect to the interfaces described in this document, providing an extra security measure.

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

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

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

12. References

12.1. Normative References

[Claims50]
ICANN, "Implementation Notes: Trademark Claims Protection for Previously Abused Names", July 2013, <https://newgtlds.icann.org/en/about/trademark-clearinghouse/previously-abused-16jul13-en.pdf>.

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

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

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

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

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

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

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

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

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

[W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition REC-xmlschema-2-20041028", October 2004, <https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>.

12.2. Informative References

[ICANN-GTLD-AGB-20120604]
ICANN, "gTLD Applicant Guidebook Version 2012-06-04", June 2012, <http://newgtlds.icann.org/en/applicants/agb/guidebook-full-04jun12-en.pdf>.

[ISO3166-2]
ISO, "International Standard for country codes and codes for their subdivisions", 2006, <http://www.iso.org/iso/home/standards/country_codes.htm>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

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

[RFC4180] Shafranovich, Y., "Common Format and MIME Type for Comma-Separated Values (CSV) Files", RFC 4180, DOI 10.17487/RFC4180, October 2005, <https://www.rfc-editor.org/info/rfc4180>.

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

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <https://www.rfc-editor.org/info/rfc4880>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

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

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

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

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

[WIPO-NICE-CLASSES] WIPO, "WIPO Nice Classification", 2015, <http://www.wipo.int/classifications/nice/en>.

[WIPO.ST3] WIPO, "Recommended standard on two-letter codes for the representation of states, other entities and intergovernmental organizations", March 2007, <http://www.iso.org/iso/home/standards/country_codes.htm>.

Author’s Address

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

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

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

J. Gould
VeriSign, Inc.
M. Casanova
SWITCH

15 November 2020

Extensible Provisioning Protocol (EPP) Unhandled Namespaces draft-ietf-regext-unhandled-namespaces-05

Abstract

The Extensible Provisioning Protocol (EPP), as defined in RFC 5730, includes a method for the client and server tof determine the objects to be managed during a session and the object extensions to be used during a session. The services are identified using namespace URIs. How should the server handle service data that needs to be returned in the response when the client does not support the required service namespace URI, which is referred to as an unhandled namespace? An unhandled namespace is a significant issue for the processing of RFC 5730 poll messages, since poll messages are inserted by the server prior to knowing the supported client services, and the client needs to be capable of processing all poll messages. This document defines an operational practice that enables the server to return information associated with unhandled namespace URIs that is compliant with the negotiated services defined in RFC 5730.

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 19 May 2021.

Copyright Notice

Table of Contents

1. Introduction

The Extensible Provisioning Protocol (EPP), as defined in [RFC5730], includes a method for the client and server to determine the objects to be managed during a session and the object extensions to be used during a session. The services are identified using namespace URIs. How should the server handle service data that needs to be returned in the response when the client does not support the required service namespace URI, which is referred to as an unhandled namespace? An unhandled namespace is a significant issue for the processing of [RFC5730] poll messages, since poll messages are inserted by the server prior to knowing the supported client services, and the client needs to be capable of processing all poll messages. An unhandled namespace is an issue also for general EPP responses when the server has information that it cannot return to the client due to the client's supported services. The server should be able to return unhandled namespace information that the client can process later. This document defines an operational practice that enables the server to return information associated with unhandled namespace URIs that is compliant with the negotiated services defined in [RFC5730].

1.1. Conventions Used in This Document

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

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

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

The examples reference XML namespace prefixes that are used for the associated XML namespaces. Implementations MUST NOT depend on the example XML namespaces and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents. The example namespace prefixes used and their associated XML namespaces include:

"changePoll": urn:ietf:params:xml:ns:changePoll-1.0
"domain": urn:ietf:params:xml:ns:domain-1.0
"secDNS": urn:ietf:params:xml:ns:secDNS-1.1

In the template example XML, placeholder content is represented by the following variables:

"[NAMESPACE-XML]": XML content associated with a login service namespace URI. An example is the <domain:infData> element content in [RFC5731].

"[NAMESPACE-URI]": XML namespace URI associated with the [NAMESPACE-XML] XML content. An example is "urn:ietf:params:xml:ns:domain-1.0" in [RFC5731].

2. Unhandled Namespaces

An Unhandled Namespace is an XML namespace that is associated with a response extension that is not included in the client-specified EPP login services of [RFC5730]. The EPP login services consists of the set of XML namespace URIs included in the <objURI> or <extURI> elements of the [RFC5730] EPP <login> command. The services supported by the server are included in the <objURI> and <extURI> elements of the [RFC5730] EPP <greeting>, which should be a superset of the login services included in the EPP <login> command. A server may have information associated with a specific namespace that it needs to return in the response to a client. The unhandled namespaces problem exists when the server has information, that it needs to return to the client, that is not supported by the client based on the negotiated EPP <login> command services.

3. Use of EPP <extValue> for Unhandled Namespace Data

In [RFC5730], the <extValue> element is used to provide additional error diagnostic information, including the <value> element that identifies the client-provided element that caused a server error condition, and the <reason> element containing the human-readable message that describes the reason for the error. This operational practice extends the use of the <extValue> element for the purpose of returning unhandled namespace information in a successful response.

When a server has data to return to the client, that the client does not support based on the login services, the server MAY return a successful response, with the data for each unsupported namespace moved into an [RFC5730] <extValue> element. The unhandled namespace will not cause an error response, but the unhandled namespace data will instead be moved to an <extValue> element, along with a reason why the unhandled namespace data could not be included in the appropriate location of the response. The <extValue> element XML will not be processed by the XML processor. The <extValue> element contains the following child elements:

<value>: Contains a child-element with the unhandled namespace XML.

The XML namespace and namespace prefix of the child element MUST be defined, which MAY be defined in the <value> element or in the the child element. XML processing of the <value> element is disabled in [RFC5730], so the information can safely be returned in the <value> element.

<reason>: A formatted human-readable message that indicates the reason the unhandled namespace data was not returned in the appropriate location of the response. The formatted reason SHOULD follow the Augmented Backus-Naur Form (ABNF) grammar [RFC5234] format: NAMESPACE-URI "not in login services", where NAMESPACE-URI is the unhandled XML namespace like "urn:ietf:params:xml:ns:domain-1.0" for [RFC5731].

This document supports handling of unsupported namespaces for [RFC3735] object-level extensions and command-response extensions. This document does not support [RFC3735] protocol-level extensions or authentication information extensions. Refer to the following sections on how to handle an unsupported object-level extension namespace or an unsupported command-response extension namespace.

3.1. Unhandled Object-Level Extension

An object-level extension in [RFC5730] is a child element of the <resData> element. If the client does not handle the namespace of the object-level extension, then the <resData> element is removed and its object-level extension child element is moved into a [RFC5730] <extValue> <value> element, with the namespace URI included in the corresponding <extValue> <reason> element. The response becomes a general EPP response without the <resData> element.

Template response for a supported object-level extension. The [NAMESPACE-XML] variable represents the object-level extension XML.

```
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:      [NAMESPACE-XML]
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Template unhandled namespace response for an unsupported object-level extension. The [NAMESPACE-XML] variable represents the object-level extension XML and the [NAMESPACE-URI] variable represents the object-level extension XML namespace URI.

```
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:      <extValue>
S:        <value>
S:          [NAMESPACE-XML]
S:        </value>
S:        <reason>
S:          [NAMESPACE-URI] not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The EPP response is converted from an object response to a general EPP response by the server when the client does not support the object-level extension namespace URI. Below is example of converting the <transfer> query response example in [RFC5731] to an unhandled namespace response.

[RFC5731] example <transfer> query response converted into an unhandled namespace 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:      <extValue>
S:        <value>
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>2020-06-06T22:00:00.0Z</domain:reDate>
S:            <domain:acID>ClientY</domain:acID>
S:            <domain:acDate>2020-06-11T22:00:00.0Z</domain:acDate>
S:            <domain:exDate>2021-09-08T22:00:00.0Z</domain:exDate>
S:          </domain:trnData>
S:        </value>
S:        <reason>
S:          urn:ietf:params:xml:ns:domain-1.0 not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.2. Unhandled Command-Response Extension

A command-response extension in [RFC5730] is a child element of the <extension> element. If the client does not handle the namespace of the command-response extension, the command-response child element is moved into a [RFC5730] <extValue> <value> element, with the namespace URI included in the corresponding <extValue> <reason> element. If after moving the command-response child element there are no additional command-response child elements, the <extension> element MUST be removed.

Template response for a supported command-response extension. The [NAMESPACE-XML] variable represents the command-response extension XML.

```
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:      [NAMESPACE-XML]
S:    </extension>
S:    <trID>
S:     <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Template unhandled namespace response for an unsupported command-response extension. The [NAMESPACE-XML] variable represents the command-response extension XML and the [NAMESPACE-URI] variable represents the command-response extension XML namespace URI.

```
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:      <extValue>
S:        <value>
S:         [NAMESPACE-XML]
S:        </value>
S:        <reason>
S:          [NAMESPACE-URI] not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The EPP response is converted to an unhandled namespace response by moving the unhandled command-response extension from under the <extension> to an <extValue> element. Below is example of converting the DS Data Interface <info> response example in [RFC5910] to an unhandled namespace response.

[RFC5910] DS Data Interface <info> response converted into an unhandled namespace 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:      <extValue>
S:        <value>
S:        <secDNS:infData
S:          xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1">
S:          <secDNS:dsData>
S:          <secDNS:keyTag>12345</secDNS:keyTag>
S:          <secDNS:alg>3</secDNS:alg>
S:          <secDNS:digestType>1</secDNS:digestType>
S:          <secDNS:digest>49FD46E6C4B45C55D4AC</secDNS:digest>
S:          </secDNS:dsData>
S:          </secDNS:infData>
S:        </value>
S:        <reason>
S:          urn:ietf:params:xml:ns:secDNS-1.1 not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:ns>
S:          <domain:hostObj>ns1.example.com</domain:hostObj>
S:          <domain:hostObj>ns2.example.com</domain:hostObj>
S:        </domain:ns>
S:        <domain:host>ns1.example.com</domain:host>
S:        <domain:host>ns2.example.com</domain:host>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:upID>ClientX</domain:upID>
S:        <domain:upDate>2020-12-03T09:00:00.0Z</domain:upDate>
S:        <domain:exDate>2021-04-03T22:00:00.0Z</domain:exDate>
S:        <domain:trDate>2000-04-08T09:00:00.0Z</domain:trDate>
S:        <domain:authInfo>
```

```
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

4. Signaling Client and Server Support

This document does not define new protocol but an operational practice using the existing EPP protocol, where the client and the server can signal support for the operational practice using a namespace URI in the login and greeting extension services. The namespace URI "urn:ietf:params:xml:ns:epp:unhandled-namespaces-1.0" is used to signal support for the operational practice. The client includes the namespace URI in an <svcExtension> <extURI> element of the [RFC5730] <login> Command. The server includes the namespace URI in an <svcExtension> <extURI> element of the [RFC5730] Greeting.

A client that receives the namespace URI in the server's Greeting extension services, can expect the following supported behavior by the server:

1. Support unhandled namespace object-level extensions and command-response extensions in EPP poll messages, per Section 6.
2. Support the option of unhandled namespace command-response extensions in general EPP responses, per Section 5.

A server that receives the namespace URI in the client's <login> Command extension services, can expect the following supported behavior by the client:

1. Support monitoring the EPP poll messages and general EPP responses for unhandled namespaces.

5. Usage with General EPP Responses

The unhandled namespace approach defined in Section 3 MAY be used for a general EPP response to an EPP command. A general EPP response includes any non-poll message EPP response. The use of the unhandled namespace approach for poll message EPP responses is defined in Section 6. The server MAY exclude the unhandled namespace information in the general EPP response or MAY include it using the unhandled namespace approach.

The unhandled namespace approach for general EPP responses SHOULD only be applicable to command-response extensions, defined in Section 3.2, since the server SHOULD NOT accept an object-level EPP command if the client did not include the object-level namespace URI in the login services. An object-level EPP response extension is returned when the server successfully executes an object-level EPP command extension. The server MAY return an unhandled object-level extension to the client as defined in Section 3.1.

Returning domain name Redemption Grace Period (RGP) data, based on [RFC3915], provides an example of applying the unhandled namespace approach for a general EPP response. If the client does not include the "urn:ietf:params:xml:ns:rgp-1.0" namespace URI in the login services, and the domain <info> response of a domain name does have RGP information, the server MAY exclude the <rgp:infData> element from the EPP response or MAY include it under in the <extValue> element per Section 3.2.

[RFC5731] domain name <info> response with the unhandled [RFC3915] <rgp:infData> element included under an <extValue> element:

```
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:      <extValue>
S:        <value>
S:          <rgp:infData xmlns:rgp="urn:ietf:params:xml:ns:rgp-1.0">
S:            <rgp:rgpStatus s="redemptionPeriod"/>
S:          </rgp:infData>
S:        </value>
S:        <reason>
S:          urn:ietf:params:xml:ns:rgp-1.0 not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="pendingDelete"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:ns>
S:          <domain:hostObj>ns1.example.com</domain:hostObj>
```

```
S:          <domain:hostObj>ns1.example.net</domain:hostObj>
S:        </domain:ns>
S:        <domain:host>ns1.example.com</domain:host>
S:        <domain:host>ns2.example.com</domain:host>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:upID>ClientX</domain:upID>
S:        <domain:upDate>2020-12-03T09:00:00.0Z</domain:upDate>
S:        <domain:exDate>2021-04-03T22:00:00.0Z</domain:exDate>
S:        <domain:trDate>2000-04-08T09:00:00.0Z</domain:trDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

6. Usage with Poll Message EPP Responses

The unhandled namespace approach, defined in Section 3, MUST be used if there is unhandled namespace information included in an EPP <poll> message response. The server inserts poll messages into the client's poll queue independent of knowing the supported client login services, therefore there may be unhandled object-level and command-response extensions included in a client's poll queue. In [RFC5730], the <poll> command is used by the client to retrieve and acknowledge poll messages that have been inserted by the server. The <poll> message response is an EPP response that includes the <msgQ> element that provides poll queue meta-data about the message. The unhandled namespace approach, defined in Section 3, is used for an unhandled object-level extension and for each of the unhandled command-response extensions attached to the <poll> message response. The resulting EPP <poll> message response MAY have either or both the object-level extension or command-response extensions moved to <extValue> elements, as defined in Section 3.

The Change Poll Message, as defined in [RFC8590], which is an extension of any EPP object, is an example of applying the unhandled namespace approach for EPP <poll> message responses. The object that will be used in the examples is a [RFC5731] domain name object.

[RFC5731] domain name <info> <poll> message response with the unhandled [RFC8590] <changePoll:changeData> element included under an <extValue> element:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:      <extValue>
S:        <value>
S:          <changePoll:changeData
S:          xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0"
S:            state="after">
S:            <changePoll:operation>update</changePoll:operation>
S:            <changePoll:date>
S:              2020-11-22T05:00:00.000Z</changePoll:date>
S:            <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:            <changePoll:who>URS Admin</changePoll:who>
S:            <changePoll:caseId type="urs">urs123
S:            </changePoll:caseId>
S:            <changePoll:reason>URS Lock</changePoll:reason>
S:          </changePoll:changeData>
S:        </value>
S:        <reason>
S:      urn:ietf:params:xml:ns:changePoll-1.0 not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <msgQ
S:      count="15"
S:      id="1"
S:    >
S:      <qDate>2020-11-22T05:00:00.000Z</qDate>
S:      <msg>Registry initiated update of domain.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>change-poll.tld</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="serverUpdateProhibited"/>
S:        <domain:status s="serverDeleteProhibited"/>
S:        <domain:status s="serverTransferProhibited"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:clID>ClientX</domain:clID>
```

```
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2012-05-03T04:00:00.000Z</domain:crDate>
S:        <domain:upID>ClientZ</domain:upID>
S:        <domain:upDate>2020-11-22T05:00:00.000Z</domain:upDate>
S:        <domain:exDate>2021-05-03T04:00:00.000Z</domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Unhandled [RFC5731] domain name <info> <poll> message response and the unhandled [RFC8590] <changePoll:changeData> element included under an <extValue> element:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:      <extValue>
S:        <value>
S:          <domain:infData
S:            xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:            <domain:name>change-poll.tld</domain:name>
S:            <domain:roid>EXAMPLE1-REP</domain:roid>
S:            <domain:status s="serverUpdateProhibited"/>
S:            <domain:status s="serverDeleteProhibited"/>
S:            <domain:status s="serverTransferProhibited"/>
S:            <domain:registrant>jd1234</domain:registrant>
S:            <domain:contact type="admin">sh8013</domain:contact>
S:            <domain:contact type="tech">sh8013</domain:contact>
S:            <domain:clID>ClientX</domain:clID>
S:            <domain:crID>ClientY</domain:crID>
S:            <domain:crDate>2012-05-03T04:00:00.000Z</domain:crDate>
S:            <domain:upID>ClientZ</domain:upID>
S:            <domain:upDate>2020-11-22T05:00:00.000Z</domain:upDate>
S:            <domain:exDate>2021-05-03T04:00:00.000Z</domain:exDate>
S:          </domain:infData>
S:        </value>
S:        <reason>
S:        urn:ietf:params:xml:ns:domain-1.0 not in login services
S:        </reason>
S:      </extValue>
S:      <extValue>
```

```
S:        <value>
S:          <changePoll:changeData
S:          xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0"
S:            state="after">
S:            <changePoll:operation>update</changePoll:operation>
S:            <changePoll:date>
S:              2020-11-22T05:00:00.000Z</changePoll:date>
S:            <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:            <changePoll:who>URS Admin</changePoll:who>
S:            <changePoll:caseId type="urs">urs123
S:            </changePoll:caseId>
S:            <changePoll:reason>URS Lock</changePoll:reason>
S:          </changePoll:changeData>
S:        </value>
S:        <reason>
S:      urn:ietf:params:xml:ns:changePoll-1.0 not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <msgQ
S:      count="15"
S:      id="1"
S:    >
S:      <qDate>2020-11-22T05:00:00.000Z</qDate>
S:      <msg>Registry initiated update of domain.</msg>
S:    </msgQ>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

7. Implementation Considerations

There are implementation considerations for the client and the server to help address the risk of the client ignoring unhandled namespace information included in an EPP response that is needed to meet technical, policy, or legal requirements.

7.1. Client Implementation Considerations

To reduce the likelihood of a client receiving unhandled namespace information, the client should consider the following:

1. Ensure that the login services is accurate with what is supported by the client. If there are gaps between the services supported by the client and the login services included in the login command, the client may receive unhandled namespace information that the client could have supported.
2. Support all of the services included in the server greeting services that may be included in an EPP response, including the poll queue responses. The client should evaluate the gaps between the greeting services and the login services provided in the login command to identify extensions that need to be supported.
3. Proactively monitor for unhandled namespace information in the EPP responses, by looking for the inclusion of the <extValue> element in successful responses, recording the unsupported namespace included in the <reason> element, and recording the unhandled namespace information included in the <value> element for later processing. The unhandled namespace can be implemented by the client to ensure that information is processed fully in future EPP responses.

7.2. Server Implementation Considerations

To assist the clients in recognizing unhandled namespaces, the server should consider the following:

1. Monitor for returning unhandled namespace information to clients and report it to the clients out-of-band to EPP so the clients can add support for the unhandled namespaces.
2. Look for the unhandled namespace support in the login services when returning optional unhandled namespace information in General EPP Responses.

8. IANA Considerations

8.1. XML Namespace

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

Registration request for the unhandled namespaces namespace:

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

8.2. EPP Extension Registry

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

Name of Extension: "Extensible Provisioning Protocol (EPP) Unhandled Namespaces"

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 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. Verisign EPP SDK

Organization: Verisign Inc.

Name: Verisign EPP SDK

Description: The Verisign EPP SDK includes an implementation of the unhandled namespaces for the processing of the poll queue messages.

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

9.2. SWITCH Automated DNSSEC Provisioning Process

Organization: SWITCH

Name: Registry of .CH and .LI

Description: SWITCH uses poll messages to inform the registrar about DNSSEC changes at the registry triggered by CDS records. These poll messages are enriched with the 'urn:ietf:params:xml:ns:changePoll-1.0' and the 'urn:ietf:params:xml:ns:secDNS-1.1' extension that are rendered in the poll msg response according to this draft.

Level of maturity: Operational

Coverage: All aspects of the protocol are implemented.

Licensing: Proprietary

Contact: martin.casanova@switch.ch

URL: https://www.nic.ch/cds

10. Security Considerations

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

11. Acknowledgements

The authors wish to thank the following persons for their feedback and suggestions: Thomas Corte, Scott Hollenbeck, Patrick Mevzek, and Marcel Parodi.

12. References

12.1. 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>.

[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>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

[RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <https://www.rfc-editor.org/info/rfc5731>.

[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>.

[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>.

[RFC8590] Gould, J. and K. Feher, "Change Poll Extension for the Extensible Provisioning Protocol (EPP)", RFC 8590, DOI 10.17487/RFC8590, May 2019, <https://www.rfc-editor.org/info/rfc8590>.

12.2. Informative References

[RFC3735] Hollenbeck, S., "Guidelines for Extending the Extensible Provisioning Protocol (EPP)", RFC 3735, DOI 10.17487/RFC3735, March 2004, <https://www.rfc-editor.org/info/rfc3735>.

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

Appendix A. Change History

A.1. Change from 00 to 01

1. Removed xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" reference from examples.
2. removed <extension></extension> block from example.
3. added SWITCH Automated DNSSEC Provisioning Process at Implementation Status

A.2. Change from 01 to 02

1. Ping update

A.3. Change from 02 to REGEXT 00

1. Changed to regext working group draft by changing draft-gould-casanova-regext-unhandled-namespaces to draft-ietf-regext-unhandled-namespaces.

A.4. Change from REGEXT 00 to REGEXT 01

1. Added the "Signaling Client and Server Support" section to describe the mechanism to signal support for the BCP by the client and the server.
2. Added the IANA Considerations section with the registration of the unhandled namespaces XML namespace and the registration of the EPP Best Current Practice (BCP) in the EPP Extension Registry.

A.5. Change from REGEXT 01 to REGEXT 02

1. Filled in the acknowledgements section.
2. Changed the reference from RFC 5730 to RFC 5731 for the transfer example in section 3.1 "Unhandled Object-Level" Extension.
3. Updated the XML namespace to urn:ietf:params:xml:ns:epp:unhandled-namespaces-1.0, which removed bcp from the namespace and bumped the version from 0.1 and 1.0. Inclusion of bcp in the XML namespace was discussed at the REGEXT interim meeting.

A.6. Change from REGEXT 02 to REGEXT 03

1. Converted from xml2rfc v2 to v3.
2. Updated Acknowledgements to match the approach taken by the RFC Editor with draft-ietf-regext-login-security.
3. Changed reference of ietf-regext-change-poll to RFC 8590.

A.7. Change from REGEXT 03 to REGEXT 04

1. Changed from Best Current Practice (BCP) to Standards Track based on mailing list discussion.
2. Revised the dates in the examples to be more up-to-date.

A.8. Change from REGEXT 04 to REGEXT 05

1. Based on feedback from Thomas Corte, added a description of the <extValue> element in RFC 5730 and it being extended to support returning unhandled namespace information.
2. Based on feedback from Thomas Corte, added a Implementation Considerations section to cover client and server implementation recommendations such as monitoring unhandled namespaces in the server to report to the clients out-of-band and monitoring for responses containing unhanded namespace information in the client to proactively add support for the unhandled namespaces.
3. Moved RFC 3735 and RFC 7451 to informative references to address down reference errors in idnits.

Authors' Addresses

James Gould
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States of America

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

Martin Casanova
SWITCH
P.O. Box
CH-8021 Zurich
Switzerland

Email: martin.casanova@switch.ch
URI: http://www.switch.ch

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: February 1, 2021

M. Loffredo
IIT-CNR/Registro.it
G. Brown
CentralNic Group plc

July 31, 2020

Using JSContact in Registration Data Access Protocol (RDAP) JSON Responses
draft-loffredo-regext-rdap-jcard-deprecation-03

Abstract

This document describes an RDAP extension which represents entity contact information in JSON responses using JSContact.

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 February 1, 2021.

Table of Contents

1. Introduction

This document specifies an extension to the Registration Data Access Protocol (RDAP) that allows RDAP servers to use JSContact ([draft-ietf-jmap-jscontact]) to represent the contact information associated with entities in RDAP responses, instead of jCard ([RFC7095]). It also describes the process by which an RDAP server can transition from jCard to JSContact. RDAP query and response extensions are defined to facilitate the transition process.

1.1. Rationale

According to the feedback from RDAP Pilot Working Group ([RDAP-PILOT-WG], a group of RDAP server implementers representing registries and registrars of generic TLDs), the most commonly raised implementation concern, for both servers and client implementers, related to the use of jCard ([RFC7095]) to represent the contact information associated with entities. Working Group members reported jCard to be unintuitive, complicated to implement for both clients and servers, and incompatible with best practices for RESTful APIs.

JSContact ([draft-ietf-jmap-jscontact]) provides a simpler and more efficient representation for contact information. In addition, similarly to jCard, it provides a means to represent internationalised and unstructured contact information. Support for internationalised contact information has been recognised being necessary to facilitate the future internationalisation of registration data directory services.

1.2. Conventions Used in This Document

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

2. JSContact

The JSContact specification defines a data model and JSON representation of contact information that can be used for data storage and exchange in address book or directory applications. It aims to be an alternative to the vCard data format ([RFC6350]) and to be unambiguous, extendable and simple to process. In contrast with jCard, it is not a direct mapping from the vCard data model and expands semantics where appropriate.

The JSContact specification declares two main object types: "JSCard", which represents a single contact "card", and "JSCardGroup" which represents a collection of JSCard objects. For the purpose of this document, only JSCard objects are considered.

JSCard differs from jCard in that it:

- follows an object-oriented rather than array-oriented approach;
- is simple to process;

- requires no extra work in serialization/deserialization from/to a data model;
- includes no "jagged" arrays;
- prefers maps rather than arrays to implement collections;
- is able to represent redacted contacts (both "name" and"fullName" properties are optional).

[draft-ietf-jmap-jscontact-vcard] provides informational guidance on the conversion of jCard objects into JSCard objects, and vice versa.

3. Using JSCard objects in RDAP Responses

Entity objects in RDAP responses MAY include a "jscard" property whose value is a JSCard object instead of the "vCardArray" property defined in [RFC7483].

Servers returning the "jscard" property in their response MUST include "jscard" in the "rdapConformance" array.

An example of an RDAP response containing a "jscard" property is shown in Figure 1. The "jscard" object in this example has been converted from the example included in section 5.1 of [RFC7483].

```
{
   "rdapConformance": [
      "rdap_level_0",
      "jscard"
   ],
   "objectClassName" : "entity",
   "handle":"XXXX",
   "jscard":{
     "uid": "XXXX",
     "fullName": { "value": "Joe User" },
     "kind": "individual",
     "preferredContactLanguages": {
       "fr": { "preference": 1 },
       "en": { "preference": 2 }
     },
     "organization": [ { "value": "Example" } ],
     "jobTitle": [ { "value": "Research Scientist" } ],
     "role": [ { "value": "Project Lead" } ],
     "addresses": [
                    {
                      "context": "work",
                      "extension": "Suite 1234",
```

```
                        "street": "4321 Rue Somewhere",
                        "locality": "Quebec",
                        "region": "QC",
                        "postcode": "G1V 2M2",
                        "country": "Canada",
                        "coordinates": "geo:46.772673,-71.282945",
                        "timeZone": "Canada/Eastern"
                      },
                      {
                        "context": "private",
                        "fullAddress": {
                          "value": "123 Maple Ave\nSuite 90001\nVancouver\nBC\n1
239\n"
                        }
                      }
        ],
        "phones": [
                    {
                      "context": "work",
                      "type": "voice",
                      "labels": {
                        "cell": true,
                        "video": true,
                        "text": true
                      },
                      "isPreferred": true,
                      "value": "tel:+1-555-555-1234;ext=102"
                    }
        ],
        "emails": [
                    {
                      "context": "work",
                      "value": "joe.user@example.com"
                    }
        ],
        "online": [
                    {
                      "context": "work",
                      "type": "uri",
                      "labels": { "key": true },
                      "value": "http://www.example.com/joe.user/joe.asc"
                    },
                    {
                      "context": "private",
                      "type": "uri",
                      "labels": { "url": true },
                      "value": "http://example.org"
                    }
        ]
```

```
      }
      "roles":[ "registrar" ],
      "publicIds":[
        {
          "type":"IANA Registrar ID",
          "identifier":"1"
        }
      ],
      "remarks":[
        {
          "description":[
            "She sells sea shells down by the sea shore.",
            "Originally written by Terry Sullivan."
          ]
        }
      ],
      "links":[
        {
          "value":"http://example.com/entity/XXXX",
          "rel":"self",
          "href":"http://example.com/entity/XXXX",
          "type" : "application/rdap+json"
        }
      ],
      "events":[
        {
          "eventAction":"registration",
          "eventDate":"1990-12-31T23:59:59Z"
        }
      ],
      "asEventActor":[
        {
          "eventAction":"last changed",
          "eventDate":"1991-12-31T23:59:59Z"
        }
      ]
   }
```

Figure 1: Example of "jscard" in RDAP response

3.1. RDAP Query Parameters

Two new query parameters are defined for the purpose of this document.

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

- "jscard": a boolean value that allows a client to request the "jscard" property in the RDAP response;

- "jcard": a boolean value that allows a client to request the "vcardArray" property in the RDAP response.

These parameters are furtherly explained in Section 4.

4. Transition Considerations

4.1. RDAP Features Supporting a Transition Process

4.1.1. Notices and Link Relationships

RDAP allows servers to communicate service information to clients through notices. An RDAP response may contain one or more notice objects ([RFC7483], Section 4.3), each of which may include a set of link objects, which can be used to provide clients with references and documentation. These link objects may have a "rel" property which defines the relationship type, as described in [RFC8288], Section 4. The transition process outlined in this document uses two types of link relation:

- "deprecation", as described in [draft-dalal-deprecation-header];

- "alternate", as described in [RFC8288].

4.1.2. rdapConformance Property

The information about the specifications used in the construction of the response is also described by the strings which appear in the "rdapConformance" property of the RDAP response.

4.1.3. Query Parameters

Clients are able to ask servers to use specific RDAP features by using appropriate query parameters as described in [RFC7482].

4.2. Transition Procedure

The procedure for jCard to JSCard transition consists of four contiguous stages. During the procedure, the presence of "jscard" tag in the rdapConformance array indicates that JSCard is returned instead of jCard. The time format used to notify clients about this procedure is defined in [RFC3339].

Some elements of the following procedure are based on the best practices in [API-DEPRECATION].

4.2.1. Transition Stages

4.2.1.1. Stage 1: only jCard provided

This stage corresponds to providing jCard as default contact card ([RFC7483]). The RDAP server is not able to provide an alternate contact card. The rdapConformance array MUST NOT contain the "jscard" tag.

4.2.1.2. Stage 2: jCard sunset

During this stage, the server uses jCard by default, but the RDAP server will return JSCard if the client sets the query parameter "jscard" to a true value. The rdapConformance array MUST contain the "jscard" tag if JSCard is requested.

The RDAP server SHOULD include a notice titled "jCard sunset end". Such a notice should include a description reporting the jCard sunset end time and two links:

- "deprecation": a link to a URI-identified resource documenting the jCard deprecation;
- "alternate": if JSCard is not requested, a link to the JSCard version of same resource as identified by the current query string plus the parameter "jscard" set to a true value (Figure 2); otherwise, only the "deprecation" link is provided (Figure 3).

```
"notices": [
  {
    "title": "jCard sunset end",
    "description": ["2020-07-01T00:00:00Z"],
    "links": [{
        "value": "http://example.net/entity/XXXX",
        "rel": "deprecation",
        "type": "text/html",
        "href": "http://www.example.com/jcard_deprecation.html"
      },
      {
        "value": "http://example.net/entity/XXXX",
        "rel": "alternate",
        "type": "application/rdap+json",
        "href": " http://example.net/entity/XXXX?jscard=1"
      }
    ]
  }
]
```

Figure 2: jCard sunset - JSCard not requested

```
"notices": [
  {
    "title": "jCard sunset end",
    "description": ["2020-07-01T00:00:00Z"],
    "links": [
      {
        "value": "http://example.net/entity/XXXX?jscard=1",
        "rel": "deprecation",
        "type": "text/html",
        "href": "http://www.example.com/jcard_deprecation.html"
      }
    ]
  }
]
```

Figure 3: jCard sunset - JSCard requested

4.2.1.3. Stage 3: jCard deprecation

This stage corresponds to the provisioning of JSCard by default, but the RDAP will return jCard if the client sets the query parameter "jcard" to a true value. The rdapConformance array contains the "jscard" tag unless jCard is requested. The "jscard" query parameter is ignored.

The RDAP server SHOULD to return a notice titled "jCard deprecation end". Such a notice should include a description reporting the jCard deprecation end time and two links:

- "deprecation": a link to a URI-identified resource documenting the jCard deprecation;

- "alternate": if jCard is not requested, a link to the jCard version of the same resource as identified by the current query string plus the parameter "jcard" set to 1/true/yes (Figure 4); otherwise, a link to the JSCard version of the same resource as identified by the current query string without the parameter "jcard" (Figure 5).

```
"notices": [
  {
    "title": "jCard deprecation end",
    "description": ["2020-12-31T23:59:59Z"],
    "links": [
      {
        "value": "http://example.net/entity/XXXX",
        "rel": "deprecation",
        "type": "text/html",
        "href": "http://www.example.com/jcard_deprecation.html"
      },
      {
        "value": "http://example.net/entity/XXXX",
        "rel": "alternate",
        "type": "application/rdap+json",
        "href": " http://example.net/entity/XXXX?jcard=1"
      }
    ]
  }
]
```

Figure 4: jCard deprecation - jCard not requested

```
"notices": [
  {
    "title": "jCard deprecation end",
    "description": ["2020-12-31T23:59:59Z"],
    "links": [
      {
        "value": "http://example.net/entity/XXXX?jcard=1",
        "rel": "deprecation",
        "type": "text/html",
        "href": "http://www.example.com/jcard_deprecation.html"
      },
      {
        "value": "http://example.net/entity/XXXX?jcard=1",
        "rel": "alternate",
        "type": "application/rdap+json",
        "href": " http://example.net/entity/XXXX"
      }
    ]
  }
]
```

Figure 5: jCard deprecation - jCard requested

4.2.1.4. Stage 4: jCard deprecated

This stage corresponds to providing JSCard as default contact card. The RDAP server is not able to provide an alternate contact card. The rdapConformance array always contains "jscard" tag. The RDAP server doesn't include any notice about the jCard deprecation process. Both "jscard" and "jcard" query parameters are ignored.

4.2.1.5. Length

The length of both jCard sunset and jCard deprecation periods are not fixed by this specification. Best practices in REST API deprecation suggest that, depending on the deprecated API's reach, user base and service offering, a convenient time could be anywhere between 3 - 8 months. Anyway, RDAP providers are recommended to monitor the server log to figure out whether declared times need to be changed to meet client requirements.

4.2.1.6. Goals

The procedure described in this document achieves the following goals:

- only one contact representation would be included in the response;

- the response would always be compliant to [RFC7483];
- clients would be informed about the transition timeline;
- the backward compatibility would be guaranteed throughout the transition;
- servers and clients could execute their transitions independently.

5. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942 [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

5.1. IIT-CNR/Registro.it

Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it

Location: https://rdap.pubtest.nic.it/

Description: This implementation includes support for RDAP queries using data from the public test environment of .it ccTLD.

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

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

Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

6. IANA Considerations

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

Extension identifier: jscard

Registry operator: Any

Published specification: This document.

Contact: IETF <iesg@ietf.org>

Intended usage: This extension represents a contact card provided in an RDAP response according to the JSContact specification ([draft-ietf-jmap-jscontact]).

7. Security Considerations

Unlike jCard, the formatted name as well as any other personally identifiable information is not required in JSCard. The only mandatory property, namely "uid", is usually an opaque string. Therefore, redacted properties can be merely excluded without using placeholder values.

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>.

[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>.

[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>.

[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>.

8.2. Informative References

[API-DEPRECATION]
Sandoval, K., "How to Smartly Sunset and Deprecate APIs", August 2019, <https://web.archive.org/web/20200417084255/https://nordicapis.com/how-to-smartly-sunset-and-deprecate-apis/>.

[draft-dalal-deprecation-header]
Dalal, S. and E. Wilde, "The Deprecation HTTP Header Field", <https://datatracker.ietf.org/doc/draft-dalal-deprecation-header/>.

[draft-ietf-jmap-jscontact]
Stepanek, R. and M. Loffredo, "JSContact: A JSON representation of contact data", <https://datatracker.ietf.org/doc/draft-ietf-jmap-jscontact/>.

[draft-ietf-jmap-jscontact-vcard]
Loffredo, M. and R. Stepanek, "JSContact: Converting from and to vCard", <https://datatracker.ietf.org/doc/draft-ietf-jmap-jscontact-vcard/>.

[RDAP-PILOT-WG]
ICANN RDAP Pilot WG, "RDAP Pilot Report", April 2019, <https://www.icann.org/en/system/files/files/rdap-pilot-report-25apr19-en.pdf>.

Appendix A. Change Log

A.1. Change from 00 to 01

1. Changed category from "Best Current Practice" to "Standards Track"
2. Replaced the example of Figure 1
3. Changed the title of the "Migration from JCard to JSCard" section to "Transition Considerations"
4. Added Section 3.1
5. Updated Section 6
6. Updated Section 7
7. Rearranged the description of stage 1 in Section 4.2.1
8. Changed the names of the transition stages 1 and 2
9. Corrected Figure 2, Figure 4, Figure 5
10. Changed the rdapConformance tag "jscard_level_0" to "jscard"
11. Removed the "Best Practices for deprecating a REST API features" section, but added a useful reference.

A.2. Change from 01 to 02

1. Removed the sentence "which cannot be represented using jCard" in Section 1.1.

A.3. Change from 02 to 023

1. Updated section "Conventions Used in This Document".

2. Updated the contact in "IANA Considerations" section.

3. Changed the reference draft-loffredo-jmap-jscontact-vcard to draft-ietf-jmap-jscontact-vcard.

4. Added reference to RFC8174.

5. 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

Gavin Brown
CentralNic Group plc
Saddlers House, 44 Gutter Lane
London, England EC2V 6BR
GB

Phone: +44 20 33 88 0600
Email: gavin.brown@centralnic.com
URI: https://www.centralnic.com