Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 26 August 2021

D. Belyavskiy

J. Gould
VeriSign, Inc.

22 February 2021

Use of Internationalized Email Addresses in EPP protocol draft-belyavskiy-epp-eai-04

Abstract

This document describes an EPP extension that permits usage of Internationalized Email Addresses in the EPP protocol and specifies the terms when it can be used by EPP clients and servers. The Extensible Provisioning Protocol (EPP), being developed before appearing the standards for Internationalized Email Addresses (EAI), does not support such email addresses.

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 26 August 2021.

Copyright Notice

Table of Contents

1. Introduction

[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 that permits usage of Internationalized Email Addresses in the EPP protocol and specifies the terms when it can be used by EPP clients and servers. A new form of EPP extension, referred to as a Functional Extension, is defined and used to apply the rules for the handling of email address elements in all of the [RFC5730] extensions negotiated in the EPP session, which include the object and command-responses extensions. The described mechanism can be applied to any object or command-response extension that uses an email address.

The Extensible Provisioning Protocol (EPP) specified in [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], and command-response extensions applied to a managed object.

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. 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. Email Address Specification

Support of non-ASCII email address syntax is defined in RFC 6530 [RFC6530]. This mapping does not prescribe minimum or maximum lengths for character strings used to represent email addresses. The exact syntax of such addresses is described in Section 3.3 of [RFC6531]. The validation rules introduced in RFC 6531 are considered to be followed.

The definition of email address in the EPP RFCs, including Section 2.6 of [RFC5733] and Section 4.1.2, 4.2.1, and 4.2.5 of [RFC8543], references [RFC5322] for the email address syntax. The

XML schema definition in Section 4 of [RFC5733] and Section 5 of [RFC8543] defines the "email" element using the type "eppcom:minTokenType", which is defined in Section 4.2 of [RFC5730] as an XML schema "token" type with minimal length of one. The XML schema "token" type will fully support the use of EAI addresses, so the primary application of the EAI extension is to apply the use of [RFC6531] instead of [RFC5322] for the email address syntax. Other EPP extensions may follow the formal syntax definition using the XML schema type "eppcom:minTokenType" and the [RFC5322] format specification, where this extension applies to all EPP extensions with the same or similar definitions.

The email address format is formally defined in Section 3.4.1 of [RFC5322], which only consists of printable US-ASCII characters for both the local-part and the domain ABNF rules. In [RFC6531], the extends the Mailbox, Local-part and Domain ABNF rules in [RFC5321] to support "UTF8-non-ascii", defined in Section 3.1 of [RFC6532], for the local-part and U-label, defined in Section 2.3.2.1 of [RFC5890], for the domain. By applying the syntax rules of [RFC5322], the EPP extensions will change from supporting only ASCII characters to supporting Internationailzed characters in the email address local-part and domain-part.

4. Functional Extension

[RFC5730] defines three types of extensions at the protocol, object, and command-response level, which impact the structure of the EPP messages. A Functional Extension applies a functional capability to an existing set of EPP extensions and properties. The scope of the applicable EPP extensions and applicable extension properties are defined in the Functional Extension along with the requirements for the servers and clients that support it. The Functional Extension needs to cover the expected behavior of the supporting client or server when interfacing with an unsupporting client or server. Negotiating support for a Functional Extension is handled using the EPP Greeting and EPP Login services.

5. Internationalized Email Addresses (EAI) Functional Extension

5.1. Scope of Functional Extension

The functional extension applies to all object extensions and command-response extensions negotiated in the EPP session that include email address properties. Examples include the <contact:email> element in the EPP contact mapping [RFC5733] or the <org:email> element in the EPP organization mapping [RFC8543]. All registry zones (e.g., top-level domains) authorized for the client in the EPP session apply. There is no concept of a per-client, per-

zone, per-extension, or per-field setting that is used to indicate support for EAI, but instead it's a global setting that applies to the EPP session.

5.2. Signaling Client and Server Support

The client and the server can signal support for the functional extension using a namespace URI in the login and greeting extension services. The namespace URI "urn:ietf:params:xml:ns:epp:eai-0.2" is used to signal support for the functional extension. 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.

5.3. Functional Extension Behavior

5.3.1. EAI Functional Extension Negotiated

If both client and server have indicated the support of the EAI addresses during the session establishment, it implies possibility to process the EAI address in any message having an email property during the established EPP session. Below are the server and client obligations when the EAI extension has been successfuly negotiated in the EPP session.

The server MUST satisfy the following obligations when the EAI extension has been negotiated:

* Accept EAI compatible addresses for all email properties in the EPP session negotiated object extensions and command-response extensions. For example the <contact:email> element in [RFC5733] and the <org:email> element in [RFC8543].

* Accept EAI compatible addresses for all registry zones (e.g., top-level domains) authorized for the client in the EPP session.

* Email address validation based on EAI validation rules defined in Section 3

* Storage of email properties that supports internationalized characters.

* Return EAI compatible addresses for all email properties in the EPP responses.

The server MUST satisfy the following obligations when THE EAI extension has been negotiated:

* Provide EAI compatible addresses for all e-mail properties in the EPP session negotiated object extensions and command-response extensions. For example the <contact:email> element in [RFC5733] and the <org:email> element in [RFC8543].

* Provide EAI compatible addresses for all registry zones (e.g., top-level domains) authorized for the client in the EPP session.

* Accept EAI compatible addresses in the EPP responses for all email properties in the EPP session negotiated object extensions and command-response extensions.

5.3.2. EAI Functional Extension Not Negotiated

The lack of EAI support can cause data and functional issues, so an EAI supporting client or server needs to handle cases where the opposite party doesn't support EAI. Below are the server and client obligations when the EAI extension is not negotiated due to the lack of support by the opposite party.

The EAI supporting server MUST satisfy the following obligations when the client does not support the EAI extension:

* When the email property is required in the EPP extension command, the server SHOULD validate the email property by the client using the ASCII email validation rules.

* When the email property is optional according the EPP extension command, if the client supplies the email property the server SHOULD validate the email property using the ASCII email validation rules.

* When the email property is required in the EPP extension response, the server MUST validate whether the email property is an EAI address and if so return the predefined placeholder email TBD and otherwise return the email property that has been set.

* When the email property is optional in the EPP extension response, the server MUST validate whether the email property is an EAI address and if so don't return the email property in the response and otherwise return the email property that has been set based on server policy.

The EAI supporting client MUST satisfy the following obligations when the server does not support the EAI extension:

* When the email property is required in the EPP extension command and the email property is an EAI address with no alternative ASCII address, the client MUST provide the predefined placeholder email address TBD.

* When the email property is optional in the EPP extension command and the email property is an EAI address with no alternative ASCII address, the client SHOULD omit the email property.

6. 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].

7. IANA Considerations

7.1. XML Namespace

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in 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-0.2
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-0.2
Registrant Contact: IESG
XML: See the "Formal Syntax" section of this document.

7.2. EPP Extension Registry

The EPP extension described in this document should be registered by 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
```

8. Implementation Considerations

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.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.27487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.27487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/info/rfc5321>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.27487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

[RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.27487/RFC5733, August 2009, <https://www.rfc-editor.org/info/rfc5733>.

[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>.

[RFC6530] Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, DOI 10.27487/RFC6530, February 2012, <https://www.rfc-editor.org/info/rfc6530>.

[RFC6531] Yao, J. and W. Mao, "SMTP Extension for Internationalized Email", RFC 6531, DOI 10.17487/RFC6531, February 2012, <https://www.rfc-editor.org/info/rfc6531>.

[RFC6532] Yang, A., Steele, S., and N. Freed, "Internationalized Email Headers", RFC 6532, DOI 10.17487/RFC6532, February 2012, <https://www.rfc-editor.org/info/rfc6532>.

[RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.27487/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 10.27487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.27487/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 10.27487/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 "0.2" instead of "1.0".

A.3. Change from 02 to 03

1. The approach has changed to use the concept of Functional EPP Extension.

2. The examples are removed

A.4. Change from 03 to 04

1. More detailed reference to email syntax is provided

2. The shortened eai namespace reference is removed

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

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

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

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on June 19, 2021.

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/>
    <csvDomain:fContactType/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file
    cksum="6B976A6C">
    domainContacts-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
  </rdeCsv:csv>
...
</csvDomain:contents>
...
```

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

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

5.1.2.1.3. "domainStatuses" CSV File Definition

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

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

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

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

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

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

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

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

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

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

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

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

5.1.2.1.4. "domainNameServers" CSV File Definition

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

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

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

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

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

- <csvHost:fName> Host name field with type="eppcom:labelType" and isRequired="true".
- <rdeCsv:fRoid> Host object Registry Object IDentifier (ROID) assigned to the host object with isRequired="true".

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

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

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

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

5.1.2.1.5. "domainNameServersAddresses" CSV File Definition

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

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

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

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

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

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

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

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

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

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

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

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

5.1.2.1.6. "dnssec" CSV File Definition

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.1.2.1.7. "domainTransfer" CSV File Definition

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.1.2.2. <csvDomain:deletes>

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

5.1.2.2.1. "domain" Deletes CSV File Definition

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

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

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

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

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

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

5.2. Host Object

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

5.2.1. XML Model

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

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

5.2.1.1. <rdeHost:host> element

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

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

- 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

16.23. Changes REGEXT 10 to REGEXT 11

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

17. Example of a Full Deposit using the XML model

Example of a Full Deposit using the XML model:

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

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

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

     <!-- Contents -->
     <rde:contents>
       <!-- Header -->
       <rdeHeader:header>
         <rdeHeader:tld>test</rdeHeader:tld>
         <rdeHeader:count
           uri="urn:ietf:params:xml:ns:rdeDomain-1.0">2
           </rdeHeader:count>
         <rdeHeader:count
           uri="urn:ietf:params:xml:ns:rdeHost-1.0">1
           </rdeHeader:count>
         <rdeHeader:count
```

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

    <!-- Domain: example1.example -->
    <rdeDomain:domain>
      <rdeDomain:name>example1.example</rdeDomain:name>
      <rdeDomain:roid>Dexample1-TEST</rdeDomain:roid>
      <rdeDomain:status s="ok"/>
      <rdeDomain:registrant>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: October 14, 2021

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

April 19, 2021

Registry Maintenance Notifications for the Extensible Provisioning Protocol (EPP) draft-ietf-regext-epp-registry-maintenance-14

Abstract

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

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 October 14, 2021.

Table of Contents

1. Introduction

Registries usually conduct maintenances and inform registrars in different ways. Given the DNS namespace expansion, it is now desirable to provide methods for EPP servers to notify EPP clients and EPP clients can 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 upcoming maintenances.

1.1. Terminology and Definitions

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 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:epp: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. 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.

Servers SHOULD (for a temporary migration period up to server policy) provide support for older versions of the extension in parallel to the newest version and allow clients to execute their preferred version of the <info> command based on the maintenance <objURI> elements of the server <greeting>. The version of the maintenance <info> response MUST match the version of the maintenance <info> command executed by the server.

Servers MUST return a Registry Maintenance Notification poll message matching the newest version of the maintenance extension, based on an intersection of the maintenance <objURI> elements in the server <greeting> and the client <login> command. If the intersection of the maintenance <objURI> elements of the server <greeting> and the

client <login> command results in an empty set, the server MUST return the newest version of the Registry Maintenance Notification poll message supported by the server based on section 6 "Usage with Poll Message EPP Response" of [draft-ietf-regext-unhandled-namespaces].

3. Object Attributes

3.1. Internationalized Domain Names

Names of affected hosts MUST be provided as an A-label according to [RFC5891].

3.2. 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.

3.3. Maintenance Elements

The <maint:item> element describes a single registry maintenance event during a specific period. This element is used in a maintenance item EPP <info> response and <poll> message.

<maint:id>
: The server unique identifier for the maintenance with the OPTIONAL "name" attribute that includes a human-readable name of the maintenance. The server unique identifier SHALL NOT be changed if the maintenance is updated or deleted. When the "name" attribute is set, the OPTIONAL "lang" attribute MAY be present to identify the language if the negotiated value is something other than the default value of "en" (English).

<maint:type>
: Zero or more OPTIONAL types of the maintenance that has the possible set of values defined by server policy, such as "Routine Maintenance", "Software Update", "Software Upgrade", or "Extended Outage".

<maint:pollType>
: The OPTIONAL <maint:pollType> element for a Registry Maintenance Notification poll message; values MUST either be "create", "update", "delete", "courtesy", or "end". For the "create" and "update" types, the server includes the state of the maintenance after the create or update. For the "delete" type, the server includes the state of the maintenance prior to the delete. The "courtesy" provides a reminder of a maintenance and the "end" provides a notification of the end of the maintenance without updating the maintenance object and includes the latest state of the maintenance. This element MUST be present only for poll messages.

<maint:systems>
One or more <maint:system> elements that are affected by the maintenance.

<maint:system>
The <maint:system> element contains the following child elements:

<maint:name>
The name of the affected system, such as "EPP", "WHOIS", "DNS", "Portal", etc.

<maint:host>
The OPTIONAL affected maintained system's hostname, which SHALL be an A-label according to [RFC5891].

<maint:impact>
The impact level; the values MUST either be "full", "partial", or "none". If access is intermittently unavailable, it is "partial". If access is completely unavailable, it is "full". If access is not affected, it is "none".

<maint:environment>
The type of the affected system; the attribute "type" is REQUIRED and MUST either be "production", "ote", "staging", "dev" or "custom". For extensibility, the <maint:environment> element includes the OPTIONAL "name" attribute that can define the name of the custom environment when the <maint:environment> element "type" attribute has the "custom" value. For example, for the custom "marketing" environment, the <maint:environment> element should be:
<maint:environment type="custom" name="marketing"/>

<maint:start>
The date and time of the start of the maintenance.

<maint:end>
The date and time of the end of the maintenance. The <maint:end> element MUST be equal to or greater than the <maint:start> element.

<maint:reason>
The reason behind the maintenance; the values MUST either be "planned" or "emergency".

<maint:detail>
The OPTIONAL URI to detailed maintenance description, formatted according to [RFC8820].

<maint:description>
Zero or more OPTIONAL free-form descriptions of the maintenance without having to create and traverse an external resource defined by the <maint:detail> element. The OPTIONAL "lang" attribute MAY be present to identify the language if the negotiated value is something other then the default value of "en" (English). The OPTIONAL "type" attribute MAY be present to identify the format of the description. It MUST either be "plain" for plain text or "html" HTML text that is defined in [W3C-HTML5] and XML-escaped, with a default value of "plain".

<maint:tlds>
The OPTIONAL <maint:tlds> element contains one or more <maint:tld> child elements. If the <maint:tlds> is not present, the entire system is affected.

<maint:tld>
The affected top-level domain or registry zone, which SHALL be an A-label according to [RFC5891].

<maint:intervention>
The OPTIONAL <maint:intervention> element contains the following child elements:

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

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

<maint:crDate>
The date and time of the maintenance object creation.

<maint:upDate>
The OPTIONAL date and time of the most recent maintenance object modification. This element MUST NOT be present if the maintenance object has never been modified.

4. EPP Command Mapping

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

4.1. 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.

4.1.1. EPP <check> Command

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

4.1.2. EPP <transfer> Command

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

4.1.3. 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 the <maint:id> child element, described in Section 4.1.3.1, to query for a specific maintenance item or the <maint:list> child element, described in Section 4.1.3.2, to query all maintenance items.

4.1.3.1. Info Maintenance Item

The information on a specific maintenance item can be retrieved by using the <info> command with the <maint:info> element and the <maint:id> child element, defined in Section 3.3. If the maintenance identifier does not exist, the server MUST return an EPP error result code of 2303 [RFC5730].

Example to retrieve a specific maintenance item in an <info> command.

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

When an <info> command has been processed successfully, the EPP <resData> element MUST contain a child <maint:infData> element that identifies the maintenance namespace. The <maint:infData> element contains the <maint:item> element defined in Section 3.3.

Example of returning a specific maintenance item in an <info> response.

```
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:epp:maintenance-1.0">
S:          <maint:item>
S:            <maint:id>2e6df9b0-4092-4491-bcc8-9fb2166dcee6
S:            </maint:id>
S:            <maint:type lang="en">Routine Maintenance</maint:type>
S:            <maint:systems>
S:              <maint:system>
S:                <maint:name>EPP</maint:name>
S:                <maint:host>epp.registry.example
S:                </maint:host>
S:                <maint:impact>full</maint:impact>
S:              </maint:system>
S:            </maint:systems>
S:            <maint:environment type="production"/>
S:            <maint:start>2021-12-30T06:00:00Z</maint:start>
S:            <maint:end>2021-12-30T14:25:57Z</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:description lang="de">Freitext
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:crDate>2021-09-08T22:10:00Z</maint:crDate>
S:          </maint:item>
S:        </maint:infData>
S:      </resData>
S:      <trID>
S:         <clTRID>ABC-12345</clTRID>
S:         <svTRID>54321-XYZ</svTRID>
S:      </trID>
S:   </response>
S:</epp>
```

4.1.3.2. Info Maintenance List

The information for a list of maintenance items can be retrieved by

using the <info> command with the <maint:info> element and the empty <maint:list> child element. Server policy determines if previous maintenances will be included in the list of maintenance items.

Example to retrieve the maintenance list in an <info> command.

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

When an <info> command has been processed successfully, the EPP <resData> element MUST contain a child <maint:infData> element that identifies the maintenance namespace. The <maint:infData> element contains the <maint:list> element with zero or more <maint:listItem> child elements. The <maint:listItem> element contains the following child elements:

<maint:id>
The <maint:id> element defined in Section 3.3.

<maint:start>
The <maint:start> element defined in Section 3.3.

<maint:end>
The <maint:end> element defined in Section 3.3.

<maint:crDate>
The <maint:crDate> element defined in Section 3.3.

<maint:upDate>
The OPTIONAL <maint:upDate> element defined in Section 3.3.

Example of returning the maintenance list in an <info> response.

```
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:epp:maintenance-1.0">
S:          <maint:list>
S:            <maint:listItem>
```

```
S:              <maint:id>2e6df9b0-4092-4491-bcc8-9fb2166dcee6
S:              </maint:id>
S:              <maint:start>2021-12-30T06:00:00Z</maint:start>
S:              <maint:end>2021-12-30T07:00:00Z</maint:end>
S:              <maint:crDate>2021-09-08T22:10:00Z</maint:crDate>
S:            </maint:listItem>
S:            <maint:listItem>
S:              <maint:id>91e9dabf-c4e9-4c19-a56c-78e3e89c2e2f
S:              </maint:id>
S:              <maint:start>2021-12-15T04:30:00Z</maint:start>
S:              <maint:end>2021-12-15T05:30:00Z</maint:end>
S:              <maint:crDate>2021-09-08T22:11:00Z</maint:crDate>
S:              <maint:upDate>2021-10-17T15:00:00Z</maint:upDate>
S:            </maint:listItem>
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>
```

4.1.4. 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 five types of poll messages, defined by the <maint:pollType> element in Section 3.3. A poll message applies when a maintenance is created, updated or deleted. A courtesy poll message can be sent as a reminder of an impending maintenance. An end poll message can be sent when the maintenance is completed. 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 contains the <maint:item> element defined in Section 3.3.

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:

```
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>2021-10-08T22:10:00Z</qDate>
S:    <msg lang="en">Registry Maintenance Notification</msg>
S:  </msgQ>
S:  <resData>
S:    <maint:infData
S:      xmlns:maint="urn:ietf:params:xml:ns:epp:maintenance-1.0">
S:      <maint:item>
S:        <maint:id>2e6df9b0-4092-4491-bcc8-9fb2166dcee6</maint:id>
S:        <maint:pollType>create</maint:pollType>
S:        <maint:systems>
S:          <maint:system>
S:            <maint:name>EPP</maint:name>
S:            <maint:host>epp.registry.example
S:            </maint:host>
S:            <maint:impact>full</maint:impact>
S:          </maint:system>
S:        </maint:systems>
S:        <maint:environment type="production"/>
S:        <maint:start>2021-12-30T06:00:00Z</maint:start>
S:        <maint:end>2021-12-30T14:25:57Z</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:crDate>2021-10-08T22:10:00Z</maint:crDate>
S:      </maint:item>
S:    </maint:infData>
S:  </resData>
S:  <trID>
S:    <clTRID>ABC-12345</clTRID>
S:    <svTRID>54321-XYZ</svTRID>
S:  </trID>
S:  </response>
S:</epp>
```

4.2. 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.

4.2.1. EPP <create> Command

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

4.2.2. EPP <delete> Command

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

4.2.3. EPP <renew> Command

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

4.2.4. EPP <transfer> Command

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

4.2.5. EPP <update> Command

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

5. Formal Syntax

The EPP Registry Maintenance schema is presented here.
The formal syntax presented here is a complete 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 schema; they are used to note the beginning and end of the schema for URI registration purposes.

5.1. Registry Maintenance EPP Mapping Schema

```
<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
  <schema targetNamespace="urn:ietf:params:xml:ns:epp:
    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:epp: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
    Registry 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="name" 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="item" type="maint:maintDataType"/>
  </choice>
</complexType>
```

```
    <!--
     Attributes associated with the list info response
    -->
    <complexType name="listDataType">
      <sequence>
        <element name="listItem" 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"/>
        <element name="end" type="dateTime"/>
        <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="type" type="maint:typeType" minOccurs="0"
         maxOccurs="unbounded"/>
        <element name="pollType" type="maint:pollType" minOccurs="0"/>
        <element name="systems" type="maint:systemsType"/>
        <element name="environment" type="maint:envType"/>
        <element name="start" type="dateTime"/>
        <element name="end" type="dateTime"/>
        <element name="reason" type="maint:reasonEnum"/>
        <element name="detail" type="anyURI" minOccurs="0"/>
        <element name="description" type="maint:descriptionType"
         minOccurs="0" maxOccurs="unbounded"/>
        <element name="tlds" type="maint:tldsType" minOccurs="0"/>
        <element name="intervention" type="maint:interventionType"
         minOccurs="0"/>
        <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 poll types
    -->
    <simpleType name="pollType">
      <restriction base="token">
        <enumeration value="create"/>
        <enumeration value="update"/>
        <enumeration value="delete"/>
        <enumeration value="courtesy"/>
        <enumeration value="end"/>
      </restriction>
    </simpleType>
    <!--
     Enumerated list of impacts
    -->
    <simpleType name="impactEnum">
      <restriction base="token">
        <enumeration value="none"/>
        <enumeration value="partial"/>
        <enumeration value="full"/>
      </restriction>
    </simpleType>
    <!--
     description element
    -->
    <complexType name="descriptionType">
      <simpleContent>
        <extension base="string">
          <attribute name="lang" type="language" default="en"/>
          <attribute name="type" type="maint:descEnum" default="plain"
          />
        </extension>
      </simpleContent>
    </complexType>
    <!--
     Enumerated list of description mime types
    -->
    <simpleType name="descEnum">
      <restriction base="token">
        <enumeration value="plain"/>
        <enumeration value="html"/>
      </restriction>
    </simpleType>
    <!--
     type element
    -->
    <complexType name="typeType">
      <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="eppcom:labelType" minOccurs="0"/>
        <element name="impact" type="maint:impactEnum"/>
      </sequence>
    </complexType>
    <!--
     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>
     <!--
      End of schema.
     -->
     </schema>
    <CODE ENDS>
```

6. IANA Considerations

6.1. 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:epp: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.

6.2. 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

7. Security Considerations

A server MUST only provide maintenance information for clients that are authorized. If a client queries for a maintenance identifier, per Section 4.1.3.1 "Info Maintenance Item", that it is not authorized to access, the server MUST return an EPP error result code of 2201 [RFC5730]. The list of top-level domains or registry zones returned in the "Info Maintenance Item" response SHOULD be filtered based on the top-level domains or registry zones the client is authorized. Authorization of poll messages is done at the time of poll message insertion and not at the time of poll message consumption.

8. Implementation Status

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

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

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

8.1. GoDaddy Registry

Organization: GoDaddy Registry

Name: GoDaddy Registry

Description: GoDaddy Registry provides maintenance notifications to their registrars.

Level of maturity: Production

Coverage: All aspects of the protocol according to the draft version 2 are implemented with further updates to come.

Licensing: Proprietary

Contact: quoc@registry.godaddy

URL: https://registry.godaddy

8.2. TANGO Registry Services

Name: TANGO Registry Services

Description: TANGO Registry Services provides maintenance notifications to their registrars.

Level of maturity: Beta

Coverage: All aspects of the protocol according to the draft version 12 are implemented with further updates to come.

Licensing: Proprietary

Contact: Michael.Bauland@knipp.de

URL: https://tango-rs.com

9. References

9.1. Normative References

[I.D.draft-ietf-regext-unhandled-namespaces]
Gould, J. and Casanova, M., "Extensible Provisioning Protocol (EPP) Unhandled Namespaces", <https://datatracker.ietf.org/doc/draft-ietf-regext-unhandled-namespaces/> (work in progress), February 2021.

[W3C-HTML5] Hickson, I., Berjon, R., Faulkner, S., Leithead, T., Doyle Navara, E., O'Connor, E., and S. Pfeiffer, "HTML5", W3C Recommendation REC-html5-20141028, October 2014, <http://www.w3.org/TR/2014/REC-html5-20141028/>.

Latest version available at <http://www.w3.org/TR/html5/>.

[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>.

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

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

[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>.

[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>.

[RFC8820] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 8820, DOI 10.17487/RFC8820, June 2020, <https://www.rfc-editor.org/info/rfc8820>.

9.2. Informative References

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

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.

A.7. Change from 04 to 05

Changed XML namespace from urn:ietf:params:xml:ns:maintenance-1.0 to urn:ietf:params:xml:ns:epp:maintenance-0.1. Removed <maint:status>. Clarified <maint:info> for retrieving maintenance items and the list.

A.8. Change from 05 to 06

Changed dates in examples to more recent dates. Renamed Query Maintenance Item and List to Info Maintenance Item and List. Removed blackout in favor of full. Added GoDaddy Registry implementation.

A.9. Change from 06 to 07

Removed IP addresses for <maint:host>. Editoral changes.

A.10. Change from 07 to 08

Editoral changes. Changed XML namespace and schema from 0.1 to 0.2. Added pollType to reflect create, update, or delete maintenance poll messages.

A.11. Change from 08 to 09

Editoral changes. Added new section "Migrating to Newer Versions of This Extension".

A.12. Change from 09 to 10

Editoral changes. Renamed "msg" to "name". Added "courtesy" and "end" to pollType.

A.13. Change from 10 to 11

Editoral changes. Added mime type to description.

A.14. Change from 11 to 12

Editoral changes. Changed XML namespace from 0.2 to 0.3.

A.15. Change from 12 to 13

Editoral changes. Added TANGO Registry Services to Section 8. Added Michael Bauland to acknowledgments. Added "none" to <maint:impact>.

A.16. Change from 13 to 14

Accepted in WGLC. Changed XML namespace from 0.3 to 1.0.

Acknowledgments

The authors wish to thank the following persons for their feedback and suggestions: James Gould, Michael Bauland, Patrick Mevzek, Quoc-Anh Pham, Raymond Zylstra, Christopher Martens, Anthony Eden, Neal McPherson, Craig Marchant, and Andreas Huber.

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

REGEXT Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 9, 2021

S. Hollenbeck
Verisign Labs

January 5, 2021

Federated Authentication for the Registration Data Access Protocol (RDAP) using OpenID Connect
draft-ietf-regext-rdap-openid-06

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on July 9, 2021.

Table of Contents

1. Introduction

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

RDAP is specified in multiple documents, including "HTTP Usage in the Registration Data Access Protocol (RDAP)" [RFC7480], "Security Services for the Registration Data Access Protocol (RDAP)" [RFC7481], "Registration Data Access Protocol Query Format" [RFC7482], and "JSON Responses for the Registration Data Access Protocol (RDAP)" [RFC7483]. RFC 7481 describes client identification and authentication services that can be used with RDAP, but it does not specify how any of these services can (or should) be used with RDAP.

1.1. Problem Statement

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

1.2. Proposal

A basic level of RDAP service can be provided to users who possess an identifier issued by a recognized provider who is able to authenticate and validate the user. The identifiers issued by social media services, for example, can be used. Users who require higher levels of service (and who are willing to share more information

about them self to gain access to that service) can secure identifiers from specialized providers who are or will be able to provide more detailed information about the user. Server operators can then make access control decisions based on the identification information provided by the user.

A federated authentication system would make it easier to operate and use RDAP by re-using existing identifiers to provide a basic level of access. It can also provide the ability to collect additional user identification information, and that information can be shared with the consent of the user. This document describes a federated authentication system for RDAP based on OpenID Connect [OIDC] that meets all of these needs.

2. Conventions Used in This Document

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

3. Federated Authentication for RDAP

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

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

3.1. RDAP and OpenID Connect

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

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

3.1.1. Terminology

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

3.1.2. Overview

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

1. An RDAP client (acting as an OpenID End-User) sends an HTTP (or HTTPS) query containing OAuth 2.0 request parameters to an RDAP server.
2. The RDAP server (acting as an OpenID Relying Party (RP)) prepares an Authentication Request containing the desired request parameters.
3. The RDAP server sends the RDAP client and Authentication Request to an Authorization Server operated by an OpenID Provider (OP) using an HTTP redirect.
4. The Authorization Server authenticates the RDAP Client.
5. The Authorization Server obtains RDAP Client consent/ authorization.

6. The Authorization Server sends the RDAP Client back to the RDAP server with an Authorization Code using an HTTP redirect.
7. The RDAP server requests a response using the Authorization Code at the Token Endpoint.
8. The RDAP server receives a response that contains an ID Token and Access Token in the response body.
9. The RDAP server validates the ID Token and retrieves the RDAP client's Subject Identifier.

The RDAP server can then make identification, authorization, and access control decisions based on local policies, the ID Token received from the OP, and the received Claims. Note that OpenID Connect describes different process flows for other types of clients, such as script-based or command line clients.

3.1.3. RDAP Authentication and Authorization Steps

End-Users MUST possess an identifier (an OpenID) issued by an OP to use OpenID Connect with RDAP. An OP MUST include support for the claims described in Section 3.1.4 to provide additional information needed for RDAP End-User authorization. OpenID Connect requires RPs to register with OPs to use OpenID Connect services for an End-User. That process is described by the "OpenID Connect Dynamic Client Registration" protocol [OIDCR].

3.1.3.1. Provider Discovery

An RDAP server/RP needs to receive an identifier from an End-User that can be used to discover the End-User's OP. That process is required and is documented in the "OpenID Connect Discovery" protocol [OIDCD].

3.1.3.2. Authentication Request

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

The benefits of using the Authorization Code Flow for authenticating a human user are described in Section 3.1 of the OpenID Connect Core protocol. The Implicit Flow is more commonly used by clients implemented in a web browser using a scripting language; it is described in Section 3.2 of the OpenID Connect Core protocol. The

Hybrid Flow (described in Section 3.3 of the OpenID Connect Core protocol) combines elements of the Authorization and Implicit Flows by returning some tokens from the Authorization Endpoint and others from the Token Endpoint.

An Authentication Request can contain several parameters. REQUIRED parameters are specified in Section 3.1.2.1 of the OpenID Connect Core protocol [OIDCC]. Other parameters MAY be included.

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

3.1.3.3. End-User Authorization

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

3.1.3.4. Authorization Response and Validation

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

3.1.3.5. Token Processing

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

3.1.3.6. Delivery of User Information

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

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

3.1.4. Specialized Claims for RDAP

OpenID Connect claims are pieces of information used to make assertions about an entity. Section 5 of the OpenID Connect Core protocol [OIDCC] describes a set of standard claims that can be used to identify a person. Section 5.1.2 notes that additional claims MAY be used, and it describes a method to create them.

3.1.4.1. Stated Purpose

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

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

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

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

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

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

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

This registry is operated under the "Specification Required" policy defined in RFC 5226 ([RFC5226]). The set of initial values used to populate the registry as described in Section 6.3 are taken from the final report [1] produced by the Expert Working Group on gTLD Directory Services chartered by the Internet Corporation for Assigned Names and Numbers (ICANN).

3.1.4.2. Do Not Track

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

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

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

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

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

4. Protocol Parameters

This specification adds the following protocol parameters to RDAP:

1. A query parameter to request authentication for a specific end-user identity.
2. A path segment to request an ID Token and an Access Token for a specific end-user identity.

3. A query parameter to deliver an ID Token and an Access Token for use with an RDAP query.

4.1. Client Authentication Request and Response

Client authentication is requested by adding a query component to an RDAP request URI using the syntax described in Section 3.4 of RFC 3986 [RFC3986]. The query used to request client authentication is represented as a "key=value" pair using a key value of "id" and a value component that contains the client identifier issued by an OP. An example:

https://example.com/rdap/domain/example.com?id=user.idp.example

The response to an authenticated query MUST use the response structures specified in RFC 7483 [RFC7483]. Information that the end-user is not authorized to receive MUST be omitted from the response.

4.2. Token Request and Response

Clients MAY send a request to an RDAP server to authenticate an end-user and return an ID Token and an Access Token from an OP that can be then be passed to the RP/RDAP server to authenticate and process subsequent queries. Identity provider authentication is requested using a "tokens" path segment and a query parameter with key value of "id" and a value component that contains the client identifier issued by an OP. An example:

https://example.com/rdap/tokens?id=user.idp.example

In addition to any core RDAP response elements, the response to this query MUST contain four name-value pairs, in any order, representing the returned ID Token and Access Token. The ID Token is represented using a key value of "id_token". The Access Token is represented using a key value of "access_token". The access token type is represented using a key value of "token_type" and a value of "bearer" as described in Sections 4.2.2 and 7.1 of RFC 6749 [RFC6749]. The lifetime of the access token is represented using a key value of "expires_in" and a numerical value that describes the lifetime in seconds of the access token as described in Section 4.2.2 of RFC 6749 [RFC6749]. The token values returned in the RDAP server response MUST be Base64url encoded as described in RFCs 7515 [RFC7515] and 7519 [RFC7519].

An example (the encoded tokens have been abbreviated for clarity):

```
{
 "access_token" : "eyJ0...NiJ9",
 "id_token" : "eyJ0...EjXk",
 "token_type" : "bearer",
 "expires_in" : "3600"
}
```

Figure 1

An RDAP server that processes this type of query MUST determine if the identifier is associated with an OP that is recognized and supported by the server. Servers MUST reject queries that include an identifier associated with an unsupported OP with an HTTP 501 (Not Implemented) response. An RDAP server that receives a query containing an identifier associated with a recognized OP MUST perform the steps required to authenticate the user with the OP using a browser or browser-like client and return encoded tokens to the client. Note that tokens are typically valid for a limited period of time and new tokens will be required when an existing token's validity period has expired.

The tokens can then be passed to the server for use with an RDAP query by passing the encoded ID Token as a query parameter with a key value of "id_token" and the encoded Access Token in an HTTP Bearer authorization header [RFC6750]. An example (the encoded tokens have been abbreviated and the URI split across multiple lines for clarity):

https://example.com/rdap/domain/example.com?id_token=eyJ0...EjXk

Authorization: Bearer eyJ0...NiJ9

The response to an authenticated query MUST use the response structures specified in RFC 7483 [RFC7483]. Information that the end-user is not authorized to receive MUST be omitted from the response.

4.3. Token Refresh and Revocation

An access token can be refreshed as described in Section 12 of the OpenID Connect Core protocol [OIDCC] and Section 6 of OAuth 2.0 [RFC6749]. Clients can take advantage of this functionality if it is supported by the OP and accepted by the RDAP server.

A refresh token is requested using a "tokens" path segment and two query parameters. The first query parameter includes a key value of

"id" and a value component that contains the client identifier issued by an OP. The second query parameter includes a key value of "refresh" and a value component of "true". A value component of "false" MUST be processed to return a result that is consistent with not including a "refresh" parameter at all as described in Section 4.2. An example using "refresh=true":

https://example.com/rdap/tokens?id=user.idp.example
&refresh=true

The response to this query MUST contain all of the response elements described in Section 4.2. In addition, the response MUST contain a name-value pair that represents a refresh token. The name-value pair includes a key value of "refresh_token" and a Base64url-encoded value that represents the refresh token.

Example refresh token request response (the encoded tokens have been abbreviated for clarity):

```
{
 "access_token" : "eyJ0...NiJ9",
 "id_token" : "eyJ0...EjXk",
 "token_type" : "bearer",
 "expires_in" : "3600",
 "refresh_token" : "eyJ0...c8da"
}
```

Figure 2

Once acquired, a refresh token can be used to refresh an access token. An access token is refreshed using a "tokens" path segment and two query parameters. The first query parameter includes a key value of "id" and a value component that contains the client identifier issued by an OP. The second query parameter includes a key value of "refresh_token" and a Base64url-encoded value that represents the refresh token. An example:

https://example.com/rdap/tokens?id=user.idp.example
&refresh_token=eyJ0...f3jE

In addition to any core RDAP response elements, the response to this query MUST contain four name-value pairs, in any order, representing a returned Refresh Token and Access Token. The Refresh Token is represented using a key value of "refresh_token". The Access Token is represented using a key value of "access_token". The access token type is represented using a key value of "token_type" and a value of "bearer" as described in Sections 4.2.2 and 7.1 of RFC 6749 [RFC6749]. The lifetime of the access token is represented using a

key value of "expires_in" and a numerical value that describes the lifetime in seconds of the access token as described in Section 4.2.2 of RFC 6749 [RFC6749]. The token values returned in the RDAP server response MUST be Base64url encoded as described in RFCs 7515 [RFC7515] and 7519 [RFC7519].

Example access token refresh response (the encoded tokens have been abbreviated for clarity):

```
{
 "access_token" : "0dac...13b0",
 "refresh_token" : "f735...d30c",
 "token_type" : "bearer",
 "expires_in" : "3600"
}
```

Figure 3

Access and refresh tokens can be revoked as described in RFC 7009 [RFC7009] by sending a request to an RDAP server that contains a "tokens/revoke" path segment and two query parameters. The first query parameter includes a key value of "id" and a value component that contains the client identifier issued by an OP. The second query parameter includes a key value of "token" and a Base64url-encoded value that represents either the current refresh token or the associated access token. An example:

```
https://example.com/rdap/tokens/revoke?id=user.idp.example
&token=f735...d30c
```

Note that this command will revoke both access and refresh tokens at the same time. In addition to any core RDAP response elements, the response to this query MUST contain a description of the result of processing the revocation request within the RDAP "notices" data structure.

Example token revocation success:

```
"notices" :
[
  {
    "title" : "Token Revocation Result",
    "description" : "Token revocation succeeded.",
  }
],
"lang" : "en-US"
```

Figure 4

Example token revocation failure:

```
"notices" :
[
  {
    "title" : "Token Revocation Result",
    "description" : "Token revocation failed.",
  }
],
"errorCode" : 400,
"lang" : "en-US"
```

Figure 5

4.4. Token Exchange

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

4.5. Parameter Processing

Unrecognized query parameters MUST be ignored. An RDAP request that does not include an "id" query component MUST be processed as an unauthenticated query. An RDAP server that processes an authenticated query MUST determine if the identifier is associated with an OP that is recognized and supported by the server. Servers MUST reject queries that include an identifier associated with an unsupported OP with an HTTP 501 (Not Implemented) response. An RDAP server that receives a query containing an identifier associated with a recognized OP MUST perform the steps required to authenticate the user with the OP, process the query, and return an RDAP response that is appropriate for the end user's level of authorization and access.

An RDAP server that receives a query containing tokens associated with a recognized OP and authenticated end user MUST process the query and return an RDAP response that is appropriate for the end user's level of authorization and access. Errors based on processing either the ID Token or the Access Token MUST be signaled with an appropriate HTTP status code as described in Section 3.1 of RFC 6750 [RFC6750].

On receiving a query containing tokens, the RDAP server MUST validate the ID Token. It can do this independently of the OP, because the ID Token is a JWT that contains all the data necessary for validation. The Access Token, however, is an opaque value, and can only be validated by sending a request using it to the UserInfo Endpoint and confirming that a successful response is received. This is different from the OpenID Connect Authorization Code and Implicit flows, where the Access Token can be validated against the at_hash claim from the ID Token. With a query containing tokens, the Access Token might not validate against the at_hash claim because the Access Token may have been refreshed since the ID Token was issued.

An RDAP server that processes requests without needing the UserInfo claims does not need to retrieve the claims merely in order to validate the Access Token. Similarly, an RDAP server that has cached the UserInfo claims for an end user, in accordance with the HTTP headers of a previous UserInfo Endpoint response, does not need to retrieve those claims again in order to revalidate the Access Token.

4.6. RDAP Conformance

RDAP responses that contain values described in this document MUST indicate conformance with this specification by including an rdapConformance ([RFC7483]) value of "rdap_openidc_level_0". The information needed to register this value in the RDAP Extensions Registry is described in Section 6.1.

Example rdapConformance structure with extension specified:

```
"rdapConformance" :
[
  "rdap_level_0",
  "rdap_openidc_level_0"
]
```

Figure 6

5. Clients with Limited User Interfaces

The flow described in Section 3.1.3 requires a client to interact with a server using a web browser. This will not work well in situations where the client is automated or an end-user is using a command line user interface such as curl [2] or wget [3]. There are multiple ways to address this limitation using a web browser on a second device. Two are described here.

5.1. OAuth 2.0 Device Authorization Grant

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

5.2. Manual Token Management

A second method of requesting user authorization from a constrained device is possible by producing and managing tokens manually as follows:

1. Authenticate with the OP as described in Section 4.2 using a browser or browser-like client.
2. Store the returned ID Token and Access Token locally.
3. Send a request to the content provider/RP along with the ID Token and Access Token received from the OP.

The Access Token MAY be passed to the RP in an HTTP "Authorization" header [RFC7235] or as a query parameter. The Access Token MUST be specified using the "Bearer" authentication scheme [RFC6750] if it is passed in an "Authorization" header. The ID Token MUST be passed to the RP as a query parameter.

Here are two examples using the curl and wget utilities. Start by authenticating with the OP:

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

Save the token information and pass it to the RP along with the URI representing the RDAP query. Using curl (encoded tokens have been abbreviated for clarity:

```
curl -H "Authorization: Bearer eyJ0...NiJ9"\
-k https://example.com/rdap/domain/example.com\
?id_token=eyJ0...EjXk

curl -k https://example.com/rdap/domain/example.com\
?id_token=eyJ0...EjXk&access_token=eyJ0...NiJ9
```

Using wget:

```
wget --header="Authorization: Bearer eyJ0...NiJ9"\
https://example.com/rdap/domain/example.com\
```

```
?id_token=eyJ0...EjXk

wget https://example.com/rdap/domain/example.com\
?id_token=eyJ0...EjXk&access_token=eyJ0...NiJ9
```

Refresh tokens can be useful to automated or command line clients who wish to continue a session without explicitly re-authenticating an end user. See Section 4.3 for more information.

6. IANA Considerations

6.1. RDAP Extensions Registry

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

Extension identifier: rdap_openidc_level_0
Registry operator: Any
Published specification: This document.
Contact: IESG <iesg@ietf.org>
Intended usage: This extension describes a federated authentication method for RDAP using OAuth 2.0 and OpenID Connect.

6.2. JSON Web Token Claims Registry

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

Claim Name: "purpose"
Claim Description: This claim describes the stated purpose for submitting a request to access a protected RDAP resource.
Change Controller: IESG
Specification Document(s): Section 3.1.4.1 of this document.

Claim Name: "dnt"
Claim Description: This claim contains a JSON boolean literal that describes an End-User's "do not track" preference for identity tracking, logging, or recording when accessing a protected RDAP resource.
Change Controller: IESG
Specification Document(s): Section 3.1.4.2 of this document.

6.3. RDAP Query Purpose Registry

IANA is requested to create a new protocol registry to manage RDAP query purpose values. This registry should appear under its own heading on IANA's protocol listings, using the same title as the name of the registry. The information to be registered and the procedures

to be followed in populating the registry are described in Section 3.1.4.1.

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

Section at http://www.iana.org/protocols:

Registry Title: Registration Data Access Protocol (RDAP) Query Purpose Values

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

Registration Procedure: Specification Required

Reference: This draft

Required information: See Section 3.1.4.1.

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

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

Initial assignments and reservations:

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

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

-----BEGIN FORM-----
Value: technicalIssueResolution

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

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

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

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

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

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

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

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

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

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 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".

7.1. Verisign Labs

Responsible Organization: Verisign Labs
Location: https://rdap.verisignlabs.com/
Description: This implementation includes support for domain registry RDAP queries using live data from the .cc and .tv country code top-level domains and the .career generic top-level domain. Three access levels are provided based on the authenticated identity of the client:

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

Level of Maturity: This is a "proof of concept" research implementation.
Coverage: This implementation includes all of the features described in this specification.
Contact Information: Scott Hollenbeck, shollenbeck@verisign.com

7.2. Viagenie

Responsible Organization: Viagenie
Location: https://auth.viagenie.ca
Description: This implementation is an OpenID identity provider enabling users and registries to connect to the federation. It also includes a barebone RDAP client and RDAP server in order to test the authentication framework. Various level of purposes are available for testing.
Level of Maturity: This is a "proof of concept" research implementation.
Coverage: This implementation includes most features described in this specification as an identity provider.
Contact Information: Marc Blanchet, marc.blanchet@viagenie.ca

8. Security Considerations

Security considerations for RDAP can be found in RFC 7481 [RFC7481]. Security considerations for OpenID Connect Core [OIDCC] and OAuth 2.0 [RFC6749] can be found in their reference specifications. OpenID Connect defines optional mechanisms for robust signing and encryption that can be used to provide data integrity and data confidentiality services as needed. Security services for ID Tokens and Access Tokens (with references to the JWT specification) are described in the OpenID Connect Core protocol.

8.1. Authentication and Access Control

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

9. Acknowledgements

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

10. References

10.1. Normative References

[OIDC] OpenID Foundation, "OpenID Connect", <http://openid.net/connect/>.

[OIDCC] OpenID Foundation, "OpenID Connect Core incorporating errata set 1", November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.

[OIDCD] OpenID Foundation, "OpenID Connect Discovery 1.0 incorporating errata set 1", November 2014, <http://openid.net/specs/openid-connect-discovery-1_0.html>.

[OIDCR] OpenID Foundation, "OpenID Connect Dynamic Client Registration 1.0 incorporating errata set 1", November 2014, <http://openid.net/specs/openid-connect-registration-1_0.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

[RFC7009] Lodderstedt, T., Ed., Dronia, S., and M. Scurtescu, "OAuth 2.0 Token Revocation", RFC 7009, DOI 10.17487/RFC7009, August 2013, <https://www.rfc-editor.org/info/rfc7009>.

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

[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>.

[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>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

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

[RFC8628] Denniss, W., Bradley, J., Jones, M., and H. Tschofenig, "OAuth 2.0 Device Authorization Grant", RFC 8628, DOI 10.17487/RFC8628, August 2019, <https://www.rfc-editor.org/info/rfc8628>.

[RFC8693] Jones, M., Nadalin, A., Campbell, B., Ed., Bradley, J., and C. Mortimore, "OAuth 2.0 Token Exchange", RFC 8693, DOI 10.17487/RFC8693, January 2020, <https://www.rfc-editor.org/info/rfc8693>.

10.2. Informative References

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

10.3. URIs

[1] https://www.icann.org/en/system/files/files/final-report-06jun14-en.pdf

[2] http://curl.haxx.se/

[3] https://www.gnu.org/software/wget/

Appendix A. Change Log

00: Initial working group version ported from draft-hollenbeck-regext-rdap-openid-10.
01: Modified ID Token delivery approach to note proper use of an HTTP bearer authorization header.
02: Modified token delivery approach (access token is the bearer token) to note proper use of an HTTP bearer authorization header, fixing the change made in -01.
03: Updated OAuth 2.0 Device Authorization Grant description and reference due to publication of RFC 8628.
04: Updated OAuth 2.0 token exchange description and reference due to publication of RFC 8693. Corrected the RDAP conformance identifier to be registered with IANA.
05: Keepalive refresh.
06: Keepalive refresh.

Author's Address

Scott Hollenbeck
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
USA

Email: shollenbeck@verisign.com
URI: http://www.verisignlabs.com/

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: October 11, 2021

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

April 9, 2021

Registration Data Access Protocol (RDAP) Reverse search capabilities
draft-ietf-regext-rdap-reverse-search-06

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 October 11, 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 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. 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. This functionality SHOULD be only accessible to authorized users and only for a specified use case.

Already the request for this functionality could contain Personal Identifiable Information and SHOULD therefore only be available over HTTPS.

Providing reverse search in RDAP carries the following threats as described in [RFC6973]:

- o Correlation
- o Disclosure
- o Misuse of information

Therefore, RDAP providers are REQUIRED to mitigate the risk of those threats by implementing appropriate measures supported by security services (see Section 8).

8. Security Considerations

Security services required to provide controlled access to the operations specified in this document are described in [RFC7481]. A non exhaustive list of access control paradigms an RDAP provider can implement is presented in Appendix A.

The specification of the 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 the following individuals for their contributions to this document: Tom Harrison, Scott Hollenbeck, Francisco Arias, Gustavo Lozano, Eduardo Alvarez and Ulrich Wisser.

10. References

10.1. Normative References

[OIDCC] OpenID Foundation, "OpenID Connect Core incorporating errata set 1", November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, DOI 10.17487/RFC3912, September 2004, <https://www.rfc-editor.org/info/rfc3912>.

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

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <https://www.rfc-editor.org/info/rfc6350>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

[RFC7095] Kewisch, P., "jCard: The JSON Format for vCard", RFC 7095, DOI 10.17487/RFC7095, January 2014, <https://www.rfc-editor.org/info/rfc7095>.

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

[RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", STD 95, RFC 7480, DOI 10.17487/RFC7480, March 2015, <https://www.rfc-editor.org/info/rfc7480>.

[RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", STD 95, RFC 7481, DOI 10.17487/RFC7481, March 2015, <https://www.rfc-editor.org/info/rfc7481>.

[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

[draft-ietf-regext-rdap-openid]
Hollenbeck, S., "Federated Authentication for the Registration Data Access Protocol (RDAP) using OpenID Connect", <https://datatracker.ietf.org/doc/draft-ietf-regext-rdap-openid/>.

[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. Paradigms to Enforce Access Control on Reverse Search in RDAP

Access control can be implemented according to different paradigms introducing increasingly stringent rules. The paradigms reported here in the following leverage the capabilities either supported natively or provided as extensions by the OpenID Connect [OIDCC]:

- Role-Based Access Control: access rights are granted depending on roles. Generally, this is done by grouping users into fixed categories and assigning each category with static grants. A more dynamic approach can be implemented by using the OpenID Connect "scope" claim;

- Purpose-Based Access Control: access rules are based on the notion of purpose which means the intended usage of some data by a user. It can be implemented by tagging a request with the usage purpose and making the RDAP server check the compliance between the given purpose and the control rules applied to data to be returned. The purpose can be stated within an out-of-band process by setting the OpenID Connect RDAP specific "purpose" claim as defined in [draft-ietf-regext-rdap-openid];

- Attribute-Based Access Control: rules to manage access rights are evaluated and applied according to specific attributes describing the context within which data are requested. It can be implemented by setting within an out-of-band process additional OpenID Connect claims describing the request context and making the RDAP server check the compliance between the given context and the control rules applied to data to be returned;

- Time-Based Access Control: data access is allowed for limited time only. It can be implemented by assigning the users with temporary credentials linked to access grants whose scope is limited.

Appendix B. Change Log

00: Initial working group version ported from draft-loffredo-regext-rdap-reverse-search-04
01: Updated "Privacy Considerations" section.
02: Revised the text.
03: Refactored the query model.
04: Keepalive refresh.
05: Reorganized "Abstract". Corrected "Conventions Used in This Document" section. Added "RDAP Conformance" section. Changed "IANA Considerations" section. Added references to RFC7095 and RFC8174. Other minor edits.
06: Updated "Privacy Considerations", "Security Considerations" and "Acknowledgements" sections. Added some normative and informative references. Added Appendix A.

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

Network Working Group
Internet-Draft
Obsoletes: 7484 (if approved)
Intended status: Standards Track
Expires: September 30, 2021

M. Blanchet
Viagenie

March 29, 2021

Finding the Authoritative Registration Data (RDAP) Service
draft-ietf-regext-rfc7484bis-03

Abstract

This document specifies a method to find which Registration Data Access Protocol (RDAP) server is authoritative to answer queries for a requested scope, such as domain names, IP addresses, or Autonomous System numbers. This document obsoletes RFC7484.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 30, 2021.

Table of Contents

1. Introduction

Querying and retrieving registration data from registries are defined in Registration Data Access Protocol (RDAP) [RFC7480] [RFC7481] [RFC7482] [RFC7483]. These documents do not specify where to send the queries. This document specifies a method to find which server is authoritative to answer queries for the requested scope.

Top-Level Domains (TLDs), Autonomous System (AS) numbers, and network blocks are delegated by IANA to Internet registries such as TLD registries and Regional Internet Registries (RIRs) that then issue further delegations and maintain information about them. Thus, the bootstrap information needed by RDAP clients is best generated from data and processes already maintained by IANA; the relevant

registries already exist at [ipv4reg], [ipv6reg], [asreg], and [domainreg]. This document obsoletes [RFC7484].

Per this document, IANA has created new registries based on a JSON format specified in this document, herein named RDAP Bootstrap Service Registries. These new registries are based on the existing entries of the above mentioned registries. An RDAP client fetches the RDAP Bootstrap Service Registries, extracts the data, and then performs a match with the query data to find the authoritative registration data server and appropriate query base URL.

2. Conventions Used in This Document

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

3. Structure of the RDAP Bootstrap Service Registries

The RDAP Bootstrap Service Registries, as specified in Section 13 below, have been made available as JSON [RFC8259] objects, which can be retrieved via HTTP from locations specified by IANA. The JSON object for each registry contains a series of members containing metadata about the registry such as a version identifier, a timestamp of the publication date of the registry, and a description. Additionally, a "services" member contains the registry items themselves, as an array. Each item of the array contains a second-level array, with two elements, each of them being a third-level array.

Each element of the Services Array is a second-level array with two elements: in order, an Entry Array and a Service URL Array.

The Entry Array contains all entries that have the same set of base RDAP URLs. The Service URL Array contains the list of base RDAP URLs usable for the entries found in the Entry Array. Elements within these two arrays are not sorted in any way.

An example structure of the JSON output of a RDAP Bootstrap Service Registry is illustrated:

```
{
    "version": "1.0",
    "publication": "YYYY-MM-DDTHH:MM:SSZ",
    "description": "Some text",
    "services": [
      [
        ["entry1", "entry2", "entry3"],
        [
          "https://registry.example.com/myrdap/",
          "http://registry.example.com/myrdap/"
        ]
      ],
      [
        ["entry4"],
        [
          "https://example.org/"
        ]
      ]
    ]
}
```

The formal syntax is described in Section 10.

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

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

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

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

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

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

Internationalized Domain Name labels used as entries or base RDAP URLs in the registries defined in this document MUST be only represented using their A-label form as defined in [RFC5890].

All Domain Name labels used as entries or base RDAP URLs in the registries defined in this document MUST be only represented in lowercase.

4. Bootstrap Service Registry for Domain Name Space

The JSON output of this registry contains domain label entries attached to the root, grouped by base RDAP URLs, as shown in this example.

```
{
    "version": "1.0",
    "publication": "YYYY-MM-DDTHH:MM:SSZ",
    "description": "Some text",
    "services": [
      [
        ["net", "com"],
        [
          "https://registry.example.com/myrdap/"
        ]
      ],
      [
        ["org", "mytld"],
        [
          "https://example.org/"
        ]
      ],
      [
        ["xn--zckzah"],
        [
          "https://example.net/rdap/xn--zckzah/",
          "http://example.net/rdap/xn--zckzah/"
        ]
      ]
    ]
}
```

The domain name's authoritative registration data service is found by doing the label-wise longest match of the target domain name with the domain values in the Entry Arrays in the IANA Bootstrap Service Registry for Domain Name Space. The match is done per label, from right to left. If the longest match results in multiple entries, then those entries are considered equivalent. The values contained in the Service URL Array of the matching second-level array are the valid base RDAP URLs as described in [RFC7482].

For example, a domain RDAP query for a.b.example.com matches the com entry in one of the arrays of the registry. The base RDAP URL for

this query is then taken from the second element of the array, which is an array of base RDAP URLs valid for this entry. The client chooses one of the base URLs from this array; in this example, it chooses the only one available, "https://registry.example.com/myrdap/". The segment specified in [RFC7482] is then appended to the base URL to complete the query. The complete query is then "https://registry.example.com/myrdap/domain/a.b.example.com".

If a domain RDAP query for a.b.example.com matches both com and example.com entries in the registry, then the longest match applies and the example.com entry is used by the client.

If the registry contains entries such as com and goodexample.com, then a domain RDAP query for example.com only matches the com entry because matching is done on a per-label basis.

The entry for the root of the domain name space is specified as "".

5. Bootstrap Service Registries for Internet Numbers

This section discusses IPv4 and IPv6 address space and Autonomous System numbers.

For IP address space, the authoritative registration data service is found by doing a longest match of the target address with the values of the arrays in the corresponding RDAP Bootstrap Service Registry for Address Space. The longest match is done the same way as for routing: the addresses are converted in binary form and then the binary strings are compared to find the longest match up to the specified prefix length. The values contained in the second element of the array are the base RDAP URLs as described in [RFC7482]. The longest match method enables covering prefixes of a larger address space pointing to one base RDAP URL while more specific prefixes within the covering prefix are being served by another base RDAP URL.

5.1. Bootstrap Service Registry for IPv4 Address Space

The JSON output of this registry contains IPv4 prefix entries, specified in Classless Inter-domain Routing (CIDR) format [RFC4632] and grouped by RDAP URLs, as shown in this example.

```
{
    "version": "1.0",
    "publication": "2024-01-07T10:11:12Z",
    "description": "RDAP Bootstrap file for example registries.",
    "services": [
      [
        ["198.51.100.0/24", "192.0.0.0/8"],
        [
          "https://rir1.example.com/myrdap/"
        ]
      ],
      [
        ["203.0.113.0/24", "192.0.2.0/24"],
        [
          "https://example.org/"
        ]
      ],
      [
        ["203.0.113.0/28"],
        [
          "https://example.net/rdaprir2/",
          "http://example.net/rdaprir2/"
        ]
      ]
    ]
}
```

For example, a query for "192.0.2.1/25" matches the "192.0.0.0/8" entry and the "192.0.2.0/24" entry in the example registry above. The latter is chosen by the client given the longest match. The base RDAP URL for this query is then taken from the second element of the array, which is an array of base RDAP URLs valid for this entry. The client chooses one of the base URLs from this array; in this example, it chooses the only one available, "https://example.org/". The {resource} specified in [RFC7482] is then appended to the base URL to complete the query. The complete query is then "https://example.org/ip/192.0.2.1/25".

5.2. Bootstrap Service Registry for IPv6 Address Space

The JSON output of this registry contains IPv6 prefix entries, using [RFC4291] text representation of the address prefixes format, grouped by base RDAP URLs, as shown in this example.

```
{
    "version": "1.0",
    "publication": "2024-01-07T10:11:12Z",
    "description": "RDAP Bootstrap file for example registries.",
    "services": [
      [
        ["2001:db8::/34"],
        [
          "https://rir2.example.com/myrdap/"
        ]
      ],
      [
        ["2001:db8:4000::/36", "2001:db8:ffff::/48"],
        [
          "https://example.org/"
        ]
      ],
      [
        ["2001:db8:1000::/36"],
        [
          "https://example.net/rdaprir2/",
          "http://example.net/rdaprir2/"
        ]
      ]
    ]
}
```

For example, a query for "2001:db8:1000::/48" matches the "2001:db8::/34" entry and the "2001:db8:1000::/36" entry in the example registry above. The latter is chosen by the client given the longest match. The base RDAP URL for this query is then taken from the second element of the array, which is an array of base RDAP URLs valid for this entry. The client chooses one of the base URLs from this array; in this example, it chooses "https://example.net/rdaprir2/" because it's the secure version of the protocol. The segment specified in [RFC7482] is then appended to the base URL to complete the query. The complete query is, therefore, "https://example.net/rdaprir2/ip/2001:0db8:1000::/48". If the target RDAP server does not answer, the client can then use another URL prefix from the array.

5.3. Bootstrap Service Registry for AS Number Space

The JSON output of this registry contains Autonomous Systems number ranges entries, grouped by base RDAP URLs, as shown in this example. The Entry Array is an array containing the list of AS number ranges served by the base RDAP URLs found in the second element. The array always contains two AS numbers represented in decimal format that

represents the range of AS numbers between the two elements of the array. A single AS number is represented as a range of two identical AS numbers. AS numbers are represented as 'asplain' as defined in [RFC5396].

```
{
    "version": "1.0",
    "publication": "2024-01-07T10:11:12Z",
    "description": "RDAP Bootstrap file for example registries.",
    "services": [
      [
        ["64496-64496"],
        [
          "https://rir3.example.com/myrdap/"
        ]
      ],
      [
        ["64497-64510", "65536-65551"],
        [
          "https://example.org/"
        ]
      ],
      [
        ["64512-65534"],
        [
          "http://example.net/rdaprir2/",
          "https://example.net/rdaprir2/"
        ]
      ]
    ]
}
```

For example, a query for AS 65411 matches the 64512-65534 entry in the example registry above. The base RDAP URL for this query is then taken from the second element of the array, which is an array of base RDAP URLs valid for this entry. The client chooses one of the base URLs from this array; in this example, it chooses "https://example.net/rdaprir2/". The segment specified in [RFC7482] is then appended to the base URL to complete the query. The complete query is, therefore, "https://example.net/rdaprir2/autnum/65411". If the server does not answer, the client can then use another URL prefix from the array.

6. Entity

Entities (such as contacts, registrants, or registrars) can be queried by handle as described in [RFC7482]. Since there is no global namespace for entities, this document does not describe how to

find the authoritative RDAP server for entities. However, it is possible that, if the entity identifier was received from a previous query, the same RDAP server could be queried for that entity, or the entity identifier itself is a fully referenced URL that can be queried. The mechanism described in [RFC8521] MAY also be used.

7. Non-existent Entries or RDAP URL Values

The registries may not contain the requested value. In these cases, there is no known RDAP server for that requested value, and the client SHOULD provide an appropriate error message to the user.

8. Deployment and Implementation Considerations

This method relies on the fact that RDAP clients are fetching the IANA registries to then find the servers locally. Clients SHOULD NOT fetch the registry on every RDAP request. Clients SHOULD cache the registry, but use underlying protocol signaling, such as the HTTP Expires header field [RFC7234], to identify when it is time to refresh the cached registry.

Some authorities of registration data may work together on sharing their information for a common service, including mutual redirection [REDIRECT-RDAP].

When a new object is allocated, such as a new AS range, a new TLD, or a new IP address range, there is no guarantee that this new object will have an entry in the corresponding bootstrap RDAP registry, since the setup of the RDAP server for this new entry may become live and registered later. Therefore, the clients should expect that even if an object, such as TLD, IP address range, or AS range is allocated, the existence of the entry in the corresponding bootstrap registry is not guaranteed.

9. Limitations

This method does not provide a direct way to find authoritative RDAP servers for any other objects than the ones described in this document. In particular, the following objects are not bootstrapped with the method described in this document:

- entities
- queries using search patterns that do not contain a terminating string that matches some entries in the registries
- nameservers

- help

10. Formal Definition

This section is the formal definition of the registries. The structure of JSON objects and arrays using a set of primitive elements is defined in [RFC8259]. Those elements are used to describe the JSON structure of the registries.

10.1. Imported JSON Terms

- OBJECT: a JSON object, defined in Section 4 of [RFC8259]
- MEMBER: a member of a JSON object, defined in Section 4 of [RFC8259]
- MEMBER-NAME: the name of a MEMBER, defined as a "string" in Section 4 of [RFC8259]
- MEMBER-VALUE: the value of a MEMBER, defined as a "value" in Section 4 of [RFC8259]
- ARRAY: an array, defined in Section 5 of [RFC8259]
- ARRAY-VALUE: an element of an ARRAY, defined in Section 5 of [RFC8259]
- STRING: a "string", as defined in Section 7 of [RFC8259]

10.2. Registry Syntax

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

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

- services-list: a MEMBER with MEMBER-NAME "services" and MEMBER-VALUE a services-array
- services-array: an ARRAY, where each ARRAY-VALUE is a service
- service: an ARRAY of 2 elements, where the first ARRAY-VALUE is a an entry-list and the second ARRAY-VALUE is a service-uri-list
- entry-list: an ARRAY, where each ARRAY-VALUE is an entry
- entry: a STRING
- service-uri-list: an ARRAY, where each ARRAY-VALUE is a service-uri
- service-uri: a STRING

11. Security Considerations

By providing a bootstrap method to find RDAP servers, this document helps to ensure that the end users will get the RDAP data from an authoritative source, instead of from rogue sources. The method has the same security properties as the RDAP protocols themselves. The transport used to access the registries can be more secure by using TLS [RFC8446], which IANA supports.

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

12. 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 [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. RDAP Browser Mobile Application

Responsible Organization: Viagenie

Author: Marc Blanchet

Location: https://viagenie.ca/rdapbrowser/

Description: RDAP Browser is an RDAP client for domain names, IP addresses and AS numbers fetching the IANA registries described in this document to find the right authoritative RDAP server. End user can query any domain name, IP address or AS number and the registration data will be shown on the screen.

Level of Maturity: Production (i.e. in the Android and iOS App stores since August 2019)

Contact Information: rdapbrowser@viagenie.ca

Information last updated: March 2021

12.2. ICANN Lookup Web Application

Responsible Organization: ICANN

Location: https://lookup.icann.org

Description: ICANN's Domain Name Registration Data Lookup is an RDAP client for domain names fetching the IANA regis tries described in this document to find the right authoritative RDAP server. End user can query any domain name and the registration data will be shown on the screen.

Level of Maturity: Production

Information last updated: March 2021

12.3. ARIN Implementation

Responsible Organization: ARIN

Base URL: https://rdap-bootstrap.arin.net/bootstrap (Sample query: https://rdap-bootstrap.arin.net/bootstrap/autnum/1)

Description: ARIN RDAP Bootstrap server aids clients by reading the bootstrapping information published by IANA and using it to send HTTP redirects to RDAP queries. RDAP clients https://search.arin.net/ and NicInfo (https://github.com/arineng/nicinfo) use this bootstrap service. The underlying server software is open-sourced at https://github.com/arineng/rdap_bootstrap_server .

Level of Maturity: Production

Contact Information: info@arin.net

Information Last Updated: Nov 2020

13. IANA Considerations

IANA has created the RDAP Bootstrap Services Registries, listed below, and made them available as JSON objects. The contents of these registries are described in Section 3, Section 4, and Section 5, with the formal syntax specified in Section 10.

The process for adding or updating entries in these registries differs from the normal IANA registry processes: these registries are generated from the data, processes, and policies maintained by IANA in their allocation registries ([ipv4reg], [ipv6reg], [asreg], and [domainreg]), with the addition of new RDAP server information.

IANA updates RDAP Bootstrap Services Registries entries from the allocation registries as those registries are updated.

This document does not change any policies related to the allocation registries; IANA has provided a mechanism for collecting the RDAP server information. The RDAP Bootstrap Services Registries will start empty and will be gradually populated as registrants of domains and address spaces provide RDAP server information to IANA.

IANA has created a new top-level category on the Protocol Registries page, <https://www.iana.org/protocols>. The group is called "Registration Data Access Protocol (RDAP)". Each of the RDAP Bootstrap Services Registries has been made available for general public on-demand download in the JSON format, and that registry's URI is listed directly on the Protocol Registries page.

Other normal registries will be added to this group by other documents, but the reason the URIs for these registries are clearly listed on the main page is to make those URIs obvious to implementers -- these are registries that will be accessed by software, as well as by humans using them for reference information.

Because these registries will be accessed by software, the download demand for the RDAP Bootstrap Services Registries may be unusually high compared to normal IANA registries. The technical infrastructure by which registries are published has been put in place by IANA to support the load. Since the first publication of this RFC, no issue have been reported regarding the load or the service.

As discussed in Section 8, software that accesses these registries will depend on the HTTP Expires header field to limit their query rate. It is, therefore, important for that header field to be properly set to provide timely information as the registries change, while maintaining a reasonable load on the IANA servers.

The HTTP Content-Type returned to clients accessing these JSON-formatted registries MUST be "application/json", as defined in [RFC8259].

Because of how information in the RDAP Bootstrap Services Registries is grouped and formatted, the registry entries may not be sortable. It is, therefore, not required or expected that the entries be sorted in any way.

NOTE TO IANA: Please update the registries to reference this new RFC instead of RFC 7484 once this document is approved by the IESG and published by the RFC Editor". RFC-Editor, please remove this paragraph before publication

13.1. Bootstrap Service Registry for IPv4 Address Space

Entries in this registry contain at least the following:

- a CIDR [RFC4632] specification of the network block being registered.
- one or more URLs that provide the RDAP service regarding this registration.

13.2. Bootstrap Service Registry for IPv6 Address Space

Entries in this registry contain at least the following:

- an IPv6 prefix [RFC4291] specification of the network block being registered.
- one or more URLs that provide the RDAP service regarding this registration.

13.3. Bootstrap Service Registry for AS Number Space

Entries in this registry contain at least the following:

- a range of Autonomous System numbers being registered.
- one or more URLs that provide the RDAP service regarding this registration.

13.4. Bootstrap Service Registry for Domain Name Space

Entries in this registry contain at least the following:

- a domain name attached to the root being registered.
- one or more URLs that provide the RDAP service regarding this registration.

14. References

14.1. 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>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <https://www.rfc-editor.org/info/rfc4632>.

[RFC5396] Huston, G. and G. Michaelson, "Textual Representation of Autonomous System (AS) Numbers", RFC 5396, DOI 10.17487/RFC5396, December 2008, <https://www.rfc-editor.org/info/rfc5396>.

[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>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/info/rfc7258>.

[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>.

14.2. Informative References

[asreg] IANA, "Autonomous System (AS) Numbers", <https://www.iana.org/assignments/as-numbers>.

[domainreg] IANA, "Root Zone Database", <https://www.iana.org/domains/root/db>.

[ipv4reg] IANA, "IPv4 Address Space Registry", <https://www.iana.org/assignments/ipv4-address-space>.

[ipv6reg] IANA, "IPv6 Global Unicast Address Assignments", <https://www.iana.org/assignments/ipv6-unicast-address-assignments>.

[REDIRECT-RDAP] Martinez, C., Zhou, L., and G. Rada, "Redirection Service for Registration Data Access Protocol", Work in Progress, draft-ietf-weirds-redirects-04, July 2014.

[RFC7071] Borenstein, N. and M. Kucherawy, "A Media Type for Reputation Interchange", RFC 7071, DOI 10.17487/RFC7071, November 2013, <https://www.rfc-editor.org/info/rfc7071>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <https://www.rfc-editor.org/info/rfc7234>.

[RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", STD 95, RFC 7480, DOI 10.17487/RFC7480, March 2015, <https://www.rfc-editor.org/info/rfc7480>.

[RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", STD 95, RFC 7481, DOI 10.17487/RFC7481, March 2015, <https://www.rfc-editor.org/info/rfc7481>.

[RFC7482] Newton, A. and S. Hollenbeck, "Registration Data Access Protocol (RDAP) Query Format", RFC 7482, DOI 10.17487/RFC7482, March 2015, <https://www.rfc-editor.org/info/rfc7482>.

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

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

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8521] Hollenbeck, S. and A. Newton, "Registration Data Access Protocol (RDAP) Object Tagging", BCP 221, RFC 8521, DOI 10.17487/RFC8521, November 2018, <https://www.rfc-editor.org/info/rfc8521>.

Acknowledgements

The WEIRDS working group had multiple discussions on this topic, including a session during IETF 84, where various methods such as in-DNS and others were debated. The idea of using IANA registries was discovered by the author during discussions with his colleagues as well as by a comment from Andy Newton. All the people involved in these discussions are herein acknowledged. Linlin Zhou, Jean-Philippe Dionne, John Levine, Kim Davies, Ernie Dainow, Scott Hollenbeck, Arturo Servin, Andy Newton, Murray Kucherawy, Tom Harrison, Naoki Kambe, Alexander Mayrhofer, Edward Lewis, Pete Resnick, Alessandro Vesely, Bert Greevenbosch, Barry Leiba, Jari Arkko, Kathleen Moriaty, Stephen Farrell, Richard Barnes, and Jean-Francois Tremblay have provided input and suggestions to this document. Guillaume Leclanche was a coauthor of this document for some revisions; his support is therein acknowledged and greatly appreciated. The section on formal definition was inspired by Section 6.2 of [RFC7071]. This new version got comments and suggestions from: Gavin Brown, Patrick Mevzek, John Levine, Jasdip Singh, George Michaelson and Scott Hollenbeck.

Author's Address

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

EMail: Marc.Blanchet@viagenie.ca
URI: https://viagenie.ca

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 9 September 2021

J. Gould
R. Wilhelm
VeriSign, Inc.

8 March 2021

Extensible Provisioning Protocol (EPP) Secure Authorization Information for Transfer
draft-ietf-regext-secure-authinfo-transfer-06

Abstract

The Extensible Provisioning Protocol (EPP), in RFC 5730, defines the use of authorization information to authorize a transfer. Object-specific, password-based authorization information (see RFC 5731 and RFC 5733) is commonly used, but raises issues related to the security, complexity, storage, and lifetime of authentication information. This document defines an operational practice, using the EPP RFCs, that leverages the use of strong random authorization information values that are short-lived, not stored by the client, and stored by the server using a cryptographic hash that provides for secure authorization information that can safely be used for object 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 9 September 2021.

Copyright Notice

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 an empty authorization information value 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 [FIPS-180-4], 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", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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

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

The 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 operational practice 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 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 an empty authorization information value 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 an empty authorization information value to the 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 value 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 ([RFC5731] with the <domain:pw> element and [RFC5733] with the <contact:pw> element). 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 generated using a strong random value and 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). Given a target entropy, the required length can be calculated after deciding on the set of characters that will be randomized.

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

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 [FIPS-140-2] 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 [FIPS-180-4], and with a random salt.
2. 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, nor otherwise recorded where it will persist beyond the transfer process.
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. Authorization information that is unset 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, authorization information that is set 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 set 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 an empty authorization information value, 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 value and MAY disallow for the passing of a non-empty authorization information value. By having an empty authorization information value 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 value passed. Examples of such extensions are [RFC5731] and [RFC5733].

Example of passing an empty authorization information value 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 an empty authorization information value 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.

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 importance of generating strong authorization information values cannot be overstated: secure transfers are very important to the Internet to mitigate damage in the form of theft, fraud, and other abuse. It is critical that registrars only use strong, randomly generated authorization information values.

Because of this, registries may validate the randomness of the authorization information based on the length and character set required by the registry. For example, validating an authorization value contains a combination of upper-case, lower-case, and non-alphanumeric characters, in an attempt to assess the strength of the value, and return an EPP error result of 2202 if the check fails.

Such checks are, by their nature, heuristic and imperfect, and may identify well-chosen authorization information values as being not sufficiently strong. Registrars, therefore, must be prepared for an error response of 2202, "Invalid authorization information", and respond by generating a new value and trying again, possibly more than once.

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 value passed. Examples of such extensions are [RFC5731] and [RFC5733]. Setting an empty authorization information value unsets the authorization information. [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 value 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 value 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 value 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 value 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 value 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 value 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 value 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 value, to the registry. The registry stores the authorization information as an encrypted value and requires a non-empty authorization information value 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 value 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 128 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, Barry Leiba, Patrick Mevzek, Matthew Pozun, Srikanth Veeramachaneni, and Ulrich Wisser.

11. References

11.1. Normative References

[FIPS-140-2]
National Institute of Standards and Technology, U.S. Department of Commerce, "NIST Federal Information Processing Standards (FIPS) Publication 140-2", May 2001, <https://csrc.nist.gov/publications/detail/fips/140/2/final>.

[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>.

[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>.

[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>.

11.2. Informative References

[FIPS-180-4] National Institute of Standards and Technology, U.S. Department of Commerce, "Secure Hash Standard, NIST Federal Information Processing Standards (FIPS) Publication 180-4", August 2015, <https://csrc.nist.gov/publications/detail/fips/180/4/final>.

[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. 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.

A.9. Change from REGEXT 04 to REGEXT 05

1. Fixed IDNITS issues, including moving RFC7451 to Informative References section.

A.10. Change from REGEXT 05 to REGEXT 06

Updates based on the Barry Leiba (AD) feedback:

1. Simplified the abstract based on the proposal provided.
2. In the Introduction, split the first paragraph by starting a new paragraph at "This document".
3. In section 1.1, updated to use the new BCP 14 boilerplate and add a normative reference to RFC 8174.
4. In section 4, Updated the phrasing to "For the authorization information to be secure it must be generated using a strong random value and have a short time-to-live (TTL).".
5. In section 4.1, removed the first two unnecessary calculations and condensed the introduction of the section.
6. In section 4.1, added the use of the normative SHOULD for use of at least 128 bits of entropy.
7. Added an informative reference to FIPS 180-4 for the SHA-256 references.
8. Normalized the way that the "empty and non-empty authorization information values" are referenced, which a few exceptions.
9. In section 4, revised the first sentence to explicitly reference the use of the <domain:pw> and <contact:pw> elements for password-based authorization information.
10. In section 4.4, revised the language associated with the storage of the authorization information to be cleaner.
11. In section 4.4, added "set" in the sentence "An empty input authorization information value MUST NOT match any set authorization information value."
12. In section 5.1 and 5.2, clarified the references to RFC5731 and RFC5733 as examples of object extensions that use the "eppcom:pwAuthInfoType" element.
13. In section 5.2, updated language for the validation of the randomness of the authorization information, based on an offline review by Barrry Leiba, Benjamin Kaduk, and Roman Danyliw.
14. In section 9, changed "49 bits of entropy" to "128 bits of entropy".

In section 3, replaced the reference to BCP with operational practice, since the draft is not defined as a BCP.

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: 26 August 2021

J. Yee, Ed.
J. Galvin
Afilias

22 February 2021

Simple Registration Reporting
draft-ietf-regext-simple-registration-reporting-03

Abstract

Domain name registries (the producer) and registrars (the consumer) report to each other by sharing bulk information through files. This document creates two IANA registries to establish a standard reporting mechanism between domain name registries and registrars. The first IANA registry lists standard data elements and their syntax for inclusion in the files. The second IANA registry lists standard reports based on the standard data elements. Each report is a file formatted as a CSV file. The advantage of this reporting mechanism is that report, each file, can be imported by recipients without any prior knowledge of their contents, although reporting is enhanced with a minimum of knowledge about the files. The mechanism for the transmission and reception of the files is a matter of local policy.

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 26 August 2021.

Copyright Notice

Table of Contents

1. Introduction

Currently, domain name registry operators (the producer) create and set their own domain name registration reports for use by their registrars (the consumer). Among the distinctions that vary by producer is the syntax of the data provided, e.g., date formats, and the format of the collection of the data provided, e.g., the report may be a CSV file that tends to allow for straightforward importation or a PDF file that can be problematic to import. In addition, although there are a number of best practice reports that have evolved, these are not currently documented as such, which results in a fair amount of customization on the part of the consumers to import data.

This document standardizes the name and syntax of the data elements to be used across all existing domain name registration reports and creates an IANA registry of them to facilitate their evolution, including adding additional data elements as needed. In addition, a known set of existing standard reports using the aforementioned data elements is specified in another IANA registry to facilitate the evolution of the reports and adding additional report definitions as needed.

Each report definition MUST use the data elements defined here, including all future reports. Future reports and future data elements may be specified in their own individual documents, updating the IANA registries as needed.

1.1. Requirements Language

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

2. Data Element Specification

Data elements are grouped into categories for convenience. There is no other significance to the groupings.

Each data element conceptually represents the column heading in a printed report. It is a single unit of information that can be passed from the producer to the consumer. The primary purposes of the IANA registry of data elements are to ensure that each data element is assigned a unique name and that the syntax of each data element is specified.

The name of the data element MUST be unique and this characteristic MUST be enforced by the registry. The name is used in the report definition (in the next section) to alert the consumer as to what to expect in the file and how to import the data element. Character encoding recommendation for data delements is specificied in Section 7.

The subsections below comprise an initial list of known data elements commonly being used between producers and consumers as of the date of publication of this document. The title of the subsection is the field name for the data element. Field names in the IANA registry MUST be unique and case insensitive.

2.1. General Information Fields

2.1.1. TLD

The string of the top level domain involved that MUST be in A-LABEL format as defined by RFC 5890 [RFC5890].

2.1.2. Server_TRID

The transaction identifier issued by an EPP Server. The format MUST conform to "type:trIDStringType" as specified in RFC 5730 [RFC5730].

2.1.3. Domain

This is the domain name in an EPP RFC 5731 [RFC5731] domain object and it SHOULD be in A-Label format.

2.1.4. Transaction_Type

The type of transform action made to the domain object (CREATE, DELETE, UPDATE, TRANSFER, RENEW) as specified in RFC 5730 [RFC5730] Section 2.9.3.

2.1.5. Object_Type

The object type involved in the report. In the EPP environment, an object could be domain RFC 5731 [RFC5731], contact RFC 5733 [RFC5733], or host RFC 5732 [RFC5732].

2.1.6. DateTime

The timestamp of the transaction recorded in the system. The date and time format follows the "type=dateTime" specification as defined in RFC 5731 [RFC5731].

2.1.7. Term

The number of units added to the domain registration period in "domain:period" RFC 5731 [RFC5731] in create command, renew or transfer command. If there’s no "domain:period", it should take the default value set out of band by the registry.

2.1.8. Fee

The amount of money charged or returned (shown as a negative value) to the registrar. The numeric format MUST conform to the currency specified below in Section 2.1.9. The format must conform to "balanceType" as defined in RFC 8748 [RFC8748].

2.1.9. Currency

The currency used in the money charged as documented above in Section 2.1.8. The currency code should follow the ISO 4217 [ISO4217] standard.

2.1.10. Status

The status of the domain object. It MUST be one of the values specified in RFC 5731 [RFC5731] Section 2.3.

2.1.11. Registrar

The name of the registrar. This field is text/string with no naming convention enforced.

2.1.12. Period

The type of time (year, month) in 'Term' described above in Section 2.1.7. The value of 'year' and 'month' are referenced to pUnitType value 'y' and 'm' respectively. pUnitType is specified in RFC 5731 [RFC5731].

2.1.13. Description

Additional information regarding the current entry in the report. It is provided by the producer and its actual value is a matter of local policy.

2.2. Domain Price Fields

2.2.1. Domain_Create

The fee charged to create the domain. The format must conform to "balanceType" as defined in RFC 8748 [RFC8748].

2.2.2. Domain_Renew

The fee charged to renew the domain. The format must conform to "balanceType" as defined in RFC 8748 [RFC8748].

2.2.3. Domain_Transfer

The fee charged to transfer the domain. The format must conform to "balanceType" as defined in RFC 8748 [RFC8748].

2.2.4. Domain_Restore

The fee charged to restore the domain. The format must conform to "balanceType" as defined in RFC 8748 [RFC8748].

2.2.5. Trade

The fee charged to trade the domain. The format must conform to "balanceType" as defined in RFC 8748 [RFC8748].

2.3. Timestamp Fields

2.3.1. Start_Date

The timestamp of when the domain object becomes available. The date and time format follows the "type=dateTime" specification as defined in RFC 5731 [RFC5731].

2.3.2. Deleted_Date

The timestamp of when the domain was deleted by the end user. The date and time format follows the "type=dateTime" specification as defined in RFC 5731 [RFC5731].

2.3.3. RGP_Date

The timestamp of when the domain will complete its redemption grace period. The date and time format follows the "type=dateTime" specification as defined in RFC 5731 [RFC5731].

2.3.4. Purge_Date

The timestamp of when the domain will be purged and become available again. The date and time format follows the "type=dateTime" specification as defined in RFC 5731 [RFC5731].

2.3.5. Updated_Date

The timestamp of the last time the domain object was updated. The date and time format follows the "type=dateTime" specification as defined in RFC 5731 [RFC5731].

2.3.6. Create_Date

The timestamp of when the domain object was allocated. The date and time format follows the "type=dateTime" specification as defined in RFC 5731 [RFC5731].

2.3.7. Expiry_Date

The timestamp of when the domain object will expire. The date and time format follows the "type=dateTime" specification as defined in RFC 5731 [RFC5731].

2.4. Registration Information Fields

2.4.1. Registrar_ID

The identifier assigned to the registrar. If the registrar is accredited under ICANN, it MUST be the registrar's IANA ID [IANA_Registrar_IDs]. Otherwise it is a value known between the producer and the consumer, set via an out-of-band mechanism and unique within all reports of the producer.

2.4.2. Server_Registrant_ID

The identifier, issued by EPP server, assigned to the contact object that is associated as registrant of the domain name that MUST conform to "clIDType" specified in RFC 5730 [RFC5730].

2.4.3. DNSSEC

The value MUST be either 'YES' or 'NO' to indicate whether the domain is DNSSEC signed.

2.4.4. Server_Contact_ID

The identifier of the contact object assigned by the registry system and MUST conform to "clIDType" specified in RFC 5730 [RFC5730].

2.4.5. Contact_Type

The value MUST be one of value as defined by "contactAttrType" in RFC 5731 [RFC5731].

2.4.6. Contact_Name

The name of the contact object. Usually it is the name of an individual or an organization as described in RFC 5733 [RFC5733] Section 2.3.

2.4.7. INUSE

The value MUST be either "YES" or "NO" to indicate whether the contact object is associated with a domain object.

2.4.8. Nameserver_Host

The full domain name of the host object as defined in RFC 5732 [RFC5732] Section 2.1. The name SHOULD be in A-Label format.

2.4.9. Nameserver_IP

The IP address of the host object. The syntax of the IPv4 address MUST conform to RFC 791 [RFC0791]. The syntax of the IPv6 address MUST conform to RFC 4291 [RFC4291].

2.4.10. Client_Contact_ID

The identifier of the contact object assigned by the registrar and MUST conform to "clIDType" specified in RFC 5730 [RFC5730].

3. Report Definition Specification

Each report specification conceptually represents a file of comma separated values [RFC4180] (commonly called a CSV file) where the values are selected from the data elements specified above. The first row of the file is a comma separated list of field names as specified in the data element registry. The remaining rows of the file are the unordered sets of data elements, one set per row, where each row is one transaction in the report.

Each data element in a set conceptually represents the column heading in a printed report.

A consumer MUST be able to receive data elements that are not recognized and MAY skip them accordingly, both in the header row and in the transaction rows.

A report is specified in the report registry with two pieces of information. First is the name of the report. This can be whatever is appropriate as defined by the producer of the report. The name of the report MUST be unique and this characteristic MUST be enforced by registry.

Second is the ordered list of field names of what is included in the report. The field names MUST be listed in the data element registry specified above. The field names and the data MUST appear in the report in the order listed in the report registry.

The subsections below comprise an initial list of standard reports commonly being used between producers and consumers as of the date of publication of this document. The title of the subsection is the report name. The report name in the IANA registry must be unique and case insensitive.

3.1. Transaction Report

[Note] There is a new column in this report that is not added to other reports. It has been suggested that the report definition indicate if a value is required for the field or if it can be empty. The column was added to this report only for visualization and to further discussion. This paragraph will be removed once the working group discusses the issue and reaches consensus on direction.

Name of report: domain_transaction

Field	Reference	Mandatory
TLD	RFC XXXX Section 2.1.1	N
Server_TRID	Section 2.1.2	Y
Domain	Section 2.1.3	Y
DateTime	Section 2.1.6	Y
Registrar_ID	Section 2.4.1	Y
Registrar	Section 2.1.11	Y
Transaction_Type	Section 2.1.4	Y
Period	Section 2.1.12	Y
Term	Section 2.1.7	N
Fee	Section 2.1.8	Y
Currency	Section 2.1.9	Y
Description	Section 2.1.13	N

Table 1: Transaction Report Definition Table

3.2. Premium Name Report

Name of report: premium_name

Field	Reference
TLD	RFC XXXX Section 2.1.1
Domain	Section 2.1.3
Status	Section 2.1.10
Description	Section 2.1.13
Currency	Section 2.1.9
Domain_Create	Section 2.2.1
Domain_Renew	Section 2.2.2
Domain_Transfer	Section 2.2.3
Domain_Restore	Section 2.2.4
Start_Date	Section 2.3.1

Table 2: Premium Name Report Definition Table

3.3. Domain RGP Report

Name of report: domain_rgp

Field	Reference
TLD	RFC XXXX Section 2.1.1
Domain	Section 2.1.3
Deleted_Date	Section 2.3.2
RGP_Date	Section 2.3.3
Purge_Date	Section 2.3.4

Table 3: Domain RGP Report Definition Table

3.4. Reserved Domain Report

Name of report: reserved_domain

Field	Reference
TLD	RFC XXXX Section 2.1.1
Domain	Section 2.1.3
Status	Section 2.1.10

Table 4: Reserved Domain Report Definition Table

3.5. Domain Inventory Report

Name of report: domain_inventory

Field	Reference
TLD	RFC XXXX Section 2.1.1
Domain	Section 2.1.3
Updated_Date	Section 2.3.5
Registrar_ID	Section 2.4.1
Create_Date	Section 2.3.6
Expiry_Date	Section 2.3.7
Server_Registrant_ID	Section 2.4.2
DNSSEC	Section 2.4.3
Status	Section 2.1.10

Table 5: Domain Inventory Report Definition Table

3.6. Contact Inventory Report

Name of report: contact_inventory

Field	Reference
Server_Contact_ID	Section 2.4.4
Client_Contact_ID	Section 2.4.10
TLD	Section 2.1.1
Domain	Section 2.1.3
Contact_Type	Section 2.4.5
Contact_Name	Section 2.4.6
Updated_Date	Section 2.3.5
INUSE	Section 2.4.7
Registrar_ID	Section 2.4.1

Table 6: Contact Inventory Report Definition Table

3.7. Host Inventory Report

Name of report: host_inventory

Field	Reference
TLD	RFCXXXX Section 2.1.1
Nameserver_Host	Section 2.4.8
Nameserver_IP	Section 2.4.9

Table 7: Host Inventory Report Definition Table

4. IANA Considerations

This section describes the format of the IANA Registration Report Registry, which has two tables described below, and the procedures used to populate and manage the registry entries.

4.1. Report Specification

This registry uses the "Specification Required" policy described in RFC 8126 [RFC8126]. An English language version of the extension specification is required in the registry, though non-English versions of the specification may also be provided.

The "Specification Required" policy implies review by a "designated expert". Section 5.2 of RFC 8126 [RFC8126] describes the role of designated experts and the function they perform.

4.1.1. Designated Expert Evaluation Criteria

A high-level description of the role of the designated expert is described in Section 5.2 of RFC 8126 [RFC8126]. Specific guidelines for the appointment of designated experts and the evaluation of a Registration Report is provided here.

The IESG SHOULD appoint a small pool of individuals (perhaps 3 - 5) to serve as designated experts, as described in Section 5.2 of RFC 8126 [RFC8126]. The pool should have a single administrative chair who is appointed by the IESG. The designated experts should use the existing regext mailing list (regext@ietf.org) for public discussion of registration requests. This implies that the mailing list should remain open after the work of the REGEXT working group has concluded.

The results of the evaluation should be shared via email with the registrant and the regext mailing list. Issues discovered during the evaluation can be corrected by the registrant, and those corrections can be submitted to the designated experts until the designated experts explicitly decide to accept or reject the registration request. The designated experts must make an explicit decision and that decision must be shared via email with the registrant and the regext mailing list. If the specification for a field or report is an IETF Standards Track document, no review is required by the designated expert.

Designated experts should be permissive in their evaluation of requests for fields and reports that have been implemented and deployed by at least one registry. This implies that it may indeed be possible to register multiple fields or reports that provide the same functionality. Requests to register fields or reports that have not been deployed should be evaluated with a goal of reducing duplication. A potential registrant who submits a request to register a new field or report that includes similar functionality to existing fields or reports should be made aware of the existing fields and reports. The registrant should be asked to reconsider their request given the existence of similar fields or reports.

Should they decline to do so, perceived similarity should not be a sufficient reason for rejection as long as all other requirements are met.

4.1.2. Registration Procedure

The registry contains information describing each registered field and report. Registry entries are created and managed by sending forms to IANA that describe the field and report on the registry entry.

4.1.2.1. Required Information

The required information must be formatted consistently using the following registration form. Form field names and values may appear on the same line.

4.1.2.1.1. Field Definition

The field name must be unique and evaluated in a case insensitive way.

Name of field
: Enforced to be case-insensitive and unique

Reference document defining the field, including section number
: SHOULD be a URL to a RFC, MAY be a URL to any document available under equivalent terms

Registrant
: Will be IESG for initial entries

Status
: MUST be active, inactive, unknown

4.1.2.1.2. Report Definition

The report name must be unique and case insensitive.

Name of Report

Document Status

Reference

Registrant:IESG

TLD:any

Status:active

4.1.2.2. Registration Processing

Registrants should send each registration form to IANA with a single record for incorporation into the registry. Send the form via email to iana@iana.org or complete the online form found on the IANA web site. The subject line should indicate whether the enclosed form represents an insertion of a new record (indicated by the word "INSERT" in the subject line) or a replacement of an existing record (indicated by the word "MODIFY" in the subject line). At no time can a record be deleted from the registry. On receipt of the registration request, IANA will initiate review by the designated expert(s) if appropriate, who will evaluate the request using the criteria in Section 4.1.1 in consultation with the regext mailing list.

4.1.2.3. Updating Report Definition Registry Entries

When submitting changes to existing registry entries, include text in the "Notes" field of the registration form describing the change. Under normal circumstances, registry entries are only to be updated by the registrant. If the registrant becomes unavailable or otherwise unresponsive, the designated expert can submit a registration form to IANA to update the registrant information. Entries can change state from "Active" to "Inactive" and back again as long as state-change requests conform to the processing requirements identified in this document. In addition to entries that become "Inactive" due to a lack of implementation, entries for which a specification becomes consistently unavailable over time should be marked "Inactive" by the designated expert until the specification again becomes reliably available.

4.2. Initial assignments

4.2.1. Field Definition in IANA Registry

---- BEGIN FORM ----

Name of field:
TLD

Reference:
This RFC Section 2.1.1

Registrant:
IESG, iesg@ietf.org

```
   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Server_TRID

   Reference:
           This RFC Section 2.1.2

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Domain

   Reference:
           This RFC Section 2.1.3

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Transaction_Type

   Reference:
           This RFC Section 2.1.4

   Registrant:
           IESG, iesg@ietf.org
```

```
   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Ojbect_Type

   Reference:
           This RFC Section 2.1.5

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           DateTime

   Reference:
           This RFC Section 2.1.6

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Term

   Reference:
           This RFC Section 2.1.7

   Registrant:
           IESG, iesg@ietf.org
```

```
   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Currency

   Reference:
           This RFC Section 2.1.9

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Status

   Reference:
           This RFC Section 2.1.10

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Registrar

   Reference:
           This RFC Section 2.1.11

   Registrant:
           IESG, iesg@ietf.org
```

```
   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Period

   Reference:
           This RFC Section 2.1.12

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Description

   Reference:
           This RFC Section 2.1.13

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Domain_Create

   Reference:
           This RFC Section 2.2.1

   Registrant:
           IESG, iesg@ietf.org
```

```
   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Domain_Renew

   Reference:
           This RFC Section 2.2.2

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Domain_Transfer

   Reference:
           This RFC Section 2.2.3

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Domain_Restore

   Reference:
           This RFC Section 2.2.4

   Registrant:
           IESG, iesg@ietf.org
```

```
   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Start_Date

   Reference:
           This RFC Section 2.3.1

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Deleted_Date

   Reference:
           This RFC Section 2.3.2

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           RGP_Date

   Reference:
           This RFC Section 2.3.3

   Registrant:
           IESG, iesg@ietf.org
```

```
   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Purge_Date

   Reference:
           This RFC Section 2.3.4

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Updated_Date

   Reference:
           This RFC Section 2.3.5

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Create_Date

   Reference:
           This RFC Section 2.3.6

   Registrant:
           IESG, iesg@ietf.org
```

```
   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Expiry_Date

   Reference:
           This RFC Section 2.3.7

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Registrar_ID

   Reference:
           This RFC Section 2.4.1

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Server_Registrant_ID

   Reference:
           This RFC Section 2.4.2

   Registrant:
           IESG, iesg@ietf.org
```

```
   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           DNSSEC

   Reference:
           This RFC Section 2.4.3

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Server_Contact_ID

   Reference:
           This RFC Section 2.4.4

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Contact_Type

   Reference:
           This RFC Section 2.4.5

   Registrant:
           IESG, iesg@ietf.org
```

```
   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Contact_Name

   Reference:
           This RFC Section 2.4.6

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           INUSE

   Reference:
           This RFC Section 2.4.7

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Nameserver_Host

   Reference:
           This RFC Section 2.4.8

   Registrant:
           IESG, iesg@ietf.org
```

```
   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Nameserver_IP

   Reference:
           This RFC Section 2.4.9

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of field:
           Client_Contact_ID

   Reference:
           This RFC Section 2.4.10

   Registrant:
           IESG, iesg@ietf.org

   Status:
           Active

   ---- END FORM ----
```

4.2.2. Report Definition in IANA Registry

```
   ---- BEGIN FORM ----

   Name of report:
           domain_transaction

   Reference:
           This RFC Table 1
```

```
   Registrant:
           IESG, iesg@ietf.org

   TLD:
           any

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of report:
           premium_name

   Reference:
           This RFC Section 3.2

   Registrant:
           IESG, iesg@ietf.org

   TLD:
           any

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of report:
           domain_rgp

   Reference:
           This RFC Section 3.3

   Registrant:
           IESG, iesg@ietf.org

   TLD:
           any

   Status:
           Active

   ---- END FORM ----
```

```
   ---- BEGIN FORM ----

   Name of report:
           reserved_domain

   Reference:
           This RFC Section 3.4

   Registrant:
           IESG, iesg@ietf.org

   TLD:
           any

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of report:
           domain_inventory

   Reference:
           This RFC Section 3.5

   Registrant:
           IESG, iesg@ietf.org

   TLD:
           any

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of report:
           contact_inventory

   Reference:
           This RFC Section 3.6

   Registrant:
           IESG, iesg@ietf.org
```

```
   TLD:
           any

   Status:
           Active

   ---- END FORM ----

   ---- BEGIN FORM ----

   Name of report:
           host_inventory

   Reference:
           This RFC Section 3.7

   Registrant:
           IESG, iesg@ietf.org

   TLD:
           any

   Status:
           Active

   ---- END FORM ----
```

5. Security Considerations

Items to be covered in a future version:

SHOULD NOT expose secret information, e.g., authInfo

This report does not address delivery mechanisms but due consideration should be given to access of reports. Note encryption is something to consider for controlling access.

6. Privacy Considerations

Items to be covered in a future version:

Note that reports may contain Personally Identifiable Information

7. Internationalization Considerations

The character encoding for the file contents MUST use UTF-8.

Throughout this document A-LABEL is indicated as a SHOULD and that MUST be interpreted as follows. All domain name labels MUST be in A-LABEL format if it is possible to represent it as an A-LABEL, otherwise U-LABEL MAY be used.

8. References

8.1. Normative References

[ISO4217] International Organization for Standardization, "ISO 4217 Currency Codes", 2018, <https://www.currency-iso.org/dam/downloads/lists/list_one.xml>.

[RFC0791] Postel, J., "Internet Protocol", 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>.

[RFC4180] Shafranovich, Y., "IP Version 6 Addressing Architecture", February 2006, <https://www.rfc-editor.org/info/rfc4180>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", February 2006, <https://www.rfc-editor.org/info/rfc4291>.

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

[RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", August 2009, <https://www.rfc-editor.org/info/rfc5731>.

[RFC5732] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Host Mapping", August 2009, <https://www.rfc-editor.org/info/rfc5732>.

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

[RFC5890] Klensin, J., "Extensible Provisioning Protocol (EPP) Contact Mapping", August 2009, <https://www.rfc-editor.org/info/rfc5890>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8748] Carney, R. and J. Frakes, "Registry Fee Extension for the Extensible Provisioning Protocol", March 2021, <https://www.rfc-editor.org/info/rfc8748>.

8.2. Informative References

[IANA_Registrar_IDs] Internet Assigned Numbers Authority, "IANA Assignments - Registrar IDs", 2020, <https://www.iana.org/assignments/registrar-ids/registrar-ids.xhtml>.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <https://www.rfc-editor.org/info/rfc2629>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

Appendix A. Acknowledgements

The authors would like to thank Roger Carney, Jody Kolker, and bestpractice.domains for their reviews and suggestions.

Appendix B. File Naming Convention

TBD on file naming convention suggestion

Authors' Addresses

Joseph Yee (editor)
Afilias
Toronto
Canada

Email: jyee@afilias.info

James Galvin
Afilias
Horsham, PA
United States

Email: jgalvin@afilias.info

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 23 August 2021

J. Gould
VeriSign, Inc.
M. Casanova
SWITCH

19 February 2021

Extensible Provisioning Protocol (EPP) Unhandled Namespaces draft-ietf-regext-unhandled-namespaces-08

Abstract

The Extensible Provisioning Protocol (EPP), as defined in RFC 5730, 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, and an "unhandled namespace" is one that is associated with a service not supported by the client. 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 23 August 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. Returning an unhandled namespace poll message is not compliant with the negotiated services defined in [RFC5730] and returning an error makes the unhandled namespace poll message a poison message by halting the processing of the poll queue. 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", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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

In examples, "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 but the namespace of the information 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 unhandled namespace MUST be declared in the child element or any containing element including the root element. XML processing of the <value> element is disabled by the XML schema 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 applies to the handling of unsupported namespaces for [RFC3735] object-level extensions and command-response extensions. This document does not apply to the handling of unsupported namespaces for [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 for an 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 an example of converting the <transfer> query response example in Section 3.1.3 of [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>2000-06-06T22:00:00.0Z</domain:reDate>
S:            <domain:acID>ClientY</domain:acID>
S:            <domain:acDate>2000-06-11T22:00:00.0Z</domain:acDate>
S:            <domain:exDate>2002-09-08T22:00:00.0Z</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 an [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 Section 5.1.2 of [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:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
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>1999-12-03T09:00:00.0Z</domain:upDate>
S:        <domain:exDate>2005-04-03T22:00:00.0Z</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 EPP protocol elements but rather specifies 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 the <extValue> element per Section 3.2. Below is example of converting the domain name <info> response example in Section 4.1.2 of [RFC3915] to an unhandled namespace response.

[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:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
S:     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
S:     epp-1.0.xsd">
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:           xsi:schemaLocation="urn:ietf:params:xml:ns:rgp-1.0
S:           rgp-1.0.xsd">
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:        xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0
S:        domain-1.0.xsd">
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>1999-12-03T09:00:00.0Z</domain:upDate>
S:        <domain:exDate>2005-04-03T22:00:00.0Z</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 Section 3.1.2 of [RFC8590], which is an extension of any EPP object, is an example of applying the unhandled namespace approach for EPP <poll> message responses. Below are examples of converting the domain name <info> response example in Section 3.1.2 of [RFC8590] to an unhandled namespace response. 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 lang="en-US">
S:        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:              2013-10-22T14:25:57.0Z</changePoll:date>
S:            <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:            <changePoll:who>URS Admin</changePoll:who>
S:            <changePoll:caseId type="urs">urs123
```

```
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 count="201" id="1">
S:      <qDate>2013-10-22T14:25:57.0Z</qDate>
S:      <msg>Registry initiated update of domain.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:exDate>2014-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <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>domain.example</domain:name>
S:            <domain:roid>EXAMPLE1-REP</domain:roid>
S:            <domain:status s="ok"/>
S:            <domain:registrant>jd1234</domain:registrant>
S:            <domain:contact type="admin">sh8013</domain:contact>
S:            <domain:contact type="tech">sh8013</domain:contact>
S:            <domain:clID>ClientX</domain:clID>
S:            <domain:crID>ClientY</domain:crID>
S:            <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
S:            <domain:exDate>2014-04-03T22:00:00.0Z</domain:exDate>
S:          </domain:infData>
S:        </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=
S:              "urn:ietf:params:xml:ns:changePoll-1.0"
S:            state="after">
S:            <changePoll:operation>update</changePoll:operation>
S:            <changePoll:date>
S:              2013-10-22T14:25:57.0Z</changePoll:date>
S:            <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:            <changePoll:who>URS Admin</changePoll:who>
S:            <changePoll:caseId type="urs">urs123
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 count="201" id="1">
S:      <qDate>2013-10-22T14:25:57.0Z</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 implementing the following:

1. Ensure that the client presents the complete set of what it supports when presenting its login services. 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 should 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 implementing 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: IETF, <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

This document does not provide any security services beyond those described by EPP [RFC5730] and protocol layers used by EPP. The security considerations described in these other specifications apply to this specification as well. Since the unhandled namespace context is XML that is not processed in the first pass by the XML parser, the client SHOULD validate the XML when the content is processed to protect against the inclusion of malicious content.

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>.

[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>.

[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>.

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>.

[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>.

[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>.

[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>.

[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>.

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.

A.9. Change from REGEXT 05 to REGEXT 06

1. Nit updates made based on the feedback provided by the Document Shepherd, David Smith.

A.10. Change from REGEXT 06 to REGEXT 07

Updates based on the Barry Leiba (AD) feedback:

1. Simplified the abstract based on the proposal provided by the AD.
2. In section 1.1, updated to use the new BCP 14 boilerplate and add a normative reference to RFC 8174.
3. In section 1.1, changed "REQUIRED feature of this protocol" to "required feature of this protocol".
4. In section 3, added "by the XML schema" in "disabled by the XML schema in [RFC5730]" to clarify the statement.
5. In section 8.2, changed the Registrant Name from "IESG" to "IETF".
6. In section 10, changed "The document do not provide" to "This document does not provide".
7. In section 10, added the sentence "Since the unhandled namespace context is XML that is not processed in the first pass by the XML parser, the client SHOULD consider validating the XML when the content is processed to protect against the inclusion of malicious content.".

A.11. Change from REGEXT 07 to REGEXT 08

1. Nit updates made based on the feedback provided by Peter Yee.
2. Update to the definition of the <value> element based on feedback from Sabrina Tanamal.
3. Added a sentence in the Introduction section to cover the poison poll message motivation based on feedback from Qin Wu.

4. Changed "does not define new protocol" to "does not define new EPP protocol elements" based on feedback from Erik Kline.
5. Changed to use "apply" instead of "support" language in Section 3 based on feedback from Benjamin Kaduk.
6. Updated the examples that reference RFC examples to reference the RFC section of the example and have the starting XML match based on feedback from Benjamin Kaduk.
7. Changed "SHOULD consider validating" to "SHOULD validate" in the Security Considerations section based on feedback from Benjamin Kaduk.
8. Moved RFC 3915, RFC 5910, and RFC 8590 as informational references based on feedback from Benjamin Kaduk.

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: July 28, 2021

M. Loffredo
IIT-CNR/Registro.it
G. Brown
CentralNic Group plc

January 24, 2021

Using JSContact in Registration Data Access Protocol (RDAP) JSON Responses
draft-loffredo-regext-rdap-jcard-deprecation-04

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 July 28, 2021.

Copyright Notice

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-ietf-httpapi-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-ietf-httpapi-deprecation-header]
Dalal, S. and E. Wilde, "The Deprecation HTTP Header Field", <https://datatracker.ietf.org/doc/draft-ietf-httpapi-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 03

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.

A.4. Change from 03 to 04

1. Updated the reference draft-dalal-deprecation-header to draft-ietf-httpapi-deprecation-header.

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