Workgroup:	rpp
Internet-Draft:	draft-simmen-rpp-dns-data-00
Published:	7 July 2025
Intended Status:	Informational
Expires:	8 January 2026

Authors: C. Simmen, *DENIC eG*; P. Kowalik, *DENIC eG*

DNS data representation for use in RESTful Provisioning Protocol (RPP)

Abstract

This document proposes a unified, extensible JSON representation for DNS resource records for use in the RESTful Provisioning Protocol (RPP). The aim is to create a single, consistent structure for provisioning all DNS-related data - including delegation, DNSSEC, and other record types - that directly mirrors the DNS data model and being mappable to existing EPP model of requests and responses same time. This approach simplifies the adoption of both current and future DNS features by aligning the provisioning format with the target system, thereby streamlining the management of domain names and related objects within RPP.

The RFC Editor will remove this note

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://github.com/christian-simmen/draft-simmen-rpp-dns-data. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-simmen-rpp-dns-data/.

Discussion of this document takes place on the WG Working Group mailing list (mailto:rpp@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/rpp/. Subscribe at https://www.ietf.org/mailman/listinfo/rpp/.

Source for this draft and an issue tracker can be found at https://github.com/christian-simmen/draft-simmen-rpp-dns-data.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Extensible Provisioning Protocol (EPP) manages DNS delegation data using distinct object types and extensions. Host Objects [RFC5732] are used for name servers (NS records) and their associated addresses (glue A/AAAA records), while DNSSEC data is handled via a separate security extension [RFC5910]. Name server information can be also directly attached to a domain name as a set of Host Attributes [RFC5731]. More recently, control over Time-to-Live (TTL) values was added through another extension [RFC9803].

While functional, this segmented approach creates complexity. The DNS landscape itself is evolving, with new transport protocols like DNS-over-HTTPS [RFC8484] and DNS-over-QUIC [RFC9250] driving the need for more sophisticated delegation information, such as the proposed DELEG record type [I-D.draft-ietf-deleg].

Some registry operators have developed their own proprietary solutions. These include grouping name servers into "sets" for easier management or allowing domains to be provisioned with arbitrary DNS resource records without formal delegation, which is expanding on Host Attribute model with other Resource Record types.

The development of the RESTful Provisioning Protocol (RPP) provides an opportunity to address this fragmentation. This document proposes a unified data representation for all DNS-related information, specified in a format that directly mirrors DNS resource records. This approach is not intended to influence existing registry data models, but rather to offer a flexible and consistent structure for the data in the protocol. By unifying the representation of delegation data (NS, A/AAAA glue), DNSSEC information, and other record types, this model can be applied across various contexts. It is designed to be equally applicable whether a registry uses separate host objects, host attributes within a domain, or more abstract concepts like name server sets, thereby simplifying implementation and ensuring adaptability for future developments in the DNS.

2. Domain Names in DNS

DNS domain names are hierarchically ordered label separated by a dot ".". Each label represent the delegation of a subordinate namespace or a host name. DNS resource records [RFC1035] are expressed as a dataset containing:

"NAME" "CLASS" "TYPE" "TTL" "RDLENGTH" "RDATA"

A set of resource records describes the behavior of a namespace. Each resource record shares the same top level format.

NAME The owner name of the DNS entry which **MAY** be the domain itself or a subordinate hostname.

CLASS The RR CLASS

TYPE The RR TYPE of data present in the RDATA field.

TTL Time interval a RR may be cached by name servers

RDLENGTH The length of the RDATA field. RDLENGTH will be safely ignored in RPP

RDATA The actual payload data. Structures defer for each type.

3. JSON representation

3.1. Rules

3.1.1. DNS data extending an domain object

Delegation data, as well as DNSSEC data, is intended to find it's way into the parent side DNS servers. Because of the strong connection to the provisioned domain object and DNS servers both aspects should be visible in the RPP data model. Therefore the domain object is extended by an array of DNS entries. The properties of an object in this array **MUST** be a representation of the top level format as described in section 3.2.1 of [RFC1035]. All keys **MUST** be lowercase. Whitespaces **MUST** be translated to underscores ("_").

```
{
  "@type": "Domain",
  "name": "example.com",
  "dns": [
    {
      "name": "",
      "class": "",
      "type": "",
      "ttl": "",
      "rdata": {}
    }
  ]
}
```

3.1.2. DNS record structure representation

3.1.2.1. name

The owner name of the DNS entry which **MAY** be the domain itself or a subordinate hostname. A server **MUST NOT** accept a NAME which is not a subordinate label to the provisioned domain name.

A server **MUST** accept values as "@", "relative names" and fully qualified domain names (FQDN).

"@" **MUST** be interpreted as the provisioned domain name.

"relative names" **MUST** be appended by the server with the provisioned domain name.

"FQDN" identified by a trailing dot (".") **MUST NOT** be interpreted by the server. A server **MUST** check if the provided name is a subordinate to the provisioned domain, or the domain itself.

Example:

```
    {
      "@type": "Domain",
      "name": "example.com",
      "dns": [
        {
          "name": "@",
          "type": "a",
          "rdata": {
            "address": "192.0.2.1"
          }
        },
        {
          "name": "www",
          "type": "a",
          "rdata": {
            "address": "192.0.2.2"
          }
        },
        {
          "name": "web.example.com.",
          "type": "a",
          "rdata": {
            "address": "192.0.2.3"
          }
        }
      ]
    }
```

would imply three resulting records: An A RR for "example.com" ("@") set to 192.0.2.1. An A RR for "www.example.com" ("www" relative) set to 192.0.2.2. An A RR for "web.example.com" (FQDN) set to 192.0.2.3.

3.1.2.2. class

A client **SHOULD** omit the class. The server **MUST** assume "IN" as class of a transferred dataset and **MAY** decline other values. If present the value **MUST** be chosen from section 3.2.4. CLASS values of [RFC1035].

3.1.2.3. type

The TYPE of data present in the RDATA. This also implies the expected fields in RDATA. If present the value **MUST** chosen from section 3.2.2. TYPE values of [RFC1035] or other RFC describing the RR type. Valuse **MUST** be converted to lower case.

3.1.2.4. ttl

TTL is considered a operational control (see section 3.1.3 Operational controls and section 4.3.1 TTL). A server **MUST** set a default value as TTL and **MAY** ignore other values send by a client.

3.1.2.5. rdlength

RDLENGTH specifies the length of the RDATA field and will be ignored in RPP. A client **MUST NOT** include this field. A server **MUST** ignore this field if present.

3.1.2.6. rdata

The RDATA structure depends on the TYPE and **MUST** be expressed as a JSON object. Property names **MUST** follow the definition of the RDATA described by the corresponding RFC. Property names **MUST** be translated to lowercase. Whitespaces **MUST** be translated to underscores ("_").

Example: Section 3.3.11 NS RDATA format of [RFC1035] describes the RDATA of a NS RR as "NSDNAME". Section 3.3.9 MX RDATA format of [RFC1035] describes the RDATA of a MX RR as "PREFERENCE", "EXCHANGE". The resulting structure is therefore:

```
{
  "@type": "Domain",
  "name": "example.com",
  "dns": [
    {
      "name": "@",
      "type": "ns",
      "rdata": {
        "nsdname": "ns1.example.net."
      }
    },
    {
      "name": "@",
      "type": "mx",
      "rdata": {
        "preference": "10",
        "exchange": "mx1.example.net"
      }
    }
  ]
}
```

3.1.3. Operational controls

In addition to the regular data a server **MAY** allow a client to control specific operational behavior. A client **MAY** add an JSON object with a number of "dns_controls" to the domain object.

```
{
  "@type": "Domain",
  "name": "example.com",
  "dns": [
    {
      "name": "<name>",
      "type": "<type>",
      "rdata": {
        "rdata_key": "<rdata_value>"
      }
    }
  ],
  "dns_controls": {
    "<named_control>": "<named_control_value>"
  }
}
```

3.1.4. Future DNS record types

With respect to an evolving DNS landscape new record types - including delegation - may emerge. Usually these record type will be defined and standardized for the DNS in first. Adopting future record types **MUST** be done using the rules described in section 3.1.2.6 of this document.

4. Use cases

4.1. Domain delegation (Host Attribute)

To enable domain delegation a server **MUST** support the "NS", "A" and "AAAA" record types ([RFC1035], [RFC3596]).

In this delegation model the delegation information and corresponding DNS configuration is attached directly to a domain object. This is corresponding to Host Attribute delegation model of [RFC5731].

A minimal delegation can be expressed by adding an array of name servers to the DNS data of a domain:

```
    {
      "@type": "Domain",
      "name": "example.com",
      "dns": [
        {
          "name": "@",
          "type": "ns",
          "rdata": {
            "nsdname": "ns1.example.net."
          }
        },
        {
          "name": "@",
          "type": "ns",
          "rdata": {
            "nsdname": "ns2.example.net."
          }
        }
      ]
    }
```

If GLUE records are needed the client may add records of type "A" or "AAAA" :

```
{
  "@type": "Domain",
  "name": "example.com",
  "dns": [
    {
      "name": "@",
      "type": "ns",
      "rdata": {
        "nsdname": "ns1.example.net."
      }
    },
    {
      "name": "@",
      "type": "ns",
      "rdata": {
        "nsdname": "ns.example.com"
      }
    },
    {
      "name": "ns.example.com.",
      "type": "a",
      "rdata": {
        "address": "192.0.2.1"
      }
    },
    {
      "name": "ns.example.com.",
      "type": "aaaa",
      "rdata": {
        "address": "2001:DB8::1"
      }
    }
  ]
}
```

4.2. Host Object

[RFC5731] specifies how domain delegation can be expressed as a relation to a separate provisioning object (Host Object), which carries the DNS configuration (name and glue records), with details specified in [RFC5732].

To enable specification of Host Objexts, similar to direct domain delegation, a server **MUST** support the "NS", "A" and "AAAA" record types ([RFC1035], [RFC3596]).

DNS configuration of Host Object is specified by NS, A and AAAA configuration within "dns" data structure:

```
{
  "@type": "Host",
  "name": "ns.example.com",
  "dns": [
    {
      "name": "@",
      "type": "ns",
      "rdata": {
        "nsdname": "ns.example.com"
      }
    },
    {
      "name": "ns.example.com.",
      "type": "a",
      "rdata": {
        "address": "192.0.2.1"
      }
    },
    {
      "name": "ns.example.com.",
      "type": "aaaa",
      "rdata": {
        "address": "2001:DB8::1"
      }
    }
  ]
}
```

4.3. DNSSEC

To enable DNSSEC provisioning a server **SHOULD** support either "DS" or "DNSKEY" or both record types. The records **MUST** be added to the "dns" array of the domain. If provided with only "DNSKEY" a server **MUST** calculate the DS record. If both record types are provided a server **MAY** use the DNSKEY to validate the DS record.

```
{
  "@type": "Domain",
  "name": "example.com",
  "dns": [
    {
      "name": "@",
      "type": "ns",
      "rdata": {
        "nsdname": "ns1.example.net."
      }
    },
    {
      "name": "@",
      "type": "ns",
      "rdata": {
        "nsdname": "ns2.example.net."
      }
    },
    {
      "name": "@",
      "type": "ds",
      "rdata": {
        "key_tag": 12345,
        "algorithm": 13,
        "digest_type": 2,
        "digest": "BE74359954660069D5C632B56F120EE9F3A86764247C"
      }
    }
  ]
}
```

```
{
  "@type": "Domain",
  "name": "example.com.",
  "dns": [
    {
      "name": "@",
      "type": "ns",
      "rdata": {
        "nsdname": "ns1.example.net."
      }
    },
    {
      "name": "@",
      "type": "ns",
      "rdata": {
        "nsdname": "ns2.example.net."
      }
    },
    {
      "name": "@",
      "type": "dnskey",
      "rdata": {
        "flags": 257,
        "protocol": 3,
        "algorithm": 5,
        "public_key": "AwEAAddt2AkL4RJ9Ao6LCWheg8"
      }
    }
  ]
}
```

4.4. Operational controls

4.4.1. TTL

The TTL controls the caching behavior of DNS resource records (see Section 5 of [RFC9499]). Typically a default TTL is defined by the registry operator. In some use cases it is desirable for a client to change the TTL value.

A client **MAY** assign "ttl" to the dns_controls of an RR set which is intended to be signed on the parent side. A server **MAY** ignore these values, e.g. for policy reasons.

Example:

```
    {
      "@type": "Domain",
      "name": "example.com",
      "dns": [
        {
          "name": "@",
          "type": "a",
          "rdata": {
            "address": "192.0.2.1"
          }
        },
        {
          "name": "@",
          "type": "aaaa",
          "rdata": {
            "address": "2001:DB8::1"
          }
        }
      ],
      "dns_controls": {
        "ttl": {
          "a": 86400,
          "aaaa": 3600
       }
      }
    }
```

4.4.2. Maximum signature lifetime

Maximum signature lifetime (maximum_signature_lifetime) describes the maximum number of seconds after signature generation a parents signature on signed DNS information should expire. The maximum_signature_lifetime value applies to the RRSIG resource record (RR) over the signed DNS RR. See Section 3 of [RFC4034] for information on the RRSIG resource record (RR).

A client **MAY** assign "maximum_signature_lifetime" to the dns_controls of an RR set which is intended to be signed on the parent side. A server **MAY** ignore these values, e.g. for policy reasons.

Example:

```
    {
      "@type": "Domain",
      "name": "example.com",
      "dns": [
        {
          "name": "@",
          "type": "ns",
          "rdata": {
            "nsdname": "ns1.example.net."
          }
        },
        {
          "name": "@",
          "type": "ns",
          "rdata": {
            "nsdname": "ns2.example.net."
          }
        },
        {
          "name": "@",
          "type": "ds",
          "rdata": {
            "key_tag": 12345,
            "algorithm": 13,
            "digest_type": 2,
            "digest": "BE74359954660069D5C632B56F120EE9F3A86764247C"
          }
        }
      ],
      "dns_controls": {
        "maximum_signature_lifetime": {
          "ds": 86400
        }
      }
    }
```

4.5. Authoritative DNS data

A server **MAY** support additional RR types, e.g. to support delegation-less provisioning. By doing this the registry operators name servers becomes authoritative for the registered domain. A server **MUST** consider resource records designed for delegation - including DNSSEC - and resource records representing authoritative data - except for GLUE RR - mutual exclusive.

```
{
  "@type": "Domain",
  "name": "example.com",
  "dns": [
    {
      "name": "@",
      "type": "a",
      "rdata": {
        "address": "192.0.2.1"
      }
    },
    {
      "name": "www.example.com.",
      "type": "a",
      "rdata": {
        "address": "192.0.2.1"
      }
    },
    {
      "name": "@",
      "type": "aaaa",
      "rdata": {
        "address": "2001:DB8::1"
      }
    },
    {
      "name": "www.example.com.",
      "type": "a",
      "rdata": {
        "address": "2001:DB8::1"
      }
    },
    {
      "name": "@",
      "type": "mx",
      "rdata": {
        "preference": "10",
        "exchange": "mx1.example.com"
      }
    },
    {
      "name": "mx1.example.com.",
      "type": "a",
      "rdata": {
        "address": "192.0.2.2"
      }
    },
    {
      "name": "@",
      "type": "mx",
      "rdata": {
        "preference": "20",
        "exchange": "mx2.example.net"
      }
    },
    {
      "name": "@",
      "type": "txt",
      "rdata": {
        "txt_data": "v=spf1 -all"
      }
    }
  ]
}
```

5. Discoverability

The server **MUST** provide the following information per profile in the discovery document in section 10 of [I-D.draft-ietf-rpp-requirements]: * a list of supported resource record types * a list of applicable dns_controls * minimum, maximum and default values for dns_controls

TODO: Needs rewrite after definition of the discovery document

6. EPP compatibility considerations

TODO

7. Conventions and Definitions

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

8. Security Considerations

8.1. Authoritative data

Allowing to store authoritative resource records (see section 4.4) in the registry provides faster resolution. However, if not done properly situations may occur where the data served authoritative should have been delegated. RPP servers **MUST** take precautions to not store authoritative and non-authoritative data at the same time.

The types and number of authoritative records can result in uncontrolled growth of the registries zone file and eventually exhaust the hardware resources of the registries name server. RPP servers **SHOULD** consider limiting the amount of authoritative records and carefully choose which record types are allowed.

8.2. Host references within the rdata field

Some RR types (NS, MX and others) use references to host names which can be categorized into three categories:

Domain internal references are references to a subordinate host name of the domain. E.g. "ns.example.com" is an domain internal reference when used as a name server for "example.com".

Registry internal references are references to a host name within the same registry. E.g. "ns.example.com" is an domain internal reference when used as a name server for "example2.com".

Registry external references are references to a host name outside of the registry. E.g. "ns.example.net" is an domain internal reference when used as a name server for "example.com".

Deletion of a host name while still being referenced may lead to severe security risks for the referencing domain.

9. IANA Considerations

This document has no IANA actions.

10. Appendix A. Examples from current implementations

10.1. EPP

10.1.1. Create domain using host attributes example

EPP XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <domain:create
          xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
        <domain:period unit="y">1</domain:period>
        <domain:ns>
          <domain:hostAttr>
            <domain:hostName>ns1.example.com</domain:hostName>
            <domain:hostAddr ip="v4">192.0.2.1</domain:hostAddr>
            <domain:hostAddr ip="v6">2001:db8::1</domain:hostAddr>
          </domain:hostAttr>
          <domain:hostAttr>
            <domain:hostName>ns2.example.com</domain:hostName>
            <domain:hostAddr ip="v4">192.0.2.2</domain:hostAddr>
          </domain:hostAttr>
        </domain:ns>
        <domain:registrant>registrantID</domain:registrant>
        <domain:contact type="admin">adminID</domain:contact>
        <domain:contact type="tech">techID</domain:contact>
      </domain:create>
    </create>
    <extension>
      <secDNS:create
          xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1">
        <secDNS:maxSigLife>604800</secDNS:maxSigLife>
        <secDNS:dsData>
          <secDNS:keyTag>12345</secDNS:keyTag>
          <secDNS:alg>13</secDNS:alg>
          <secDNS:digestType>2</secDNS:digestType>
          <secDNS:digest>
            BE74359954660069D5C632B56F120EE9F3A86764247
          </secDNS:digest>
        </secDNS:dsData>
      </secDNS:create>
      <ttl:create xmlns:ttl="urn:ietf:params:xml:ns:epp:ttl-1.0">
        <ttl:ttl for="NS">3600</ttl:ttl>
      </ttl:create>
    </extension>
    <clTRID>clTRID-1234</clTRID>
  </command>
</epp>
```

RPP JSON representation:

```
{
  "@type": "Domain",
  "name": "example.com",
  "...": "",
  "dns": [
    {
      "name": "@",
      "type": "ns",
      "rdata": {
        "nsdname": "ns1.example.com"
      }
    },
    {
      "name": "ns1.example.com",
      "type": "a",
      "rdata": {
        "address": "192.0.2.1"
      }
    },
    {
      "name": "ns1.example.com",
      "type": "aaaa",
      "rdata": {
        "address": "2001:db8::1"
      }
    },
    {
      "name": "@",
      "type": "ns",
      "rdata": {
        "nsdname": "ns2.example.com"
      }
    },
    {
      "name": "ns2.example.com",
      "type": "a",
      "rdata": {
        "address": "192.0.2.2"
      }
    },
    {
      "name": "@",
      "type": "ds",
      "rdata": {
        "key_tag": 12345,
        "algorithm": 13,
        "digest_type": 2,
        "digest": "BE74359954660069D5C632B56F120EE9F3A86764247"
      }
    }
  ],
  "dns_controls": {
    "maximum_signature_lifetime": {
      "ds": 604800
    },
    "ttl": {
      "ns": 3600
    }
  }
}
```

10.1.2. Create domain using host object example

EPP XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <domain:create
          xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
        <domain:period unit="y">1</domain:period>
        <domain:ns>
          <domain:hostObj>ns1.example.net</domain:hostObj>
          <domain:hostObj>ns2.example.net</domain:hostObj>
        </domain:ns>
        <domain:registrant>registrantID</domain:registrant>
        <domain:contact type="admin">adminID</domain:contact>
        <domain:contact type="tech">techID</domain:contact>
      </domain:create>
    </create>
    <extension>
      <secDNS:create
          xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1">
        <secDNS:maxSigLife>604800</secDNS:maxSigLife>
        <secDNS:dsData>
          <secDNS:keyTag>12345</secDNS:keyTag>
          <secDNS:alg>13</secDNS:alg>
          <secDNS:digestType>2</secDNS:digestType>
          <secDNS:digest>
            BE74359954660069D5C632B56F120EE9F3A86764247C
          </secDNS:digest>
        </secDNS:dsData>
      </secDNS:create>
      <ttl:create xmlns:ttl="urn:ietf:params:xml:ns:epp:ttl-1.0">
        <ttl:ttl for="NS">3600</ttl:ttl>
      </ttl:create>
    </extension>
    <clTRID>clTRID-1234</clTRID>
  </command>
</epp>
```

RPP JSON representation:

```
{
  "@type": "Domain",
  "name": "example.com",
  "...": "",
  "_object_references": {
    "nameserver": [
      {
        "name": "ns1.example.net.",
        "href": "https://rpp.example/nameservers/ns1.example.net",
        "rel": "nameserver"
      },
      {
        "name": "ns2.example.net.",
        "href": "https://rpp.example/nameservers/ns2.example.net",
        "rel": "nameserver"
      }
    ]
  },
  "dns": [
    {
      "name": "@",
      "type": "ds",
      "rdata": {
        "key_tag": 12345,
        "algorithm": 13,
        "digest_type": 2,
        "digest": "BE74359954660069D5C632B56F120EE9F3A86764247C"
      }
    }
  ],
  "dns_controls": {
    "maximum_signature_lifetime": {
      "ds": 604800
    },
    "ttl": {
      "ns": 3600
    }
  }
}
```

10.1.3. Create host object example

EPP XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <host:create
       xmlns:host="urn:ietf:params:xml:ns:host-1.0">
        <host:name>ns1.example.com</host:name>
        <host:addr ip="v4">192.0.2.2</host:addr>
        <host:addr ip="v4">192.0.2.29</host:addr>
        <host:addr ip="v6">1080:0:0:0:8:800:200C:417A</host:addr>
      </host:create>
    </create>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

RPP JSON representation:

```
{
    "@type": "Host",
    "...": "",
    "name": "ns1.example.com",
    "dns": [
        {
            "name": "@",
            "type": "a",
            "rdata": {
                "address": "192.0.2.2"
            }
        },
        {
            "name": "@",
            "type": "a",
            "rdata": {
                "address": "192.0.2.29"
            }
        },
        {
            "name": "@",
            "type": "aaaa",
            "rdata": {
                "address": "1080:0:0:0:8:800:200C:417A"
            }
        }
    ]
}
```

10.2. Free Registry for ENUM and Domains (FRED)

FRED is an open source registry software developed by CZ.NIC

10.2.1. Create domain example

EPP XML:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <domain:create
          xmlns:domain="http://www.nic.cz/xml/epp/domain-1.4">
        <domain:name>example.cz</domain:name>
        <domain:registrant>registrantID</domain:registrant>
        <domain:admin>adminID</domain:admin>
        <domain:nsset>nssetID</domain:nsset>
        <domain:keyset>keysetID</domain:keyset>
      </domain:create>
    </create>
    <clTRID>clTRID-1234</clTRID>
  </command>
</epp>
```

10.2.2. Create nsset example

EPP XML:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <nsset:create
          xmlns:nsset="http://www.nic.cz/xml/epp/nsset-1.2">
        <nsset:id>nssetID</nsset:id>
        <nsset:ns>
          <nsset:name>ns1.example.cz</nsset:name>
          <nsset:addr>192.0.2.1</nsset:addr>
          <nsset:addr>192.0.2.2</nsset:addr>
        </nsset:ns>
        <nsset:ns>
          <nsset:name>nameserver-example.cz</nsset:name>
        </nsset:ns>
        <nsset:tech>techID</nsset:tech>
        <nsset:reportlevel>1</nsset:reportlevel>
      </nsset:create>
    </create>
    <clTRID>clTRID-1234</clTRID>
  </command>
</epp>
```

10.2.3. Create keyset example

EPP XML:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <keyset:create
          xmlns:keyset="http://www.nic.cz/xml/epp/keyset-1.3">
        <keyset:id>keysetID</keyset:id>
        <keyset:dnskey>
          <keyset:flags>257</keyset:flags>
          <keyset:protocol>3</keyset:protocol>
          <keyset:alg>5</keyset:alg>
          <keyset:pubKey>AwEAAddt2AkL4RJ9Ao6LCWheg8</keyset:pubKey>
        </keyset:dnskey>
        <keyset:dnskey>
          <keyset:flags>257</keyset:flags>
          <keyset:protocol>3</keyset:protocol>
          <keyset:alg>5</keyset:alg>
          <keyset:pubKey>AwEAAddt2AkL4RJ9Ao6LCWheg9</keyset:pubKey>
        </keyset:dnskey>
        <keyset:tech>techID</keyset:tech>
      </keyset:create>
    </create>
    <clTRID>clTRID-1234</clTRID>
  </command>
</epp>
```

RPP JSON representation:

TODO

```
{}
```

10.3. Realtime Registry Interface (RRI)

RRI is a proprietary protocol developed by DENIC

10.3.1. Create domain with name servers example

RRI XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<registry-request
    xmlns="http://registry.denic.de/global/5.0"
    xmlns:domain="http://registry.denic.de/domain/5.0"
    xmlns:dnsentry="http://registry.denic.de/dnsentry/5.0">
  <domain:create>
    <domain:handle>example.de</domain:handle>
    <domain:contact role="holder">registrantID</domain:contact>
    <dnsentry:dnsentry xsi:type="dnsentry:NS">
      <dnsentry:owner>example.de</dnsentry:owner>
      <dnsentry:rdata>
        <dnsentry:nameserver>ns1.example.com</dnsentry:nameserver>
      </dnsentry:rdata>
    </dnsentry:dnsentry>
    <dnsentry:dnsentry xsi:type="dnsentry:NS">
      <dnsentry:owner>example.de</dnsentry:owner>
      <dnsentry:rdata>
        <dnsentry:nameserver>ns1.example.de</dnsentry:nameserver>
        <dnsentry:address>192.0.2.1</dnsentry:address>
      </dnsentry:rdata>
    </dnsentry:dnsentry>
    <dnsentry:dnsentry xsi:type="dnsentry:DNSKEY">
      <dnsentry:owner>example.de.</dnsentry:owner>
      <dnsentry:rdata>
        <dnsentry:flags>257</dnsentry:flags>
        <dnsentry:protocol>3</dnsentry:protocol>
        <dnsentry:algorithm>5</dnsentry:algorithm>
        <dnsentry:publicKey>
          AwEAAddt2AkL4RJ9Ao6LCWheg8
        </dnsentry:publicKey>
      </dnsentry:rdata>
    </dnsentry:dnsentry>
  </domain:create>
  <ctid>clTRID-1234</ctid>
</registry-request>
```

RPP JSON representation:

```
{
  "@type": "Domain",
  "name": "example.de",
  "...": "",
  "dns": [
    {
      "name": "@",
      "type": "ns",
      "rdata": {
        "nsdname": "ns1.example.com"
      }
    },
    {
      "name": "@",
      "type": "ns",
      "rdata": {
        "nsdname": "ns1.example.de"
      }
    },
    {
      "name": "ns1.example.de",
      "type": "a",
      "rdata": {
        "address": "192.0.2.1"
      }
    },
    {
      "name": "@",
      "type": "dnskey",
      "rdata": {
        "flags": 257,
        "protocol": 3,
        "algorithm": 5,
        "public_key": "AwEAAddt2AkL4RJ9Ao6LCWheg8"
      }
    }
  ]
}
```

10.3.2. Create domain without delegation example

RRI XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<registry-request
    xmlns="http://registry.denic.de/global/5.0"
    xmlns:domain="http://registry.denic.de/domain/5.0"
    xmlns:dnsentry="http://registry.denic.de/dnsentry/5.0">
  <domain:update>
    <domain:handle>example.de</domain:handle>
    <domain:contact role="holder">registrantID</domain:contact>
    <dnsentry:dnsentry xsi:type="dnsentry:A">
      <dnsentry:owner>example.de</dnsentry:owner>
      <dnsentry:rdata>
        <dnsentry:address>192.0.2.1</dnsentry:address>
      </dnsentry:rdata>
    </dnsentry:dnsentry>
  </domain:update>
  <ctid>clTRID-1234</ctid>
</registry-request>
```

RPP JSON representation:

```
{
  "@type": "Domain",
  "name": "example.de",
    "...": "",
  "dns": [
    {
      "name": "@",
      "type": "a",
      "rdata": {
        "address": "192.0.2.1"
      }
    }
  ]
}
```

10.4. RDAP

10.4.1. Domain object

Registration Data Access Protocol (RDAP) is described in [RFC9083]. An extention proposing Time-to-Live (TTL) values is described in [I-D.draft-brown-rdap-ttl-extension] and is close to adoption in the regext working group.

RDAP JSON:

```
{
  "objectClassName": "domain",
  "ldhName": "example.com",
  "nameservers": [
    {
      "objectClassName": "nameserver",
      "ldhName": "ns1.example.com",
      "ipAddresses": {
        "v4": ["192.0.2.1"],
        "v6": ["2001:db8::1"]
      }
    },
    {
      "objectClassName": "nameserver",
      "ldhName": "ns2.example.com",
      "ipAddresses": {
        "v4": ["192.0.2.2"]
      }
    }
  ],
  "secureDNS": {
    "delegationSigned": true,
    "maxSigLife": 604800,
    "dsData": [
      {
        "keyTag": 12345,
        "algorithm": 13,
        "digestType": 2,
        "digest": "BE74359954660069D5C632B56F120EE9F3A86764247C"
      }
    ]
  },
  "ttl": [
       {
         "types": [ "NS" ],
         "value": 3600
       }
  ],
  "events": [
    {
      "eventAction": "registration",
      "eventDate": "2025-01-01T00:00:00Z"
    },
    {
      "eventAction": "expiration",
      "eventDate": "2035-01-01T00:00:00Z"
    }
  ],
  "status": ["active"]
}
```

11. References

11.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/rfc/rfc1035>.

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

[RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", STD 88, RFC 3596, DOI 10.17487/RFC3596, October 2003, <https://www.rfc-editor.org/rfc/rfc3596>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/rfc/rfc4034>.

[RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <https://www.rfc-editor.org/rfc/rfc5731>.

[RFC5732] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Host Mapping", STD 69, RFC 5732, DOI 10.17487/RFC5732, August 2009, <https://www.rfc-editor.org/rfc/rfc5732>.

[RFC5910] Gould, J. and S. Hollenbeck, "Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)", RFC 5910, DOI 10.17487/RFC5910, May 2010, <https://www.rfc-editor.org/rfc/rfc5910>.

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

[RFC9083] Hollenbeck, S. and A. Newton, "JSON Responses for the Registration Data Access Protocol (RDAP)", STD 95, RFC 9083, DOI 10.17487/RFC9083, June 2021, <https://www.rfc-editor.org/rfc/rfc9083>.

[RFC9803] Brown, G., "Extensible Provisioning Protocol (EPP) Mapping for DNS Time-to-Live (TTL) Values", RFC 9803, DOI 10.17487/RFC9803, June 2025, <https://www.rfc-editor.org/rfc/rfc9803>.

11.2. Informative References

[I-D.draft-brown-rdap-ttl-extension] Brown, G., "RDAP Extension for DNS Time-To-Live (TTL Values)", Work in Progress, Internet-Draft, draft-brown-rdap-ttl-extension-03, 17 June 2025, <https://datatracker.ietf.org/doc/html/draft-brown-rdap-ttl-extension-03>.

[I-D.draft-ietf-deleg] April, T., Špaček, P., Weber, R., and Lawrence, "Extensible Delegation for DNS", Work in Progress, Internet-Draft, draft-ietf-deleg-01, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-deleg-01>.

[I-D.draft-ietf-rpp-requirements] Wullink, M. and P. Kowalik, "RESTful Provisioning Protocol (RPP) - Requirements", Work in Progress, Internet-Draft, draft-ietf-rpp-requirements-01, 30 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rpp-requirements-01>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/rfc/rfc8484>.

[RFC9250] Huitema, C., Dickinson, S., and A. Mankin, "DNS over Dedicated QUIC Connections", RFC 9250, DOI 10.17487/RFC9250, May 2022, <https://www.rfc-editor.org/rfc/rfc9250>.

[RFC9499] Hoffman, P. and K. Fujiwara, "DNS Terminology", BCP 219, RFC 9499, DOI 10.17487/RFC9499, March 2024, <https://www.rfc-editor.org/rfc/rfc9499>.

Acknowledgments

Authors' Addresses

Christian Simmen
DENIC eG
Germany
Email: simmen@denic.de

Pawel Kowalik
DENIC eG
Germany
Email: pawel.kowalik@denic.de