Abstract

Registration Access Data Protocol (RDAP) is being deployed in domain and IP address registries. This document describes issues and findings while interfacing with the known server implementations and deployments. It also provides recommendations for the specifications.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 15, 2019.

Copyright Notice

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

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

While developing various tools and software related to RDAP, issues have been found and are documented below. This document should help in writing future version of the specifications and provide better conformant deployment. It is split in various sections based on where the fix should be applied. Obviously, there are different levels of severity of the issues, including nits or very minor. The actual instances and organisations running the RDAP servers where the issues were found are not listed.

2. IANA RDAP Registries Related Issues

This section describes issues related to the IANA non-Bootstrap registries as specified in [RFC7483].
### 2.1. Values not Registered or Similar

The IANA RDAP JSON Values registry [1] contains various values expected in JSON responses. The following table shows values not registered in the registry but seen in the field. The second column shows the possible corresponding values already registered.

Recommendation: implementations should replace their custom values with the registered ones, when one exist. Implementors should register their values when there is no corresponding registered one.

#### Remarks Type

<table>
<thead>
<tr>
<th>Unregistered Values</th>
<th>Possibly Corresponding Registered Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>object truncated due to server policy</td>
<td>object truncated due to authorization</td>
</tr>
<tr>
<td>Response truncated due to authorization</td>
<td>object truncated due to authorization</td>
</tr>
<tr>
<td>Object truncated due to authorization</td>
<td>object truncated due to authorization</td>
</tr>
<tr>
<td>object redacted due to authorization</td>
<td>object truncated due to authorization</td>
</tr>
</tbody>
</table>

#### Event Action

<table>
<thead>
<tr>
<th>Unregistered Values</th>
<th>Possibly Corresponding Registered Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>delegation check</td>
<td>last changed</td>
</tr>
<tr>
<td>last correct delegation check</td>
<td></td>
</tr>
<tr>
<td>last update</td>
<td></td>
</tr>
</tbody>
</table>
Status Value

<table>
<thead>
<tr>
<th>Unregistered Values</th>
<th>Possibly Corresponding Registered Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>server deleted</td>
<td>server delete prohibited</td>
</tr>
<tr>
<td>ok</td>
<td>active</td>
</tr>
</tbody>
</table>

Role Value

<table>
<thead>
<tr>
<th>Unregistered Values</th>
<th>Possibly Corresponding Registered Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>registrant</td>
</tr>
</tbody>
</table>

2.1.1. Registry Entity

The (domain or IP) registry itself is currently not modeled in entities in RDAP. In an whois query for a TLD itself, the Remarks contains the URL of the registry entity (for registration information) and the whois entry of the registry is returned. In RDAP context, the RDAP server URL of the TLD registry should also be returned. Therefore, IANA RDAP server should send this data for the TLDs as part of its RDAP response. These semantics are currently not modeled.

This document proposes that RDAP servers may send an entity with role "registry" in the top-level of the RDAP response. This entity would have embedded [links] to its web server ("rel": "self", "type": "text/html") and rdap server ("rel": "self", "type": "application/rdap+json").

IANA Action: add a new row "registry", "role" to the RDAP JSON Values registry.

2.2. RDAP Extensions not Registered

The IANA RDAP Extensions registry [2] contains various extensions values expected in RDAP JSON responses in the rdapCconformance member. It is our understanding from [RFC7483] section 4.1 and [RFC7480] section 8.1 that only the prefix of the extension (i.e. "rdap_ObjectTag"), not the whole string ("rdap_objectTag_level_0"), need to be registered in the IANA registry. However, some entries in
the IANA RDAP extensions registry seem to imply a 0 version as part of the registered value.

The following table shows values seen in the field in the first column, corresponding prefix (guessed as there is no normalized delimiter) in the second column and if the prefix is registered in IANA registry in the third column.

This registry may end up listing all names of all registries if each one has his own extension. Moreover, there is no normalized delimiter of the prefix in the full string, which may not help the RDAP client to parse and interpret correctly. As with [RFC6350], we may instead use the First Come First Serve (FCFS) private enterprise numbers (PEN) registry to automatically have an organisation prefix defined without creating another set of org names within this registry and have the delimeter be "_" following the PEN.

**Recommendation (short term):** implementations should replace their custom values with the registered ones, when one exist. Implementors should register their values when there is no corresponding registered one.

<table>
<thead>
<tr>
<th>Values Seen</th>
<th>Corresponding Assumed Prefix</th>
<th>Prefix Already Registered in IANA</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdap_objectTag_level_0</td>
<td>rdap_objectTag</td>
<td>Y</td>
</tr>
<tr>
<td>fred_version_0</td>
<td>fred</td>
<td>Y</td>
</tr>
<tr>
<td>rdap_openidc_level_0</td>
<td>rdap_openidc</td>
<td>N</td>
</tr>
<tr>
<td>icann_rdap_technical_implementation_guide_0</td>
<td>icann_rdap_technical_imple</td>
<td>N</td>
</tr>
<tr>
<td>icann_rdap_response_profile_e_0</td>
<td>icann_rdap_response_profile</td>
<td>N</td>
</tr>
<tr>
<td>itNic_level_0</td>
<td>itNic</td>
<td>N</td>
</tr>
<tr>
<td>nicbr_level_0</td>
<td>nicbr</td>
<td>N</td>
</tr>
<tr>
<td>ur_domain_check_level_0</td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>history_version_0</td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>registrar_api_0</td>
<td></td>
<td>N</td>
</tr>
</tbody>
</table>

3. RDAP Responses

This section discusses issues found related to RDAP responses, specified in [RFC7483].
3.1. Cross-origin resource sharing (CORS)

As specified in [RFC7480], the HTTP "Access-Control-Allow-Origin: *" header should be included in the responses, to enable Web clients to work properly. Some RDAP servers do not set this header. RFC7480 says "it is RECOMMENDED that servers". It should be updated to "for any public Internet deployment, servers MUST".

3.2. Object Class Name empty

A non-conformant server sends the following answer, where the value of "objectClassName" is an empty string (as well as "handle" also empty). As per [RFC7483] section 4.9, this "objectClassName" value is required. Extract of the seen response:

```
{
    entities: [
        {
            "entities": [
                {
                    "objectClassName": ",",
                    "handle": ",",
                }
            ],
        },
    ],
}
```

3.3. Links Relation Values

The links relation values as specified in [RFC7483] section 4.3 refer to [RFC5988] which creates the IANA Link Relations registry [3]. This registry contains a large number of values where most of them do not apply to the RDAP deployment. As seen with other values above that are similar to registered ones but not used, we list here the ones we have seen. It would be appropriate to further describes the main ones in the RFC so implementors focus on ones that are expected instead of picking the wrong ones in the IANA registry or to define new ones and do not register them.
As shown in the table, an implementation put an URL as the value of the "rel", instead of an actual registered value.

3.4. Related link pointing to self causes infinite loop

An RDAP server returns a link of "rel": "related" is pointing to itself, therefore causing the RDAP client to fetch the object again, then read the related link and then fetch again, creating an infinite loop. Extract of the seen response:

```
{
    "links": [
        {
            "title": "Self",
            "rel": "self",
            "type": "application/rdap+json",
            "href": "https://rdapserver.example.com/domain/example.net"
        },
        {
            "title": "Registrar Data for this object",
            "rel": "related",
            "href": "https://rdapserver.example.com/domain/example.net",
            "type": "application/rdap+json"
        }
    ],
}
```

Recommendation: do not put related link same as self. RFC7483 section 4.2 should be updated to add the following text: "A link of "rel": "related" should not have the "href" value the same as the value of "href" of link of "rel": "self"."
3.5. Link without rel

An RDAP server returns a link with no "rel" property, so the client parser has no clue what is this data and what to do with it. Extract of the seen response:

```
{  
  "links": [  
    {  
      "title": "My Corporation",  
      "href": "http://mycorp.mytld"  
    }  
  ],
}
```

Recommendation: Any link must have a "rel" value. RFC7483 says that only href is mandatory. RFC7483 should be updated to have both rel and href mandatory. The original text "The "href" JSON value MUST be specified." should be changed to "The "href" and "rel" JSON values MUST be specified."

3.6. Value and href for IDNs in links

An RDAP server should return a link with "rel": "self" with a href corresponding to the target URL and value as context URI. In case of idn, there are at least two possible representations of the URI: with the A-label or U-label in the URI, the latter known as IRI ([RFC3987]). Moreover, the query may be of a U-Label or A-Label or combination of these types of labels. Therefore, there is an ambiguity in which representation should be the canonical one sent. This also applies to any type of "rel" for links. Extract of the seen response:

```
{  
  "links": [  
    {  
      "rel": "self",  
      "href": "http://myrdapserver.xn--abcd/domain/example.ULABEL"  
    }  
  ],
}
```

Recommendation: All links of any "rel" types should always be returned in the A-Label form for IDNs in the href or value members, independent of if the query was a U-Label or A-Label or a mix. This should be added to [RFC7483].
3.7. Registrant Entity Too Deep

An RDAP server returns the registrant entity in a subentity, which makes difficult to parse given the expectation is the registrant would be at the top level. Extract of the seen response:

```
{  
  entities: [
  {   
    "objectClassName": "entity",  
    "handle": "HANDLE1",  
    "roles": [ "abuse" ],  
    "vcardArray": [ ... ],  
    "entities": [
      {   
        "objectClassName": "entity",  
        "handle": "HANDLE2",  
        "roles": [ "registrant" ],  
        "vcardArray": [ ... ],  
      }
    ],  
  },  
],
```

Recommendation: put the registrant in the top-level entities as follows:

```
{  
  entities: [
  {   
    "objectClassName": "entity",  
    "handle": "HANDLE1",  
    "roles": [ "abuse" ],  
    "vcardArray": [ ... ]  
  },  
  {   
    "objectClassName": "entity",  
    "handle": "HANDLE2",  
    "roles": [ "registrant" ],  
    "vcardArray": [ ... ]  
  }
],
```

4. Queries

This section talks about support of RFC7482 queries and the RDAP server behaviors seen.
4.1. URL encoding of :

For RIR registries, the ip query may include an IPv6 address which then includes one or many ":". Clients may decide to do percent-encoding of the query. In one RDAP server, the server rejected the percent-encoded query of an IPv6 address. For example, https://rdapserver.example.com/ip/2001%3Adb8%3A0%3A3A/48 is rejected, while https://rdapserver.example.com/ip/2001:db8:0::/48 is accepted.

Recommendation: accept both percent-encoded queries or non-percent encoded queries.

5. Domain Registrar RDAP Server Location

The ICANN RDAP Profile [4] section 3.2 requires the domain registries who do not have registrant information (so-called thin registries) to put a specific link of "rel": "related" pointing to the domain registrar responsible for the domain being queried, so that a client can get the registrant information using a second query to the related link. However, the semantics seems ambiguous as other RDAP servers may use the "rel": "related" for other related means, but not the specific semantic of finding the registrant data. Therefore, a possible mitigation is to define a new "rel" type of "registrantInfo" (mnemonic TBD) to carry the specific semantic of registrant info.

6. Issues related to RFC7482

6.1. Search patterns that are not

Section 3.2.1 of [RFC7482] says: "domains?nsIp=ZZZZ. ZZZZ is a search pattern representing an IPv4 [RFC1166] or IPv6 [RFC5952] address.". Search pattern has been used throughout the document as something that can include ‘*’, while here, it does not. The syntax statement is also misleading. Similarly, section 3.2.2 says:

"nameservers?ip=YYYY YYYY is a search pattern representing an IPv4 [RFC1166] or IPv6 [RFC5952] address."

Recommendation: in [RFC7482], replace: "ZZZZ is a search pattern representing an IPv4" by "ZZZZ is an IPv4", "Syntax: domains?nsIp=<domain search pattern>" by "Syntax: domains?nsIp=<nameserver IP address>", "YYYY is a search pattern representing an IPv4" by "YYYY is an IPv4", "Syntax: nameservers?ip=<nameserver search pattern>" by "Syntax: nameservers?ip=<nameserver IP address>"
7. IANA RDAP Bootstrap Registries Related Issues

This section describes issues related to the IANA Bootstrap registries as specified in [RFC7484].

7.1. Missing Trailing Char in Bootstrap Registries

[RFC7484] section 3 says: "Base RDAP URLs MUST have a trailing "/" character". However, some values in the various IANA Bootstrap registries do not have the trailing "/" character. These should be added to provide consistency.

7.2. Single target value

[RFC7484] provides a way to list multiple RDAP servers for an entry. This flexibility was designed initially to support multiple URI types, such as http: and https, and to provide some level of redundancy. However, given that security deployment policy is to use https everywhere and redundancy can be accomplished in other ways, deployment has shown that all entries in all bootstrap registries have a single target RDAP URL value. Therefore, we can consider updating the RFC to provide only one target value. However, this should be done carefully to avoid breaking current deployed clients.

8. Security Considerations

Proper conformance to specifications helps security. However, no security issues have been found in the context of this draft.

9. IANA Considerations

This document request IANA to add the following values to this registry. TBD. See ‘IANA Action:’ within the document.

10. Acknowledgements

Audric Schiltknecht, Mario Loffredo, Justin Mack, James Gould have provided input and suggestions to this document.

11. References

11.1. Normative References


11.2. Informative References


11.3. URIs

[1] https://www.iana.org/assignments/rdap-json-values/rdap-json-values.xhtml


[3] https://www.iana.org/assignments/link-relations/link-relations.xhtml


Author’s Address
Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on February 6, 2020.
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.

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

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

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

"Storing Authorization Information Securely": The EPP RFCs don’t specify where and how the authorization information is stored in
the client or the server, so there are no restrictions to define an operational practice for storing the authorization information securely. The operational practice will not require the client to store the authorization information and will require the server to store the authorization information using a cryptographic hash. Returning the authorization information set in an EPP info response will not be supported.

1.1. Conventions Used in This Document

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

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

In examples, "C:" represents lines returned by a protocol client. 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 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. Secure Authorization Information

The authorization information in the EPP RFCs ([RFC5731] and [RFC5733]) that support transfer use password-based authorization information. Other EPP objects that support password-based authorization information for transfer can use the Secure Authorization Information defined in this document. For the authorization information to be secure, it must be a strong random value and must have a short time-to-live (TTL). The security of the authorization information is defined in the following sections.
3.1. 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 \( L = \text{ROUNDUP}(H / \log_2 N) \). With a target entropy of 49, the required length can be calculated after deciding on the set of characters that will be randomized. The following are a set of possible character sets and the calculation of the required length.

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

\[
\text{ROUNDUP}(49 / \log_2 94) = \text{ROUNDUP}(49 / 6.55) = \text{ROUNDUP}(7.48) = 8
\]

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

\[
\text{ROUNDUP}(49 / \log_2 36) = \text{ROUNDUP}(49 / 5.17) = \text{ROUNDUP}(9.48) = 10
\]

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

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

\[
\text{ROUNDUP}(128 / \log_2 94) = \text{ROUNDUP}(128 / 6.55) = \text{ROUNDUP}(19.54) = 20
\]

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).

\[
\text{ROUNDUP}(128 / \log_2 36) = \text{ROUNDUP}(128 / 5.17) = \text{ROUNDUP}(24.76) = 25
\]

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 [1] 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.

3.2. 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.

3.3. Authorization Information Storage and Transport

To protect the disclosure of the authorization information, the authorization information MUST be stored by the registry using a strong one-way cryptographic hash, MUST NOT be stored by the losing registrar, and MUST only be stored by the gaining registrar as a "transient" value in support of the transfer process. The plain text version of the authorization information MUST NOT be written to any logs by the registrar or the registry. All communication that includes the authorization information MUST be over an encrypted channel, such as [RFC5734] for EPP. The registrar's interface for communicating the authorization information with the registrant MUST be over an authenticated and encrypted channel.

4. Create, Transfer, and Secure Authorization Information

To make the transfer process secure using secure authorization information, as defined in Section 3, the client and server need to implement steps where the authorization information is set only when a transfer is actively in process and ensure that the authorization information is stored securely and transported only over secure channels. The steps in management of the authorization information for transfers include:
1. Registrant requests to register the object with the registrar.
   Registrar sends the create command, with empty authorization
   information, to the registry, as defined in Section 4.1.

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 4.2, 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 4.3.

6. Gaining registrar sends the transfer request with the
   authorization information to the registry, as defined in
   Section 4.4.

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 4.2.

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.

4.1. Create Command

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

C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:authInfo>
C:          <domain:pw/>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:  </command>
C: </epp>
Example of passing empty authorization information in an [RFC5733] contact create command.

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

4.2. 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 3.1, and setting it in the registry in the update command. The registry SHOULD validate the randomness of the authorization information based on the length and character set required by the registry. For example, a registry that requires 20 random printable ASCII characters except space
(0x20), should validate that the authorization information contains at least one upper case alpha character, one lower case alpha character, and one non-alpha numeric character. If the authorization information fails the randomness validation, the registry MUST return an EPP error result code of 2202.

Often the registrar has the "clientTransferProhibited" status set, so to start the transfer process, the "clientTransferProhibited" status needs to be removed, and the strong, random authorization information value needs to be set. The registrar MUST define a time-to-live (TTL), as defined in Section 3.2, 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</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 unseting the authorization information value and may add back statuses like the "clientTransferProhibited" status. Any EPP object extension that supports setting the authorization information with a "eppcom:pwAuthInfoType" element, can have an empty authorization information passed, such as [RFC5731] and [RFC5733]. Setting an empty authorization information unsets the value. [RFC5731] supports an explicit mechanism of unsetting the authorization information, by
passing the `<domain:null>` authorization information value. The registry MUST support unsetting the authorization information by accepting an empty authorization information value and accepting an explicit unset element if it is supported by the object extension.

Example of unsetting the authorization information explicitly in an [RFC5731] domain name update command.

```xml
C: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:add>
C:          <domain:status s="clientTransferProhibited"/>
C:        </domain:add>
C:        <domain:chg>
C:          <domain:authInfo>
C:            <domain:null/>
C:          </domain:authInfo>
C:        </domain:chg>
C:      </domain:update>
C:    </update>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C: </epp>
```
Example of unsetting the authorization information with an empty authorization information in an [RFC5731] domain name update command.

```xml
<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 xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <C:domain:name>example.com</C: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</C:clTRID>
  </C:command>
</C:epp>
```

Example of unsetting the authorization information with an empty authorization information in an [RFC5733] contact update command.

```xml
<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 xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
        <C:contact:id>sh8013</C: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</C:clTRID>
  </C:command>
</C:epp>
```
4.3. Info Command and Response

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

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

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

The Info Response in object extensions, such as [RFC5731] and [RFC5733], MUST NOT include the optional authorization information element. The authorization information is stored as a hash in the registry, so returning the plain text authorization information is not possible. The registry MUST NOT return any indication of whether the authorization information is set or unset by not returning the authorization information element in the response.

4.4. Transfer Request Command

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

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

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

Upon successful completion of the transfer, the registry MUST
automatically unset the authorization information. If the transfer
does not complete within the time-to-live (TTL) (Section 3.2), the
registrar MUST unset the authorization information as defined in
Section 4.2.

5. 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".

5.1. 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-gould-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


5.2. 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

6. Security Considerations

TBD

7. Acknowledgements

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

- Scott Hollenbeck
- Jody Kolker
- Patrick Mevzek
- Matthew Pozun
- Srikanth Veeramachaneni

8. References

8.1. Normative References


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.

Authors' Addresses

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

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

The Registration Data Access Protocol (RDAP) is used by Regional Internet Registries (RIRs) and Domain Name Registries (DNRs) to provide access to their resource registration information. RDAP uses jCard to convey information about individuals and other entities: for example, for the technical contact for a domain name. In practice, server operators are only using a small subset of jCard’s functionality, so in an effort to simplify the requirements on the client side, this document defines a jCard profile for use with RDAP.

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 January 8, 2020.

Copyright Notice

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

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

The Registration Data Access Protocol (RDAP) [RFC7480] is used by Regional Internet Registries (RIRs) and Domain Name Registries (DNRs) to provide access to their resource registration information. RDAP uses jCard ([RFC7095]) to convey information about individuals and other entities (e.g. organisations and groups). jCard is in turn a way of representing vCard ([RFC6350]) information in JSON.

The core vCard specification defines 36 properties, 11 parameters, 12 data types, and 31 parameter values. More of each are defined in subsequent specifications (see the IANA vCard Elements registry). Due to the lack of supporting libraries for jCard, RDAP client developers often have to implement jCard support themselves, and handling the entire specification is a substantial burden.

This document defines a jCard profile for use with RDAP that will reduce the implementation complexity for client developers. The profile is primarily based on response data from the implementations that are currently included in the IANA RDAP bootstrap files.

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. Profile Definition

The following properties may be used in jCards included in RDAP responses:

kind;
fn;
n;
adr;
tel;
email;
lang;
geo;
title;
role;
org;
version; and
contact-uri.

Each jCard MUST contain a "kind" property. The value of that property MUST be "individual", "group", or "org".

A "geo" property MUST have a type of "uri". A "tel" property MUST have a type of "uri" or "text". A "lang" property MUST have a type of "language-tag". A "contact-uri" property MUST have a type of "uri". All other properties MUST have a type of "text".

An "adr" property MUST include a "label" parameter, containing the content of the delivery address as a single string.

Properties may include "language", "altid", and "pref" parameters.

"email", "org" and "adr" properties may include a "type" parameter. The "type" parameter values that may be used are "work" and "home".
A "tel" property may include a "type" parameter. The "type" parameter values that may be used are those defined in [RFC6350].

An "adr" property may include a "cc" parameter. This parameter may be set even when the "country name" component of the property’s value is not set.

Properties, types, and parameters not expressly permitted by way of this profile MUST NOT be used.

3. Operational Considerations

An RDAP client that encounters a jCard that is not in conformance with this specification SHOULD treat the jCard as if any non-conforming properties, parameters, or types were not present.

Various server implementations are currently using the non-standard "ISO-3166-1-alpha-2" property for the ISO-3166-1 alpha 2 country code of the country from the "adr" property in the jCard. This behaviour appears to be based on guidance from section 1.4.1 of ICANN’s current RDAP response profile ([ICANN-RDAP-PROFILE]). However, that section states that it only applies when the "ISO-3166-1-alpha-2" property "has been published in the vCardProperties registry defined in Section 10.3.1 of RFC 6350", and that property has not yet been published in that way.

Various server implementations are currently setting the "country name" component of the "adr" property to be the ISO-3166-1 alpha 2 country code of the country. This behaviour appears to be based on guidance from section 1.4.2 of ICANN’s current RDAP response profile ([ICANN-RDAP-PROFILE]). The "cc" parameter approach to this problem, defined in [RFC8605], is preferred in this profile.

4. Security Considerations

TBD

5. IANA Considerations

TBD

6. References

6.1. Normative References
6.2. Informative References


Authors’ Addresses

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

Email: tomh@apnic.net

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

Email: ggm@apnic.net
Extensible Provisioning Protocol (EPP) Domain Name Mapping Extension for Strict Bundling Registration
draft-ietf-regext-bundling-registration-09

Abstract

This document describes an extension of Extensible Provisioning Protocol (EPP) domain name mapping for the provisioning and management of strict bundling registration of domain names. Specified in XML, this mapping extends the EPP domain name mapping to provide additional features required for the provisioning of bundled domain names.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 2, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents
Bundled domain names are those which share the same TLD but whose second level labels are variants, or those which have identical second level labels for which certain parameters are shared in different TLDs. For example, Public Interest Registry, request to implement technical bundling of second level domains for .NGO and .ONG. So we have two kinds of bundled domain names. First one is in the form of "V-label.TLD" in which the second level labels (V-label) are variants sharing the same TLD; Second one is in the form of "LABEL.V-tld" in which the second level labels (LABEL) are same ending with the different TLDs (V-tld);

Bundled domain names normally share some attributes. There are three types of bundling. First one is strict bundling, which requires all bundled names to share many same attributes. When creating, updating, or transferring of any of the bundled domain names, all bundled domain names will be created, updated or transferred. Second one is partial bundling, which requires that at least the bundled domain names if registered should be registered by the same registrant. Third one is relax bundling, which has not specific requirements to the domain registration. This document mainly focus on strict bundling names registration.

For the name variants, some registries adopt the policy that variant IDNs which are identified as equivalent are allocated or delegated to the same registrant. For example, the specified registration policy of Chinese Domain Name (CDN) is that a registrant can apply an original CDN in any forms: Simplified Chinese (SC) form, Traditional Chinese (TC) form, or other variant forms, then the corresponding variant CDN in SC form and that in TC form will also be delegated to the same registrant. All variant names in the same TLD contain same attributes.

The basic Extensible Provisioning Protocol (EPP) domain name mapping [RFC5731] provides the domain name registration one by one. It does not specify how to register the strict bundled names which share many same attributes.

In order to meet above requirements of the strict bundled names registration, this document describes an extension of the EPP domain name mapping [RFC5731] for the provisioning and management of bundled names. This document is specified using the Extensible Markup Language (XML) 1.0 as described in [W3C.REC-xml-20040104] and XML Schema
notation as described in [W3C.REC-xmlschema-1-20041028] and [W3C.REC-xmlschema-2-20041028].

The EPP core protocol specification [RFC5730] provides a complete description of EPP command and response structures. A thorough understanding of the base protocol specification is necessary to understand the extension of mapping described in this document.

This document uses lots of the concepts of the IDN, so a thorough understanding of the IDNs for Application (IDNA, described in [RFC5890], [RFC5891], and [RFC5892]) and a thorough understanding of variant approach discussed in [RFC4290] are both required.

2. Terminology

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].

uLabel in this document is used to express U-label of the internationalized domain name into series of characters where non-ASCII characters will be represented with the format of U+XXXX where XXXX is a UNICODE point. U-Label is defined in [RFC5890].

"b-dn-1.0" in this document is used as an abbreviation for urn:ietf:params:xml:ns:epp:b-dn-1.0.

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

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

3. Definitions

The following definitions are used in this document:

- Registered Domain Name (RDN), represents the valid domain name that users submitted for registration by the first time.

- Bundled Domain Name (BDN), represents the bundled domain name produced according to the bundled domain name registration policy.
4. Overview

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

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

The bundled names share many same properties, such as expiration date and sponsoring registrar, by sharing one domain object. So when users update any property of a domain object within a bundle package, that property of all other domain objects in the bundle package will be updated at the same time.

5. Requirement for Bundling Registration of Names

The bundled names whether they are in the form of "V-label.TLD" or in the form of "LABEL.V-tld" should share some parameter or attributes associated with domain names. Typically, Bundled names will share the following parameters or attributes:
- Registrar Ownership
- Registration and Expiry Dates
- Registrant, Admin, Billing, and Technical Contacts
- Name Server Association
- Domain Status
- Applicable grace periods (Add Grace Period, Renewal Grace Period, Auto-Renewal Grace Period, Transfer Grace Period, and Redemption Grace Period)

Because the domain names are bundled and share the same parameters or attributes, the EPP command should do some processing for these requirements:
- When performing a domain check, either BDN or RDN can be queried for the EPP command, and will return the same response.
- When performing a domain info, either BDN or RDN can be queried, the same response will include both BDN and RDN information with the same attributes.
o When performing a domain Create, either BDN or RDN will be accepted. If the domain name is available, both BDN and RDN will be registered.

o When performing a domain Delete, either BDN or RDN will be accepted. If the domain name is available, both BDN and RDN will be deleted.

o When performing a domain renew, either BDN or RDN will be accepted. Upon a successful domain renewal, both BDN and RDN will have their expiry date extended by the requested term. Upon a successful domain renewal, both BDN and RDN will conform to the same renew grace period.

o When performing a domain transfer, either BDN or RDN will be accepted. Upon successful completion of a domain transfer request, both BDN and RDN will enter a pendingTransfer status. Upon approval of the transfer request, both BDN and RDN will be owned and managed by the same new registrant.

o When performing a domain update, either BDN or RDN will be accepted. Any modifications to contact associations, name server associations, domain status values and authorization information will be applied to both BDN and RDN.

6. Object Attributes

This extension defines following additional elements to the EPP domain name mapping [RFC5731]. All of these additional elements can be got from <domain:info> command.

6.1. RDN

The RDN is an ASCII name or an IDN with the A-label [RFC5890] form. In this document, its corresponding element is <b-dn:rdn>. An optional attribute "uLabel" associated with <b-dn:rdn> is used to represent the U-label [RFC5890] form. An optional boolean "activated" attribute, with a default true value, is used to indicate the presence of the label in the zone file.

For example: <b-dn:rdn uLabel="U+5B9E"U+4F8B".example> xn--fsq270a.example</b-dn:rdn>

6.2. BDN

The BDN is an ASCII name or an IDN with the A-label [RFC5890] form which is converted from the corresponding BDN. In this document, its corresponding element is <b-dn:bdn>. An optional attribute "uLabel" associated with <b-dn:bdn> is used to represent the U-label [RFC5890] form.
For example: <b-dn:bdn uLabel="U+5BE6"U+4F8B".example> xn--fsql41a.example</b-dn:bdn>

7. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730]. The command mappings described here are specifically for use in provisioning and managing bundled names via EPP.

7.1. EPP Query Commands

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

7.1.1. EPP <check> Command

This extension does not add any element to the EPP <check> command or <check> response described in the EPP domain name mapping [RFC5731]. However, when either RDN or BDN is sent for check, response SHOULD contain both RDN and BDN information, which may also give some explanation in the reason field to tell the user that the associated domain name is a produced name according to some bundle domain name policy.
Example <check> Response for an authorized client:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <resData>
      <domain:chkData
        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:cd>
          <domain:name avail="1">
            xn--fsq270a.example
          </domain:name>
        </domain:cd>
        <domain:cd>
          <domain:name avail="1">
            xn--fsqz41a.example
          </domain:name>
          <domain:reason>This associated domain name is a produced name based on bundle name policy.</domain:reason>
        </domain:cd>
      </domain:chkData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54322-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

7.1.2. EPP <info> Command

This extension does not add any element to the EPP <info> command described in the EPP domain mapping [RFC5731]. However, additional elements are defined for the <info> response.

When an <info> command has been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. In addition, the EPP <extension> element SHOULD contain a child <b-dn:infData> element that identifies the extension namespace if the domain object has data associated with this extension and based on its service policy. The <b-dn:infData> element contains the <b-dn:bundle> which has the following child elements:

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

The above elements contain the following attributes:

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

Example <info> Response for an authorized client:

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <resData>
S:       <domain:infData
S:           xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>xn--fsq270a.example</domain:name>
S:         <domain:roid>58812678-domain</domain:roid>
S:         <domain:status s="ok"/>
S:         <domain:registrant>123</domain:registrant>
S:         <domain:contact type="admin">123</domain:contact>
S:         <domain:contact type="tech">123</domain:contact>
S:         <domain:ns>
S:           <domain:hostObj>ns1.example.cn</domain:hostObj>
S:         </domain:ns>
S:         <domain:clID>ClientX</domain:clID>
S:         <domain:crID>ClientY</domain:crID>
S:         <domain:crDate>2011-04-03T22:00:00.0Z</domain:crDate>
S:         <domain:exDate>2012-04-03T22:00:00.0Z</domain:exDate>
S:         <domain:authInfo>
S:           <domain:pw>2fooBAR</domain:pw>
S:         </domain:authInfo>
S:       </domain:infData>
S:     </resData>
S:   </extension>
S: </epp>
<info> Response for the unauthorized client has not been changed, see [RFC5731] for detail.

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

7.1.3. EPP <transfer> Query Command

This extension does not add any element to the EPP <transfer> command or <transfer> response described in the EPP domain mapping [RFC5731].

7.2. EPP Transform Commands

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

When these commands have been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. This EPP <extension> element SHOULD contain the <b-dn:bundle> which has the following child elements:

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

The above elements contain the following attribute:

- An optional "uLabel" attribute represents the U-label of the element.
7.2.1. EPP <create> Command

This extension defines additional elements to extend the EPP <create> command described in the EPP domain name mapping [RFC5731] for bundled names registration.

In addition to the EPP command elements described in the EPP domain mapping [RFC5731], the <create> command SHALL contain an <extension> element. The <extension> element SHOULD contain a child <b-dn:create> element that identifies the bundle namespace and the location of the bundle name schema.

Example <create> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:      <command>
C:        <create>
C:          <domain:create
C:            xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:            <domain:name>xn--fsq270a.example</domain:name>
C:            <domain:period unit="y">2</domain:period>
C:            <domain:registrant>123</domain:registrant>
C:            <domain:contact type="admin">123</domain:contact>
C:            <domain:contact type="tech">123</domain:contact>
C:            <domain:authInfo>
C:              <domain:pw>2fooBAR</domain:pw>
C:            </domain:authInfo>
C:          </domain:create>
C:        </create>
C:        <extension>
C:          <b-dn:create
C:            xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn-1.0">
C:            <b-dn:rdn uLabel="U+5B9E"U+4F8B".example>
C:              xn--fsq270a.example</b-dn:rdn>
C:          </b-dn:create>
C:        </extension>
C:        <clTRID>ABC-12345</clTRID>
C:      </command>
C:</epp>
```

When an <create> command has been processed successfully, the EPP <creData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. In addition, the EPP <extension> element SHOULD contain a child <b-dn:creData> element that identifies the extension namespace if the domain object has data associated with this extension and based on its service policy. The <b-dn:creData> element contains the <b-dn:bundle> element.
Example <create> Response for an authorized client:

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:     <response>
S:       <result code="1000">
S:         <msg>Command completed successfully</msg>
S:       </result>
S:       <resData>
S:         <domain:creData
S:             xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:           <domain:name>xn--fsq270a.example</domain:name>
S:           <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:           <domain:exDate>2001-04-03T22:00:00.0Z</domain:exDate>
S:         </domain:creData>
S:       </resData>
S:       <extension>
S:         <b-dn:creData
S:             xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn-1.0">
S:           <b-dn:bundle>
S:             <b-dn:rdn uLabel="U+5B9E"U+4F8B.example
S:               >xn--fsq270a.example</b-dn:rdn>
S:             <b-dn:bdn uLabel="U+5BE6"U+4F8B.example
S:               >xn--fsqz41a.example</b-dn:bdn>
S:           </b-dn:bundle>
S:         </b-dn:creData>
S:       </extension>
S:       <trID>
S:         <clTRID>ABC-12345</clTRID>
S:         <svTRID>54322-XYZ</svTRID>
S:       </trID>
S:     </response>
S:</epp>

=create> Response for the unauthorized client has not been changed, see [RFC5731] for detail.

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

7.2.2. EPP <delete> Command

This extension does not add any element to the EPP <delete> command described in the EPP domain mapping [RFC5731]. However, additional elements are defined for the <delete> response.

When a <delete> command has been processed successfully, the EPP <delData> element MUST contain child elements as described in the EPP
domain mapping [RFC5731]. In addition, the EPP <extension> element SHOULD contain a child <b-dn:delData> element that identifies the extension namespace if the domain object has data associated with this extension and based on its service policy. The <b-dn:delData> element SHOULD contain the <b-dn:bundle> element.

Example <delete> response:

S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <extension>
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="U+5B9E"U+4F8B".example xn--fsq270a.example</b-dn:rdn>
S:          <b-dn:bdn uLabel="U+5BE6"U+4F8B".example xn--fsqz41a.example</b-dn:bdn>
S:        </b-dn:bundle>
S:      </b-dn:delData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>

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

7.2.3.  EPP <renew> Command

This extension does not add any element to the EPP <renew> command described in the EPP domain name mapping [RFC5731]. However, when either RDN or BDN is sent for renew, response SHOULD contain both RDN and BDN information. When the command has been processed successfully, the EPP <extension> element SHOULD be contained in the response if the domain object has data associated with bundled names. This EPP <extension> element SHOULD contain the <b-dn:renData> which contains <b-dn:bundle> element.
Example <renew> Response for an authorized client:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:renData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>xn--fsq270a.example</domain:name>
S:        <domain:exDate>2012-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:renData>
S:    </resData>
S:     <extension>
S:       <b-dn:renData
S:        xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn-1.0">
S:         <b-dn:bundle>
S:            <b-dn:rdn uLabel="U+5B9E"u+4F8B".example
S:              >xn--fsq270a.example</b-dn:rdn>
S:            <b-dn:bdn uLabel="U+5BE6"u+4F8B".example
S:              >xn--fsgz41a.example</b-dn:bdn>
S:         </b-dn:bundle>
S:       </b-dn:renData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

7.2.4. EPP <transfer> Command

This extension does not add any element to the EPP <transfer> command described in the EPP domain name mapping [RFC5731]. However, additional elements are defined for the <transfer> response in the EPP object mapping. When the command has been processed successfully, the EPP <extension> element SHOULD be contained in the response if the domain object has data associated with bundled names. This EPP <extension> element SHOULD contain the <b-dn:trnData> which contains <b-dn:bundle> element.
Example <transfer> Response for an authorized client:

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1001">
S:       <msg>Command completed successfully; action pending</msg>
S:     </result>
S:     <resData>
S:       <domain:trnData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>xn--fsq270a.example</domain:name>
S:         <domain:trStatus>pending</domain:trStatus>
S:         <domain:reID>ClientX</domain:reID>
S:         <domain:reDate>2011-04-03T22:00:00.0Z</domain:reDate>
S:         <domain:acID>ClientY</domain:acID>
S:         <domain:acDate>2011-04-08T22:00:00.0Z</domain:acDate>
S:         <domain:exDate>2012-04-03T22:00:00.0Z</domain:exDate>
S:       </domain:trnData>
S:     </resData>
S:     <extension>
S:       <b-dn:trnData
S:         xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn-1.0">
S:         <b-dn:bundle>
S:           <b-dn:rdn uLabel="U+5B9E"U+4F8B".example
S:             >xn--fsq270a.example</b-dn:rdn>
S:           <b-dn:bdn uLabel="U+5BE6"U+4F8B".example
S:             >xn--fsqz41a.example</b-dn:bdn>
S:         </b-dn:bundle>
S:       </b-dn:trnData>
S:     </extension>
S:   </trID>
S: </response>
S:</epp>

7.2.5. EPP <update> Command

This extension does not add any element to the EPP <update> command described in the EPP domain name mapping [RFC5731]. However, additional elements are defined for the <update> response in the EPP object mapping. When the command has been processed successfully, the EPP <extension> element SHOULD be contained in the response if the domain object has data associated with bundled names. This EPP <extension> element SHOULD contain the <b-dn:upData> which contains <b-dn:bundle> element.
Example <update> Response for an authorized client:

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <extension>
S:       <b-dn:upData
S:         xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn-1.0">
S:         <b-dn:bundle>
S:           <b-dn:rdn uLabel="U+5B9E"U+4F8B".example
S:             >xn--fsq270a.example</b-dn:rdn>
S:           <b-dn:bdn uLabel="U+5B9E"U+4F8B".example
S:             >xn--fsqz41a.example</b-dn:bdn>
S:         </b-dn:bundle>
S:       </b-dn:upData>
S:     </extension>
S:   </response>
S: </epp>

8. Formal Syntax

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

BEGIN
<?xml version="1.0" encoding="UTF-8"?>

<schema targetNamespace="urn:ietf:params:xml:ns:epp:b-dn-1.0"
xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn-1.0"
xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
xmlns="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">

<!--
Import common element types.
-->

<import namespace="urn:iana:xml:ns:eppcom-1.0"
schemaLocation="eppcom-1.0.xsd"/>
<import namespace="urn:iana:xml:ns:epp-1.0"
schemaLocation="epp-1.0.xsd"/>
<annotation>
  <documentation>
    Extensible Provisioning Protocol v1.0
    Bundle Domain Extension Schema v1.0
  </documentation>
</annotation>

<!--
Child elements found in EPP commands.
-->
<element name="create" type="b-dn:createDataType"/>

<!--
Child elements of the <b-dn:create> command
All elements must be present at time of creation
-->
<complexType name="createDataType">
  <sequence>
    <element name="rdn" type="b-dn:rdnType"
      minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>

<!--
Child elements of the <b-dn:update> command
All elements must be present at time of creation
-->

<!--
Child elements found in EPP commands.
-->
<element name="infData" type="b-dn:trnDataType"/>
<element name="delData" type="b-dn:trnDataType"/>
<element name="creData" type="b-dn:trnDataType"/>
<element name="renData" type="b-dn:trnDataType"/>
<element name="trnData" type="b-dn:trnDataType"/>
<element name="upData" type="b-dn:trnDataType"/>
<complexType name="trnDataType">
  <sequence>
    <element name="bundle" type="b-dn:bundleType" />
  </sequence>
</complexType>
<!--
<transfer> response elements. All elements must be present at time of poll query -->
<complexType name="bundleType">
  <sequence>
    <element name="rdn" type="b-dn:rdnType" />
    <element name="bdn" type="b-dn:rdnType"
      minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>
</complexType>
<complexType name="rdnType">
  <simpleContent>
    <extension base="eppcom:labelType">
      <attribute name="uLabel" type="eppcom:labelType"/>
    </extension>
  </simpleContent>
</complexType>

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

END

9. Internationalization Considerations

EPP is represented in XML, which provides native support for encoding information using the Unicode character set and its more compact representations including UTF-8. Conformant XML processors recognize both UTF-8 and UTF-16. Though XML includes provisions to identify and use other character encodings through use of an "encoding" attribute in an <?xml?> declaration, use of UTF-8 is RECOMMENDED.

As an extension of the EPP domain name mapping, the elements, element content described in this document MUST inherit the internationalization conventions used to represent higher-layer domain and core protocol structures present in an XML instance that includes this extension.
10. IANA Considerations

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

Registration request for the IDN namespace:
- Registrant Contact: See the "Author’s Address" section of this document.
- XML: None. Namespace URI does not represent an XML specification.

Registration request for the IDN XML schema:
- Registrant Contact: See the "Author’s Address" section of this document.
- XML: See the "Formal Syntax" section of this document.

The EPP extension described in this document should be registered by IANA in the "Extensions for the Extensible Provisioning Protocol (EPP)" registry described in [RFC7451]. The details of the registration are as follows:

- Name of Extension: "Domain Name Mapping Extension for Strict Bundling Registration"
- Document status: Informational
- Reference: This document
- Registrant Name and Email Address: IESG, iesg@ietf.org
- Top-Level Domains (TLDs): Any
- IPR Disclosure: https://datatracker.ietf.org/ipr/
- Status: Active
- Notes: None
11. Security Considerations

Some registries and registrars have more than 15 years of the bundled registration of domain names (especially Chinese domain names). They have not found some significant security issues. One principle that the registry and registrar should let the registrants know is that bundled registered domain names will be created, transferred, updated, and deleted together as a group. The registrants for bundled domain names should remember this principle when doing some operations to these domain names. [RFC5730] also introduces some security consideration.

This document does not take a position regarding whether or not the bundled domain names share a DS/DNSKEY key. The DNS administrator can choose whether DS/DNSKEY information can be shared or not. If a DS/DNSKEY key is shared then the bundled domain names share fate if there is a key compromise.

12. Implementation Status

Note to RFC Editor: Please remove this section before publication.

- The Chinese Domain Name Consortium (CDNC) including CNNIC, TWNIC, HKIRC, MONIC, SGNIC and more have followed the principles defined in this document for many years.
- CNNIC and TELEINFO have implemented this extension in their EPP based Chinese domain name registration system.
- Public Interest Registry, has requested to implement technical bundling of second level domains for .NGO and .ONG. This means that by registering and purchasing a domain in the .ngo TLD, for example, the NGO registrant is also registering and purchasing the corresponding name in the .ong TLD (and vice-versa for registrations in .ong).
- Patrick Mevzek has released a new version of Net::DRI, an EPP client (Perl library, free software) implementing this extension.

13. Acknowledgements

The authors especially thank the authors of [RFC5730] and [RFC5731] and the following ones of CNNIC: Weiping Yang, Chao Qi.

Useful comments were made by John Klensin, Scott Hollenbeck, Patrick Mevzek and Edward Lewis.
14. Change History

RFC Editor: Please remove this section.

14.1. draft-kong-epp-bundle-mapping: Version 00
   - EPP extension for bundled domain name registrations.

14.2. draft-kong-epp-bundle-mapping: Version 01
   - Change the proposed category from EXP to STD.
   - Add the section of Implementation Status.
   - Refine the text, and update the examples.

14.3. draft-kong-epp-bundle-mapping: Version 02
   - Refine the texts.

14.4. draft-ietf-regext-bundle-mapping: Version 00
   - accepted as WG document.

14.5. draft-ietf-regext-bundle-mapping: Version 01
   - make this document to focus on the restrict bundled domain name registration.

14.6. draft-ietf-regext-bundle-mapping: Version 02
   - Update the section of implementation status.

14.7. draft-ietf-regext-bundle-mapping: Version 03
   - This document is changed to informational category.
   - Refine the text.

14.8. draft-ietf-regext-bundle-mapping: Version 04
   - Update the implementation section.
   - Refine the text.
14.9. draft-ietf-regext-bundle-mapping: Version 05
  o Scope the XML namespaces to include ‘epp’.

14.10. draft-ietf-regext-bundle-mapping: Version 06
  o add some examples for the transfer, update and renew command
  o add some text to security consideration

14.11. draft-ietf-regext-bundle-mapping: Version 07
  o Update IANA consideration section based on Scott’s comments
  o Update security consideration based on Chair and Patrick Mevzek’s comments

15. References

15.1. Normative References


15.2. Informative References


Authors’ Addresses

Ning Kong
Consultant
Email: ietfing@gmail.com

Jiankang Yao (editor)
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China
Phone: +86 10 5881 3007
Email: yaojk@cnnic.cn

Linlin Zhou
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China
Phone: +86 10 5881 2677
Email: zhoulinlin@cnnic.cn

Wil Tan
Cloud Registry
Suite 32 Seabridge House, 377 Kent St
Sydney, NSW 2000
Australia
Phone: +61 414 710899
Email: wil@cloudregistry.net

Jiagui Xie
Email: jiagui1984@163.com
Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on December 20, 2019.

Copyright Notice

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

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

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

The goal of data escrow is higher resiliency of registration services, for the benefit of Internet users. The beneficiaries of a registry are not just those registering information there, but all relying parties that need to identify the owners of objects.

In the context of domain name registries, registration data escrow is a requirement for generic top-level domains and some country code top-level domain managers are also currently escrowing data. There is also a similar requirement for ICANN-accredited domain registrars.

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

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119].

DEPOSIT. Deposits can be of three kinds: Full, Differential or Incremental. For all kinds of Deposits, the Universe of Registry objects to be considered for data escrow are those objects necessary in order to offer the Registry Services.

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

ESCROW AGENT. The organization designated by the Registry or the Third-Party Beneficiary to receive and guard Data Escrow Deposits from the Registry.

FULL DEPOSIT. Contains the Registry Data that reflects the current and complete Registry Database and will consist of data that reflects the state of the registry as of a defined Timeline Watermark for the deposit.

INCREMENTAL DEPOSIT. Contains data that reflects all transactions involving the database that were not reflected in the last previous Full Deposit. Incremental Deposit files will contain information from all database objects that were added, modified or deleted since
the previous Full Deposit was completed as of its defined Timeline Watermark. If the Timeline Watermark of an Incremental Deposit were to cover the Watermark of another (Incremental or Differential) Deposit since the last Full Deposit, the more recent Deposit MUST contain all the transactions of the earlier Deposit.

REGISTRY. A registration organization providing registration services for a certain type of objects, e.g., domain names, IP number resources, routing information.

THIRD-PARTY BENEFICIARY. Is the organization that, under extraordinary circumstances, would receive the escrow Deposits the Registry transferred to the Escrow Agent. This organization could be a backup Registry, Registry regulator, contracting party of the Registry, etc.

TIMELINE WATERMARK. Point in time on which to base the collecting of database objects for a Deposit. Deposits are expected to be consistent to that point in time.

3. Problem Scope

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

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

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

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

While the main motivation for developing this solution is rooted on the domain name industry, the specification has been designed to be as general as possible. This allows other types of registries to use the base specification and develop their own specifications covering the objects used by other registration organizations.

A solution to the problem at hand SHALL clearly identify the format and contents of the deposits a Registry has to make, such that a
different Registry would be able to rebuild the registration services of the former, without its help, in a timely manner, with minimum disruption to its users.

Since the details of the registration services provided vary from Registry to Registry, the solution SHALL provide mechanisms that allow its extensibility to accommodate variations and extensions of the registration services.

Given the requirement for confidentiality and the importance of accuracy of the information that is handled in order to offer registration services, the solution SHALL define confidentiality and integrity mechanisms for handling the registration data.

The solution SHALL NOT include in the specification transient objects that can be recreated by the new Registry, particularly those of delicate confidentiality, e.g., DNSSEC KSK/ZSK private keys.

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

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

4. General Conventions

4.1. Date and Time

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

5. Protocol Description

The following is a format for Data Escrow deposits as produced by a Registry. The deposits are represented in XML. Only the format of the objects deposited is defined, nothing is prescribed about the method used to transfer such deposits between the Registry and the Escrow Agent or vice versa.

The protocol intends to be object agnostic allowing the "overload" of abstract elements using the "substitutionGroup" attribute to define the actual elements of an object to be escrowed.
5.1. Root element <deposit>

The container or root element for a Registry Data Escrow deposits is <deposit>. This element contains the following child elements: watermark, deletes and contents. This element also contains the following attributes:

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

- A REQUIRED "id" attribute that is used to uniquely identify the escrow deposit. Each registry is responsible for maintaining its own escrow deposits identifier space to ensure uniqueness, e.g., using identifiers as described in Section 2.8 of [RFC5730].

- An OPTIONAL "prevId" attribute that can be used to identify the previous incremental, differential or full escrow deposit. This attribute MUST be used in Differential Deposits ("DIFF" type).

- An OPTIONAL "resend" attribute that is incremented each time the escrow deposit failed the verification procedure at the receiving party and a new escrow deposit needs to be generated by the Registry for that specific date. The first time a deposit is generated the attribute is either omitted or MUST be "0". If a deposit needs to be generated again, the attribute MUST be set to "1", and so on.

Example of root element object:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rde:deposit
    xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
    type="FULL"
    id="20101017001" prevId="20101010001">
    <rde:watermark>2010-10-18T00:00:00Z</rde:watermark>
    <rde:deletes>...
    </rde:deletes>
    <rde:contents>...
    </rde:contents>
</rde:deposit>
```
5.2. Child <watermark> element

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

Example of <watermark> element object:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rde:deposit
    xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
    ...;
type="FULL"
id="20101017001" prevId="20101010001">
    <rde:watermark>2010-10-18T00:00:00Z</rde:watermark>
    ...
</rde:deposit>
```

5.3. Child <rdeMenu> element

This element contains auxiliary information of the data escrow deposit.

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

- A REQUIRED <version> element that identifies the RDE protocol version.
- One or more <objURI> elements that contain namespace URIs representing the <contents> and <deletes> element objects.

Example of <rdeMenu> element object:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rde:deposit
    xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
    ...
<rde:rdeMenu>
    <rde:version>1.0</rde:version>
    <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:rdeMenu>
...
</rde:deposit>
```
5.4. Child <deletes> element

This element SHOULD be present in deposits of type Incremental or Differential. It contains the list of objects that were deleted since the base previous deposit. Each object in this section SHALL contain an ID for the object deleted.

This section of the deposit SHOULD NOT be present in Full deposits. When rebuilding a registry it SHOULD be ignored if present in a Full deposit.

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

Example of <deletes> element:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rde:deposit
 xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
 ...>
 <rde:deletes>
  <rdeObj1:delete>
   <rdeObj1:name>foo.test</rdeObj1:name>
   <rdeObj1:name>bar.test</rdeObj1:name>
  </rdeObj1:delete>
  <rdeObj2:delete>
   <rdeObj2:id>sh8013-TEST</rdeObj2:id>
   <rdeObj2:id>co8013-TEST</rdeObj2:id>
  </rdeObj2:delete>
 </rde:deletes>
 ... 
</rde:deposit>
```

5.5. Child <contents> element

This element of the deposit contains the objects in the deposit. It MUST be present in all type of deposits. It contains the data for the objects to be escrowed. The actual objects have to be specified individually.

In the case of Incremental or Differential deposits, the objects indicate whether the object was added or modified after the base previous deposit. In order to distinguish between one and the other, it will be sufficient to check existence of the referenced object in the base previous deposit.

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

If an object is present in the <contents> section of several Deposits (e.g. Full and Differential) the registry data from the latest Deposit (as defined by the Timeline Watermark) SHOULD be used when rebuilding the registry.

Example of <contents> element object:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rde:deposit
    xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
    ...
    <rde:contents>
        ...
        <rdeObj1:contents>
            <rdeObj1:element1>
                <rdeObj1:child1>Object1 specific.</rdeObj1:child1>
                ...
            </rdeObj1:element1>
            ...
        </rdeObj1:contents>
        ...
    </rde:contents>
    ...
</rde:deposit>
```

6. Formal Syntax

6.1. RDE Schema

Copyright (c) 2011 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Lozano
Expires December 20, 2019
Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote products derived from this software without specific prior written permission.

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

BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:rde-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">
  <annotation>
    <documentation>
      Registry Data Escrow schema
    </documentation>
  </annotation>
  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0"/>

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

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

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

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

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

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

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

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

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

<!-- A RDE version number is a dotted pair of decimal numbers -->
<simpleType name="versionType">
  <restriction base="token">
    <pattern value="[1-9]\.[0-9]="/>
    <enumeration value="1.0"/>
  </restriction>
</simpleType>

<complexType name="rrType">
  <simpleContent>
    <extension base="eppcom:clIDType">
      <attribute name="client" type="eppcom:clIDType"/>
    </extension>
  </simpleContent>
</complexType>
END

7. Internationalization Considerations

Data Escrow deposits are represented in XML, which provides native support for encoding information using the Unicode character set and its more compact representations including UTF-8. Conformant XML processors recognize both UTF-8 and UTF-16. Though XML includes provisions to identify and use other character encodings through use of an "encoding" attribute in an <?xml?> declaration, use of UTF-8 is RECOMMENDED.

8. IANA Considerations

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

Registration request for the RDE namespace:

URI: urn:ietf:params:xml:ns:rde-1.0
Registrant Contact: See the "Author’s Address" section of this document.

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

Registration request for the RDE XML schema:

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

Registrant Contact: See the "Author’s Address" section of this document.

See the "Formal Syntax" section of this document.

9. Implementation Status

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

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942 [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942 [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

9.1. Implementation in the gTLD space

Organization: ICANN

Name: ICANN Registry Agreement

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

Level of maturity: production.

Coverage: all aspects of this specification are implemented.

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

Contact: gustavo.lozano@icann.org

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

10. Security Considerations

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

Depending on local policies, some elements or most likely, the whole deposit will be considered confidential. As such the Registry transmitting the data to the Escrow Agent must take all the necessary precautions like encrypting the data itself and/or the transport channel to avoid inadvertent disclosure of private data.

Mutual authentication of both parties passing data escrow deposit files is of the utmost importance. The Escrow Agent should properly authenticate the identity of the Registry before accepting data escrow deposits. In a similar manner, the Registry should authenticate the identity of the Escrow Agent before submitting any data.

Additionally, the Registry and the Escrow Agent should use integrity checking mechanisms to ensure the data transmitted is what the source intended. It is recommended that specifications defining format and semantics for particular business models define an algorithm that Escrow Agents and Third-Party Beneficiaries could use to validate the contents of the data escrow deposit.
11. Acknowledgments

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

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

12. Change History

12.1. Changes from version 00 to 01

1. Included DNSSEC elements as part of the basic <domain> element as defined in RFC 5910.

2. Included RGP elements as part of the basic <domain> element as defined in RFC 3915.

3. Added support for IDNs and IDN variants.

4. Eliminated the <summary> element and all its subordinate objects, except <watermarkDate>.

5. Renamed <watermarkDate> to <watermark> and included it directly under root element.

6. Renamed root element to <deposit>.

7. Added <authinfo> element under <registrar> element.

8. Added <roid> element under <registrar> element.

9. Reversed the order of the <deletes> and <contents> elements.

10. Removed <rdeDomain:status> minOccurs="0".

11. Added <extension> element under root element.

12. Added <extension> element under <contact> element.

13. Removed <period> element from <domain> element.
15. Populated the "Internationalization Considerations" section.
16. Populated the "Extension Example" section.
17. Added <deDate> element under <domain> element.
18. Added <icannID> element under <registrar> element.
19. Added <eppParams> element under root element.
20. Fixed some typographical errors and omissions.

12.2. Changes from version 01 to 02
1. Added definition for "canonical" in the "IDN variants Handling" section.
2. Clarified that "blocked" and "reserved" IDN variants are optional.
4. Introduced substitutionGroup as the mechanism for extending the protocol.
5. Moved <eppParams> element to be child of <contents>
7. Removed <trDate> from <rdeDomain> and added <trnData> instead, which include all the data from the last (pending/processed) transfer request
8. Removed <trDate> from <rdeContact> and added <trnData> instead, which include all the data from the last (pending/processed) transfer request
9. Fixed some typographical errors and omissions.

12.3. Changes from version 02 to 03
1. Separated domain name objects from protocol.
2. Moved <extension> elements to be child of <deletes> and <contents>, additionally removed <extension> element from
<rdeDomain>, <rdeHost>, <rdeContact>, <rdeRegistrar> and <rdeIDN> elements.

3. Modified the definition of <rde:id> and <rde:prevId>.
4. Added <rdeMenu> element under <deposit> element.
5. Fixed some typographical errors and omissions.

12.4. Changes from version 03 to 04
1. Removed <eppParams> objects.
2. Populated the "Extension Guidelines" section.
3. Fixed some typographical errors and omissions.

12.5. Changes from version 04 to 05
1. Fixes to the XSD
2. Extension Guidelines moved to dnrd-mappings draft
3. Fixed some typographical errors and omissions.

12.6. Changes from version 05 to 06
1. Fix resend definition.

12.7. Changes from version 06 to 07
1. Editorial updates.
2. schemaLocation removed from RDE Schema.

12.8. Changes from version 07 to 08
1. Ping update

12.9. Changes from version 08 to 09
1. Ping update.

12.10. Changes from version 09 to 10
1. Implementation Status section was added
12.11. Changes from version 10 to 11

1. Ping update.

12.12. Changes from version 11 to 00

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

13. References

13.1. Normative References


13.2. Informative References


Author’s Address

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

Phone: +1.310.823.9358
Email: gustavo.lozano@icann.org
Domain Name Registration Data (DNRD) Objects Mapping
draft-ietf-regext-dnrd-objects-mapping-00

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 December 20, 2019.

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

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

1. Introduction ........................................... 3  
2. Models ................................................. 4  
3. Terminology ............................................ 5  
4. General Conventions ...................................... 5  
   4.1. Date and Time .................................. 5  
   4.2. Country names .................................. 5  
   4.3. Telephone numbers ................................. 6  
   4.4. Checksum ......................................... 6  
   4.5. IP addresses .................................... 6  
   4.6. CSV Parent Child Relationship .................... 6  
   4.7. CSV elements .................................... 7  
   4.8. Internationalized and Localized Elements ........ 12  
5. Object Description ....................................... 14  
   5.1. Domain Name Object ............................... 14  
   5.2. Host Object ...................................... 33  
   5.3. Contact Object .................................. 43  
   5.4. Registrar Object .................................. 60  
   5.5. IDN Table Reference Object ....................... 68  
   5.6. NNDN Object ...................................... 72  
   5.7. EPP Parameters Object ............................. 77  
   5.8. Policy Object .................................... 79  
   5.9. Header Object .................................... 79  
6. RDE IDN Variants handling ................................. 82  
7. Profile ................................................ 83  
8. Data escrow agent extended verification process .......... 83  
9. Formal Syntax ........................................... 84  
   9.1. RDE CSV Schema .................................... 84  
   9.2. RDE Domain Object ................................ 94  
   9.3. CSV Domain Object ................................ 97  
   9.4. RDE Host Object .................................. 101  
   9.5. CSV Host Object .................................. 104  
   9.6. RDE Contact Object ............................... 107  
   9.7. CSV Contact Object ............................... 110  
   9.8. RDE Registrar Object .............................. 116  
   9.9. CSV Registrar Object .............................. 120  
   9.10. RDE IDN Table Reference Objects ................ 124  
   9.11. CSV IDN Language Object ........................ 126  
   9.12. EPP Parameters Object ........................... 128  
   9.13. NNDN Object ..................................... 130  
   9.14. CSV NNDN Object ................................ 132  
   9.15. Policy Object .................................... 135  
   9.16. Header Object .................................... 136  
10. Internationalization Considerations ...................... 138  
11. IANA Considerations .................................... 138  
12. Implementation Status .................................. 144  
   12.1. Implementation in the gTLD space ................. 144
1. Introduction

This document defines the data escrow structure of the standard set of objects for a Domain Name Registry which 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): A lightweight domain-like object that is not linked to a Registrar.

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: Definition of the specific 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.

2. Models

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

- 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).

- 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).

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

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119].

REGISTRY. In the context of this draft the definition will be overloaded (from the definition in the base protocol) to indicate an organization providing Registry Services for a REGISTRY-CLASS DOMAIN NAME.

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 and affiliate company) provides Registry Services for other organizations or individuals. For example: .COM, .ORG, .BIZ, .CO.JP, .B.BR.

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; and responding to queries for contact and other information concerning DNS registrations in the RCDN. 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.

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 represents the first step on the way to delegation in the DNS.

4. General Conventions

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 0x002B), followed by a country code defined in [ITU-E164], followed by a dot (".", ASCII value 0x002E), followed by a sequence of digits representing the telephone number.

4.4. Checksum

Checksum of the CSV data escrow files MUST use CRC32, that is the algorithm used in the ISO 3309 standard and in section 8.1.1.6.2 of ITU-T recommendation V.42.

4.5. IP addresses

IP addresses syntax MUST conform to the text representation of either of, Internet Protocol [RFC0791], for IPv4 addresses, or IP Version 6 Addressing Architecture [RFC4291], for IPv6 addresses.

4.6. 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.

The following is an example of the CSV file definitions 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>

4.7.  CSV elements

4.7.1.  <rdeCsv:csv> element

To support a CSV model with the Registry Data Escrow Specification [1], 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 is 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). 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 like "gzip" or "zip".

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

cksum Defines the checksum of the CSV file using CRC32, as defined in Section 4.4. This attribute is used to validate that the full CSV file exists and has not been tampered with.

The <rdeCsv:csv> elements require 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 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.test,Domain2-TEST,,registrantid,registrarX,registrarX,clientY,2009-04-03T22:00:00.0Z,registrarX,clientY,2009-12-03T09:05:00.0Z,2015-04-03T22:00:00.0Z

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

```xml
<csvDomain:deletes>
  ...
  <rdeCsv:csv name="domain">
    <rdeCsv:fields>
      <csvDomain:fName/>
    </rdeCsv:fields>
    <rdeCsv:files>
      <rdeCsv:file checksum="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.test
domain2.test
domainN.test

4.7.2. CSV common field elements

The `<rdeCsv:fields>` element defined in the `<rdeCsv:csv>` element (Section 4.7.1) 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 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 client that created the object with type="eppcom:clIDType".

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

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

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

<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".

4.8. Internationalized and Localized Elements

Some elements MAY be provided in either internationalized form ("int") or provided in 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/>
  <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>
...
</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>
```

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], 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. If the domain name is an IDN, the ASCII Compatible Encoding (ACE) MUST be used.
- 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 name of the 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>` element 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 contain 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].
o An OPTIONAL <trDate> element that contains the date and time of the most recent domain object successful transfer. This element MUST NOT be present if the domain name object has never been transferred.

o An OPTIONAL <trnData> element that contains the following child elements related to the last transfer request of the domain name object. This element MUST NOT be present if a transfer request for the domain name has never been created.

* A <trStatus> element that contains the state of the most recent transfer request.

* A <reRr> element that contains the identifier of the registrar that requested the domain name object transfer. An OPTIONAL client attribute is used to specify the client that performed the operation.

* A <reDate> element that contains the date and time that the transfer was requested.

* An <acRr> element that contains the identifier of the registrar that SHOULD act upon a PENDING transfer request. For all other status types, the value identifies the registrar that took the indicated action. An OPTIONAL client attribute is used to specify the client that performed the operation.

* An <acDate> element that contains the date and time of a required or completed response. For a PENDING request, the value identifies the date and time by which a response is required before an automated response action will be taken by the registry. For all other status types, the value identifies the date and time when the request was completed.

* An OPTIONAL <exDate> element that contains the end of the domain name object’s validity period (expiry date) if the transfer caused or causes a change in the validity period.
Example of a domain object:

...<rdeDom:domain>
  <rdeDom:name>example1.test</rdeDom:name>
  <rdeDom:roid>Dexample1-TEST</rdeDom:roid>
  <rdeDom:status s="ok"/>
  <rdeDom:registrant>jd1234</rdeDom:registrant>
  <rdeDom:contact type="admin">sh8013</rdeDom:contact>
  <rdeDom:contact type="tech">sh8013</rdeDom:contact>
  <rdeDom:ns>
    <domain:hostObj>ns1.example.com</domain:hostObj>
    <domain:hostObj>ns1.example1.test</domain:hostObj>
  </rdeDom:ns>
  <rdeDom:clID>RegistrarX</rdeDom:clID>
  <rdeDom:crRr client="jdoe">RegistrarX</rdeDom:crRr>
  <rdeDom:crDate>1999-04-03T22:00:00.0Z</rdeDom:crDate>
  <rdeDom:exDate>2015-04-03T22:00:00.0Z</rdeDom:exDate>
</rdeDom: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.test</rdeDomain:name>
    <rdeDomain:name>bar.test</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" fields, defined in section CSV common field elements (Section 4.7.2), 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> Identifier of client that sponsors the domain name object with isRequired="true".

<rdeCsv:fCrDate> Created date and time of the domain name object with isRequired="true".

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

<rdeCsv:fUpDate> Date and time of the last update to the domain name object.

<rdeCsv:fExDate> Expiration date and time for the domain name object. The attribute isRequired MUST equal "true".

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.7.2), 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 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:fCrID> Identifier of client that created the domain name object.

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

<rdeCsv:fTrDate> Date and time of the last transfer for the domain name object.
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:fCIID/>
  <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.test,Ddomain1-TEST,,,registrantid,registrarX,registrarX,clientY,2009-04-03T22:00:00.0Z,registrarX,clientY,2009-12-03T09:05:00.0Z,2015-04-03T22:00:00.0Z
domain2.test,Ddomain2-TEST,,,registrantid,registrarX,registrarX,clientY,1999-04-03T22:00:00.0Z,registrarX,clientY,2009-12-03T09:05:00.0Z,2015-04-03T22:00:00.0Z
xn--abc123.test,Dxnabc123-TEST,LANG-1,,registrantid,registrarX,registrarX,clientY,2009-04-03T22:00:00.0Z,registrarX,clientY,2009-12-03T09:05:00.0Z,2015-04-03T22:00:00.0Z
xn--abc321.test,Dxnabc321-TEST,LANG-1,registrantid,registrarX,registrarX,clientY,2009-04-03T22:00:00.0Z,registrarX,clientY,2009-12-03T09:05:00.0Z,2015-04-03T22:00:00.0Z
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 MUST be used in the "domainContacts" <rdeCsv:csv> <rdeCsv:fields> element:

<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 "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".

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

<csvContact:fId> The server-unique contact identifier with isRequired="true".
Example of a "domainContacts" <csvDomain:contents> <rdeCsv:csv> element.

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

Example of the corresponding domainContacts-YYYYMMDD.csv file. The file contains an admin, tech, and billing contact for the four domain names domain1.test, domain2.test, xn--abc123.test and xn--abc321.test.

```
domain1.test,domain1admin,admin
domain1.test,domain1tech,tech
domain1.test,domain1billing,billing
domain2.test,domain2admin,admin
domain2.test,domain2tech,tech
domain2.test,domain2billing,billing
xn--abc123.test,xnabc123admin,admin
xn--abc123.test,xnabc123tech,tech
xn--abc123.test,xnabc123billing,billing
xn--abc321.test,xnabc123admin,admin
xn--abc321.test,xnabc123tech,tech
xn--abc321.test,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" field elements MUST be used in the "domainStatuses" <rdeCsv:csv> <rdeCsv:fields> element:
<csvDomain:fStatus> The status of the domain name with type="domain:statusValueType" and isRequired="true".

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

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".

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.7.2), MAY be used in the "domainStatuses" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fStatusDescription> Domain 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.test, domain2.test, xn--abc123.test and xn--abc321.test.

domain1.test,clientUpdateProhibited,"Disallow update", en,
domain1.test,clientDeleteProhibited,"Disallow delete", en,
domain2.test,ok,,
xn--abc123.test,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 host <rdeCsv:fRoid> and isRequired="true".

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.7.2), MUST be used in the "domainNameServers" <rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fRoid> Registry Object IDentifier (ROID) used to uniquely reference a delegated host record with isRequired="true".
Example of a "domainNameServers" <csvDomain:contents> <rdeCsv:csv> element.

...<csvDomain:contents>
...<rdeCsv:csv name="domainNameServers">
<rdeCsv:fields>
<rdeCsv:file cksum="8FE6E9E1">
domainNameServers-YYYYMMDD.csv
</rdeCsv:file>
</rdeCsv:fields>
</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.test, domain2.test, xn--abc123.test and xn--abc321.test referenced via the <rdeCsv:fRoid> field element.

domain1.test,Hns1_domain1_test-TEST
domain1.test,Hns2_domain1_test-TEST
domain2.test,Hns1_domain2_test-TEST
domain2.test,Hns2_domain2_test-TEST
xn--abc123.test,Hns1_example_test-TEST
xn--abc123.test,Hns2_example_test-TEST
xn--abc321.test,Hns1_example_test-TEST
xn--abc321.test,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 host as domain 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.

...
Example of the corresponding domainNameServersAddresses-YYYYMMDD.csv file. The file contains the delegated hosts (name servers) for the four domain names domain1.test, domain2.test, xn--abc123.test and xn--abc321.test.

domain1.test,ns1.domain1.test,192.0.2.1,v4
domain1.test,ns2.domain1.test,2001:DB8::1,v6
domain2.test,ns1.domain2.test,192.0.2.2,v4
domain2.test,ns2.domain2.test,2001:DB8::2,v6
xn--abc123.test,ns1.example.example,,
xn--abc123.test,ns2.example.example,,
xn--abc321.test,ns1.example.example,,
xn--abc321.test,ns2.example.example,,

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 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.test.

domain1.test,604800,12345,3,1,49FD46E6C4B45C55D4AC
domain1.test,604800,12346,3,1,38EC35D5B3A34B44C39B

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

<csvDomain:contents>
...  
<csvDomain:fields>
<rdeCsv:fields>
<csvDomain:fName/>
<csvDomain:fMaxSigLife/>
<csvDomain:fFlags/>
<csvDomain:fProtocol/>
<csvDomain:fKeyAlg/>
<csvDomain:fPubKey/>
</rdeCsv:fields>
</csvDomain:fields>
</rdeCsv:csv>
...

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

domain1.test,604800,257,3,1,AQPJ////4Q==
domain1.test,604800,257,3,1,AQPJ////4QQQ

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 4.7.2), 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 4.7.2), MAY be used in the "domainTransfer" element:

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

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

<rdeCsv:fAcID> Identifier of 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" element:

<csvDomain:fName> Domain name of the domain 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 checksum="2E5A9ACD">
    domainTransfer-YYYYMMDD.csv
  </rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>

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

domain1.test,pending,registrarX,clientY,
2011-03-08T19:38:00.0Z,registrarY,,2011-03-13T23:59:59.0Z,
2016-04-03T22:00:00.0Z

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.test
domain2.test

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

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

Example of <host> object:

```
...<rdeHost:host>
  <rdeHost:name>ns1.example1.test</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">1080:0:0:0:8:800:200C:417A</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.test</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 4.7.2), 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" fields, defined in section CSV common field elements (Section 4.7.2), MAY be used in the "host" <rdeCsv:csv> <rdeCsv:fields> element:

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

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

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

<rdeCsv:fCrDate> Date and time that the host object was created with isRequired="true".

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

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

<rdeCsv:fUpDate> Date and time that the host object was last updated.

<rdeCsv:fTrDate> Date and time that the host was last 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.test,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.test,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.test,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.test,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.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.example.example,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 4.7.2), 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 4.7.2), 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. The attribute isRequired MUST equal "true".

<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:file cksum="0DAE0583">
hostStatuses-YYYYMMDD.csv
</rdeCsv:file>
</rdeCsv:fields>
</rdeCsv:csv>
...<csvHost:contents>
...

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

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 4.7.2), MUST be used in the "hostAddresses"
<rdeCsv:csv> <rdeCsv:fields> element:

<rdeCsv:fRoid> Host object Registry Object IDentifier (ROID) assigned to the host object.

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 hostAddressesObj-YYYYMMDD.csv file. The file contains the IP addresses for the host names ns1.domain1.test, ns2.domain1.test, ns1.domain2.test and ns2.domain2.test.

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. \texttt{<csvHost:deletes>}

The \texttt{<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 \texttt{<csvHost:deletes>} is split into separate CSV file definitions using named \texttt{<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 4.7.2), MUST be used in the "host" \texttt{<rdeCsv:csv>} element:

- \texttt{<rdeCsv:fRoid>} Repository Object IDentifier (ROID) assigned to the host object with isRequired="true".

Example of a "host" \texttt{<csvHost:deletes>} \texttt{<rdeCsv:csv>} element.

```xml
...<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> element, 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 A <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 A <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:roid>Csh8013-TEST</rdeContact:roid>
  <rdeContact:id>sh8013</rdeContact:id>
  <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.test</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:

```xml
<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 contact 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" fields, defined in section CSV common field elements (Section 4.7.2), 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> Identifier of client (registrar) that sponsors the contact object with isRequired="true".

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

<rdeCsv:fCrDate> Created date and time of the contact object with isRequired="true".

<rdeCsv:fUpRr> Identifier of the registrar, defined in Section 5.4, of the client that updated the object.
<rdeCsv:fUpDate> Date and time of the last update to the contact object.

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.7.2), MAY be used in the "contact" <rdeCsv:csv> <rdeCsv:fields> element:

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

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

<rdeCsv:fTrDate> Date and time of the last transfer for the contact object.

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.test,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.test,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.test,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.test,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.test,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.test,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.test,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.test,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.test,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".

<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 4.7.2), 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 4.8, 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 to used to indicate the localized or internationalized form as defined in section Section 4.8.

<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 index "2" for the last line. An OPTIONAL "isLoc" attribute to used to indicate the localized or internationalized form as defined in section Section 4.8.

<csvContact:fCity> Contains the contact’s city with type="contact:postalLineType" and isRequired="true". An OPTIONAL "isLoc" attribute to used to indicate the localized or internationalized form as defined in section Section 4.8.
<csvContact:fcCc> Contains the contact’s country code with type="contact:ccType" and isRequired="true". An OPTIONAL "isLoc" attribute to used to indicate the localized or internationalized form as defined in section Section 4.8.

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 to used to indicate the localized or internationalized form as defined in section Section 4.8.

<csvContact:fSp> Contains the contact’s state or province with type="contact:optPostalLineType". An OPTIONAL "isLoc" attribute to used to indicate the localized or internationalized form as defined in section Section 4.8.

<csvContact:fPc> Contains the contact’s postal code with type="contact:pcType". An OPTIONAL "isLoc" attribute to used to indicate the localized or internationalized form as defined in section Section 4.8.

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". 
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

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 4.7.2), 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 4.7.2), MAY be used in the "contactTransfer" element:

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

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

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

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

Example of a "contactTransfer" element:

```xml
...<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 for 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.

```xml
...<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, for operational purposes MAY be the registry operator. 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 ID assigned by ICANN.

- An OPTIONAL <status> element that contains the operational status of the registrar. Possible values are: ok, readonly and terminated.
Zero 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 RDE registrar-object modification. This element MUST NOT be present if the rdeRegistrar object has never been modified.

Example of <registrar> object:

```xml
<rdeRegistrar:registrar>
  <rdeRegistrar:id>RegistrarX</rdeRegistrar:id>
  <rdeRegistrar:name>Registrar X</rdeRegistrar:name>
  <rdeRegistrar:gurid>123</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:addr>
  </rdeRegistrar:postalInfo>
  <rdeRegistrar:voice x="1234">+1.7035555555</rdeRegistrar:voice>
  <rdeRegistrar:fax>+1.7035555556</rdeRegistrar:fax>
  <rdeRegistrar:email>jdoe@example.test</rdeRegistrar:email>
  <rdeRegistrar:url>http://www.example.test</rdeRegistrar:url>
  <rdeRegistrar:whoisInfo>
    <rdeRegistrar:name>whois.example.test</rdeRegistrar:name>
    <rdeRegistrar:url>http://whois.example.test</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:

```xml
does not exist
```

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> Contains the server-unique registrar identifier with type="eppcom:clIDType" 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".

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

The following "rdeCsv" fields, defined in section CSV common field elements (Section 4.7.2), 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.

<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 4.8.
<csvContact:fCity> Registrar city with an OPTIONAL "isLoc" attribute that is used to indicate the localized or internationalized form, as defined in Section 4.8.

<csvContact:fCc> Registrar country code with an OPTIONAL "isLoc" attribute that is used to indicate the localized or internationalized form, as defined in Section 4.8.

<csvContact:fEmail> Registrar email address.

<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 4.8.

<csvContact:fPc> Registrar postal code with an OPTIONAL "isLoc" attribute that is used to indicate the localized or internationalized form, as defined in Section 4.8.

<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/>
    <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 three registrar records.

registrarX,"Example Inc.",1234,ok,"123 Example Dr.","Suite 100",Dulles,VA,20166-6503,US,+1.7035555555,1234,
+1.7035555556,,jdoe@example.test,http://www.example.test,
http://whois.example.test,2005-04-23T11:49:00.0Z,
2009-02-17T17:51:00.0Z
registrarY,"Example2 Inc.",1234,ok,"123 Example Dr.","Suite 100",Dulles,VA,20166-6503,US,+1.7035555555,1234,
+1.7035555556,,jdoe@example.test,http://www.example.test,
http://whois.example.test,2005-04-23T11:49:00.0Z,
2009-02-17T17:51:00.0Z
registrarZ,"Example2 Inc.",1234,ok,"123 Example Dr.","Suite 100",Dulles,VA,20166-6503,US,+1.7035555555,1234,
+1.7035555556,,jdoe@example.test,http://www.example.test,
http://whois.example.test,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> Contains the server-unique registrar identifier with type="eppcom:clIDType" and isRequired="true".
Example of a "registrar" `<csvRegistrar:deletes> <rdeCsv:csv>` element.

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

Example of <idnTableRef> object:

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

5.5.2. CSV Model

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

5.5.2.1. <csvIDN:contents>

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

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

The following "rdeCsv" fields, defined in Section 4.7.2, 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 the language defined by the <rdeCsv:fLang> field element. The attribute isRequired MUST equal "true".

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

...<csvIDN:contents>
...<rdeCsv:csv name="idnLanguage">
  <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

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

A NNDN is a lightweight domain-like object that is not linked directly to a Registrar (a mirroring NNDN is linked to a Registrar via the original name).

A domain name can only exist as a domain name object or 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 a 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 name of the NNDN. If the NNDN is an IDN, the ASCII Compatible Encoding (ACE) MUST be used.

- 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 a NNDN is considered undesirable for registration (i.e., unavailable for allocation to anyone), then the NNDN will be tagged as "blocked".

* If a NNDN is considered a potential registration of a domain 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 a NNDN is considered a mirrored IDN variant of a domain object, then the NNDN will be tagged as "mirrored". A `mirroringNS` attribute is used to specify if the mirrored IDN variant use the NS mirror mechanism. The default value of `mirroringNS` is true. If another mechanism such as DNAME is used, the value of `mirroringNS` attribute MUST be false.

A `<crDate>` element that contains the date and time of the NNDN object creation.

Example of `<rdeNNDN:NNDN>` object:

```
...<rdeNNDN:NNDN>
  <rdeNNDN:aName>xn--exampl-gva.test</rdeNNDN:aName>
  <rdeNNDN:idnTableId>pt-BR</rdeNNDN:idnTableId>
  <rdeNNDN:originalName>example1.test</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 ACE of a NNDN that was deleted, i.e., the `<aName>`.
Example of `<rdeNNDN::delete>` object:

```
<...>
<rde:deletes>
    <rdeNNDN:delete>
        <rdeNNDN:aName>xn--pingino-q2a.test</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>` ASCII Compatible Encoding (ACE) field of the NNDN with type="eppcom:labelType" and isRequired="true".
- `<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 4.7.2), MUST be used in the "domain" <rdeCsv:csv>
<rdeCsv:fields> element:

<rdeCsv:fCrDate>  Created date and time of the NNDN object with
isRequired="true".

The following "rdeCsv" fields, defined in section CSV common field
elements (Section 4.7.2), MAY be used in the "domain" <rdeCsv:csv>
<rdeCsv:fields> element:

<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 a "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--abc456.test,LANG-1,xn--abc123.test,
blocked,,2005-04-23T11:49:00.0Z
xn--abc789.test,LANG-1,xn--abc123.test,
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:
Example of a "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--abc456.test

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. Only 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:

```xml
<rdeEppParams:eppParams>
  <rdeEppParams:version>1.0</rdeEppParams:version>
  <rdeEppParams:lang>en</rdeEppParams:lang>
  <rdeEppParams:svcExtension>
    <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:prov/>
      </epp:purpose>
      <epp:recipient>
        <epp:ours/>
        <epp:public/>
      </epp:recipient>
      <epp:retention>
        <epp:stated/>
      </epp:retention>
    </epp:statement>
  </rdeEppParams:dcp>
</rdeEppParams:eppParams>
```
Internet-Draft
5.8.

DNRD Objects Mapping

Jun 2019

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 4.7.1, is used to specify which OPTIONAL fields are REQUIRED
based on the business model of the registry.
5.8.1.

<rdePolicy:policy> object

The OPTIONAL <policy> contains the following attributes:
o

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

o

<scope> that defines the XPath of the element referenced by
<element>.

Example of <rdePolicy:policy> object:
...
<rdePolicy:policy scope="//rde:deposit/rde:contents/rdeDomain:domain"
element="rdeDom: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. The Header Object is only defined as XML but one header
object MUST always be present per escrow deposit regardless of using
XML Model or CSV Model. The Header Object is defined using the
<rdeHeader:header> element.
5.9.1.

<rdeHeader:header> object

The <rdeHeader:header> contains the following elements:
o

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

A <rdeHeader:tld> element that defines TLD or the REGISTRYCLASS DOMAIN NAME (RCDN) being escrowed in the case of a

Lozano, et al.

Expires December 20, 2019

[Page 79]


Registry data escrow deposit. For IDNs the A-Label is used [RFC5891].

* 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.

  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 REGISTRY-CLASS DOMAIN NAME (RCDN) of the objects included in the `<count>` element. For IDNs the A-Label is used [RFC5891]. 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.
Example of <rdeHeader:header> object referencing only the XML Model objects:

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

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

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 a NNDN object, but not both.

If using domain objects to represent IDN variants, the normal behavior during restoration of a 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.

A 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 to define 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 used 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 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

The Data Escrow Agent MUST perform a extended verification process using the contents of the data escrow deposits to a point in time (watermark), last full plus all differentials or last full plus last incremental escrow deposits. The following are the minimum suggested tests:

   o Validate the escrow deposits using the definition agreed with the registry.
     * In the case of the XML model, the contents of the escrow deposits MUST be validated using the XML schemas of the profile.
   
   o Count the objects and validate that the number of objects is equal to the number objects reported in the <header> element of the escrow deposit of that point in time (watermark).
o All contact objects linked to domain names MUST be present.

o All registrars objects linked to other objects MUST be present.

o A domain name exists as a domain name or NNDN, but not both.

o The elements listed as required in the <policy> element MUST be present.

o All idnTableRef definitions linked from other objects MUST be present.

o If an EPP Parameters Object was escrowed in the past, one and only one EPP Parameters Object MUST be present.

o The watermark is not in the future.

9. Formal Syntax

Schemas are presented here.

9.1. RDE CSV Schema

Copyright (c) 2012 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

o Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

o Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

o Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
BEGIN
<?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"
           schemaLocation="eppcom-1.0.xsd"/>
  <import namespace="urn:ietf:params:xml:ns:rde-1.0"
           schemaLocation="rde-1.0.xsd"/>

  <annotation>
    <documentation>
      Registry Data Escrow Comma-Separated Values (CSV)
    </documentation>
  </annotation>

  <!-- csv content element -->
  <element name="csv" type="rdeCsv:csvType"/>

  <!-- Definition of CSV file -->
  <complexType name="csvType">
    <sequence>
      <element name="fields" type="rdeCsv:fieldsType"/>
      <element name="files" type="rdeCsv:filesType"/>
    </sequence>
    <attribute name="name" type="token" use="required"/>
    <attribute name="sep" type="rdeCsv:sepType" default=",”/>
  </complexType>

  <!-- field separator must be a single character -->
  <simpleType name="sepType">
    <restriction base="string">
      <minLength value="1"/>
    </restriction>
  </simpleType>
</schema>
<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/>
</complexContent>
</complexType>
​
<!-- Identifier of 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:fClIDRequiredType"
        substitutionGroup="rdeCsv:field"/>

<!-- Identifier of 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 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:fClIDType"
        substitutionGroup="rdeCsv:field"/>

<!-- Identifier of 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 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>
<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>
<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>
</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/>
    </extension>
  </complexContent>
</complexType>
<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"/>
    </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>

<!-- -->
9.2. RDE Domain Object

Copyright (c) 2011 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
<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="rde:rrType" minOccurs="0"/>
<element name="crDate"
    type="dateTime" minOccurs="0"/>
<element name="exDate"
    type="dateTime" minOccurs="0"/>
<element name="upRr"
    type="rde: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>
</complexType>
<complexType name="transferDataType">
    <sequence>
        <element name="trStatus"
            type="eppcom:trStatusType"/>
        <element name="reRr"
            type="rde:rrType"/>
        <element name="reDate"
            type="dateTime"/>
        <element name="acRr"
            type="rde:rrType"/>
        <element name="acDate"
            type="dateTime"/>
        <element name="exDate"
            type="dateTime" minOccurs="0"/>
    </sequence>
</complexType>

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

9.3. CSV Domain Object

Copyright (c) 2012 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
<?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"
schemaLocation="eppcom-1.0.xsd"/>
<import namespace="urn:ietf:params:xml:ns:domain-1.0"
schemaLocation="domain-1.0.xsd"/>
<import namespace="urn:ietf:params:xml:ns:secDNS-1.1"
schemaLocation="secDNS-1.1.xsd"/>
<import namespace="urn:ietf:params:xml:ns:rgp-1.0"
schemaLocation="rgp-1.0.xsd"/>
<import namespace="urn:ietf:params:xml:ns:rde-1.0"
schemaLocation="rde-1.0.xsd"/>
<import namespace="urn:ietf:params:xml:ns:rdeCsv-1.0"
schemaLocation="rdeCsv-1.0.xsd"/>

<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/>
    </extension>
  </complexContent>
</complexType>
<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"/>

<!-- Variant group / tag field -->
<element name="fVariantGroup"
    type="rdeCsv:fTokenType"
    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>
END

9.4. RDE Host Object

Copyright (c) 2011 IETF Trust and the persons identified as authors of the code. All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

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

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

BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns: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="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified">
<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="rde:rrType"/>
                <element name="crDate" type="dateTime"/>
                <element name="upRr" type="rde: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>
9.5. CSV Host Object

Copyright (c) 2012 IETF Trust and the persons identified as authors
of the code. All rights reserved.

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

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

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
<import namespace="urn:ietf:params:xml:ns:eppcom-1.0"
schemaLocation="eppcom-1.0.xsd"/>
<import namespace="urn:ietf:params:xml:ns:host-1.0"
schemaLocation="host-1.0.xsd"/>
<import namespace="urn:ietf:params:xml:ns:rde-1.0"
schemaLocation="rde-1.0.xsd"/>
<import namespace="urn:ietf:params:xml:ns:rdeCsv-1.0"
schemaLocation="rdeCsv-1.0.xsd"/>

<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>
</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\:statusValue"/>
</extension>
</complexContent>
</complexType>
<extension>
</complexContent>
</complexType>

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

9.6. RDE Contact Object

Copyright (c) 2011 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote products derived from this software without specific prior written permission.

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

&lt;?xml version="1.0" encoding="UTF-8"?&gt;
&lt;schema targetNamespace="urn:ietf:params:xml:ns:rdeContact-1.0"
 xmlns:rdeContact="urn:ietf:params:xml:ns:rdeContact-1.0"
<element name="abstractContact">
  <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"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
type="rde:rrType"/>
<element name="crDate"
    type="dateTime"/>
<element name="upRr"
    type="rde: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="rde:rrType"/>
    <element name="reDate" type="dateTime"/>
    <element name="acRr" type="rde: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>
END
9.7. CSV Contact Object

Copyright (c) 2012 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

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

BEGIN
<?xml version="1.0" encoding="UTF-8"?>

<schema targetNamespace="urn:ietf:params:xml:ns: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.

Contact Comma-Separated Values (CSV) Object

Child elements of the `<rde:contents>` object

```xml
<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

```xml
<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">
  <extension base="rdeCsv:fieldRequiredType">
    <sequence/>
  </extension>
</complexType>
<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"
<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"/>
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>
END

9.8. RDE Registrar Object

Copyright (c) 2011 IETF Trust and the persons identified as authors
of the code. All rights reserved.

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

- Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote products derived from this software without specific prior written permission.

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

BEGIN
<?xml version="1.0" encoding="UTF-8"?><!-- 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>

9.9. CSV Registrar Object

Copyright (c) 2012 IETF Trust and the persons identified as authors of the code. All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote products derived from this software without specific prior written permission.

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

BEGIN

<?xml version="1.0" encoding="UTF-8"?>

<schema targetNamespace="urn:ietf:params:xml:ns: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"
schemaLocation="eppcom-1.0.xsd"/>
<import namespace="urn:ietf:params:xml:ns:domain-1.0"

<annotation>
  <documentation>
    Registrar Comma-Separated Values (CSV) Object
  </documentation>
</annotation>

<!-- Child elements of the <rde:contents> object -->
<element name="contents" type="csvRegistrar:contentType"
         substitutionGroup="rde:content"/>

<complexType name="contentType">
  <complexContent>
    <extension base="rde:contentType">
      <sequence>
        <element ref="rdeCsv:csv" maxOccurs="unbounded"/>
        <sequence>
          </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>
      </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:fieldRequiredType">
            <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>
END

9.10.  RDE IDN Table Reference Objects

Copyright (c) 2011 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

  o Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

  o Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

  o Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
<?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>
</schema>
9.11. CSV IDN Language Object

Copyright (c) 2012 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
<?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"
      schemaLocation="rde-1.0.xsd"/>
  <import namespace="urn:ietf:params:xml:ns:rdeCsv-1.0"
      schemaLocation="rdeCsv-1.0.xsd"/>
  <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>
  </complexContent>
</schema>
<extension base="rde:deleteType">
  <sequence>
    <element ref="rdeCsv:csv" maxOccurs="unbounded"/>
  </sequence>
</extension>
</complexContent>
</complexType>

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

9.12. EPP Parameters Object

Copyright (c) 2011 IETF Trust and the persons identified as authors
of the code. All rights reserved.

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

 o Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

 o Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.

 o Neither the name of Internet Society, IETF or IETF Trust, nor the
   names of specific contributors, may be used to endorse or promote
   products derived from this software without specific prior written
   permission.

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

Copyright (c) 2011 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

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

BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns: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"/>

END
<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"/>
</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"/>
</extension>
</simpleContent>
</complexType>
<complexType name="deleteType">
  <complexContent>
    <extension base="rde:deleteType">
      <sequence>
        <element name="aName"
          type="eppcom:labelType" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
</schema>
END

9.14. CSV NNDN Object

Copyright (c) 2013 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

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

BEGIN
<?xml version="1.0" encoding="UTF-8"?>

<schema targetNamespace="urn:ietf:params:xml:ns: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"
schemaLocation="rde-1.0.xsd"/>
<import namespace="urn:ietf:params:xml:ns:rdeCsv-1.0"
schemaLocation="rdeCsv-1.0.xsd"/>
<import namespace="urn:ietf:params:xml:ns:rdeNNDN-1.0"
schemaLocation="rde-nndn-1.0.xsd"/>

<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

```xml
<element name="deletes" type="csvNNDN:deleteType"
  substitutionGroup="rde:delete"/>
```

```xml
<complexType name="deleteType">
  <complexContent>
    <extension base="rde:deleteType">
      <sequence>
        <element ref="rdeCsv:csv" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

<!-- ASCII Compatible Encoding (ACE) name field -->

```xml
<element name="fAName" type="rdeCsv:fNameRequiredType"
  substitutionGroup="rdeCsv:field"/>
```

<!-- domain name used to generate the IDN variant field -->

```xml
<element name="fOriginalName" type="rdeCsv:fNameType"
  substitutionGroup="rdeCsv:field"/>
```

<!-- Variant group / tag field -->

```xml
<element name="fVariantGroup"
  type="rdeCsv:fTokenType"
  substitutionGroup="rdeCsv:field"/>
```

<!-- RGP status field -->

```xml
<element name="fNameState"
  type="csvNNDN:fNameStateType"
  substitutionGroup="rdeCsv:field"/>
```

```xml
<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? -->

```xml
<element name="fMirroringNS"
  type="rdeCsv:fBooleanType"
  substitutionGroup="rdeCsv:field"/>
```
9.15. Policy Object

Copyright (c) 2011 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns: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>
END

9.16. Header Object

Copyright (c) 2011 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote

products derived from this software without specific prior written permission.

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

BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns: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"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</schema>
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]. Fourteen URI assignments have been registered by the IANA.

Registration request for the RDE CSV namespace:

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

Registrant Contact: See the "Author’s Address" section of this document.

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: See the "Author's Address" section of this document.

See the "Formal Syntax" section of this document.

Registration request for the RDE domain namespace:

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

Registrant Contact: See the "Author’s Address" section of this document.

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: See the "Author’s Address" section of this document.

See the "Formal Syntax" section of this document.

Registration request for the CSV domain namespace:

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

Registrant Contact: See the "Author’s Address" section of this document.

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: See the "Author’s Address" section of this document.

See the "Formal Syntax" section of this document.

Registration request for the RDE host namespace:

URI: urn:ietf:params:xml:ns:rdeHost-1.0
Registrant Contact: See the "Author’s Address" section of this document.

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: See the "Author’s Address" section of this document.

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

Registration request for the CSV host namespace:

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

Registrant Contact: See the "Author’s Address" section of this document.

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: See the "Author’s Address" section of this document.

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

Registration request for the RDE contact namespace:

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

Registrant Contact: See the "Author’s Address" section of this document.

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: See the "Author’s Address" section of this document.
See the "Formal Syntax" section of this document.

Registration request for the CSV contact namespace:

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

Registrant Contact: See the "Author’s Address" section of this document.

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: See the "Author’s Address" section of this document.

See the "Formal Syntax" section of this document.

Registration request for the RDE registrar namespace:

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

Registrant Contact: See the "Author’s Address" section of this document.

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: See the "Author’s Address" section of this document.

See the "Formal Syntax" section of this document.

Registration request for the CSV registrar namespace:

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

Registrant Contact: See the "Author’s Address" section of this document.

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: See the "Author’s Address" section of this document.
See the "Formal Syntax" section of this document.
Registration request for the RDE IDN namespace:
URI: urn:ietf:params:xml:ns:rdeIDN-1.0
Registrant Contact: See the "Author’s Address" section of this document.
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: See the "Author’s Address" section of this document.
See the "Formal Syntax" section of this document.
Registration request for the CSV IDN namespace:
URI: urn:ietf:params:xml:ns:csvIDN-1.0
Registrant Contact: See the "Author’s Address" section of this document.
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: See the "Author’s Address" section of this document.
See the "Formal Syntax" section of this document.
Registration request for the RDE NNDN namespace:
URI: urn:ietf:params:xml:ns:rdeNNDN-1.0
Registrant Contact: See the "Author’s Address" section of this document.

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: See the "Author’s Address" section of this document.

See the "Formal Syntax" section of this document.

Registration request for the CSV NNDN namespace:

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

Registrant Contact: See the "Author’s Address" section of this document.

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: See the "Author’s Address" section of this document.

See the "Formal Syntax" section of this document.

Registration request for the RDE EPP parameters namespace:

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

Registrant Contact: See the "Author’s Address" section of this document.

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: See the "Author’s Address" section of this document.
See the "Formal Syntax" section 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 - 08 are known to be implemented.
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 Registry transmitting the data to the Escrow Agent SHOULD take all the necessary precautions like encrypting the data itself and/or the transport channel to avoid inadvertent disclosure of private data.

It is also of the utmost importance the authentication of the parties passing data escrow deposit files. The Escrow Agent SHOULD properly authenticate the identity of the Registry before accepting data escrow deposits. In a similar manner, the Registry SHOULD authenticate the identity of the Escrow Agent before submitting any data.

Additionally, the Registry and the Escrow Agent SHOULD use integrity checking mechanisms to ensure the data transmitted is what the source intended. Validation of the contents by the Escrow Agent is RECOMMENDED to ensure not only the file was transmitted correctly from the Registry, but also the contents are also "meaningful".

14. Acknowledgments

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

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

15. Change History

[[RFC Editor: Please remove this section.]]
15.1. 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.

15.2. Changes from version 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.

15.3. Changes from version 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.

15.4. Changes from version 02 to 03
1. Remove authinfo from the XML Schema.
2. Resend attribute is now an element
3. Scope attribute added to policy object.

15.5. Changes from version 03 to 04
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.


15. Replaced use of `<rdeCsv:fLang>` to new `<rdeCsv:fIdnTableId>` field in the "domain", "idnLanguage", and "NNDN" CSV files.


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.


20. Added definition of `<csvNNDN:fMirroringNS>` field element.

15.6. Changes from version 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.

15.7. Changes from version 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 subsections 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.

15.8. Changes from version 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

15.9. Changes from version 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.
15.10. Changes from version 08 to 09
   1. Implementation Status section was added

15.11. Changes from version 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.

15.12. Changes from version 10 to 00
   1. Internet Draft (I-D) adopted by the REGEXT WG.

16. Example of a full deposit using the XML model

Example of a full deposit using the XML model:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rde:deposit type="FULL" id="20101017001" prevId="20101010001"
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:rdeDom="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>2010-10-17T00:00:00Z</rde:watermark>
  <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:deposit>
```
<!-- Contents -->
<rde:contents>
<!-- Header -->
<rde:header>
<rde:tld>test</rde:tld>
<rde:count uri="urn:ietf:params:xml:ns:rdeHost-1.0">1</rde:count>
<rde:count uri="urn:ietf:params:xml:ns:rdeContact-1.0">1</rde:count>
<rde:count uri="urn:ietf:params:xml:ns:rdeRegistrar-1.0">1</rde:count>
<rde:count uri="urn:ietf:params:xml:ns:rdeIDN-1.0">1</rde:count>
<rde:count uri="urn:ietf:params:xml:ns:rdeNNDN-1.0">1</rde:count>
</rde:header>

<!-- Domain: example1.test -->
<rde:domain>
<rde:name>example1.test</rde:name>
<rde:roid>example1-TEST</rde:roid>
<rde:status s="ok"/>  
<rde:registrant>jd1234</rde:registrant>
<rde:contact type="admin">sh8013</rde:contact>
<rde:contact type="tech">sh8013</rde:contact>
</rde:domain>
</rde:contents>
<!-- Domain: example2.test -->
<rdeDom:domain>
  <rdeDom:name>example2.test</rdeDom:name>
  <rdeDom:roid>Dexample2-TEST</rdeDom:roid>
  <rdeDom:status s="ok"/>
  <rdeDom:status s="clientUpdateProhibited"/>
  <rdeDom:registrant>jd1234</rdeDom:registrant>
  <rdeDom:contact type="admin">sh8013</rdeDom:contact>
  <rdeDom:contact type="tech">sh8013</rdeDom:contact>
</rdeDom:domain>

<!-- Host: ns1.example.test -->
<rdeHost:host>
  <rdeHost:name>ns1.example1.test</rdeHost:name>
  <rdeHost:roid>Hns1_example_test-TEST</rdeHost:roid>
  <rdeHost:addr ip="v4">192.0.2.2</rdeHost:addr>
  <rdeHost:addr ip="v4">192.0.2.29</rdeHost:addr>
  <rdeHost:addr ip="v6">1080:0:0:0:8:800:200C:417A</rdeHost:addr>
</rdeHost:host>

<!-- Contact: sh8013 -->
<rdeContact:contact>
  <rdeContact:id>sh8013</rdeContact:id>
  <rdeContact:roid>Csh8013-TEST</rdeContact:roid>
  <rdeContact:postalInfo type="int">
    <contact:name>John Doe</contact:name>
    <contact:org>Example Inc.</contact:org>
  </rdeContact:postalInfo>
</rdeContact:contact>
<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>

<contact:postalInfo>
  <contact:voice x="1234">+1.7035555555</contact:voice>
  <contact:fax>+1.7035555556</contact:fax>
  <contact:email>jdoe@example.test</contact:email>
  <contact:clID>RegistrarX</contact:clID>
  <contact:crRr client="jdoe">RegistrarX</contact:crRr>
  <contact:crDate>2009-09-13T08:01:00.0Z</contact:crDate>
  <contact:upRr client="jdoe">RegistrarX</contact:upRr>
  <contact:upDate>2009-11-26T09:10:00.0Z</contact:upDate>
  <contact:trDate>2009-12-03T09:05:00.0Z</contact:trDate>
  <contact:disclose flag="0">
    <contact:voice/>
    <contact:email/>
  </contact:disclose>
</contact:contact>

<!-- Registrar: RegistrarX -->
<rderegistrar:registrar>
  <rderegistrar:id>RegistrarX</rderegistrar:id>
  <rderegistrar:name>Registrar X</rderegistrar:name>
  <rderegistrar:gurid>123</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:registrar>
</rdeRegistrar:addr>
</rdeRegistrar:postalInfo>
<rdeRegistrar:voice x="1234">+1.7035555555
</rdeRegistrar:voice>
<rdeRegistrar:fax>+1.7035555556
</rdeRegistrar:fax>
<rdeRegistrar:email>jdoe@example.test
</rdeRegistrar:email>
<rdeRegistrar:url>http://www.example.test
</rdeRegistrar:url>
<rdeRegistrar:whoisInfo>
<rdeRegistrar:name>whois.example.test
</rdeRegistrar:name>
<rdeRegistrar:url>http://whois.example.test
</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.test -->
<rdeNNDN:NNDN>
<rdeNNDN:aName>xn--exampl-gva.test</rdeNNDN:aName>
<rdeNNDN:idnTableId>pt-BR</rdeNNDN:idnTableId>
<rdeNNDN:originalName>example1.test</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>
</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>
<rdePolicy:policy
  scope="/rde:deposit/rde:contents/rdeDomain:domain"
  element="rdeDom:registrant" />
</rde:contents>
</rde:deposit>

17. Example of differential deposit using the XML model

Example of a differential deposit using the XML model:

<?xml version="1.0" encoding="UTF-8"?>
<rdedeposit type="DIFF" id="20101017002" prevId="20101017001"
  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:rdeDom="urn:ietf:params:xml:ns:rdeDomain-1.0"
  xmlns:rdeHost="urn:ietf:params:xml:ns:rdeHost-1.0"
18. Example of a full deposit using the CSV model

Example of a full deposit using the CSV model:

```xml
<?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="20101017001"
  prevId="20101010001">
  <rde:watermark>2010-10-18T00:00:00Z</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: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: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>
</csvDomain:contents>
<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">
    </rdeCsv:file>
  </rdeCsv:files>
</rdeCsv:csv>
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: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:csv>
  <rdeCsv:files>
    <rdeCsv:file
      cksum="120938E3">
      host-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
  <rdeCsv:csv name="hostStatuses" sep="",">
    <rdeCsv:fields>
      <rdeCsv:fRoid parent="true"/>
      <csvHost:fStatus/>
      <rdeCsv:fStatusDescription/>
      <rdeCsv:fLang/>
    </rdeCsv:fields>
  </rdeCsv:csv>
  <rdeCsv:files>
    <rdeCsv:file
      cksum="0BA504FC">
      hostStatuses-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
  <rdeCsv:csv name="hostAddresses" sep="",">
    <rdeCsv:fields>
      <rdeCsv:fRoid parent="true"/>
      <csvHost:fAddr isRequired="true"/>
      <csvHost:fAddrVersion isRequired="true"/>
    </rdeCsv:fields>
  </rdeCsv:csv>
</csvHost:contents>
</rdeCsv:fields>
</rdeCsv:files>
</rdeCsv:csv>
</csvHost:contents>
<csvContact:contents>
<rdeCsv:csv name="contact" sep="",">
<rdeCsv:files>
<rdeCsv:file
cksum="D7F106A5">
contact-YYYYMMDD.csv
</rdeCsv:file>
</rdeCsv:files>
</rdeCsv:csv>
<rdeCsv:csv name="contactStatuses" sep="",">
<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/>
  </rdeCsv:fields>
</rdeCsv:csv>
<csvContact:fDiscloseAddrInt/>
<csvContact:fDiscloseVoice/>
<csvContact:fDiscloseFax/>
<csvContact:fDiscloseEmail/>
</rdeCsv:fields>
<rdeCsv:files>
  <rdeCsv:file cksum="89043A90">
    contactDisclose-YYYYMMDD.csv
  </rdeCsv:file>
</rdeCsv:files>
</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/>
    <rdeCsv:fUrl/>
    <csvRegistrar:fWhoisUrl/>
    <rdeCsv:fCrDate/>
    <rdeCsv:fUpDate/>
  </rdeCsv:fields>
  <rdeCsv:files>
    <rdeCsv:file cksum="306178BB">
      registrar-YYYYMMDD.csv
    </rdeCsv:file>
  </rdeCsv:files>
</csvRegistrar:contents>
<csvIDN:contents>
<rdeCsv:csv name="idnLanguage" sep="",">
  <rdeCsv:fields>
  </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:svcExtension>
    <epp:extURI>urn:ietf:params:xml:ns:rgp-1.0</epp:extURI>
  </rdeEppParams:svcExtension>
</rdeEppParams:eppParams>
19. Example of differential deposit using the CSV model

Example of a differential deposit using the CSV model:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdede: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="20101017001" prevId="20101010001">
  <rdede:watermark>2010-10-18T00:00:00Z</rdede:watermark>
  <rdede:rdeMenu>
    <rdede:version>1.0</rdede:version>
    <rdede:objURI>urn:ietf:params:xml:ns:csvDomain-1.0</rdede:objURI>
    <rdede:objURI>urn:ietf:params:xml:ns:csvHost-1.0</rdede:objURI>
    <rdede:objURI>urn:ietf:params:xml:ns:csvContact-1.0</rdede:objURI>
    <rdede:objURI>urn:ietf:params:xml:ns:csvRegistrar-1.0</rdede:objURI>
    <rdede:objURI>urn:ietf:params:xml:ns:csvIDN-1.0</rdede:objURI>
  </rdede:rdeMenu>
</rdede:deposit>
```
<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">
        </rdeCsv:file>
      </rdeCsv:files>
    </rdeCsv:csv>
  </csvRegistrar:deletes>
</rde:deletes>
<rdeCsv:csv name="idnLanguage">
  <rdeCsv:fields>
    <rdeCsv:fIdnTableId/>
  </rdeCsv:fields>
  <rdeCsv:file cksum="757B573A">
    idnLanguage-delete-YYYYMMDD.csv
  </rdeCsv:file>
</rdeCsv:csv>
</csvIDN:deletes>
</rdeCsv:csv>
</csvIDN:deletes>
</rdeCsv:csv>
</csvNNDN:deletes>
</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:header>
<rdeCsv:csv name="domain" sep="",">
    <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="",">
    <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>
</rdeCsv:files>
</rdeCsv:csv>
<csvDomain:fName parent="true"/>
<csvDomain:fStatus/>
<rdeCsv:fStatusDescription/>
<rdeCsv:fLang/>
<csvDomain:fRgpStatus/>
</rdeCsv:fields>
<rdeCsv:file
  cksum="EB8C548E">
  domainStatuses-YYYYMMDD.csv
</rdeCsv:file>
</rdeCsv:files>

<rdeCsv:csv name="domainNameServers" sep="",">
<rdeCsv:fields>
  <csvDomain:fName parent="true"/>
  <csvHost:fName parent="true"/>
</rdeCsv:fields>
<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: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: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:fFlags/>
        <csvDomain:fProtocol/>
        <csvDomain:fKeyAlg/>
        <csvDomain:fPubKey/>
    </rdeCsv:fields>
</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: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/>
</rdeCsv:fields>
</rdeCsv:csv>
</csvContact:contents>
<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:files>
<rdeCsv:file cksum="02CC2504">
    contactPostal-YYYYMMDD.csv
</rdeCsv:file>
</rdeCsv:files>
<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:csv>
<rdeCsv:files>
    <rdeCsv:file cksum="D0929632">
        contactTransfer-YYYYMMDD.csv
    </rdeCsv:file>
</rdeCsv:files>
<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:csv>
<rdeCsv:files>
    <rdeCsv:file cksum="89043A90">
        contactDisclose-YYYYMMDD.csv
    </rdeCsv:file>
</rdeCsv:files>
</csvContact:contents>
<csvRegistrar:contents>
    <rdeCsv:csv name="registrar" sep="",">
    </csvCsv:csv>
</csvRegistrar:contents>
<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/>
  <rdeCsv:fUrl/>
  <csvRegistrar:fWhoisUrl/>
  <rdeCsv:fCrDate/>
  <rdeCsv:fUpDate/>
</rdeCsv:fields>

</rdeCsv:files>
  <rdeCsv:file cksum="306178BB">
    registrar-YYYYMMDD.csv
  </rdeCsv:file>
</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:fields>
  </rdeCsv:csv>
</csvNNDN:contents>
20. References

20.1. Normative References


20.2. Informative References


20.3. URIs


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
Registry Fee Extension for the Extensible Provisioning Protocol (EPP)

draft-ietf-regext-epp-fees-16

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on November 2, 2019.

Copyright Notice

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

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

Table of Contents

1. Introduction .................................................. 3
   1.1. Conventions Used in This Document .................... 3
2. Migrating to Newer Versions of This Extension ............. 4
3. Extension Elements ........................................... 4
   3.1. Client Commands ........................................ 4
   3.2. Currency Codes .......................................... 5
   3.3. Validity Periods ....................................... 5
   3.4. Fees and Credits ....................................... 6
      3.4.1. Refunds ........................................ 7
      3.4.2. Grace Periods ................................... 7
      3.4.3. Correlation between Refundability and Grace Periods . 7
      3.4.4. Applicability ................................ 7
   3.5. Account Balance ........................................ 8
   3.6. Credit Limit .......................................... 8
   3.7. Classification of Objects ............................. 8
   3.8. Phase and Subphase Attributes ....................... 9
   3.9. Reason ............................................... 10
4. Server Handling of Fee Information .......................... 11
5. EPP Command Mapping ......................................... 12
   5.1. EPP Query Commands .................................. 12
      5.1.1. EPP <check> Command ............................... 12
      5.1.2. EPP Transfer Query Command ....................... 16
   5.2. EPP Transform Commands ................................ 18
      5.2.1. EPP <create> Command ............................ 18
      5.2.2. EPP <delete> Command ............................ 20
      5.2.3. EPP <renew> Command ............................ 21
      5.2.4. EPP <transfer> Command .......................... 23
      5.2.5. EPP <update> Command ............................ 25
6. Formal Syntax ................................................. 27
   6.1. Fee Extension Schema ................................ 27
7. Security Considerations ..................................... 32
8. IANA Considerations ......................................... 32
   8.1. XML Namespace ......................................... 32
   8.2. EPP Extension Registry ................................ 33
9. Implementation Status ....................................... 33
   9.1. RegistryEngine EPP Service ........................... 34
10. Acknowledgements .......................................... 34
11. Change History ............................................ 35
   11.1. Change from 15 to 16 ................................ 35
   11.2. Change from 14 to 15 ................................ 35
   11.3. Change from 13 to 14 ................................ 35
   11.4. Change from 12 to 13 ................................ 35
1. Introduction

Historically, domain name registries have applied a simple fee structure for billable transactions, namely a basic unit price applied to domain <create>, <renew>, <transfer> and RGP [RFC3915] restore commands. Given the relatively small number of EPP servers to which EPP clients have been required to connect, it has generally been the case that client operators have been able to obtain details of these fees out-of-band by contacting the server operators.

Given the expansion of the DNS namespace, and the proliferation of novel business models, it is desirable to provide a method for EPP clients to query EPP servers for the fees and credits associated with certain commands and specific objects.

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This EPP mapping provides a mechanism by which EPP clients may query the fees and credits associated with various billable transactions, and obtain their current account balance.

1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.
XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.

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

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

2. Migrating to Newer Versions of This Extension

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

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

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

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

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

3. Extension Elements

3.1. Client Commands

The <fee:command> element is used in the EPP <check> command to determine the fee that is applicable to the given command.
The use of the <fee:command> keys off the use of the "name" attribute to define which transform fees the client is requesting information about. Here is the list of possible values for the "name" attribute:

- "create" indicating a <create> command as defined in [RFC5730];
- "delete" indicating a <delete> command as defined in [RFC5730];
- "renew" indicating a <renew> command as defined in [RFC5730];
- "update" indicating a <update> command as defined in [RFC5730];
- "transfer" indicating a <transfer> command as defined in [RFC5730];
- If the server supports the Registry Grace Period Mapping [RFC3915], then the server MUST also support the "restore" value as defined in [RFC3915];
- "custom" indicating a custom command that uses the "customName" attribute to define the custom operation.

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

3.2. Currency Codes

The <fee:currency> element is used to indicate which currency fees are charged in. This value of this element MUST be a three-character currency code from [ISO4217].

Note that ISO 4217 provides the special "XXX" code, which MAY be used if the server uses a non-currency based system for assessing fees, such as a system of credits.

The use of <fee:currency> elements in client commands is OPTIONAL: if a <fee:currency> element is not present in a command, the server MUST determine the currency based on the server default currency or based on the client’s account settings which are agreed to by the client and server via an out-of-band channel. However, the <fee:currency> element MUST be present in responses.

Servers SHOULD NOT perform a currency conversion if a client uses an incorrect currency code. Servers SHOULD return a 2004 "Parameter value range" error instead.

3.3. Validity Periods

When querying for fee information using the <check> command, the <fee:period> element is used to indicate the units to be added to the registration period of objects by the <create>, <renew> and
<transfer> commands. This element is derived from the 
<domain:period> element described in [RFC5731].

The <fee:period> element is OPTIONAL in <check> commands, if omitted, 
the server MUST determine the fee(s) using the server default period.  
The <fee:period> element MUST be present in <check> responses.

3.4. Fees and Credits

Servers which implement this extension will include elements in  
responses which provide information about the fees and/or credits  
associated with a given billable transaction.

The <fee:fee> and <fee:credit> elements are used to provide this  
information. The presence of a <fee:fee> element in a response  
indicates a debit against the client’s account balance; a  
<fee:credit> element indicates a credit. A <fee:fee> element MUST  
have a non-negative value. A <fee:credit> element MUST have a  
negative value.

A server MAY respond with multiple <fee:fee> and <fee:credit>  
elements in the same response. In such cases, the net fee or credit  
applicable to the transaction is the arithmetic sum of the values of  
each of the <fee:fee> and/or <fee:credit> elements. This amount  
applies to the total additional validity period applied to the object  
(where applicable) rather than to any incremental unit.

The following attributes are defined for the <fee:fee> element.  
These are described in detail below:

description: an OPTIONAL attribute which provides a human-readable  
description of the fee. Servers should provide documentation on the  
possible values of this attribute, and their meanings. An OPTIONAL  
"lang" attribute MAY be present to identify the language of the  
returned text and has a default value of "en" (English).

refundable: an OPTIONAL boolean attribute indicating whether the fee  
is refundable if the object is deleted.

grace-period: an OPTIONAL attribute which provides the time period  
during which the fee is refundable.

applied: an OPTIONAL attribute indicating when the fee will be  
deducted from the client’s account.

The <fee:credit> element can take a "description" attribute as  
described above. An OPTIONAL "lang" attribute MAY be present to
3.4.1. Refunds

<fee:fee> elements MAY have an OPTIONAL "refundable" attribute which takes a boolean value. Fees may be refunded under certain circumstances, such as when a domain application is rejected (as described in [RFC8334]) or when an object is deleted during the relevant Grace Period (see below).

If the "refundable" attribute is omitted, then clients SHOULD NOT make any assumption about the refundability of the fee.

3.4.2. Grace Periods

[RFC3915] describes a system of "grace periods", which are time periods following a billable transaction during which, if an object is deleted, the client receives a refund.

The "grace-period" attribute MAY be used to indicate the relevant grace period for a fee. If a server implements the Registry Grace Period extension [RFC3915], it MUST specify the grace period for all relevant transactions.

If the "grace-period" attribute is omitted, then clients SHOULD NOT make any assumption about the grace period of the fee.

3.4.3. Correlation between Refundability and Grace Periods

If a <fee:fee> element has a "grace-period" attribute then it MUST also be refundable and the "refundable" attribute MUST be true. If the "refundable" attribute of a <fee:fee> element is false then it MUST NOT have a "grace-period" attribute.

3.4.4. Applicability

Fees may be applied immediately upon receipt of a command from a client, or may only be applied once an out-of-band process (such as the processing of applications at the end of a launch phase) has taken place.

The "applied" attribute of the <fee:fee> element allows servers to indicate whether a fee will be applied immediately, or whether it will be applied at some point in the future. This attribute takes two possible values: "immediate" or "delayed".
3.5. Account Balance

The <fee:balance> element is an OPTIONAL element which MAY be included in server responses to transform commands. If present, it can be used by the client to determine the remaining credit at the server.

Whether or not the <fee:balance> is included in responses is a matter of server policy. However, if a server chooses to offer support for this element, it MUST be included in responses to all "transform" or billable commands (e.g. <create>, <renew>, <update>, <delete>, <transfer op="request">).

The value of the <fee:balance> MAY be negative. A negative balance indicates the server has extended a line of credit to the client (see below).

If a server includes a <fee:balance> element in response to transform commands, the value of the element MUST reflect the client’s account balance after any fees or credits associated with that command have been applied. If the "applied" attribute of the <fee:fee> element is "delayed", then the <fee:balance> MUST reflect the client’s account balance without any fees or credits associated with that command.

3.6. Credit Limit

As described above, if a server returns a response containing a <fee:balance> with a negative value, then the server has extended a line of credit to the client. A server MAY also include a <fee:creditLimit> element in responses that indicates the maximum credit available to a client. A server MAY reject certain transactions if the absolute value of the <fee:balance> is equal to or exceeds the value of the <fee:creditLimit> element.

Whether or not the <fee:creditLimit> is included in responses is a matter of server policy. However, if a server chooses to offer support for this element, it MUST be included in responses to all "transform" commands (e.g. <create>, <renew>, <update>, <delete>, <transfer op="request">).

3.7. Classification of Objects

Objects may be assigned to a particular class, category, or tier, each of which has a particular fee or set of fees associated with it. The <fee:class> element, which MAY appear in <check> and transform responses, is used to indicate the classification of an object.
If a server makes use of this element, it should provide clients with a list of all the values that the element may take via an out-of-band channel. Servers MUST NOT use values which do not appear on this list.

Servers that make use of this element MUST use a <fee:class> element with the value "standard" for all objects that are subject to the standard or default fee.

3.8. Phase and Subphase Attributes

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

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

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

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

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

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

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

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

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

3.9. Reason

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

The `<fee:reason>` element can be used within the server response `<fee:command>` element or within the `<fee:cd>` element.

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

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

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

2. Partial-fail - The server, upon error identification, MAY continue processing `<fee:command>` elements and return to the client a `<fee:cd>` containing successfully processed `<fee:command>` elements and failed `<fee:command>` elements. All returned failed `<fee:command>` elements MUST have a `<fee:reason>` element detailing the reason for failure, and the server MAY additionally include a `<fee:reason>` element at the `<fee:cd>` level.
S: <fee:cd avail="0">
S:   <fee:objID>example.xyz</fee:objID>
S:   <fee:command name="create">
S:     <fee:period unit="y">2</fee:period>
S:     <fee:reason>Only 1 year registration periods are valid.</fee:reason>
S:   </fee:command>
S: </fee:cd>

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

4. Server Handling of Fee Information

Depending on server policy, a client MAY be required to include the extension elements described in this document for certain transform commands. Servers must provide clear documentation to clients about the circumstances in which this extension must be used.

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

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

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

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

If the currency or total fee provided by the client is less than the server's own calculation of the fee for that command, then the server MUST reject the command with a 2004 "Parameter value range" error.
5. EPP Command Mapping

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

5.1. EPP Query Commands

This extension does not add any elements to the EPP <poll> or <info> commands or responses.

5.1.1. EPP <check> Command

This extension defines a new command called the Fee Check Command that defines additional elements for the EPP <check> command to provide fee information along with the availability information of the EPP <check> command.

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

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

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

```xml
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <check>
      <domain:check xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
        <domain:name>example.net</domain:name>
        <domain:name>example.xyz</domain:name>
      </domain:check>
    </check>
    <extension>
      <fee:check xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
        <fee:currency>USD</fee:currency>
        <fee:command name="create">
          <fee:period unit="y">2</fee:period>
        </fee:command>
        <fee:command name="renew"/>
        <fee:command name="transfer"/>
        <fee:command name="restore"/>
      </fee:check>
    </extension>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

When the server receives a <check> command that includes the extension elements described above, its response MUST contain an <extension> element, which MUST contain a child <fee:chkData> element. The <fee:chkData> element MUST contain a <fee:currency> element and a <fee:cd> for each element referenced in the client <check> command.

Each <fee:cd> element MUST contain the following child elements:

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

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

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

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

If the "avail" attribute of the <fee:cd> element is true and if no <fee:fee> elements are present in a <fee:command> element, this indicates that no fee will be assessed by the server for this command.

If the "avail" attribute is true, then the <fee:command> element MUST NOT contain a <fee:reason> element.

Example <check> response:

```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:       <domain:chkData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:cd>
S:           <domain:name avail="1">example.com</domain:name>
S:         </domain:cd>
S:       </domain:chkData>
```
<domain:name avail="1">example.net</domain:name>
<domain:cd>
<domain:name avail="1">example.xyz</domain:name>
</domain:cd>
</domain:chkData>
</resData>
<extension>
<fee:chkData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
<fee:currency>USD</fee:currency>
<fee:cd avail="1">
<fee:objID>example.com</fee:objID>
<fee:class>Premium</fee:class>
<fee:command name="create">
<fee:period unit="y">2</fee:period>
<fee:fee
description="Registration Fee"
refundable="1"
grace-period="P5D">10.00</fee:fee>
</fee:command>
<fee:command name="renew">
<fee:period unit="y">1</fee:period>
<fee:fee
description="Renewal Fee"
refundable="1"
grace-period="P5D">10.00</fee:fee>
</fee:command>
<fee:command name="transfer">
<fee:period unit="y">1</fee:period>
<fee:fee
description="Transfer Fee"
refundable="1"
grace-period="P5D">10.00</fee:fee>
</fee:command>
<fee:command name="restore">
<fee:fee
description="Redemption Fee">15.00</fee:fee>
</fee:command>
</fee:cd>
<fee:cd avail="1">
<fee:objID>example.net</fee:objID>
<fee:class>standard</fee:class>
<fee:command name="create" standard="1">
<fee:period unit="y">2</fee:period>
<fee:fee
description="Registration Fee"
refundable="1"
S:   grace-period="P5D">5.00</fee:fee>
S: </fee:command>
S: <fee:command name="renew" standard="1">
  <fee:period unit="y">1</fee:period>
  <fee:fee
    description="Renewal Fee"
    refundable="1"
    grace-period="P5D">5.00</fee:fee>
</fee:command>
S: <fee:command name="transfer" standard="1">
  <fee:period unit="y">1</fee:period>
  <fee:fee
    description="Transfer Fee"
    refundable="1"
    grace-period="P5D">5.00</fee:fee>
</fee:command>
S: <fee:command name="restore" standard="1">
  <fee:fee
    description="Redemption Fee">5.00</fee:fee>
</fee:command>
S: </fee:cd>
S: <fee:cd avail="0">
  <fee:objID>example.xyz</fee:objID>
  <fee:command name="create">
    <fee:period unit="y">2</fee:period>
    <fee:reason>Only 1 year registration periods are valid.</fee:reason>
  </fee:command>
</fee:cd>
S: </fee:chkData>
S: </extension>
S: </trID>
S: <clTRID>ABC-12345</clTRID>
S: <svTRID>54322-XYZ</svTRID>
S: </trID>
S: </response>
S: </epp>

5.1.2. EPP Transfer Query Command

This extension does not add any elements to the EPP <transfer> query command, but does include elements in the response, when the extension is included in the <login> command service extensions.

When the <transfer> query command has been processed successfully, if the client has included the extension in the <login> command service <svcExtension> element, and if the client is authorized by the server to view information about the transfer, then the server MAY include
in the <extension> section of the EPP response a <fee:trnData> element, which contains the following child elements:

- A <fee:currency> element (as described in Section 3.2).
- A <fee:period> element (as described in Section 3.3).
- Zero or more <fee:fee> elements (as described in Section 3.4) containing the fees that will be charged to the gaining client.
- Zero or more <fee:credit> elements (as described in Section 3.4) containing the credits that will be refunded to the losing client.

Servers SHOULD omit <fee:credit> when returning a response to the gaining client, and omit <fee:fee> elements when returning a response to the losing client.

If no <fee:trnData> element is included in the response, then no fee will be assessed by the server for the transfer.
Example <transfer> query response:
S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1001">
S:       <msg>Command completed successfully; action pending</msg>
S:     </result>
S:     <resData>
S:       <domain:trnData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>example.com</domain:name>
S:         <domain:trStatus>pending</domain:trStatus>
S:         <domain:reID>ClientX</domain:reID>
S:         <domain:reDate>2000-06-08T22:00:00.0Z</domain:reDate>
S:         <domain:acID>ClientY</domain:acID>
S:         <domain:acDate>2000-06-13T22:00:00.0Z</domain:acDate>
S:         <domain:exDate>2002-09-08T22:00:00.0Z</domain:exDate>
S:       </domain:trnData>
S:     </resData>
S:     <extension>
S:       <fee:trnData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:period unit="y">1</fee:period>
S:         <fee:fee>5.00</fee:fee>
S:       </fee:trnData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54322-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>

5.2. EPP Transform Commands

5.2.1. EPP <create> Command

This extension adds elements to both the EPP <create> command and response, when the extension is included in the <login> command service extensions.

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

- An OPTIONAL <fee:currency> element (as described in Section 3.2);
- One or more <fee:fee> elements (as described in Section 3.4).
The server MUST fail the <create> command if the <fee:fee> provided by the client is less than the server fee.

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

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

Example <create> command:

```
C: <?xml version="1.0" encoding="utf-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <create>
C:       <domain:create
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:period unit="y">2</domain:period>
C:         <domain:ns>
C:           <domain:hostObj>ns1.example.net</domain:hostObj>
C:           <domain:hostObj>ns2.example.net</domain:hostObj>
C:         </domain:ns>
C:         <domain:registrant>jd1234</domain:registrant>
C:         <domain:contact type="admin">sh8013</domain:contact>
C:         <domain:contact type="tech">sh8013</domain:contact>
C:         <domain:authInfo>
C:           <domain:pw>2fooBAR</domain:pw>
C:         </domain:authInfo>
C:       </domain:create>
C:     </create>
C:     <extension>
C:       <fee:create xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:fee>5.00</fee:fee>
C:       </fee:create>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```
Example <create> response:

S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <resData>
S:       <domain:creData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>example.com</domain:name>
S:         <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:         <domain:exDate>2001-04-03T22:00:00.0Z</domain:exDate>
S:       </domain:creData>
S:     </resData>
S:     <extension>
S:       <fee:creData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee
S:           description="Registration Fee"
S:           lang="en"
S:           refundable="1"
S:           grace-period="P5D">5.00</fee:fee>
S:         <fee:balance>-5.00</fee:balance>
S:         <fee:creditLimit>1000.00</fee:creditLimit>
S:       </fee:creData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>

5.2.2. EPP <delete> Command

This extension does not add any elements to the EPP <delete> command, but does include elements in the response, when the extension is included in the <login> command service extensions.

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

- A <fee:currency> element (as described in Section 3.2);
Internet-Draft           Registry Fee Extension                 May 2019

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

Example <delete> response:

S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <extension>
S:       <fee:delData
S:         xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:credit
S:           description="AGP Credit"
S:           lang="en">-5.00</fee:credit>
S:         <fee:balance>1005.00</fee:balance>
S:       </fee:delData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>

5.2.3.  EPP <renew> Command

This extension adds elements to both the EPP <renew> command and response, when the extension is included in the <login> command service extensions.

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

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

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

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

Example `<renew>` command:

```xml
C: <?xml version="1.0" encoding="utf-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <renew>
C:       <domain:renew xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:curExpDate>2000-04-03</domain:curExpDate>
C:       </domain:renew>
C:     </renew>
C:     <extension>
C:       <fee:renew xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:fee>5.00</fee:fee>
C:       </fee:renew>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```
Example <renew> response:

S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <resData>
S:       <domain:renData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>example.com</domain:name>
S:         <domain:exDate>2005-04-03T22:00:00.0Z</domain:exDate>
S:       </domain:renData>
S:     </resData>
S:     <extension>
S:       <fee:renData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee refundable="1" grace-period="P5D">5.00</fee:fee>
S:         <fee:balance>1000.00</fee:balance>
S:       </fee:renData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54322-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>

5.2.4. EPP <transfer> Command

This extension adds elements to both the EPP <transfer> command and response, when the value of the "op" attribute of the <transfer> command element is "request", and the extension is included in the <login> command service extensions.

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

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

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

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

Example <transfer> command:

C: <?xml version="1.0" encoding="utf-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <transfer op="request">
C:       <domain:transfer
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:period unit="y">1</domain:period>
C:         <domain:authInfo>
C:           <domain:pw roid="JD1234-REP">2fooBAR</domain:pw>
C:         </domain:authInfo>
C:       </domain:transfer>
C:     </transfer>
C:   </command>
C: <extension>
C:     <fee:transfer xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:       <fee:currency>USD</fee:currency>
C:       <fee:fee>5.00</fee:fee>
C:     </fee:transfer>
C:   </extension>
C: <clTRID>ABC-12345</clTRID>
C: </epp>
Example <transfer> response:

S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1001">
S:       <msg>Command completed successfully; action pending</msg>
S:     </result>
S:     <resData>
S:       <domain:trnData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>example.com</domain:name>
S:         <domain:trStatus>pending</domain:trStatus>
S:         <domain:reID>ClientX</domain:reID>
S:         <domain:reDate>2000-06-08T22:00:00.0Z</domain:reDate>
S:         <domain:acID>ClientY</domain:acID>
S:         <domain:acDate>2000-06-13T22:00:00.0Z</domain:acDate>
S:         <domain:exDate>2002-09-08T22:00:00.0Z</domain:exDate>
S:       </domain:trnData>
S:     </resData>
S:     <extension>
S:       <fee:trnData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee
S:           refundable="1"
S:           grace-period="P5D">5.00</fee:fee>
S:       </fee:trnData>
S:     </extension>
S:   </response>
S: </epp>

5.2.5. EPP <update> Command

This extension adds elements to both the EPP <update> command and response, when the extension is included in the <login> command service extensions.

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

- An OPTIONAL <fee:currency> element (as described in Section 3.2);
- One or more <fee:fee> elements (as described in Section 3.4).
When the <update> command has been processed successfully, and the client included the extension in the <login> command service extensions, the server MAY include in the <extension> section of the EPP response a <fee:updData> element, which contains the following child elements:

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

Example <update> command:

C: <?xml version="1.0" encoding="utf-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <update>
C:       <domain:update
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:chg>
C:           <domain:registrant>sh8013</domain:registrant>
C:       </domain:update>
C:     </update>
C:     <extension>
C:       <fee:update xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:fee>5.00</fee:fee>
C:       </fee:update>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
Example <update> response:

S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <extension>
S:       <fee:updData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee>5.00</fee:fee>
S:       </fee:updData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>

6. Formal Syntax

One schema is presented here that is the EPP Fee Extension schema.

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

6.1. Fee Extension Schema

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

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

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

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

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

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

  <!-- Child elements found in EPP commands and responses -->
  <element name="check" type="fee:checkType"/>
  <element name="chkData" type="fee:chkDataType"/>
  <element name="create" type="fee:transformCommandType"/>
  <element name="creData" type="fee:transformResultType"/>
  <element name="renew" type="fee:transformCommandType"/>
  <element name="renData" type="fee:transformResultType"/>
  <element name="transfer" type="fee:transformCommandType"/>
  <element name="trnData" type="fee:transformResultType"/>
  <element name="update" type="fee:transformCommandType"/>
  <element name="updData" type="fee:transformResultType"/>
  <element name="delData" type="fee:transformResultType"/>

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

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

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

<complexType name="objectCDType">
  <sequence>
    <element name="objID" type="fee:objectIdentifierType" />
    <element name="class" type="token" minOccurs="0" />
    <element name="command" type="fee:commandDataType"
      minOccurs="0" maxOccurs="unbounded" />
    <element name="reason" type="fee:reasonType" minOccurs="0" />
  </sequence>
  <attribute name="avail" type="boolean" default="1" />
</complexType>

<!-- general transform (create, renew, update, transfer) command -->
<complexType name="transformCommandType">
  <sequence>
    <element name="currency" type="fee:currencyType"
      minOccurs="0" />
    <element name="fee" type="fee:feeType"
      minOccurs="0" maxOccurs="unbounded" />
    <element name="credit" type="fee:creditType"
      minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>
<complexType name="transformResultType">
  <sequence>
    <element name="currency" type="fee:currencyType" minOccurs="0" />
    <element name="period" type="domain:periodType" minOccurs="0" />
    <element name="fee" type="fee:feeType" minOccurs="0" maxOccurs="unbounded" />
    <element name="credit" type="fee:creditType" minOccurs="0" maxOccurs="unbounded" />
    <element name="balance" type="fee:balanceType" minOccurs="0" />
    <element name="creditLimit" type="fee:creditLimitType" minOccurs="0" />
  </sequence>
</complexType>

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

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

<complexType name="commandEnum">
  <restriction base="token">
    <enumeration value="create"/>
    <enumeration value="delete"/>
    <enumeration value="renew"/>
    <enumeration value="update"/>
    <enumeration value="transfer"/>
    <enumeration value="restore"/>
    <enumeration value="custom"/>
  </restriction>
</complexType>

<simpleType name="nonNegativeDecimal">
  <restriction base="decimal">
    <minInclusive value="0" />
  </restriction>
</simpleType>

<simpleType name="negativeDecimal">
  <restriction base="decimal">
    <maxInclusive value="0" />
  </restriction>
</simpleType>

<complexType name="feeType">
  <simpleContent>
    <extension base="fee:nonNegativeDecimal">
      <attribute name="description"/>
      <attribute name="lang" type="language" default="en"/>
      <attribute name="refundable" type="boolean"/>
      <attribute name="grace-period" type="duration"/>
      <attribute name="applied">
        <simpleType>
          <restriction base="token">
            <enumeration value="immediate"/>
            <enumeration value="delayed"/>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </simpleContent>
</complexType>
7. Security Considerations

The mapping extensions described in this document do not provide any
security services beyond those described by EPP [RFC5730], the EPP
domain name mapping [RFC5731], and protocol layers used by EPP. The
security considerations described in these other specifications apply
to this specification as well.

8. IANA Considerations

8.1. XML Namespace

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

Registration request for the fee namespace:

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

Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.

Registration request for the fee schema:
URI: urn:ietf:params:xml:schema:epp:fee-1.0
Registrant Contact: IESG

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

8.2.  EPP Extension Registry

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

Name of Extension: Registry Fee Extension for the Extensible Provisioning Protocol (EPP)
Document status: Standards Track
Reference: (insert reference to RFC version of this document)
Registrant Name and Email Address: IESG, <iesg@ietf.org>
TLDs: Any
IPR Disclosure: None
Status: Active
Notes: None

9.  Implementation Status

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

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

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

9.1. RegistryEngine EPP Service

Organization: CentralNic

Name: RegistryEngine EPP Service

Description: Generic high-volume EPP service for gTLDs, ccTLDs and SLDs

Level of maturity: Deployed in CentralNic’s production environment as well as two other gTLD registry systems, and two ccTLD registry systems.

Coverage: All aspects of the protocol are implemented.

Licensing: Proprietary In-House software

Contact: epp@centralnic.com

URL: https://www.centralnic.com

10. Acknowledgements

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

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

11.1. Change from 15 to 16

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

11.2. Change from 14 to 15

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

11.3. Change from 13 to 14

Moved RFC 7451 reference from Normative to Informative section.

11.4. Change from 12 to 13


11.5. Change from 11 to 12

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

11.6. Change from 10 to 11

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

11.7. Change from 09 to 10

Updated document per Working Group Last Call comments. Updated schema to version 1.0 in anticipation of standardization, no changes were made to the latest, 0.25, schema. Made minor textual changes throughout for enhanced clarity per WGLC comments.
11.8. Change from 08 to 09

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

11.9. Change from 07 to 08

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

11.10. Change from 06 to 07

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

11.11. Change from 05 to 06

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

11.12. Change from 04 to 05

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

11.13. Change from 03 to 04

Updated scheme to version 0.19 to correct typos and to replace the
commandTypeValue type with the commandEnum type and customName
attribute for stricter validation. Updated various text for grammar
and clarity. Added text to section 4 clarifying the <check> response
when the client provided no fee extension but the server was expecting the extension.

11.14. Change from 02 to 03

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

11.15. Change from 01 to 02

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

11.16. Change from 00 to 01

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

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

Updated to be REGEXT WG document.

12. References

12.1. Normative References

12.2. Informative References


Authors’ Addresses

Carney, et al.          Expires November 2, 2019               [Page 38]
Roger Carney  
GoDaddy Inc.  
14455 N. Hayden Rd. #219  
Scottsdale, AZ  85260  
US  

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

Gavin Brown  
CentralNic Group plc  
35-39 Moorgate  
London, England  EC2R 6AR  
GB  

Phone: +44 20 33 88 0600  
Email: gavin.brown@centralnic.com  
URI:   http://www.centralnic.com

Jothan Frakes  

Email: jothan@jothan.com  
URI:   http://jothan.com
Login Security Extension for the Extensible Provisioning Protocol (EPP)
draft-ietf-regext-login-security-03

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on February 6, 2020.

Copyright Notice

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

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

Gould & Pozun Expires February 6, 2020 [Page 1]
1. Introduction

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

1.1. Conventions Used in This Document

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

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

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

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

2. Migrating to Newer Versions of This Extension

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

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

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

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

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

3. Object Attributes

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

3.1. Event

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

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

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

"exDate": Contains the date and time that a "warning" level has or will become an "error" level. At expiry there MAY be an error to connect or MAY be an error to login. An example is an expired certificate that will result in an error to connect or an expired password that may result in a failed login.

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

"duration": Defines the duration that a statistical event is associated with, ending when the login command was received. The format of the duration is defined by the duration primitive datatype in [W3C.REC-xmlschema-2-20041028].

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

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

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

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

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

3.2. "[LOGIN-SECURITY]" Password

The <loginSec:pw> element MUST override the [RFC5730] <pw> element only if the <pw> contains the predefined value of "[LOGIN-SECURITY]", which is a constant value for the server to use the <loginSec:pw> element for the password. Similarly, the <loginSec:newPW> element MUST override the [RFC5730] <newPW> element only if the <newPW>
contains the predefined value of ":[LOGIN-SECURITY]"", which is a constant value for the server to use the <loginSec:newPW> element for the new password. The ":[LOGIN-SECURITY]" pre-defined string MUST be supported by the server for the client to explicitly indicate to the server whether to use <loginSec:pw> element in place of the [RFC5730] <pw> element or to use the <loginSec:newPW> in place of the [RFC5730] <newPW> element. The server MUST NOT allow the client to set the password to the value ":[LOGIN-SECURITY]".

3.3. Dates and Times

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

4. EPP Command Mapping

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

4.1. EPP <login> Command

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

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

<loginSec:userAgent>: OPTIONAL client user agent that identifies the client application software, technology, and operating system used by the server to identify functional or security constraints, current security issues, and potential future functional or security issues for the client. The <loginSec:userAgent> element MUST contain at least one of the following child elements:

<loginSec:app>: OPTIONAL name of the client application software with version if available, such as the name of the client SDK "EPP SDK 1.0.0".

<loginSec:tech>: OPTIONAL technology used for the client software with version if available, such as "Java 11.0.2".
<loginSec:os>: OPTIONAL client operating system used with version if available, such as "x86_64 Mac OS X 10.11.6".
<loginSec:pw>: OPTIONAL plain text password that is case sensitive, has a minimum length of 6 characters, and has a maximum length that is up to server policy. All leading and trailing whitespace is removed, and all internal contiguous whitespace that includes \#x9 (tab), \#xA (linefeed), \#xD (carriage return), and \#x20 (space) is replaced with a single \#x20 (space). This element MUST only be used if the [RFC5730] <pw> element is set to the "[LOGIN-SECURITY]" value.
<loginSec:newPW>: OPTIONAL plain text new password that is case sensitive, has a minimum length of 6 characters, and has a maximum length that is up to server policy. All leading and trailing whitespace is removed, and all internal contiguous whitespace that includes \#x9 (tab), \#xA (linefeed), \#xD (carriage return), and \#x20 (space) is replaced with a single \#x20 (space). This element MUST only be used if the [RFC5730] <newPW> element is set to the "[LOGIN-SECURITY]" value.
Example login command that uses the `<loginSec:pw>` element instead of the [RFC5730] `<pw>` element to establish the session and includes the `<loginSec:userAgent>` element:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <login>
      <clID>ClientX</clID>
      <pw>[LOGIN-SECURITY]</pw>
      <options>
        <version>1.0</version>
        <lang>en</lang>
      </options>
      <svcs>
        <objURI>urn:ietf:params:xml:ns:obj1</objURI>
        <objURI>urn:ietf:params:xml:ns:obj2</objURI>
        <svcExtension>
          <extURI>urn:ietf:params:xml:ns:epp:loginSec-1.0</extURI>
        </svcExtension>
      </svcs>
    </login>
    <extension>
      <loginSec:loginSec
        xmlns:loginSec="urn:ietf:params:xml:ns:epp:loginSec-1.0">
        <loginSec:userAgent>
          <loginSec:app>EPP SDK 1.0.0</loginSec:app>
          <loginSec:tech>Java 11.0.2</loginSec:tech>
          <loginSec:os>x86_64 Mac OS X 10.11.6</loginSec:os>
        </loginSec:userAgent>
        <loginSec:pw>this is a long password</loginSec:pw>
      </loginSec:loginSec>
    </extension>
  <clTRID>ABC-12345</clTRID>
</command>
</epp>
```
Example login command that uses the `<loginSec:pw>` element instead of the [RFC5730] `<pw>` element to establish the session, and uses the `<loginSec:newPW>` element instead of the [RFC5730] `<newPW>` element to set the new password:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <login>
      <clID>ClientX</clID>
      <pw>[LOGIN-SECURITY]</pw>
      <newPW>[LOGIN-SECURITY]</newPW>
      <options>
        <version>1.0</version>
        <lang>en</lang>
      </options>
      <svcs>
        <objURI>urn:ietf:params:xml:ns:obj1</objURI>
        <objURI>urn:ietf:params:xml:ns:obj2</objURI>
        <svcExtension>
          <extURI>urn:ietf:params:xml:ns:epp:loginSec-1.0</extURI>
        </svcExtension>
      </svcs>
    </login>
    <extension>
      <loginSec:loginSec
        xmlns:loginSec="urn:ietf:params:xml:ns:epp:loginSec-1.0">
        <loginSec:pw>this is a long password</loginSec:pw>
        <loginSec:newPW>new password that is still long</loginSec:newPW>
      </loginSec:loginSec>
    </extension>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```
Example login command that uses the [RFC5730] <pw> element to establish the session, and uses the <loginSec:newPW> element instead of the [RFC5730] <newPW> element to set the new password:

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

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

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

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

The extension to the EPP response uses the <loginSec:loginSecData> element that contains the following child elements:
<loginSec:event>: One or more <loginSec:event> elements defined in Section 3.1.

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

S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <extension>
S:      <loginSec:loginSecData xmlns:loginSec="urn:ietf:params:xml:ns:epp:loginSec-1.0">
S:        <loginSec:event type="password" level="warning" exDate="2018-04-01T22:00:00.0Z" lang="en">
S:          Password expiring in a week
S:        </loginSec:event>
S:      </loginSec:loginSecData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
Example EPP response to a failed login command where the password has expired and the new password does not meet the server complexity requirements:

```
S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="2200">
S:      <msg>Authentication error</msg>
S:    </result>
S:    <extension>
S:      <loginSec:loginSecData
S:        xmlns:loginSec="urn:ietf:params:xml:ns:epp:loginSec-1.0">
S:        <loginSec:event
S:          type="password"
S:          level="error"
S:          exDate="2018-03-26T22:00:00.0Z">
S:          Password has expired
S:        </loginSec:event>
S:        <loginSec:event
S:          type="newPW"
S:          level="error">
S:          New password does not meet complexity requirements
S:        </loginSec:event>
S:      </loginSec:loginSecData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S: </epp>
```

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

```
S: <?xml version="1.0" encoding="UTF-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <extension>
S:      <loginSec:loginSecData
S:        xmlns:loginSec="urn:ietf:params:xml:ns:epp:loginSec-1.0">
S:        <loginSec:event
S:          type="password"
S:          level="error"
S:          exDate="2018-03-26T22:00:00.0Z">
S:          Password has expired
S:        </loginSec:event>
S:        <loginSec:event
S:          type="newPW"
S:          level="error">
S:          New password does not meet complexity requirements
S:        </loginSec:event>
S:      </loginSec:loginSecData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S: </epp>
```
S: type="password"
S: level="warning"
S: exDate="2018-04-01T22:00:00.0Z"
S: lang="en">
S: Password expiration soon
S: </loginSec:event>
S: <loginSec:event
S: type="certificate"
S: level="warning"
S: exDate="2018-04-02T22:00:00.0Z"/>
S: <loginSec:event
S: type="cipher"
S: level="warning"
S: value="TLS_RSA_WITH_AES_128_CBC_SHA">
S: Non-PFS Cipher negotiated
S: </loginSec:event>
S: <loginSec:event
S: type="tlsProtocol"
S: level="warning"
S: value="TLSv1.0">
S: Insecure TLS protocol negotiated
S: </loginSec:event>
S: <loginSec:event
S: type="stat"
S: name="failedLogins"
S: level="warning"
S: value="100"
S: duration="P1D">
S: Excessive invalid daily logins
S: </loginSec:event>
S: <loginSec:event
S: type="custom"
S: name="myCustomEvent"
S: level="warning">
S: A custom login security event occured
S: </loginSec:event>
S: </loginSec:loginSecData>
S: </extension>
S: <trID>
S: <clTRID>ABC-12345</clTRID>
S: <svTRID>54321-XYZ</svTRID>
S: </trID>
S: </response>
S:</epp>
5. Formal Syntax

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

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

5.1. Login Security Extension Schema

BEGIN
<?xml version="1.0" encoding="UTF-8"?><schema xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
   xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
   xmlns:loginSec="urn:ietf:params:xml:ns:epp:loginSec-1.0"
   targetNamespace="urn:ietf:params:xml:ns:epp:loginSec-1.0"
   elementFormDefault="qualified">
   <!-- Import common element types. -->
   <import namespace="urn:ietf:params:xml:ns:eppcom-1.0" />
   <import namespace="urn:ietf:params:xml:ns:epp-1.0" />
   <annotation>
     <documentation>Extensible Provisioning Protocol v1.0 Login Security Extension Schema.</documentation>
   </annotation>
   <!-- Login command extension elements -->
   <element name="loginSec" type="loginSec:loginSecType" />
   <!-- Attributes associated with the login command extension. -->
   <complexType name="loginSecType">
     <sequence>
       <element name="userAgent" type="loginSec:userAgentType" minOccurs="0" />
       <element name="pw" type="loginSec:pwType" minOccurs="0" />
       <element name="newPW" type="loginSec:pwType" minOccurs="0" />
     </sequence>
   </complexType>
</schema>
</complexType>

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

<complexType name="userAgentType">
  <sequence>
    <element name="app" type="token" minOccurs="0" />
    <element name="tech" type="token" minOccurs="0" />
    <element name="os" type="token" minOccurs="0" />
  </sequence>
</complexType>

<!-- Login response extension elements -->
<element name="loginSecData" type="loginSec:loginSecDataType" />
<complexType name="loginSecDataType">
  <sequence>
    <element name="event" type="loginSec:eventType" minOccurs="1" maxOccurs="unbounded" />
  </sequence>
</complexType>

<!-- Security event element -->
<complexType name="eventType">
  <simpleContent>
    <extension base="normalizedString">
      <attribute name="type" type="loginSec:typeEnum" use="required" />
      <attribute name="name" type="token" />
      <attribute name="level" type="loginSec:levelEnum" use="required" />
      <attribute name="exDate" type="dateTime" />
      <attribute name="value" type="token" />
      <attribute name="duration" type="duration" />
      <attribute name="lang" type="language" default="en" />
    </extension>
  </simpleContent>
</complexType>
6. IANA Considerations

6.1. XML Namespace

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

Registration request for the loginSec namespace:

URI: urn:ietf:params:xml:ns:epp:loginSec-1.0
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.
Registration request for the loginSec XML schema:

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

6.2. EPP Extension Registry

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

Name of Extension: "Login Security Extension for the Extensible Provisioning Protocol (EPP)"

Document status: Standards Track
Reference: (insert reference to RFC version of this document)
Registrant Name and Email Address: IESG, <iesg@ietf.org>
TLDs: Any
IPR Disclosure: None
Status: Active
Notes: None

7. Implementation Status

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

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942 [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.
According to RFC 7942 [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

7.1. Verisign EPP SDK

Organization: Verisign Inc.

Name: Verisign EPP SDK

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

Level of maturity: Development

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com


8. Security Considerations

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

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

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

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

- Martin Casanova
- Scott Hollenbeck
- Patrick Mevzek
- Joseph Yee

10. References

10.1. Normative References


10.2. Informative References

10.3. URIs


Appendix A. Change History

A.1. Change from 00 to 01

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

A.2. Change from 01 to 02


A.3. Change from 02 to 03

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

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

A.4. Change from 03 to REGEXT 00

1. Changed to regext working group draft by changing draft-gould-regext-login-security to draft-ietf-regext-login-security.

A.5. Change from REGEXT 00 to REGEXT 01

1. Changed the <loginSec:userAgent> element to be structured with the <loginSec:app>, <loginSec:tech>, and <loginSec:os> sub-elements. This was based on the feedback from Martin Casanova.
This resulted in the need to change the XML namespace from urn:ietf:params:xml:ns:epp:loginSec-0.3 to urn:ietf:params:xml:ns:epp:loginSec-0.4.

A.6. Change from REGEXT 01 to REGEXT 02

1. Updated the Implementation Status section from "TBD" to include the Verisign EPP SDK implementation.

A.7. Change from REGEXT 02 to REGEXT 03

1. Revised the description of the "duration" attribute to clarify that it ends when the login command was received and to clarify the format, based on the feedback from Martin Casanova.
2. Revised the sentence ‘Upon a completed login command (success or failed), the extension MUST be included in the response based on the following conditions:' to ‘Upon a completed login command (success or failed), the extension MUST be included in the response based on both of the following conditions:' based on the feedback from Patrick Mevzek.
3. Updates based on the review by Joseph Yee, that include:
   1. Revised the description of the <loginSec:event> "name" attribute read 'Used to define a sub-type when the "type" attribute is not "custom" or the full type name when the "type" attribute is "custom"'. The definition of the "stat" type was updated to 'Provides a login security statistical warning that MUST set the "name" attribute to the name of the statistic.'
   2. Added the following sentence 'The server MUST NOT allow the client to set the password to the value "[LOGIN-SECURITY]".' to address the corner case where the constant is used as the password.
   3. Revised the description of the <loginSec:userAgent> element to read 'The <loginSec:userAgent> element MUST contain at least one of the following child elements:'.
   4. Revised the description of the <loginSec:userAgent> to match the child elements that can be passed, by changing "client software" to "client application software" and change "language" to "technology".
Authors’ Addresses

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

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

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

Email: mpozun@verisign.com
URI:   http://www.verisign.com
Federated Authentication for the Registration Data Access Protocol (RDAP) using OpenID Connect
draft-ietf-regext-rdap-openid-03

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 February 20, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.
This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.  Introduction ............................................. 3
   1.1.  Problem Statement ................................. 3
   1.2.  Proposal .......................................... 3
2.  Conventions Used in This Document ........................ 4
3.  Federated Authentication for RDAP .......................... 4
   3.1.  RDAP and OpenID Connect ............................ 5
      3.1.1.  Terminology .................................... 5
      3.1.2.  Overview ...................................... 5
      3.1.3.  RDAP Authentication and Authorization Steps ....... 6
         3.1.3.1.  Provider Discovery .......................... 6
         3.1.3.2.  Authentication Request ...................... 6
         3.1.3.3.  End-User Authorization ....................... 7
         3.1.3.4.  Authorization Response and Validation ....... 7
         3.1.3.5.  Token Processing ............................. 7
         3.1.3.6.  Delivery of User Information ................. 7
      3.1.4.  Specialized Claims for RDAP ..................... 8
         3.1.4.1.  Stated Purpose .............................. 8
         3.1.4.2.  Do Not Track ................................ 9
4.  Protocol Parameters ....................................... 9
   4.1.  Client Authentication Request and Response ........... 10
   4.2.  Token Request and Response .......................... 10
   4.3.  Token Refresh and Revocation ........................ 11
   4.4.  Token Exchange .................................... 14
   4.5.  Parameter Processing ................................ 14
   4.6.  RDAP Conformance .................................. 15
5.  Clients with Limited User Interfaces ....................... 15
   5.1.  OAuth 2.0 Device Authorization Grant ................. 16
   5.2.  Manual Token Management ............................ 16
6.  IANA Considerations ..................................... 17
   6.1.  RDAP Extensions Registry ............................ 17
   6.2.  JSON Web Token Claims Registry ...................... 17
   6.3.  RDAP Query Purpose Registry ......................... 17
7.  Implementation Status .................................... 20
   7.1.  Verisign Labs ..................................... 21
   7.2.  Viagenie ......................................... 21
8.  Security Considerations .................................. 22
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.
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" privileges in accordance with service policies and regulations. Specification of these policies and regulations is beyond the scope of this document.

4. Protocol Parameters

This specification adds the following protocol parameters to RDAP:

1. 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):

```json
{
    "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):

```text
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 TBD [I-D.ietf-oauth-token-exchange].

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:

```json
"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

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
- Registry operator: Any
- Published specification: This document.
- Contact: IESG <iesg@ietf.org>
- Intended usage: This extension includes response information required for federated authentication using 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: see text
-----END FORM-----

Hollenbeck

Expires February 20, 2020
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 website 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


Hollenbeck Expires February 20, 2020 [Page 23]


10.2. Informative References


10.3. URIs


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.

Author’s Address

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

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

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

Several leading APIs providers (e.g. LinkedIn [LINKEDIN], Facebook [FACEBOOK], Google [GOOGLE]) implement the partial response feature by providing an optional query parameter by which users require the fields they wish to receive. Partial response is also considered a leading principle by many best practices guidelines in REST APIs implementation ([REST-API1], [REST-API2]) in order to improve performance, save on bandwidth and possibly accelerate the overall interaction. In other contexts, for example in digital libraries and bibliographic catalogues, servers can provide responses according to different element sets (i.e. "brief" to get back a short response and "full" to get back the complete response).
Currently, RDAP does not provide a client with any way to request a partial response: the server can only provide the client with the full response ([RFC7483]). Furthermore, servers cannot define the limits of the results according to partial responses and this causes strong inefficiencies.

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

1.1. Conventions Used in This Document

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

2. Approaches to Partial Response Implementation

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

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

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

- fields have to be declared according to a given syntax. This is a simple task when the data structure of the object is flat, but it is much more difficult when the object has a tree structure like the one of a JSON object. The presence of arrays and deep nested objects contributes to complicate both the syntax definition of the query and, consequently, the processing phase on the server side;
- clients should perfectly know the returned data structure to avoid cases when the requested fields are invalid;
- the request of some fields might not match the user access levels. Clients might put unauthorized fields in their requests and servers should define a strategy for providing a response: returning always an error response or returning a response that ignores the unauthorized fields.
In addition to those listed above, RDAP responses raise some specific issues:

- most of the relevant information of the entity object is included in the jCard but such information cannot be easily selected because it is split into the items of a jagged array;

- RDAP responses contain some properties providing service information (e.g. rdapConformance, links, notices, remarks, etc.) which are not normally selected but they are just as important. They could be returned anyway but, in this case, the server would provide unrequested data.

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

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

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

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

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

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

3. RDAP Path Segment Specification

The path segment defined in this section is an OPTIONAL extension of search path segments defined in RFC 7482 ([RFC7482]). This document defines an RDAP query parameter, "fieldSet", whose value is a string identifying a server pre-defined set of fields (Figure 2).

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

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

3.1. Subsetting Metadata

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

Therefore, servers implementing the query parameter described in this specification SHOULD provide additional information in their responses about the available field sets. Such information is collected in a new data structure named "subsetting_metadata" containing the following properties:

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

- "availableFieldSets": "AvailableFieldSet[]" (OPTIONAL) an array of objects each one describing an alternate available field set. Members are:
  - "name": "String" (REQUIRED) the field set name;
  - "default": "Boolean" (REQUIRED) whether the field set is applied by default;
  - "description": "String" (OPTIONAL) a human-readable description of the field set;
* "links": "Link[]" (OPTIONAL) an array of links as described in RFC 8288 ([RFC8288]) containing the query string that applies the field set.

3.1.1. Representing Subsetting Links

An RDAP server MAY use the "links" array of the "subsetting_metadata" element to provide ready-made references ([RFC8288]) to the available field sets (Figure 3). Each link represents a reference to an alternate view of the results.

```
{
    "rdapConformance": [
        "rdap_level_0",
        "subsetting_level_0"
    ],
    ...
    "subsetting_metadata": {
        "currentFieldSet": "afieldset",
        "availableFieldSets": [
            {
                "name": "anotherfieldset",
                "description": "Contains some fields",
                "default": false,
                "links": [
                    {
                        "value": "https://example.com/rdap/domains?name=*nr.com &fieldSet=afieldset",
                        "rel": "alternate",
                        "href": "https://example.com/rdap/domains?name=*nr.com &fieldSet=anotherfieldset",
                        "title": "Result Subset Link",
                        "type": "application/rdap+json"
                    },
                    ...
                ]
            },
            "domainSearchResults": [
                ...
            ]
        }
    }
}
```

Figure 3: Example of a "subsetting_metadata" instance
4. Dealing with Relationships

Some additional considerations can be made about how second level objects could be represented within a field set. In fact, since the topmost objects could be returned according to different field sets, the same thing could go for their related objects. As a consequence, the response could contain either no relationship or associated objects which are in turn provided according to a field set.

5. Basic Field Sets

In order to improve interoperability between clients and servers, the name, as well as the list of fields for each field set, should be shared by most of RDAP providers. This section defines three basic field sets which servers MAY implement to facilitate their interaction with clients:

- "id": the server provides only the key field, respectively: "handle" for entities, "ldhName" for domains and nameservers. If a returned domain or nameserver is an IDN ([RFC5890]), then the "unicodeName" field MUST be included in the response. This field set could be used when the client wants to simply obtain a collection of object identifiers (Figure 4);

- "brief": it contains the fields that can be included in a "short" response. This field set could be used when the client is asking for a subset of the full response which gives a basic knowledge of each object;

- "full": it contains all the information the server can provide for a particular object.

The "objectClassName" field is implicitly included in each of the above field sets. RDAP providers MAY add any property providing service information.

Fields included in "brief" and "full" field sets could be returned according to users access levels.
6. Negative Answers

Each request including an unsupported field set SHOULD obtain an HTTP 400 (Bad Request) response code.

Optionally, the response MAY include additional information regarding the negative answer in the HTTP entity body.

7. RDAP Conformance

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

8. 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".

8.1. 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 .it public test environment.
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

9. IANA Considerations

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

Extension identifier: subsetting
Registry operator: Any
Published specification: This document.
Contact: IESG <iesg@ietf.org>
Intended usage: This extension describes a best practice for partial response provisioning.

10. Security Considerations

Search query typically requires more server resources (such as memory, CPU cycles, and network bandwidth) when compared to lookup query. This increases the risk of server resource exhaustion and subsequent denial of service due to abuse. Partial response can contribute together with other strategies (e.g. restricting search functionality, limiting the rate of search requests, truncating and paging results) to mitigate this risk.

Furthermore, partial response can support RDAP operators to implement a versatile access control policy through the HTTP authentication
mechanisms as described in RFC 7481 ([RFC7481]). In fact, RDAP operators can follow different, not alternative, approaches to the building of responses according to the user access levels:

- the list of fields for each set can be different according to the user access levels;
- some field sets could be available only to some users.

Servers can also define different results limits according to the available field sets, so a more flexible truncation strategy can be realized.

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

11. Acknowledgements

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

12. References

12.1. Normative References

12.2. Informative References


Appendix A. Change Log

00: Initial working group version ported from draft-loffredo-regext-rdap-partial-response-03
01: Removed "FOR DISCUSSION" items. Changed the basic field sets from REQUIRED to OPTIONAL. Removed the definition of fields included in "brief" field set. Provided a more detailed description of "subsetting_metadata" structure. Removed some references.
02: Added the "Negative Answers" section. Changed "IANA Considerations" section.
03: Added the "unicodeName" field in the id fieldSet when a returned domain or nameserver is an IDN. Added RFC5890 to "Normative References" section.

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
Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on October 13, 2019.

Copyright Notice

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

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

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

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

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

- the loss of synchronization between the registrar database and the registry database;

- the need of such data to perform massive EPP ([RFC5730]) updates (e.g. changing the contacts of a set of domains, etc.).

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

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

1.1. Conventions Used in This Document

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

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

Syntax: domains?entityHandle=<reverse search pattern>
Syntax: domains?entityFn=<reverse search pattern>
Syntax: domains?entityEmail=<reverse search pattern>
Syntax: domains?entityAddr=<reverse search pattern>

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

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

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

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

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

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

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

where XXXX is a search pattern representing an email address as defined in RFC 5322 ([RFC5322]).
Searches for domains by related entity postal address are specified using this form:

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

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

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

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

3. Implementation Considerations

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

3.1. JSON in URLs

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

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

Sometimes, it happens that such advantages should be combined with the requirement to pass objects and arrays in the query string. JSON is the best candidate as data interchange format, but it contains
some characters that are forbidden from appearing in a URL. Anyway, escaping the invalid characters is not an issue because, on the client side, modern browsers automatically encode URLs and, on the server side, several URL encoding/decoding libraries for all web development programming languages are available. The downside of URL encoding is that it can make a pretty long URL, which, depending on the initial length and the number of invalid characters, might exceed the practical limit of web browsers (i.e. 2,000 characters).

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

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

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

4. Implementation Status

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

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942 ([RFC7942]). The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

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

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

5. Privacy Considerations

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

6. Security Considerations

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

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgements

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

9.1. Normative References


Loeffredo & Martinelli  Expires October 13, 2019  [Page 8]
9.2. Informative References

[ICANN-RA]

[ICANN-RDS1]

[ICANN-RDS2]


Appendix A. Change Log

- 00: Initial working group version ported from draft-loffredo-regext-rdap-reverse-search-04
- 01: Updated "Privacy Considerations" section.

Authors’ Addresses

Mario Loffredo
IIT-CNR/Registro.it
Via Moruzzi,1
Pisa 56124
IT

Email: mario.loffredo@iit.cnr.it
URI: http://www.iit.cnr.it

Maurizio Martinelli
IIT-CNR/Registro.it
Via Moruzzi,1
Pisa 56124
IT

Email: maurizio.martinelli@iit.cnr.it
URI: http://www.iit.cnr.it
Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on February 2, 2020.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents
1. Introduction

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

- reducing the server response bandwidth requirements;
- improving server response time;
- improving query precision and, consequently, obtaining more reliable results;
- decreasing server query processing load;
- reducing client response processing time.
Approaches to implementing features for result sorting and paging can be grouped into two main categories:

1. Sorting and paging are implemented through the introduction of additional parameters in the query string (i.e. ODATA protocol [OData-Part1]);

2. Information related to the number of results and the specific portion of the result set to be returned, in addition to a set of ready-made links for the result set scrolling, are inserted in the HTTP header of the request/response.

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

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

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

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

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

The implementation of the new parameters is technically feasible, as operators for counting, sorting and paging rows are currently supported by the major RDBMSs.
1.1. Conventions Used in This Document

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

2. RDAP Query Parameter Specification

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

- "count": a boolean value that allows a client to request the total number of objects found (that due to truncation can be different from the number of returned objects);
- "sort": a string value that allows a client to request a specific sort order for the result set;
- "cursor": a string value representing a pointer to a specific fixed size portion of the result set.

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

2.1. Sorting and Paging Metadata

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

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

Obviously, both the new data structures are OPTIONAL because their presence in the response not only depends on the implementation of sorting and paging query capabilities but also on some situations related to the results. For example, it is quite natural to expect
that the "paging_metadata" element will not be present at the last result page when the server implements only the forward pagination.

The "sorting_metadata" structure contains the following properties:

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

- "availableSorts": "AvailableSort[]" (OPTIONAL) an array of objects each one describing an alternate available sorting criterion. Members are:
  - "property": "String" (REQUIRED) the name that can be used by the client to request the sorting criterion;
  - "default": "Boolean" (REQUIRED) whether the sorting criterion is applied by default;
  - "jsonPath": "String" (OPTIONAL) the JSON Path of the RDAP field corresponding to the property;
  - "links": "Link[]" (OPTIONAL) an array of links as described in RFC 8288 [RFC8288] containing the query string that applies the sorting criterion.

At least one between "currentSort" and "availableSorts" MUST be present.

The "paging_metadata" structure contains the following fields:

- "totalCount": "Numeric" (OPTIONAL) a numeric value representing the total number of objects found. It is provided if the query string contains the "count" parameter;

- "pageCount": "Numeric" (OPTIONAL) a numeric value representing the number of objects returned in the current page. It is provided when the total number of objects exceeds the page size. This property is redundant for clients because the page size can be derived from the length of the search results array but it can be helpful if the end user interacts with the server through a web browser;

- "links": "Link[]" (OPTIONAL) an array of links as described in RFC 8288 [RFC8288] containing the reference to next page. In this specification, only the forward pagination is dealt because it is considered satisfactory in order to traverse the result set. Examples of additional references are to: the previous page, the first page, the last page.

At least one between "totalCount" and "links" MUST be present.
2.2. "count" Parameter

Currently the RDAP protocol does not allow a client to determine the total number of the results in a query response when the result set is truncated. This is rather inefficient because the user cannot evaluate the query precision and, at the same time, cannot receive information that could be relevant.

The "count" parameter provides additional functionality (Figure 1) that allows a client to request information from the server that specifies the total number of elements matching the search pattern.

https://example.com/rdap/domains?name=*nr.com&count=true

Figure 1: Example of RDAP query reporting the "count" parameter

The ABNF syntax is the following:

```
count = "count=" ( trueValue / falseValue )
trueValue = ("true" / "yes" / "1")
falseValue = ("false" / "no" / "0")
```

A trueValue means that the server MUST provide the total number of the objects in the "totalCount" field of the "paging_metadata" element (Figure 2). A falseValue means that the server MUST NOT provide this number.

```
{
  "rdapConformance": [
    "rdap_level_0",
    "paging_level_0"
  ],
  ...
  "paging_metadata": {
    "totalCount": 73
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 2: Example of RDAP response with "paging_metadata" element containing the "totalCount" field
2.3. "sort" Parameter

The RDAP protocol does not provide any capability to specify results sort criteria. A server could implement a default sorting scheme according to the object class, but this feature is not mandatory and might not meet user requirements. Sorting can be addressed by the client, but this solution is rather inefficient. Sorting features provided by the RDAP server could help avoid truncation of relevant results.

The "sort" parameter allows the client to ask the server to sort the results according to the values of one or more properties and according to the sort direction of each property. The ABNF syntax is the following:

```abnf
sort = "sort=" sortItem *( ""," sortItem )
sortItem = property-ref [":" ( "a" / "d" ) ]
property-ref = ALPHA *( ALPHA / DIGIT / "_" )
```

"a" means that the ascending sort MUST be applied, "d" means that the descending sort MUST be applied. If the sort direction is absent, an ascending sort MUST be applied (Figure 3).

https://example.com/rdap/domains?name=*nr.com&sort=name

https://example.com/rdap/domains?name=*nr.com&sort=registrationDate:d

https://example.com/rdap/domains?name=*nr.com&sort=lockedDate,name

Figure 3: Examples of RDAP query reporting the "sort" parameter

Servers MUST implement sorting according to the JSON value type of the RDAP field the sorting property refers to: the lexicographic sorting for strings and the numeric sorting for numbers.

If the "sort" parameter reports an allowed sorting property, it MUST be provided in the "currentSort" field of the "sorting_metadata" element.

2.3.1. Sorting Properties Declaration

In the "sort" parameter ABNF syntax, property-ref represents a reference to a property of an RDAP object. Such a reference could be expressed by using a JSON Path. The JSON Path in a JSON document [RFC8259] is equivalent to the XPath [W3C.CR-xpath-31-20161213] in an XML document. For example, the JSON Path to select the value of the ASCII name inside an RDAP domain object is "$\.ldhName", where $
identifies the root of the document (DOM). Another way to select a value inside a JSON document is the JSON Pointer [RFC6901]. While JSON Path or JSON Pointer are both standard ways to select any value inside JSON data, neither is particularly easy to use (e.g. "$\cdot\text{events}[\?(@.eventAction='registration')].\text{eventDate}" is the JSON Path expression of the registration date in an RDAP domain object).

Therefore, this specification provides a definition of property-ref in terms of RDAP properties. However, not all the RDAP properties are suitable to be used in sort criteria, such as:

- properties providing service information (e.g. links, notices, remarks, etc.);
- multivalued properties (e.g. status, roles, variants, etc.);
- properties modeling relationships to other objects (e.g. entities).

On the contrary, some properties expressed as values of other properties (e.g. registration date) could be used in such a context.

In the following, a list of properties an RDAP server MAY implement is presented. The properties are divided in two groups: object common properties and object specific properties.

- Object common properties. Object common properties are derived from the merge of the "eventAction" and the "eventDate" properties. The following values of the "sort" parameter are defined:
  * registrationDate
  * reregistrationDate
  * lastChangedDate
  * expirationDate
  * deletionDate
  * reinstatiationDate
  * transferDate
  * lockedDate
  * unlockedDate

- Object specific properties. With regard to the specific properties, some of them are already defined among the query paths. In the following a list of possible sorting properties, grouped by objects, is shown:
  * Domain: name
  * Nameserver: name, ipV4, ipV6.
The correspondence between the sorting properties and the RDAP fields is shown in Table 1:

<table>
<thead>
<tr>
<th>Object class</th>
<th>Sorting property</th>
<th>RDAP property</th>
<th>RFC 7483</th>
<th>RFC 6350</th>
<th>RFC 8605</th>
</tr>
</thead>
<tbody>
<tr>
<td>Searchable objects properties</td>
<td>Common properties</td>
<td>eventAction values suffixed by &quot;Date&quot;</td>
<td>4.5.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Domain</td>
<td>name</td>
<td>unicodeName/ldhName</td>
<td>5.3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nameserver</td>
<td>name</td>
<td>unicodeName/ldhName</td>
<td>5.2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ipV4</td>
<td>v4 ipAddress</td>
<td>5.2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ipV6</td>
<td>v6 ipAddress</td>
<td>5.2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entity</td>
<td>handle</td>
<td>handle</td>
<td>5.1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fn</td>
<td>vcard fn</td>
<td>5.1.0</td>
<td>6.2.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>org</td>
<td>vcard org</td>
<td>5.1.0</td>
<td>6.6.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>voice</td>
<td>vcard tel with type=&quot;voice&quot;</td>
<td>5.1.0</td>
<td>6.4.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>email</td>
<td>vcard email</td>
<td>5.1.0</td>
<td>6.4.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>country</td>
<td>country name in vcard adr</td>
<td>5.1.0</td>
<td>6.3.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cc</td>
<td>country code in vcard adr</td>
<td>5.1.0</td>
<td></td>
<td>3.1.0</td>
</tr>
<tr>
<td></td>
<td>city</td>
<td>locality in vcard adr</td>
<td>5.1.0</td>
<td>6.3.1</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Sorting properties definition

With regard to the definitions in Table 1, some further considerations must be made to disambiguate some cases:

- Since the response to a search on either domains or nameservers might include both A-labels and U-labels ([RFC5890]) in general, a consistent sorting policy shall take the unicodeName and ldhName as two formats of the same value rather than separately. Therefore, the unicodeName value MUST be taken while sorting, when unicodeName is missing, the value of ldhName MUST be considered instead;

- The jCard "sort-as" parameter MUST be ignored for the purpose of the sorting capability as described in this document;
- even if a nameserver can have multiple IPv4 and IPv6 addresses, the most common configuration includes one address for each IP version. Therefore, the assumption of having a single IPv4 and/or IPv6 value for a nameserver cannot be considered too stringent;

- with the exception of handle values, all the sorting properties defined for entity objects can be multivalued according to the definition of vCard as given in RFC6350 [RFC6350]. When more than a value is reported, sorting will be applied to the preferred value identified by the parameter pref="1". If the pref parameter is missing, sorting will be applied to the first value.

Each RDAP provider MAY define other sorting properties than those shown in this document.

The "jsonPath" field in the "sorting_metadata" element is used to clarify the RDAP field the sorting property refers to. The mapping between the sorting properties and the JSON Paths of the RDAP fields is shown in Table 2. The JSON Paths are provided according to the Goessner v.0.8.0 specification ([GOESSNER-JSON-PATH]):

<table>
<thead>
<tr>
<th>Object class</th>
<th>Sorting property</th>
<th>JSON Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Searchable objects</td>
<td>registrationDate</td>
<td>&quot;$.domainSearchResults[*].events[?(@.eventAction==&quot;registration&quot;)].eventDate&quot;</td>
</tr>
<tr>
<td></td>
<td>reregistrationDate</td>
<td>&quot;$.domainSearchResults[*].events[?(@.eventAction==&quot;reregistration&quot;)].eventDate&quot;</td>
</tr>
<tr>
<td></td>
<td>lastChangedDate</td>
<td>&quot;$.domainSearchResults[*].events[?(@.eventAction==&quot;lastChanged&quot;)].eventDate&quot;</td>
</tr>
<tr>
<td></td>
<td>expirationDate</td>
<td>&quot;$.domainSearchResults[*].events[?(@.eventAction==&quot;expiration&quot;)].eventDate&quot;</td>
</tr>
<tr>
<td></td>
<td>deletionDate</td>
<td>&quot;$.domainSearchResults[*].events[?(@.eventAction==&quot;deletion&quot;)].eventDate&quot;</td>
</tr>
<tr>
<td></td>
<td>reinstatementDate</td>
<td>&quot;$.domainSearchResults[*].events[?(@.eventAction==&quot;reinstatement&quot;)].eventDate&quot;</td>
</tr>
<tr>
<td></td>
<td>transferDate</td>
<td>&quot;$.domainSearchResults[*].events[?(@.eventAction==&quot;transfer&quot;)].eventDate&quot;</td>
</tr>
<tr>
<td></td>
<td>lockedDate</td>
<td>&quot;$.domainSearchResults[*].events[?(@.eventAction==&quot;locked&quot;)].eventDate&quot;</td>
</tr>
<tr>
<td></td>
<td>unlockedDate</td>
<td>&quot;$.domainSearchResults[*].events[?(@.eventAction==&quot;unlocked&quot;)].eventDate&quot;</td>
</tr>
<tr>
<td>Domain name</td>
<td></td>
<td>&quot;$.domainSearchResults[*].unicodeName&quot;</td>
</tr>
</tbody>
</table>
### Table 2: Sorting properties - JSON Path Mapping

<table>
<thead>
<tr>
<th>Property</th>
<th>Path Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nameserver</td>
<td>$.nameserverSearchResults[*].unicodeName</td>
</tr>
<tr>
<td></td>
<td>$.nameserverSearchResults[*].ipAddresses.v4[0]</td>
</tr>
<tr>
<td></td>
<td>$.nameserverSearchResults[*].ipAddresses.v6[0]</td>
</tr>
<tr>
<td>Entity</td>
<td>$.entitySearchResults[*].handle</td>
</tr>
<tr>
<td></td>
<td>$.entitySearchResults[*].vcardArray[1][?(@[0]=&quot;fn&quot;)][3]</td>
</tr>
<tr>
<td></td>
<td>$.entitySearchResults[*].vcardArray[1][?(@[0]=&quot;org&quot;)][3]</td>
</tr>
<tr>
<td></td>
<td>$.entitySearchResults[*].vcardArray[1][?(@[0]=&quot;tel&quot; &amp;&amp; @[1].type==&quot;voice&quot;)][3]</td>
</tr>
<tr>
<td></td>
<td>$.entitySearchResults[*].vcardArray[1][?(@[0]=&quot;email&quot;)][3]</td>
</tr>
<tr>
<td></td>
<td>$.entitySearchResults[*].vcardArray[1][?(@[0]=&quot;adr&quot;)][3][6]</td>
</tr>
<tr>
<td></td>
<td>$.entitySearchResults[*].vcardArray[1][?(@[0]=&quot;adr&quot;)][1].cc</td>
</tr>
<tr>
<td></td>
<td>$.entitySearchResults[*].vcardArray[1][?(@[0]=&quot;adr&quot;)][3][3]</td>
</tr>
</tbody>
</table>

#### 2.3.2. Representing Sorting Links

An RDAP server MAY use the "links" array of the "sorting_metadata" element to provide ready-made references [RFC8288] to the available sort criteria (Figure 4). Each link represents a reference to an alternate view of the results.
Figure 4: Example of a "sorting_metadata" instance to implement result sorting

2.4. "cursor" Parameter

An RDAP query could return a response with hundreds, even thousands, of objects, especially when partial matching is used. For that
reason, the cursor parameter addressing result pagination is defined
to make responses easier to handle.

Presently, the most popular methods to implement pagination in REST
API are: offset pagination and keyset pagination. Both two
pagination methods don’t require the server to handle the result set
in a storage area across the requests since a new result set is
generated each time a request is submitted. Therefore, they are
preferred in comparison to any other method requiring the management
of a REST session.

Using limit and offset operators represents the traditionally used
method to implement results pagination. Both of them can be used
individually:

- "limit": means that the server must return the first N objects of
  the result set;
- "offset": means that the server must skip the first N objects and
  must return objects starting from position N+1.

When limit and offset are used together, they allow to identify a
specific portion of the result set. For example, the pair
"offset=100,limit=50" returns first 50 objects starting from position
101 of the result set.

Despite its easiness of implementation, offset pagination raises some
well known drawbacks:

- when offset has a very high value, scrolling the result set could
take some time;
- it always requires to fetch all the rows before dropping as many
  rows as specified by offset;
- it may return inconsistent pages when data are frequently updated
  (i.e. real-time data) but this doesn’t seem the case of
  registration data.

The keyset pagination [SEEK] consists in adding a query condition
that enables the selection of the only data not yet returned. This
method has been taken as the basis for the implementation of a
"cursor" parameter [CURSOR] by some REST API providers (e.g.
[CURSOR-API1],[CURSOR-API2]). The cursor is an opaque URL-safe
string representing a logical pointer to the first result of the next
page (Figure 5).

Nevertheless, even keyset pagination can be troublesome:
o it needs at least one key field;

o it does not allow to sort just by any field because the sorting criterion must contain a key;

o it works best with full composite values support by DBMS (i.e. \([x,y]>[a,b]\)), emulation is possible but ugly and less performant;

o it does not allow to directly navigate to arbitrary pages because the result set must be scrolled in sequential order starting from the initial page;

o implementing the bi-directional navigation is tedious because all comparison and sort operations have to be reversed.

Furthermore, in the RDAP context, some additional considerations can be made:

o an RDAP object is a conceptual aggregation of information generally collected from more than one data structure (e.g. table) and this makes even harder for the developers the implementation of the keyset pagination that is already quite difficult. For example, the entity object can gather information from different data structures (registrars, registrants, contacts, resellers, and so on), each one with its own key field mapping the RDAP entity handle;

o depending on the number of the page results as well as the number and the complexity of the properties of each RDAP object in the response, the time required by offset pagination to skip the previous pages could be much faster than the processing time needed to build the current page. In fact, RDAP objects are usually formed by information belonging to multiple data structures and containing multivalued properties (i.e. arrays) and, therefore, data selection might be a time consuming process. This situation occurs even though the selection is supported by indexes;

o depending on the access levels defined by each RDAP operator, the increase of complexity and the decrease of flexibility of keyset pagination with respect to the offset pagination could be considered impractical.

Ultimately, both pagination methods have benefits and drawbacks.

That said, the cursor parameter defined in this specification can be used to encode information about any pagination method. For example, in the case of a simple implementation of the cursor parameter to
represent offset pagination information, the cursor value "b2Zmc2V0PTEwMCxsaW1pdD01MAo=" is the mere Base64 encoding of "offset=100,limit=50". Likewise, in a simple implementation to represent keyset pagination information, the cursor value "a2V5PXRoZWxhc3Rkb21haW5vZnRoZXBlZ2UyY29t=" represents the mere Base64 encoding of "key=thelastdomainofthepage.com" where the key value identifies the last row of the current page.

This solution lets RDAP providers to implement a pagination method according to their needs, the user access levels, the submitted queries. In addition, servers can change the method over time without announcing anything to the clients.

The ABNF syntax of the cursor parameter is the following:

```
cursor = "cursor=" 1*( ALPHA / DIGIT / "/" / ":=" / "-" / "_" )
```

https://example.com/rdap/domains?name=*nr.com
&cursor=wJlCDLl6KTwypN7t6v6nWEEmEy9Hjf1XY1xmqV-M=

Figure 5: An example of RDAP query reporting the "cursor" parameter

2.4.1. Representing Paging Links

An RDAP server SHOULD use the "links" array of the "paging_metadata" element to provide a ready-made reference [RFC8288] to the next page of the result set (Figure 6). Examples of additional "rel" values a server MAY implements are "first", "last", "prev".

Figure 6: Example of a "paging_metadata" instance to implement cursor pagination

3. Negative Answers

The value constraints for the parameters are defined by their ABNF syntax. Therefore, each request including an invalid value for a parameter SHOULD obtain an HTTP 400 (Bad Request) response code. The same response SHOULD be returned in the following cases:

- if the client provides an unsupported value for the "sort" parameter in both single and multi sort;
- if the client submits an invalid value for the "cursor" parameter.
Optionally, the response MAY include additional information regarding the negative answer in the HTTP entity body.

4. RDAP Conformance

Servers returning the "paging_metadata" element in their response MUST include "paging_level_0" in the rdapConformance array as well as servers returning the "sorting_metadata" element MUST include "sorting_level_0".

5. Implementation Considerations

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

Similar operators are completely or partially supported by the most known NoSQL databases (MongoDB, CouchDB, HBase, Cassandra, Hadoop) so the implementation of the new parameters seems to be practicable by servers working without the use of an RDBMS.

6. Implementation Status

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

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942 [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

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

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

6.2. Google Registry

Responsible Organization: Google Registry
Location: https://www.registry.google/rdap/
Description: This implementation includes support for RDAP queries for TLDs such as .google, .how, .soy, and .xn--q9jyb4c. The RDAP server implements cursor pagination. The link used to request the next page is included in the notice section of the response.
Level of Maturity: Production.
Coverage: This implementation includes the "cursor" parameter described in this specification.
Contact Information: Brian Mountford, mountford@google.com

7. IANA Considerations

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

Extension identifier: paging
Registry operator: Any
Published specification: This document.
Contact: IESG <iesg@ietf.org>
Intended usage: This extension describes a best practice for result set paging.

Extension identifier: sorting
Registry operator: Any
Published specification: This document.
Contact: IESG <iesg@ietf.org>
Intended usage: This extension describes a best practice for result set sorting.
8. Security Considerations

Security services for the operations specified in this document are described in RFC 7481 [RFC7481].

Search query typically requires more server resources (such as memory, CPU cycles, and network bandwidth) when compared to lookup query. This increases the risk of server resource exhaustion and subsequent denial of service due to abuse. This risk can be mitigated by either restricting search functionality and limiting the rate of search requests. Servers can also reduce their load by truncating the results in the response. However, this last security policy can result in a higher inefficiency if the RDAP server does not provide any functionality to return the truncated results.

The new parameters presented in this document provide the RDAP operators with a way to implement a secure server without penalizing its efficiency. The "count" parameter gives the user a measure to evaluate the query precision and, at the same time, returns a significant information. The "sort" parameter allows the user to obtain the most relevant information at the beginning of the result set. In both cases, the user doesn’t need to submit further unnecessary search requests. Finally, the "cursor" parameter enables the user to scroll the result set by submitting a sequence of sustainable queries according to the server limits.

9. Acknowledgements

The authors would like to acknowledge Brian Mountford for his contribution to the development of this document.

10. References

10.1. Normative References


10.2. Informative References


Appendix A. Change Log

00: Initial working group version ported from draft-loffredo-regext-rdap-sorting-and-paging-05

01: Removed both "offset" and "nextOffset" to keep "paging_metadata" consistent between the pagination methods. Renamed "Considerations about Paging Implementation" section in "cursor" Parameter. Removed "FOR DISCUSSION" items. Provided a more detailed description of both "sorting_metadata" and "paging_metadata" objects.

02: Removed both "offset" and "limit" parameters. Added ABNF syntax of cursor parameter. Rearranged the layout of some sections. Removed some items from "Informative References" section. Changed "IANA Considerations" section.

03: Added "cc" to the list of sorting properties in "Sorting Properties Declaration" section. Added RFC8605 to the list of "Informative References".

04: Replaced "ldhName" with "name" in the "Sorting Properties Declaration" section. Clarified the sorting logic with respect the JSON value types and the sorting policy for multivalued fields.

Authors' Addresses