Internet Engineering Task Force (IETF)
Request for Comments: 7483
Category: Standards Track
ISSN: 2070-1721

A. Newton
ARIN
S. Hollenbeck
Verisign Labs

March 2015

JSON Responses for the Registration Data Access Protocol (RDAP)

Abstract

This document describes JSON data structures representing registration information maintained by Regional Internet Registries (RIRs) and Domain Name Registries (DNRs). These data structures are used to form Registration Data Access Protocol (RDAP) query responses.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7483.

Table of Contents

1. Introduction

This document describes responses in the JSON [RFC7159] format for the queries as defined by the Registration Data Access Protocol Query Format [RFC7482]. A communication protocol for exchanging queries and responses is described in [RFC7480].

1.1. Terminology and Definitions

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

The following list describes terminology and definitions used throughout this document:

DNR: Domain Name Registry

LDH: letters, digits, hyphen

member: data found within an object as defined by JSON [RFC7159].

object: a data structure as defined by JSON [RFC7159].

object class: the definition of members that may be found in JSON objects described in this document.

object instance: an instantiation or specific instance of an object class.

RDAP: Registration Data Access Protocol

RIR: Regional Internet Registry

1.2. Data Model

The data model for JSON responses is specified in five sections:

1. simple data types conveyed in JSON strings

2. data structures specified as JSON arrays or objects that are used repeatedly when building up larger objects

3. object classes representing structured data corresponding to a lookup of a single object

4. arrays of objects representing structured data corresponding to a search for multiple objects

5. the response to an error

The object classes represent responses for two major categories of data: responses returned by RIRs for registration data related to IP addresses, reverse DNS names, and Autonomous System numbers and responses returned by DNRs for registration data related to forward DNS names. The following object classes are returned by both RIRs and DNRs:

1. domains

2. nameservers

3. entities

The information served by both RIRs and DNRs for these object classes overlap extensively and are given in this document as a unified model for both classes of service.

In addition to the object classes listed above, RIRs also serve the following object classes:

1. IP networks

2. Autonomous System numbers

Object classes defined in this document represent a minimal set of what a compliant client/server needs to understand to function correctly; however, some deployments may want to include additional object classes to suit individual needs. Anticipating this need for extension, Section 2.1 of this document defines a mechanism for extending the JSON objects that are described in this document.

Positive responses take two forms. A response to a lookup of a single object in the registration system yields a JSON object, which is the subject of the lookup. A response to a search for multiple objects yields a JSON object that contains an array of JSON objects that are the subject of the search. In each type of response, other data structures are present within the topmost JSON object.

2. Use of JSON

2.1. Naming

Clients of these JSON responses SHOULD ignore unrecognized JSON members in responses. Servers can insert members into the JSON responses, which are not specified in this document, but that does not constitute an error in the response. Servers that insert such unspecified members into JSON responses SHOULD have member names prefixed with a short identifier followed by an underscore followed by a meaningful name. It has been observed that these short identifiers aid software implementers with identifying the specification of the JSON member, and failure to use one could cause an implementer to assume the server is erroneously using a name from this specification. This allowance does not apply to jCard [RFC7095] objects. The full JSON name (the prefix plus the underscore plus the meaningful name) SHOULD adhere to the character and name limitations of the prefix registry described in [RFC7480]. Failure to use these limitations could result in slower adoption as these limitations have been observed to aid some client programming models.

Consider the following JSON response with JSON members, all of which are specified in this document.

```
{
  "handle" : "ABC123",
  "remarks" :
  [
    {
      "description" :
      [
        "She sells sea shells down by the sea shore.",
        "Originally written by Terry Sullivan."
      ]
    }
  ]
}
```

Figure 1

If The Registry of the Moon desires to express information not found in this specification, it might select "lunarNic" as its identifying prefix and insert, as an example, the member named "lunarNic_beforeOneSmallStep" to signify registrations occurring before the first moon landing and the member named "lunarNic_harshMistressNotes" that contains other descriptive text.

Consider the following JSON response with JSON names, some of which should be ignored by clients without knowledge of their meaning.

```
{
  "handle" : "ABC123",
  "lunarNic_beforeOneSmallStep" : "TRUE THAT!",
  "remarks" :
  [
    {
      "description" :
      [
        "She sells sea shells down by the sea shore.",
        "Originally written by Terry Sullivan."
      ]
    }
  ],
  "lunarNic_harshMistressNotes" :
  [
    "In space,",
    "nobody can hear you scream."
  ]
}
```

Figure 2

Insertion of unrecognized members ignored by clients may also be used for future revisions to this specification.

Clients processing JSON responses need to be prepared for members representing registration data specified in this document to be absent from a response. In other words, servers are free to not include JSON members containing registration data based on their own policies.

Finally, all JSON names specified in this document are case sensitive. Both servers and clients MUST transmit and process them using the specified character case.

3. Common Data Types

JSON [RFC7159] defines the data types of a number, character string, boolean, array, object, and null. This section describes the semantics and/or syntax reference for common, JSON character strings used in this document.

handle: DNRs and RIRs have registry-unique identifiers that may be used to specifically reference an object instance. The semantics of this data type as found in this document are to be a registry-unique reference to the closest enclosing object where the value is found. The data type names "registryId", "roid", "nic-handle", "registrationNo", etc., are terms often synonymous with this data type. In this document, the term "handle" is used. The term exposed to users by clients is a presentation issue beyond the scope of this document.

IPv4 addresses: The representation of IPv4 addresses in this document uses the dotted-decimal notation. An example of this textual representation is "192.0.2.0".

IPv6 addresses: The representation of IPv6 addresses in this document follow the forms outlined in [RFC5952]. An example of this textual representation is "2001:db8::1:0:0:1".

country codes: Where the identity of a geopolitical nation or country is needed, these identities are represented with the alpha-2 or two-character country code designation as defined in [ISO.3166.1988]. The alpha-2 representation is used because it is freely available, whereas the alpha-3 and numeric-3 standards are not.

LDH names: Textual representations of DNS names where the labels of the domain are all "letters, digits, hyphen" labels as described by [RFC5890]. Trailing periods are optional.

Unicode names: Textual representations of DNS names where one or more of the labels are U-labels as described by [RFC5890]. Trailing periods are optional.

dates and times: The syntax for values denoting dates and times is defined in [RFC3339].

URIs: The syntax for values denoting a Uniform Resource Identifier (URI) is defined by [RFC3986].

Contact information is defined using jCards as described in [RFC7095].

4. Common Data Structures

This section defines common data structures used in responses and object classes.

4.1. RDAP Conformance

The data structure named "rdapConformance" is an array of strings, each providing a hint as to the specifications used in the construction of the response. This data structure appears only in the topmost JSON object of a response.

An example rdapConformance data structure:

```
"rdapConformance" :
[
  "rdap_level_0"
]
```

Figure 3

The string literal "rdap_level_0" signifies conformance with this specification. When custom JSON values are inserted into responses, conformance to those custom specifications MUST use a string prefixed with the appropriate identifier from the IANA RDAP Extensions registry specified in [RFC7480]. For example, if the fictional Registry of the Moon wants to signify that their JSON responses are conformant with their registered extensions, the string used might be "lunarNIC_level_0". These prefixes aid the identification of specifications for software implementers, and failure to use them could result in slower adoption of extensions.

Example rdapConformance structure with custom extensions noted:

```
"rdapConformance" :
[
  "rdap_level_0",
  "lunarNic_level_0"
]
```

Figure 4

4.2. Links

The "links" array is found in data structures to signify links to other resources on the Internet. The relationship of these links is defined by the IANA registry described by [RFC5988].

The following is an example of the link structure:

```
{
  "value" : "http://example.com/context_uri",
  "rel" : "self",
  "href" : "http://example.com/target_uri",
  "hreflang" : [ "en", "ch" ],
  "title" : "title",
  "media" : "screen",
  "type" : "application/json"
}
```

Figure 5

The JSON name/values of "rel", "href", "hreflang", "title", "media", and "type" correspond to values found in Section 5 of [RFC5988]. The "value" JSON value is the context URI as described by [RFC5988]. The "href" JSON value MUST be specified. All other JSON values are OPTIONAL.

This is an example of the "links" array as it might be found in an object class:

```
"links" :
[
    {
      "value" : "http://example.com/ip/2001:db8::123",
      "rel" : "self",
      "href" : "http://example.com/ip/2001:db8::123",
      "type" : "application/rdap+json"
    },
    {
      "value" : "http://example.com/ip/2001:db8::123",
      "rel" : "up",
      "href" : "http://example.com/ip/2001:db8::/48",
      "type" : "application/rdap+json"
    }

]
```

Figure 6

4.3. Notices and Remarks

The "notices" and "remarks" data structures take the same form. The notices structure denotes information about the service providing RDAP information and/or information about the entire response, whereas the remarks structure denotes information about the object class that contains it (see Section 5 regarding object classes).

Both are arrays of objects. Each object contains an optional "title" string representing the title of the object, an optional "type" string denoting a registered type of remark or notice (see Section 10.2.1), an array of strings named "description" for the purposes of conveying any descriptive text, and an optional "links" array as described in Section 4.2.

An example of the notices data structure:

```
"notices" :
[
  {
    "title" : "Terms of Use",
    "description" :
    [
      "Service subject to The Registry of the Moon’s TOS.",
      "Copyright (c) 2020 LunarNIC"
    ],
    "links" :
    [
      {
        "value" : "http://example.net/entity/XXXX",
        "rel" : "alternate",
        "type" : "text/html",
        "href" : "http://www.example.com/terms_of_use.html"
      }
    ]
  }
]
```

Figure 7

It is the job of the clients to determine line breaks, spacing, and display issues for sentences within the character strings of the "description" array. Each string in the "description" array contains a single complete division of human-readable text indicating to clients where there are semantic breaks.

An example of the remarks data structure:

```
"remarks" :
[
  {
    "description" :
    [
      "She sells sea shells down by the sea shore.",
      "Originally written by Terry Sullivan."
    ]
  }
]
```

Figure 8

Note that objects in the "remarks" array may also have a "links" array.

While the "title" and "description" fields are intended primarily for human consumption, the "type" string contains a well-known value to be registered with IANA (see Section 10.2.1) for programmatic use.

An example of the remarks data structure:

```
"remarks" :
[
  {
    "type" : "object truncated due to authorization",
    "description" :
    [
      "Some registration data may not have been given.",
      "Use proper authorization credentials to see all of it."
    ]
  }
]
```

Figure 9

While the "remarks" array will appear in many object classes in a response, the "notices" array appears only in the topmost object of a response.

4.4. Language Identifier

This data structure consists solely of a name/value pair, where the name is "lang" and the value is a string containing a language identifier as described in [RFC5646].

```
"lang" : "mn-Cyrl-MN"
```

Figure 10

The "lang" attribute may appear anywhere in an object class or data structure except for in jCard objects.

4.5. Events

This data structure represents events that have occurred on an instance of an object class (see Section 5 regarding object classes).

This is an example of an "events" array.

```
"events" :
[
  {
    "eventAction" : "registration",
    "eventActor" : "SOMEID-LUNARNIC",
    "eventDate" : "1990-12-31T23:59:59Z"
  },
  {
    "eventAction" : "last changed",
    "eventActor" : "OTHERID-LUNARNIC",
    "eventDate" : "1991-12-31T23:59:59Z"
  }
]
```

Figure 11

The "events" array consists of objects, each with the following members:

- "eventAction" -- a string denoting the reason for the event
- "eventActor" -- an optional identifier denoting the actor responsible for the event
- "eventDate" -- a string containing the time and date the event occurred.
- "links" -- see Section 4.2

Events can be future dated. One use case for future dating of events is to denote when an object expires from a registry.

The "links" array in this data structure is provided for references to the event actor. In order to reference an RDAP entity, a "rel" of "related" and a "type" of "application/rdap+json" is used in the link reference.

See Section 10.2.3 for a list of values for the "eventAction" string. See Appendix B regarding the various ways events can be modeled.

4.6. Status

This data structure, named "status", is an array of strings indicating the state of a registered object (see Section 10.2.2 for a list of values).

4.7. Port 43 WHOIS Server

This data structure, a member named "port43", is a simple string containing the fully qualified host name or IP address of the WHOIS [RFC3912] server where the containing object instance may be found. Note that this is not a URI, as there is no WHOIS URI scheme.

4.8. Public IDs

This data structure maps a public identifier to an object class. It is named "publicIds" and is an array of objects, with each object containing the following members:

- type -- a string denoting the type of public identifier
- identifier -- a public identifier of the type denoted by "type"

The following is an example of a publicIds structure.

```
"publicIds":
[
  {
    "type":"IANA Registrar ID",
    "identifier":"1"
  }
]
```

Figure 12

4.9. Object Class Name

This data structure, a member named "objectClassName", gives the object class name of a particular object as a string. This identifies the type of object being processed. An objectClassName is REQUIRED in all RDAP response objects so that the type of the object can be interpreted.

4.10. An Example

This is an example response with both rdapConformance and notices embedded:

```
{
  "rdapConformance" :
  [
    "rdap_level_0"
  ],
  "notices" :
  [
    {
      "title" : "Content Removed",
      "description" :
      [
        "Without full authorization, content has been removed.",
        "Sorry, dude!"
      ],
      "links" :
      [
        {
          "value" : "http://example.net/ip/192.0.2.0/24",
          "rel" : "alternate",
          "type" : "text/html",
          "href" : "http://www.example.com/redaction_policy.html"
        }
      ]
    }
  ],
  "lang" : "en",
  "objectClassName" : "ip network",
  "startAddress" : "192.0.2.0",
  "endAddress" : "192.0.2.255",
  "handle" : "XXXX-RIR",
  "ipVersion" : "v4",
  "name": "NET-RTR-1",
  "parentHandle" : "YYYY-RIR",
  "remarks" :
  [
```

```
      {
        "description" :
        [
          "She sells sea shells down by the sea shore.",
          "Originally written by Terry Sullivan."
        ]
      }
    ]
  }
```

Figure 13

5. Object Classes

Object classes represent structures appropriate for a response from the queries specified in [RFC7482].

Each object class contains a "links" array as specified in Section 4.2. For every object class instance in a response, whether the object class instance is directly representing the response to a query or is embedded in other object class instances or is an item in a search result set, servers SHOULD provide a link representing a URI for that object class instance using the "self" relationship as described in the IANA registry specified by [RFC5988]. As explained in Section 5.2, this may be not always be possible for nameserver data. Clients MUST be able to process object instances without a self link. When present, clients can use the self link for caching data. Servers MAY provide more than one self link for any given object instance. Failure to provide any self link by a server may result in clients being unable to cache object class instances.

Clients using self links for caching SHOULD not cache any object class instances where the authority of the self link is different than the authority of the server returning the data. Failing to do so might result in cache poisoning.

Self links MUST contain a "type" element containing the "application/rdap+json" media type when referencing RDAP object instances as defined by this document.

This is an example of the "links" array with a self link to an object class:

```
"links" :
[
    {
      "value" : "http://example.com/ip/2001:db8::123",
      "rel" : "self",
      "href" : "http://example.com/ip/2001:db8::123",
      "type" : "application/rdap+json"
    }
]
```

Figure 14

5.1. The Entity Object Class

The entity object class appears throughout this document and is an appropriate response for the /entity/XXXX query defined in "Registration Data Access Protocol (RDAP) Query Format" [RFC7482]. This object class represents the information of organizations, corporations, governments, non-profits, clubs, individual persons, and informal groups of people. All of these representations are so similar that it is best to represent them in JSON [RFC7159] with one construct, the entity object class, to aid in the reuse of code by implementers.

The entity object class uses jCard [RFC7095] to represent contact information, such as postal addresses, email addresses, phone numbers and names of organizations and individuals. Many of the types of information that can be represented with jCard have no use in RDAP, such as birthdays, anniversaries, and gender.

The entity object is served by both RIRs and DNRs. The following is an example of an entity that might be served by an RIR.

```
{
  "objectClassName" : "entity",
  "handle":"XXXX",
  "vcardArray":[
    "vcard",
    [
      ["version", {}, "text", "4.0"],
      ["fn", {}, "text", "Joe User"],
      ["n", {}, "text",
        ["User", "Joe", "", "", ["ing. jr", "M.Sc."]]
      ],
      ["kind", {}, "text", "individual"],
```

```
        ["lang", {
          "pref":"1"
        }, "language-tag", "fr"],
        ["lang", {
          "pref":"2"
        }, "language-tag", "en"],
        ["org", {
          "type":"work"
        }, "text", "Example"],
        ["title", {}, "text", "Research Scientist"],
        ["role", {}, "text", "Project Lead"],
        ["adr",
          { "type":"work" },
          "text",
          [
            "",
            "Suite 1234",
            "4321 Rue Somewhere",
            "Quebec",
            "QC",
            "G1V 2M2",
            "Canada"
          ]
        ],
        ["adr",
          {
            "type":"home",
            "label":"123 Maple Ave\nSuite 90001\nVancouver\nBC\n1239\n"
          },
          "text",
          [
            "", "", "", "", "", "", ""
          ]
        ],
        ["tel",
          {
            "type":["work", "voice"],
            "pref":"1"
          },
          "uri",
          "tel:+1-555-555-1234;ext=102"
        ],
        ["tel",
          { "type":["work", "cell", "voice", "video", "text"] },
          "uri",
          "tel:+1-555-555-4321"
        ],
        ["email",
```

```
          { "type":"work" },
          "text",
          "joe.user@example.com"
        ],
        ["geo", {
          "type":"work"
        }, "uri", "geo:46.772673,-71.282945"],
        ["key",
          { "type":"work" },
          "uri",
          "http://www.example.com/joe.user/joe.asc"
        ],
        ["tz", {},
          "utc-offset", "-05:00"],
        ["url", { "type":"home" },
          "uri", "http://example.org"]
    ]
  ],
  "roles":[ "registrar" ],
  "publicIds":[
    {
      "type":"IANA Registrar ID",
      "identifier":"1"
    }
  ],
  "remarks":[
    {
      "description":[
        "She sells sea shells down by the sea shore.",
        "Originally written by Terry Sullivan."
      ]
    }
  ],
  "links":[
    {
      "value":"http://example.com/entity/XXXX",
      "rel":"self",
      "href":"http://example.com/entity/XXXX",
      "type" : "application/rdap+json"
    }
  ],
  "events":[
    {
      "eventAction":"registration",
      "eventDate":"1990-12-31T23:59:59Z"
    }
  ],
  "asEventActor":[
```

```
      {
        "eventAction":"last changed",
        "eventDate":"1991-12-31T23:59:59Z"
      }
    ]
}
```

Figure 15

The entity object class can contain the following members:

- objectClassName -- the string "entity"
- handle -- a string representing a registry unique identifier of the entity
- vcardArray -- a jCard with the entity's contact information
- roles -- an array of strings, each signifying the relationship an object would have with its closest containing object (see Section 10.2.4 for a list of values)
- publicIds -- see Section 4.8
- entities -- an array of entity objects as defined by this section
- remarks -- see Section 4.3
- links -- see Section 4.2
- events -- see Section 4.5
- asEventActor -- this data structure takes the same form as the events data structure (see Section 4.5), but each object in the array MUST NOT have an "eventActor" member. These objects denote that the entity is an event actor for the given events. See Appendix B regarding the various ways events can be modeled.
- status -- see Section 4.6
- port43 -- see Section 4.7
- networks -- an array of IP network objects as defined in Section 5.4
- autnums -- an array of autnum objects as defined in Section 5.5

Entities may also have other entities embedded with them in an array. This can be used to model an organization with specific individuals fulfilling designated roles of responsibility.

The following is an elided example of an entity with embedded entities.

```
{
  "objectClassName" : "entity",
  "handle" : "ANENTITY",
  "roles" : [ "registrar" ],
  ...
  "entities" :
  [
    {
      "objectClassName" : "entity",
      "handle": "ANEMBEDDEDENTITY",
      "roles" : [ "technical" ],
      ...
    },
    ...
  ],
  ...
}
```

Figure 16

The following is an example of an entity that might be served by a DNR.

```
{
  "objectClassName" : "entity",
  "handle":"XXXX",
  "vcardArray":[
    "vcard",
    [
      ["version", {}, "text", "4.0"],
      ["fn", {}, "text", "Joe User"],
      ["kind", {}, "text", "individual"],
      ["lang", {
        "pref":"1"
      }, "language-tag", "fr"],
      ["lang", {
        "pref":"2"
      }, "language-tag", "en"],
      ["org", {
        "type":"work"
      }, "text", "Example"],
```

```
        ["title", {}, "text", "Research Scientist"],
        ["role", {}, "text", "Project Lead"],
        ["adr",
          { "type":"work" },
          "text",
          [
            "",
            "Suite 1234",
            "4321 Rue Somewhere",
            "Quebec",
            "QC",
            "G1V 2M2",
            "Canada"
          ]
        ],
        ["tel",
          { "type":["work", "voice"], "pref":"1" },
          "uri", "tel:+1-555-555-1234;ext=102"
        ],
        ["email",
          { "type":"work" },
          "text", "joe.user@example.com"
        ]
      ]
    ],
    "status":[ "validated", "locked" ],
    "remarks":[
      {
        "description":[
          "She sells sea shells down by the sea shore.",
          "Originally written by Terry Sullivan."
        ]
      }
    ],
    "links":[
      {
        "value":"http://example.com/entity/XXXX",
        "rel":"self",
        "href":"http://example.com/entity/XXXX",
        "type":"application/rdap+json"
      }
    ],
    "port43":"whois.example.net",
    "events":[
      {
        "eventAction":"registration",
        "eventDate":"1990-12-31T23:59:59Z"
      },
```

```
      {
        "eventAction":"last changed",
        "eventDate":"1991-12-31T23:59:59Z",
        "eventActor":"joe@example.com"
      }
    ]
}
```

Figure 17

See Appendix A for use of the entity object class to model various types of entities found in both RIRs and DNRs. See Appendix C regarding structured vs. unstructured postal addresses in entities.

5.2. The Nameserver Object Class

The nameserver object class represents information regarding DNS nameservers used in both forward and reverse DNS. RIRs and some DNRs register or expose nameserver information as an attribute of a domain name, while other DNRs model nameservers as "first class objects".

The nameserver object class accommodates both models and degrees of variation in between.

The following is an example of a nameserver object.

```
{
  "objectClassName" : "nameserver",
  "handle" : "XXXX",
  "ldhName" : "ns1.xn--fo-5ja.example",
  "unicodeName" : "ns1.foo.example",
  "status" : [ "active" ],
  "ipAddresses" :
  {
    "v4": [ "192.0.2.1", "192.0.2.2" ],
    "v6": [ "2001:db8::123" ]
  },
  "remarks" :
  [
    {
      "description" :
      [
        "She sells sea shells down by the sea shore.",
        "Originally written by Terry Sullivan."
      ]
    }
  ],
  "links" :
  [
    {
      "value" : "http://example.net/nameserver/xxxx",
      "rel" : "self",
      "href" : "http://example.net/nameserver/xxxx",
      "type" : "application/rdap+json"
    }
  ],
  "port43" : "whois.example.net",
  "events" :
  [
    {
      "eventAction" : "registration",
      "eventDate" : "1990-12-31T23:59:59Z"
    },
    {
      "eventAction" : "last changed",
      "eventDate" : "1991-12-31T23:59:59Z",
      "eventActor" : "joe@example.com"
    }
  ]
}
```

Figure 18

Figure 18 is an example of a nameserver object with all values given. Registries using a first-class nameserver data model would embed this in domain objects as well as allowing references to it with the "/nameserver" query type (all depending on the registry operators policy). Other registries may pare back the information as needed. Figure 19 is an example of a nameserver object as would be found in RIRs and some DNRs, while Figure 20 is an example of a nameserver object as would be found in other DNRs.

The following is an example of the simplest nameserver object:

```
{
  "objectClassName" : "nameserver",
  "ldhName" : "ns1.example.com"
}
```

Figure 19

The following is an example of a simple nameserver object that might be commonly used by DNRs:

```
{
  "objectClassName" : "nameserver",
  "ldhName" : "ns1.example.com",
  "ipAddresses" : { "v6" : [ "2001:db8::123", "2001:db8::124" ] }
}
```

Figure 20

As nameservers can be modeled by some registries to be first-class objects, they may also have an array of entities (Section 5.1) embedded to signify parties responsible for the maintenance, registrations, etc., of the nameservers.

The following is an elided example of a nameserver with embedded entities.

```
{
  "objectClassName" : "nameserver",
  "handle" : "XXXX",
  "ldhName" : "ns1.xn--fo-5ja.example",
  ...
  "entities" :
  [
    ...
  ],
  ...
}
```

Figure 21

The nameserver object class can contain the following members:

- objectClassName -- the string "nameserver"
- handle -- a string representing a registry unique identifier of the nameserver
- ldhName -- a string containing the LDH name of the nameserver (see Section 3)
- unicodeName -- a string containing a DNS Unicode name of the nameserver (see Section 3)
- ipAddresses -- an object containing the following members:
  - v6 -- an array of strings containing IPv6 addresses of the nameserver
  - v4 -- an array of strings containing IPv4 addresses of the nameserver
- entities -- an array of entity objects as defined by Section 5.1
- status -- see Section 4.6
- remarks -- see Section 4.3
- links -- see Section 4.2
- port43 -- see Section 4.7
- events -- see Section 4.5

5.3. The Domain Object Class

The domain object class represents a DNS name and point of delegation. For RIRs, these delegation points are in the reverse DNS tree, whereas for DNRs, these delegation points are in the forward DNS tree.

In both cases, the high-level structure of the domain object class consists of information about the domain registration, nameserver information related to the domain name, and entities related to the domain name (e.g., registrant information, contacts, etc.).

The following is an elided example of the domain object showing the high-level structure:

```
{
  "objectClassName" : "domain",
  "handle" : "XXX",
  "ldhName" : "blah.example.com",
  ...
  "nameservers" :
  [
    ...
  ],
  ...
  "entities" :
  [
    ...
  ]
}
```

Figure 22

The domain object class can contain the following members:

- objectClassName -- the string "domain"
- handle -- a string representing a registry unique identifier of the domain object instance
- ldhName -- a string describing a domain name in LDH form as described in Section 3
- unicodeName -- a string containing a domain name with U-labels as described in Section 3

- variants -- an array of objects, each containing the following values:
  - relation -- an array of strings, with each string denoting the relationship between the variants and the containing domain object (see Section 10.2.5 for a list of suggested variant relations).
  - idnTable -- the name of the Internationalized Domain Name (IDN) table of codepoints, such as one listed with the IANA (see IDN tables [IANA_IDNTABLES]).
  - variantNames -- an array of objects, with each object containing an "ldhName" member and a "unicodeName" member (see Section 3).
- nameservers -- an array of nameserver objects as defined by Section 5.2
- secureDNS -- an object with the following members:
  - zoneSigned -- true if the zone has been signed, false otherwise.
  - delegationSigned -- boolean true if there are DS records in the parent, false otherwise.
  - maxSigLife -- an integer representing the signature lifetime in seconds to be used when creating the RRSIG DS record in the parent zone [RFC5910].
  - dsData -- an array of objects, each with the following members:
    - keyTag -- an integer as specified by the key tag field of a DNS DS record as specified by [RFC4034] in presentation format
    - algorithm -- an integer as specified by the algorithm field of a DNS DS record as described by RFC 4034 in presentation format
    - digest -- a string as specified by the digest field of a DNS DS record as specified by RFC 4034 in presentation format
    - digestType -- an integer as specified by the digest type field of a DNS DS record as specified by RFC 4034 in presentation format

      + events -- see Section 4.5

      + links -- see Section 4.2

    * keyData -- an array of objects, each with the following members:

      + flags -- an integer representing the flags field value in the DNSKEY record [RFC4034] in presentation format

      + protocol -- an integer representation of the protocol field value of the DNSKEY record [RFC4034] in presentation format

      + publicKey -- a string representation of the public key in the DNSKEY record [RFC4034] in presentation format

      + algorithm -- an integer as specified by the algorithm field of a DNSKEY record as specified by [RFC4034] in presentation format

      + events -- see Section 4.5

      + links -- see Section 4.2

    See Appendix D for background information on these objects.

- entities -- an array of entity objects as defined by Section 5.1
- status -- see Section 4.6
- publicIds -- see Section 4.8
- remarks -- see Section 4.3
- links -- see Section 4.2
- port43 -- see Section 4.7
- events -- see Section 4.5
- network -- represents the IP network for which a reverse DNS domain is referenced. See Section 5.4

The following is an example of a JSON domain object representing a reverse DNS delegation point that might be served by an RIR.

```
{
  "objectClassName" : "domain",
  "handle" : "XXXX",
  "ldhName" : "0.2.192.in-addr.arpa",
  "nameservers" :
  [
    {
      "objectClassName" : "nameserver",
      "ldhName" : "ns1.rir.example"
    },
    {
      "objectClassName" : "nameserver",
      "ldhName" : "ns2.rir.example"
    }
  ],
  "secureDNS":
  {
    "delegationSigned": true,
    "dsData":
    [
      {
        "keyTag": 12345,
        "algorithm": 3,
        "digestType": 1,
        "digest": "49FD46E6C4B45C55D4AC"
      }
    ]
  },
  "remarks" :
  [
    {
      "description" :
      [
        "She sells sea shells down by the sea shore.",
        "Originally written by Terry Sullivan."
      ]
    }
  ],
  "links" :
  [
    {
      "value": "http://example.net/domain/XXXX",
      "rel" : "self",
      "href" : "http://example.net/domain/XXXXX",
      "type" : "application/rdap+json"
```

```
      }
    ],
    "events" :
    [
      {
        "eventAction" : "registration",
        "eventDate" : "1990-12-31T23:59:59Z"
      },
      {
        "eventAction" : "last changed",
        "eventDate" : "1991-12-31T23:59:59Z",
        "eventActor" : "joe@example.com"
      }
    ],
    "entities" :
    [
      {
        "objectClassName" : "entity",
        "handle" : "XXXX",
        "vcardArray":[
          "vcard",
          [
            ["version", {}, "text", "4.0"],
            ["fn", {}, "text", "Joe User"],
            ["kind", {}, "text", "individual"],
            ["lang", {
              "pref":"1"
            }, "language-tag", "fr"],
            ["lang", {
              "pref":"2"
            }, "language-tag", "en"],
            ["org", {
              "type":"work"
            }, "text", "Example"],
            ["title", {}, "text", "Research Scientist"],
            ["role", {}, "text", "Project Lead"],
            ["adr",
              { "type":"work" },
              "text",
              [
                "",
                "Suite 1234",
                "4321 Rue Somewhere",
                "Quebec",
                "QC",
                "G1V 2M2",
                "Canada"
              ]
```

```
            ],
            ["tel",
              { "type":["work", "voice"], "pref":"1" },
              "uri", "tel:+1-555-555-1234;ext=102"
            ],
            ["email",
              { "type":"work" },
              "text", "joe.user@example.com"
            ]
          ]
        ],
        "roles" : [ "registrant" ],
        "remarks" :
        [
          {
            "description" :
            [
              "She sells sea shells down by the sea shore.",
              "Originally written by Terry Sullivan."
            ]
          }
        ],
        "links" :
        [
          {
            "value": "http://example.net/entity/xxxx",
            "rel" : "self",
            "href" : "http://example.net/entity/xxxx",
            "type" : "application/rdap+json"
          }
        ],
        "events" :
        [
          {
            "eventAction" : "registration",
            "eventDate" : "1990-12-31T23:59:59Z"
          },
          {
            "eventAction" : "last changed",
            "eventDate" : "1991-12-31T23:59:59Z",
            "eventActor" : "joe@example.com"
          }
        ]
      }
    ],
    "network" :
    {
      "objectClassName" : "ip network",
```

```
      "handle" : "XXXX-RIR",
      "startAddress" : "192.0.2.0",
      "endAddress" : "192.0.2.255",
      "ipVersion" : "v6",
      "name": "NET-RTR-1",
      "type" : "DIRECT ALLOCATION",
      "country" : "AU",
      "parentHandle" : "YYYY-RIR",
      "status" : [ "active" ]
    }
  }
```

Figure 23

The following is an example of a JSON domain object representing a forward DNS delegation point that might be served by a DNR.

```
{
  "objectClassName" : "domain",
  "handle" : "XXXX",
  "ldhName" : "xn--fo-5ja.example",
  "unicodeName" : "foo.example",
  "variants" :
  [
    {
      "relation" : [ "registered", "conjoined" ],
      "variantNames" :
      [
        {
          "ldhName" : "xn--fo-cka.example",
          "unicodeName" : "foo.example"
        },
        {
          "ldhName" : "xn--fo-fka.example",
          "unicodeName" : "foo.example"
        }
      ]
    },
    {
      "relation" : [ "unregistered", "registration restricted" ],
      "idnTable": ".EXAMPLE Swedish",
      "variantNames" :
      [
        {
          "ldhName": "xn--fo-8ja.example",
          "unicodeName" : "foo.example"
        }
      ]
```

```
      }
    ],
    "status" : [ "locked", "transfer prohibited" ],
    "publicIds":[
      {
        "type":"ENS_Auth ID",
        "identifier":"1234567890"
      }
    ],
    "nameservers" :
    [
      {
        "objectClassName" : "nameserver",
        "handle" : "XXXX",
        "ldhName" : "ns1.example.com",
        "status" : [ "active" ],
        "ipAddresses" :
        {
          "v6": [ "2001:db8::123", "2001:db8::124" ],
          "v4": [ "192.0.2.1", "192.0.2.2" ]
        },
        "remarks" :
        [
          {
            "description" :
            [
              "She sells sea shells down by the sea shore.",
              "Originally written by Terry Sullivan."
            ]
          }
        ],
        "links" :
        [
          {
            "value" : "http://example.net/nameserver/XXXX",
            "rel" : "self",
            "href" : "http://example.net/nameserver/XXXX",
            "type" : "application/rdap+json"
          }
        ],
        "events" :
        [
          {
            "eventAction" : "registration",
            "eventDate" : "1990-12-31T23:59:59Z"
          },
          {
            "eventAction" : "last changed",
```

```
            "eventDate" : "1991-12-31T23:59:59Z"
          }
        ]
      },
      {
        "objectClassName" : "nameserver",
        "handle" : "XXXX",
        "ldhName" : "ns2.example.com",
        "status" : [ "active" ],
        "ipAddresses" :
        {
          "v6" : [ "2001:db8::125", "2001:db8::126" ],
          "v4" : [ "192.0.2.3", "192.0.2.4" ]
        },
        "remarks" :
        [
          {
            "description" :
            [
              "She sells sea shells down by the sea shore.",
              "Originally written by Terry Sullivan."
            ]
          }
        ],
        "links" :
        [
          {
            "value" : "http://example.net/nameserver/XXXX",
            "rel" : "self",
            "href" : "http://example.net/nameserver/XXXX",
            "type" : "application/rdap+json"
          }
        ],
        "events" :
        [
          {
            "eventAction" : "registration",
            "eventDate" : "1990-12-31T23:59:59Z"
          },
          {
            "eventAction" : "last changed",
            "eventDate" : "1991-12-31T23:59:59Z"
          }
        ]
      }
    ],
    "secureDNS":
    {
```

```
        "zoneSigned": true,
        "delegationSigned": true,
        "maxSigLife": 604800,
        "keyData":
        [
          {
            "flags": 257,
            "protocol": 3,
            "algorithm": 1,
            "publicKey": "AQPJ////4Q==",
            "events":
            [
              {
                "eventAction": "last changed",
                "eventDate": "2012-07-23T05:15:47Z"
              }
            ]
          }
        ]
      },
      "remarks" :
      [
        {
          "description" :
          [
            "She sells sea shells down by the sea shore.",
            "Originally written by Terry Sullivan."
          ]
        }
      ],
      "links" :
      [
        {
          "value": "http://example.net/domain/XXXX",
          "rel" : "self",
          "href" : "http://example.net/domain/XXXX",
          "type" : "application/rdap+json"
        }
      ],
      "port43" : "whois.example.net",
      "events" :
      [
        {
          "eventAction" : "registration",
          "eventDate" : "1990-12-31T23:59:59Z"
        },
        {
          "eventAction" : "last changed",
```

```
        "eventDate" : "1991-12-31T23:59:59Z",
        "eventActor" : "joe@example.com"
      },
      {
        "eventAction" : "transfer",
        "eventDate" : "1991-12-31T23:59:59Z",
        "eventActor" : "joe@example.com"
      },
      {
        "eventAction" : "expiration",
        "eventDate" : "2016-12-31T23:59:59Z",
        "eventActor" : "joe@example.com"
      }
    ],
    "entities" :
    [
      {
        "objectClassName" : "entity",
        "handle" : "XXXX",
        "vcardArray":[
          "vcard",
          [
            ["version", {}, "text", "4.0"],
            ["fn", {}, "text", "Joe User"],
            ["kind", {}, "text", "individual"],
            ["lang", {
              "pref":"1"
            }, "language-tag", "fr"],
            ["lang", {
              "pref":"2"
            }, "language-tag", "en"],
            ["org", {
              "type":"work"
            }, "text", "Example"],
            ["title", {}, "text", "Research Scientist"],
            ["role", {}, "text", "Project Lead"],
            ["adr",
              { "type":"work" },
              "text",
              [
                "",
                "Suite 1234",
                "4321 Rue Somewhere",
                "Quebec",
                "QC",
                "G1V 2M2",
                "Canada"
              ]
```

```
            ],
            ["tel",
              { "type":["work", "voice"], "pref":"1" },
              "uri", "tel:+1-555-555-1234;ext=102"
            ],
            ["email",
              { "type":"work" },
              "text", "joe.user@example.com"
            ]
          ]
        ],
        "status" : [ "validated", "locked" ],
        "roles" : [ "registrant" ],
        "remarks" :
        [
          {
            "description" :
            [
              "She sells sea shells down by the sea shore.",
              "Originally written by Terry Sullivan."
            ]
          }
        ],
        "links" :
        [
          {
            "value" : "http://example.net/entity/xxxx",
            "rel" : "self",
            "href" : "http://example.net/entity/xxxx",
            "type" : "application/rdap+json"
          }
        ],
        "events" :
        [
          {
            "eventAction" : "registration",
            "eventDate" : "1990-12-31T23:59:59Z"
          },
          {
            "eventAction" : "last changed",
            "eventDate" : "1991-12-31T23:59:59Z"
          }
        ]
      }
    ]
  }
```

Figure 24

5.4. The IP Network Object Class

The IP network object class models IP network registrations found in RIRs and is the expected response for the "/ip" query as defined by [RFC7482]. There is no equivalent object class for DNRs. The high-level structure of the IP network object class consists of information about the network registration and entities related to the IP network (e.g., registrant information, contacts, etc.).

The following is an elided example of the IP network object type showing the high-level structure:

```
{
  "objectClassName" : "ip network",
  "handle" : "XXX",
  ...
  "entities" :
  [
    ...
  ]
}
```

Figure 25

The following is an example of the JSON object for the network registration information.

```
{
  "objectClassName" : "ip network",
  "handle" : "XXXX-RIR",
  "startAddress" : "2001:db8::",
  "endAddress" : "2001:db8:0:ffff:ffff:ffff:ffff:ffff",
  "ipVersion" : "v6",
  "name": "NET-RTR-1",
  "type" : "DIRECT ALLOCATION",
  "country" : "AU",
  "parentHandle" : "YYYY-RIR",
  "status" : [ "active" ],
  "remarks" :
  [
    {
      "description" :
      [
        "She sells sea shells down by the sea shore.",
        "Originally written by Terry Sullivan."
      ]
    }
  ],
```

```
"links" :
[
  {
    "value" : "http://example.net/ip/2001:db8::/48",
    "rel" : "self",
    "href" : "http://example.net/ip/2001:db8::/48",
    "type" : "application/rdap+json"
  },
  {
    "value" : "http://example.net/ip/2001:db8::/48",
    "rel" : "up",
    "href" : "http://example.net/ip/2001:C00::/23",
    "type" : "application/rdap+json"
  }
],
"events" :
[
  {
    "eventAction" : "registration",
    "eventDate" : "1990-12-31T23:59:59Z"
  },
  {
    "eventAction" : "last changed",
    "eventDate" : "1991-12-31T23:59:59Z"
  }
],
"entities" :
[
  {
    "objectClassName" : "entity",
    "handle" : "XXXX",
    "vcardArray":[
      "vcard",
      [
        ["version", {}, "text", "4.0"],
        ["fn", {}, "text", "Joe User"],
        ["kind", {}, "text", "individual"],
        ["lang", {
          "pref":"1"
        }, "language-tag", "fr"],
        ["lang", {
          "pref":"2"
        }, "language-tag", "en"],
        ["org", {
          "type":"work"
        }, "text", "Example"],
        ["title", {}, "text", "Research Scientist"],
        ["role", {}, "text", "Project Lead"],
```

```
          ["adr",
            { "type":"work" },
            "text",
            [
              "",
              "Suite 1234",
              "4321 Rue Somewhere",
              "Quebec",
              "QC",
              "G1V 2M2",
              "Canada"
            ]
          ],
          ["tel",
            { "type":["work", "voice"], "pref":"1" },
            "uri", "tel:+1-555-555-1234;ext=102"
          ],
          ["email",
            { "type":"work" },
            "text", "joe.user@example.com"
          ]
        ]
      ],
      "roles" : [ "registrant" ],
      "remarks" :
      [
        {
          "description" :
          [
            "She sells sea shells down by the sea shore.",
            "Originally written by Terry Sullivan."
          ]
        }
      ],
      "links" :
      [
        {
          "value" : "http://example.net/entity/xxxx",
          "rel" : "self",
          "href" : "http://example.net/entity/xxxx",
          "type" : "application/rdap+json"
        }
      ],
      "events" :
      [
        {
          "eventAction" : "registration",
          "eventDate" : "1990-12-31T23:59:59Z"
```

```
          },
          {
            "eventAction" : "last changed",
            "eventDate" : "1991-12-31T23:59:59Z"
          }
        ]
      }
    ]
  }
```

Figure 26

The IP network object class can contain the following members:

- objectClassName -- the string "ip network"
- handle -- a string representing an RIR-unique identifier of the network registration
- startAddress -- the starting IP address of the network, either IPv4 or IPv6
- endAddress -- the ending IP address of the network, either IPv4 or IPv6
- ipVersion -- a string signifying the IP protocol version of the network: "v4" signifies an IPv4 network, and "v6" signifies an IPv6 network
- name -- an identifier assigned to the network registration by the registration holder
- type -- a string containing an RIR-specific classification of the network
- country -- a string containing the two-character country code of the network
- parentHandle -- a string containing an RIR-unique identifier of the parent network of this network registration
- status -- an array of strings indicating the state of the IP network
- entities -- an array of entity objects as defined by Section 5.1
- remarks -- see Section 4.3

- links -- see Section 4.2
- port43 -- see Section 4.7
- events -- see Section 4.5

5.5. Autonomous System Number Entity Object Class

The Autonomous System number (autnum) object class models Autonomous System number registrations found in RIRs and represents the expected response to an "/autnum" query as defined by [RFC7482]. There is no equivalent object class for DNRs. The high-level structure of the autnum object class consists of information about the network registration and entities related to the autnum registration (e.g., registrant information, contacts, etc.) and is similar to the IP network entity object class.

The following is an example of a JSON object representing an autnum.

```
{
  "objectClassName" : "autnum",
  "handle" : "XXXX-RIR",
  "startAutnum" : 10,
  "endAutnum" : 15,
  "name": "AS-RTR-1",
  "type" : "DIRECT ALLOCATION",
  "status" : [ "active" ],
  "country": "AU",
  "remarks" :
  [
    {
      "description" :
      [
        "She sells sea shells down by the sea shore.",
        "Originally written by Terry Sullivan."
      ]
    }
  ],
  "links" :
  [
    {
      "value" : "http://example.net/autnum/xxxx",
      "rel" : "self",
      "href" : "http://example.net/autnum/xxxx",
      "type" : "application/rdap+json"
    }
  ],
  "events" :
```

```
    [
      {
        "eventAction" : "registration",
        "eventDate" : "1990-12-31T23:59:59Z"
      },
      {
        "eventAction" : "last changed",
        "eventDate" : "1991-12-31T23:59:59Z"
      }
    ],
    "entities" :
    [
      {
        "objectClassName" : "entity",
        "handle" : "XXXX",
        "vcardArray":[
          "vcard",
          [
            ["version", {}, "text", "4.0"],
            ["fn", {}, "text", "Joe User"],
            ["kind", {}, "text", "individual"],
            ["lang", {
              "pref":"1"
            }, "language-tag", "fr"],
            ["lang", {
              "pref":"2"
            }, "language-tag", "en"],
            ["org", {
              "type":"work"
            }, "text", "Example"],
            ["title", {}, "text", "Research Scientist"],
            ["role", {}, "text", "Project Lead"],
            ["adr",
              { "type":"work" },
              "text",
              [
                "",
                "Suite 1234",
                "4321 Rue Somewhere",
                "Quebec",
                "QC",
                "G1V 2M2",
                "Canada"
              ]
            ],
            ["tel",
              { "type":["work", "voice"], "pref":"1" },
              "uri", "tel:+1-555-555-1234;ext=102"
```

```
            ],
            ["email",
              { "type":"work" },
              "text", "joe.user@example.com"
            ]
          ]
        ],
        "roles" : [ "registrant" ],
        "remarks" :
        [
          {
            "description" :
            [
              "She sells sea shells down by the sea shore.",
              "Originally written by Terry Sullivan."
            ]
          }
        ],
        "links" :
        [
          {
            "value" : "http://example.net/entity/XXXX",
            "rel" : "self",
            "href" : "http://example.net/entity/XXXX",
            "type" : "application/rdap+json"
          }
        ],
        "events" :
        [
          {
            "eventAction" : "registration",
            "eventDate" : "1990-12-31T23:59:59Z"
          },
          {
            "eventAction" : "last changed",
            "eventDate" : "1991-12-31T23:59:59Z"
          }
        ]
      }
    ]
  }
```

Figure 27

The Autonomous System number object class can contain the following members:

- objectClassName -- the string "autnum"
- handle -- a string representing an RIR-unique identifier of the autnum registration
- startAutnum -- a number representing the starting number [RFC5396] in the block of Autonomous System numbers
- endAutnum -- a number representing the ending number [RFC5396] in the block of Autonomous System numbers
- name -- an identifier assigned to the autnum registration by the registration holder
- type -- a string containing an RIR-specific classification of the autnum
- status -- an array of strings indicating the state of the autnum
- country -- a string containing the name of the two-character country code of the autnum
- entities -- an array of entity objects as defined by Section 5.1
- remarks -- see Section 4.3
- links -- see Section 4.2
- port43 -- see Section 4.7
- events -- see Section 4.5

6. Error Response Body

Some non-answer responses may return entity bodies with information that could be more descriptive.

The basic structure of that response is an object class containing an error code number (corresponding to the HTTP response code) followed by a string named "title" and an array of strings named "description".

This is an example of the common response body.

```
{
  "errorCode": 418,
  "title": "Your Beverage Choice is Not Available",
  "description":
  [
    "I know coffee has more ummppphhh.",
    "Sorry, dude!"
  ]
}
```

Figure 28

This is an example of the common response body with an rdapConformance and notices data structures:

```
{
  "rdapConformance" :
  [
    "rdap_level_0"
  ],
  "notices" :
  [
    {
      "title" : "Beverage Policy",
      "description" :
      [
        "Beverages with caffeine for keeping horses awake."
      ],
      "links" :
      [
        {
          "value" : "http://example.net/ip/192.0.2.0/24",
          "rel" : "alternate",
          "type" : "text/html",
          "href" : "http://www.example.com/redaction_policy.html"
        }
      ]
    }
  ],
  "lang" : "en",
  "errorCode": 418,
  "title": "Your beverage choice is not available",
  "description":
  [
    "I know coffee has more ummppphhh.",
    "Sorry, dude!"
  ]
}
```

Figure 29

7. Responding to Help Queries

The appropriate response to /help queries as defined by [RFC7482] is to use the notices structure as defined in Section 4.3.

This is an example of a response to a /help query including the rdapConformance data structure.

```
{
  "rdapConformance" :
  [
    "rdap_level_0"
  ],
  "notices" :
  [
    {
      "title" : "Authentication Policy",
      "description" :
      [
        "Access to sensitive data for users with proper credentials."
      ],
      "links" :
      [
        {
          "value" : "http://example.net/help",
          "rel" : "alternate",
          "type" : "text/html",
          "href" : "http://www.example.com/auth_policy.html"
        }
      ]
    }
  ]
}
```

Figure 30

8. Responding To Searches

[RFC7482] specifies three types of searches: domains, nameservers, and entities. Responses to these searches take the form of an array of object instances where each instance is an appropriate object class for the search (i.e., a search for /domains yields an array of domain object instances). These arrays are contained within the response object.

The names of the arrays are as follows:

- for /domains searches, the array is "domainSearchResults"
- for /nameservers searches, the array is "nameserverSearchResults"
- for /entities searches, the array is "entitySearchResults"

The following is an elided example of a response to a /domains search.

```
{
  "rdapConformance" :
  [
    "rdap_level_0"
  ],
  ...
  "domainSearchResults" :
  [
    {
      "objectClassName" : "domain",
      "handle" : "1-XXXX",
      "ldhName" : "1.example.com",
      ...
    },
    {
      "objectClassName" : "domain",
      "handle" : "2-XXXX",
      "ldhName" : "2.example.com",
      ...
    }
  ]
}
```

Figure 31

9. Indicating Truncated Responses

In cases where the data of a response needs to be limited or parts of the data need to be omitted, the response is considered "truncated". A truncated response is still valid JSON, but some of the results in a search set or some of the data in an object are not provided by the server. A server may indicate this by including a typed notice in the response object.

The following is an elided example of a search response that has been truncated.

```
{
  "rdapConformance" :
  [
    "rdap_level_0"
  ],
  "notices" :
  [
    {
      "title" : "Search Policy",
      "type" : "result set truncated due to authorization",
      "description" :
      [
        "Search results are limited to 25 per day per querying IP."
      ],
      "links" :
      [
        {
          "value" : "http://example.net/help",
          "rel" : "alternate",
          "type" : "text/html",
          "href" : "http://www.example.com/search_policy.html"
        }
      ]
    }
  ],
  "domainSearchResults" :
  [
    ...
  ]
}
```

Figure 32

A similar technique can be used with a typed remark where a single object has been returned and data in that object has been truncated. Such an example might be an entity object with only a partial set of the IP networks associated with it.

The following is an elided example of an entity truncated data.

```
{
  "objectClassName" : "entity",
  "handle" : "ANENTITY",
  "roles" : [ "registrant" ],
  ...
  "entities" :
  [
    {
      "objectClassName" : "entity",
      "handle": "ANEMBEDDEDENTITY",
      "roles" : [ "technical" ],
      ...
    },
    ...
  ],
  "networks" :
  [
    ...
  ],
  ...
  "remarks" :
  [
    {
      "title" : "Data Policy",
      "type" : "object truncated due to unexplainable reason",
      "description" :
      [
        "Some of the data in this object has been removed."
      ],
      "links" :
      [
        {
          "value" : "http://example.net/help",
          "rel" : "alternate",
          "type" : "text/html",
          "href" : "http://www.example.com/data_policy.html"
        }
      ]
    }
  ]
}
```

Figure 33

10. IANA Considerations

10.1. RDAP JSON Media Type Registration

This specification registers the "application/rdap+json" media type.

Type name: application

Subtype name: rdap+json

Required parameters: n/a

Encoding considerations: See Section 3.1 of [RFC6839].

Security considerations: The media represented by this identifier does not have security considerations beyond that found in Section 6 of [RFC7159].

Interoperability considerations: There are no known interoperability problems regarding this media format.

Published specification: RFC 7483

Applications that use this media type: Implementations of the Registration Data Access Protocol (RDAP).

Additional information: This media type is a product of the IETF WEIRDS working group. The WEIRDS charter, information on the WEIRDS mailing list, and other documents produced by the WEIRDS working group can be found at <https://datatracker.ietf.org/wg/weirds/>.

Person & email address to contact for further information: IESG <iesg@ietf.org>

Intended usage: COMMON

Restrictions on usage: none

Author: Andy Newton

Change controller: IETF

Provisional Registration: No (upon publication of this RFC)

10.2. JSON Values Registry

IANA has created a category in the protocol registries labeled "Registration Data Access Protocol (RDAP)", and within that category, IANA has established a URL-referenceable, stand-alone registry labeled "RDAP JSON Values". This new registry is for use in the notices and remarks (Section 4.3), status (Section 4.6), role (Section 5.1), event action (Section 4.5), and domain variant relation (Section 5.3) fields specified in RDAP.

Each entry in the registry contains the following fields:

1. Value -- the string value being registered.

2. Type -- the type of value being registered. It should be one of the following:

   * "notice or remark type" -- denotes a type of notice or remark.

   * "status" -- denotes a value for the "status" object member as defined by Section 4.6.

   * "role" -- denotes a value for the "role" array as defined in Section 5.1.

   * "event action" -- denotes a value for an event action as defined in Section 4.5.

   * "domain variant relation" -- denotes a relationship between a domain and a domain variant as defined in Section 5.3.

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

4. Registrant Name -- the name of the person registering the value.

5. Registrant Contact Information -- an email address, postal address, or some other information to be used to contact the registrant.

This registry is operated under the "Expert Review" policy defined in [RFC5226].

Review of registrations into this registry by the designated expert(s) should be narrowly judged on the following criteria:

1. Values in need of being placed into multiple types must be assigned a separate registration for each type.

2. Values must be strings. They should be multiple words separated by single space characters. Every character should be lowercased. If possible, every word should be given in English and each character should be US-ASCII.

3. Registrations should not duplicate the meaning of any existing registration. That is, if a request for a registration is significantly similar in nature to an existing registration, the request should be denied. For example, the terms "maintainer" and "registrant" are significantly similar in nature as they both denote a holder of a domain name or Internet number resource. In cases where it may be reasonably argued that machine interpretation of two similar values may alter the operation of client software, designated experts should not judge the values to be of significant similarity.

4. Registrations should be relevant to the common usages of RDAP. Designated experts may rely upon the serving of the value by a DNR or RIR to make this determination.

The following sections provide initial registrations into this registry.

10.2.1. Notice and Remark Types

The following values have been registered in the "RDAP JSON Values" registry:

Value: result set truncated due to authorization
Type: notice and remark type
Description: The list of results does not contain all results due to lack of authorization. This may indicate to some clients that proper authorization will yield a longer result set.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org

```
Value: result set truncated due to excessive load
Type: notice and remark type
Description: The list of results does not contain all results due
   to an excessively heavy load on the server.  This may indicate
   to some clients that requerying at a later time will yield a
   longer result set.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: result set truncated due to unexplainable reasons
Type: notice and remark type
Description: The list of results does not contain all results for
   an unexplainable reason.  This may indicate to some clients
   that requerying for any reason will not yield a longer result
   set.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: object truncated due to authorization
Type: notice and remark type
Description: The object does not contain all data due to lack of
   authorization.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: object truncated due to excessive load
Type: notice and remark type
Description: The object does not contain all data due to an
   excessively heavy load on the server.  This may indicate to
   some clients that requerying at a later time will yield all
   data of the object.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: object truncated due to unexplainable reasons
Type: notice and remark type
Description: The object does not contain all data for an
   unexplainable reason.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org
```

10.2.2. Status

The following values have been registered in the "RDAP JSON Values" registry:

Value: validated
Type: status
Description: Signifies that the data of the object instance has been found to be accurate. This type of status is usually found on entity object instances to note the validity of identifying contact information.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org

Value: renew prohibited
Type: status
Description: Renewal or reregistration of the object instance is forbidden.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org

Value: update prohibited
Type: status
Description: Updates to the object instance are forbidden.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org

Value: transfer prohibited
Type: status
Description: Transfers of the registration from one registrar to another are forbidden. This type of status normally applies to DNR domain names.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org

Value: delete prohibited
Type: status
Description: Deletion of the registration of the object instance is forbidden. This type of status normally applies to DNR domain names.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org

```
Value: proxy
Type: status
Description: The registration of the object instance has been
   performed by a third party.  This is most commonly applied to
   entities.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: private
Type: status
Description: The information of the object instance is not
   designated for public consumption.  This is most commonly
   applied to entities.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: removed
Type: status
Description: Some of the information of the object instance has
   not been made available and has been removed.  This is most
   commonly applied to entities.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: obscured
Type: status
Description: Some of the information of the object instance has
   been altered for the purposes of not readily revealing the
   actual information of the object instance.  This is most
   commonly applied to entities.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: associated
Type: status
Description: The object instance is associated with other object
   instances in the registry.  This is most commonly used to
   signify that a nameserver is associated with a domain or that
   an entity is associated with a network resource or domain.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org
```

```
Value: active
Type: status
Description: The object instance is in use.  For domain names, it
   signifies that the domain name is published in DNS.  For
   network and autnum registrations, it signifies that they are
   allocated or assigned for use in operational networks.  This
   maps to the "OK" status of the Extensible Provisioning Protocol
   (EPP) [RFC5730] .
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: inactive
Type: status
Description: The object instance is not in use.  See "active".
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: locked
Type: status
Description: Changes to the object instance cannot be made,
   including the association of other object instances.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: pending create
Type: status
Description: A request has been received for the creation of the
   object instance, but this action is not yet complete.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: pending renew
Type: status
Description: A request has been received for the renewal of the
   object instance, but this action is not yet complete.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org
```

```
Value: pending transfer
Type: status
Description: A request has been received for the transfer of the
   object instance, but this action is not yet complete.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: pending update
Type: status
Description: A request has been received for the update or
   modification of the object instance, but this action is not yet
   complete.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: pending delete
Type: status
Description: A request has been received for the deletion or
   removal of the object instance, but this action is not yet
   complete.  For domains, this might mean that the name is no
   longer published in DNS but has not yet been purged from the
   registry database.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org
```

10.2.3. Event Actions

The following values have been registered in the "RDAP JSON Values" registry:

```
Value: registration
Type: event action
Description: The object instance was initially registered.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: reregistration
Type: event action
Description: The object instance was registered subsequently to
   initial registration.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org
```

```
Value: last changed
Type: event action
Description: An action noting when the information in the object
   instance was last changed.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: expiration
Type: event action
Description: The object instance has been removed or will be
   removed at a predetermined date and time from the registry.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: deletion
Type: event action
Description: The object instance was removed from the registry at
   a point in time that was not predetermined.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: reinstantiation
Type: event action
Description: The object instance was reregistered after having
   been removed from the registry.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: transfer
Type: event action
Description: The object instance was transferred from one
   registrant to another.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: locked
Type: event action
Description: The object instance was locked (see the "locked"
   status).
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org
```

```
   Value: unlocked
   Type: event action
   Description: The object instance was unlocked (see the "locked"
      status).
   Registrant Name: IESG
   Registrant Contact Information: iesg@ietf.org
```

10.2.4. Roles

The following values have been registered in the "RDAP JSON Values" registry:

```
   Value: registrant
   Type: role
   Description: The entity object instance is the registrant of the
      registration.  In some registries, this is known as a
      maintainer.
   Registrant Name: IESG
   Registrant Contact Information: iesg@ietf.org


   Value: technical
   Type: role
   Description: The entity object instance is a technical contact for
      the registration.
   Registrant Name: IESG
   Registrant Contact Information: iesg@ietf.org


   Value: administrative
   Type: role
   Description: The entity object instance is an administrative
      contact for the registration.
   Registrant Name: IESG
   Registrant Contact Information: iesg@ietf.org


   Value: abuse
   Type: role
   Description: The entity object instance handles network abuse
      issues on behalf of the registrant of the registration.
   Registrant Name: IESG
   Registrant Contact Information: iesg@ietf.org
```

```
Value: billing
Type: role
Description: The entity object instance handles payment and
   billing issues on behalf of the registrant of the registration.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: registrar
Type: role
Description: The entity object instance represents the authority
   responsible for the registration in the registry.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: reseller
Type: role
Description: The entity object instance represents a third party
   through which the registration was conducted (i.e., not the
   registry or registrar).
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: sponsor
Type: role
Description: The entity object instance represents a domain policy
   sponsor, such as an ICANN-approved sponsor.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: proxy
Type: role
Description: The entity object instance represents a proxy for
   another entity object, such as a registrant.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org


Value: notifications
Type: role
Description: An entity object instance designated to receive
   notifications about association object instances.
Registrant Name: IESG
Registrant Contact Information: iesg@ietf.org
```

```
   Value: noc
   Type: role
   Description: The entity object instance handles communications
      related to a network operations center (NOC).
   Registrant Name: IESG
   Registrant Contact Information: iesg@ietf.org
```

10.2.5. Variant Relations

The following values have been registered in the "RDAP JSON Values" registry:

```
   Value: registered
   Type: domain variant relation
   Description: The variant names are registered in the registry.
   Registrant Name: IESG
   Registrant Contact Information: iesg@ietf.org


   Value: unregistered
   Type: domain variant relation
   Description: The variant names are not found in the registry.
   Registrant Name: IESG
   Registrant Contact Information: iesg@ietf.org


   Value: registration restricted
   Type: domain variant relation
   Description: Registration of the variant names is restricted to
      certain parties or within certain rules.
   Registrant Name: IESG
   Registrant Contact Information: iesg@ietf.org


   Value: open registration
   Type: domain variant relation
   Description: Registration of the variant names is available to
      generally qualified registrants.
   Registrant Name: IESG
   Registrant Contact Information: iesg@ietf.org
```

```
      Value: conjoined
      Type: domain variant relation
      Description: Registration of the variant names occurs
         automatically with the registration of the containing domain
         registration.
      Registrant Name: IESG
      Registrant Contact Information: iesg@ietf.org
```

11. Security Considerations

This specification models information serialized in JSON format. As JSON is a subset of JavaScript, implementations are advised to follow the security considerations outlined in Section 6 of [RFC7159] to prevent code injection.

Though not specific to JSON, RDAP implementers should be aware of the security considerations specified in [RFC7480] and the security requirements and considerations in [RFC7481].

Clients caching data, especially clients using RDAP-specific caches (instead of HTTP-layer caches), should have safeguards to prevent cache poisoning. See Section 5 for advice on using the self links for caching.

Finally, service operators should be aware of the privacy mechanisms noted in Section 13.

12. Internationalization Considerations

12.1. Character Encoding

The default text encoding for JSON responses in RDAP is UTF-8 [RFC3629], and all servers and clients MUST support UTF-8.

12.2. URIs and IRIs

[RFC7480] defines the use of URIs and IRIs in RDAP.

12.3. Language Tags

Section 4.4 defines the use of language tags in the JSON responses defined in this document.

12.4. Internationalized Domain Names

IDNs are denoted in this specification by the separation of DNS names in LDH form and Unicode form (see Section 3). Representation of IDNs in registries is described by the "variants" object in Section 5.3 and the suggested values listed in Section 10.2.5.

13. Privacy Considerations

This specification suggests status values to denote contact and registrant information that has been marked as private and/or has been removed or obscured. See Section 10.2.2 for the complete list of status values. A few of the status values indicate that there are privacy concerns associated with the object instance. The following status codes SHOULD be used to describe data elements of a response when appropriate:

private -- The object is not be shared in query responses, unless the user is authorized to view this information.

removed -- Data elements within the object have been collected but have been omitted from the response. This option can be used to prevent unauthorized access to associated object instances without the need to mark them as private.

obscured -- Data elements within the object have been collected, but the response value has been altered so that values are not easily discernible. A value changed from "1212" to "XXXX" is an example of obscured data. This option may reveal privacy sensitive information and should only be used when data sensitivity does not require a more protective option like "private" or "removed".

See Appendix A.1 for an example of applying those values to contacts and registrants.

14. References

14.1. Normative References

[ISO.3166.1988] International Organization for Standardization, "Codes for the representation of names of countries, 3rd edition", ISO Standard 3166, August 1988.

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

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

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

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

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

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

[RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009, <http://www.rfc-editor.org/info/rfc5646>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010, <http://www.rfc-editor.org/info/rfc5890>.

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

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, October 2010, <http://www.rfc-editor.org/info/rfc5988>.

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

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

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

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

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

14.2. Informative References

[IANA_IDNTABLES]
IANA, "Repository of IDN Practices", <http://www.iana.org/domains/idn-tables>.

[JSON_ascendancy]
MacVittie, L., "The Stealthy Ascendancy of JSON", April 2011, <https://devcentral.f5.com/weblogs/macvittie/archive/2011/04/27/the-stealthy-ascendancy-of-json.aspx>.

[JSON_performance_study]
Nurseitov, N., Paulson, M., Reynolds, R., and C. Izurieta, "Comparison of JSON and XML Data Interchange Formats: A Case Study", 2009, <http://www.cs.montana.edu/izurieta/pubs/caine2009.pdf>.

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

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

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

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

[RFC6839] Hansen, T. and A. Melnikov, "Additional Media Type Structured Syntax Suffixes", RFC 6839, January 2013, <http://www.rfc-editor.org/info/rfc6839>.

Appendix A. Suggested Data Modeling with the Entity Object Class

A.1. Registrants and Contacts

This document does not provide specific object classes for registrants and contacts. Instead, the entity object class may be used to represent a registrant or contact. When the entity object is embedded inside a containing object such as a domain name or IP network, the "roles" string array can be used to signify the relationship. It is recommended that the values from Section 10.2.4 be used.

The following is an example of an elided containing object with an embedded entity that is both a registrant and administrative contact:

```
{
  ...
  "entities" :
  [
    {
      "objectClassName" : "entity",
      "handle" : "XXXX",
      "vcardArray":[
        "vcard",
        [
          ["version", {}, "text", "4.0"],
          ["fn", {}, "text", "Joe User"],
          ["kind", {}, "text", "individual"],
          ["lang", {
            "pref":"1"
          }, "language-tag", "fr"],
          ["lang", {
            "pref":"2"
          }, "language-tag", "en"],
          ["org", {
            "type":"work"
          }, "text", "Example"],
          ["title", {}, "text", "Research Scientist"],
          ["role", {}, "text", "Project Lead"],
          ["adr",
            { "type":"work" },
            "text",
            [
              "",
              "Suite 1234",
              "4321 Rue Somewhere",
              "Quebec",
              "QC",
```

```
                "G1V 2M2",
                "Canada"
              ]
            ],
            ["tel",
              { "type":["work", "voice"], "pref":"1" },
              "uri", "tel:+1-555-555-1234;ext=102"
            ],
            ["email",
              { "type":"work" },
              "text", "joe.user@example.com"
            ]
          ]
        ],
        "roles" : [ "registrant", "administrative" ],
        "remarks" :
        [
          {
            "description" :
            [
              "She sells sea shells down by the sea shore.",
              "Originally written by Terry Sullivan."
            ]
          }
        ],
        "events" :
        [
          {
            "eventAction" : "registration",
            "eventDate" : "1990-12-31T23:59:59Z"
          },
          {
            "eventAction" : "last changed",
            "eventDate" : "1991-12-31T23:59:59Z"
          }
        ]
      }
    ]
  }
```

Figure 34

In many use cases, it is necessary to hide or obscure the information of a registrant or contact due to policy or other operational matters. Registries can denote these situations with "status" values (see Section 10.2.2).

The following is an elided example of a registrant with information changed to reflect that of a third party.

```
{
  ...
  "entities" :
  [
    {
      "objectClassName" : "entity",
      "handle" : "XXXX",
      ...
      "roles" : [ "registrant", "administrative" ],
      "status" : [ "proxy", "private", "obscured" ]
    }
  ]
}
```

Figure 35

A.2. Registrars

This document does not provide a specific object class for registrars, but like registrants and contacts (see Appendix A.1), the "roles" string array maybe used. Additionally, many registrars have publicly assigned identifiers. The publicIds structure (Section 4.8) represents that information.

The following is an example of an elided containing object with an embedded entity that is a registrar:

```
{
  ...
  "entities":[
    {
      "objectClassName" : "entity",
      "handle":"XXXX",
      "vcardArray":[
        "vcard",
        [
          ["version", {}, "text", "4.0"],
          ["fn", {}, "text", "Joe's Fish, Chips, and Domains"],
          ["kind", {}, "text", "org"],
          ["lang", {
            "pref":"1"
          }, "language-tag", "fr"],
          ["lang", {
            "pref":"2"
          }, "language-tag", "en"],
```

```
          ["org", {
            "type":"work"
          }, "text", "Example"],
          ["adr",
            { "type":"work" },
            "text",
            [
              "",
              "Suite 1234",
              "4321 Rue Somewhere",
              "Quebec",
              "QC",
              "G1V 2M2",
              "Canada"
            ]
          ],
          ["tel",
            {
              "type":["work", "voice"],
              "pref":"1"
            },
            "uri", "tel:+1-555-555-1234;ext=102"
          ],
          ["email",
            { "type":"work" },
            "text", "joes_fish_chips_and_domains@example.com"
          ]
        ]
      ],
      "roles":[ "registrar" ],
      "publicIds":[
        {
          "type":"IANA Registrar ID",
          "identifier":"1"
        }
      ],
      "remarks":[
        {
          "description":[
            "She sells sea shells down by the sea shore.",
            "Originally written by Terry Sullivan."
          ]
        }
      ],
      "links":[
        {
          "value":"http://example.net/entity/XXXX",
          "rel":"alternate",
```

```
            "type":"text/html",
            "href":"http://www.example.com"
          }
        ]
      }
    ]
  }
```

Figure 36

Appendix B. Modeling Events

Events represent actions that have taken place against a registered object at a certain date and time. Events have three properties: the action, the actor, and the date and time of the event (which is sometimes in the future). In some cases, the identity of the actor is not captured.

Events can be modeled in three ways:

1. events with no designated actor

2. events where the actor is only designated by an identifier

3. events where the actor can be modeled as an entity

For the first use case, the events data structure (Section 4.5) is used without the "eventActor" object member.

This is an example of an "events" array without the "eventActor".

```
"events" :
[
  {
    "eventAction" : "registration",
    "eventDate" : "1990-12-31T23:59:59Z"
  }
]
```

Figure 37

For the second use case, the events data structure (Section 4.5) is used with the "eventActor" object member.

This is an example of an "events" array with the "eventActor".

```
"events" :
[
  {
    "eventAction" : "registration",
    "eventActor" : "XYZ-NIC",
    "eventDate" : "1990-12-31T23:59:59Z"
  }
]
```

Figure 38

For the third use case, the "asEventActor" array is used when an entity (Section 5.1) is embedded into another object class. The "asEventActor" array follows the same structure as the "events" array but does not have "eventActor" attributes.

The following is an elided example of a domain object with an entity as an event actor.

```
{
  "objectClassName" : "domain",
  "handle" : "XXXX",
  "ldhName" : "foo.example",
  "status" : [ "locked", "transfer prohibited" ],
  ...
  "entities" :
  [
    {
      "handle" : "XXXX",
      ...
      "asEventActor" :
      [
        {
          "eventAction" : "last changed",
          "eventDate" : "1990-12-31T23:59:59Z"
        }
      ]
    }
  ]
}
```

Figure 39

Appendix C. Structured vs. Unstructured Addresses

The entity (Section 5.1) object class uses jCard [RFC7095] to represent contact information, including postal addresses. jCard has the ability to represent multiple language preferences, multiple email address and phone numbers, and multiple postal addresses in both a structured and unstructured format. This section describes the use of jCard for representing structured and unstructured addresses.

The following is an example of a jCard.

```
{
  "vcardArray":[
    "vcard",
    [
      ["version", {}, "text", "4.0"],
      ["fn", {}, "text", "Joe User"],
      ["n", {}, "text",
        ["User", "Joe", "", "", ["ing. jr", "M.Sc."]]
      ],
      ["kind", {}, "text", "individual"],
      ["lang", {
        "pref":"1"
      }, "language-tag", "fr"],
      ["lang", {
        "pref":"2"
      }, "language-tag", "en"],
      ["org", {
        "type":"work"
      }, "text", "Example"],
      ["title", {}, "text", "Research Scientist"],
      ["role", {}, "text", "Project Lead"],
      ["adr",
        { "type":"work" },
        "text",
        [
          "",
          "Suite 1234",
          "4321 Rue Somewhere",
          "Quebec",
          "QC",
          "G1V 2M2",
          "Canada"
        ]
      ],
      ["adr",
        {
```

```
            "type":"home",
            "label":"123 Maple Ave\nSuite 90001\nVancouver\nBC\n1239\n"
          },
          "text",
          [
            "", "", "", "", "", "", ""
          ]
        ],
        ["tel",
          { "type":["work", "voice"], "pref":"1" },
          "uri", "tel:+1-555-555-1234;ext=102"
        ],
        ["tel",
          {
            "type":["work", "cell", "voice", "video", "text"]
          },
          "uri",
          "tel:+1-555-555-1234"
        ],
        ["email",
          { "type":"work" },
          "text", "joe.user@example.com"
        ],
        ["geo", {
          "type":"work"
        }, "uri", "geo:46.772673,-71.282945"],
        ["key",
          { "type":"work" },
          "uri", "http://www.example.com/joe.user/joe.asc"
        ],
        ["tz", {},
          "utc-offset", "-05:00"],
        ["url", { "type":"home" },
          "uri", "http://example.org"]
    ]
  ]
}
```

Figure 40

The arrays in Figure 40 with the first member of "adr" represent postal addresses. In the first example, the postal address is given as an array of strings and constitutes a structured address. For components of the structured address that are not applicable, an empty string is given. Each member of that array aligns with the positions of a vCard as given in [RFC6350]. In this example, the following data corresponds to the following positional meanings:

1. post office box -- not applicable; empty string

2. extended address (e.g., apartment or suite number) -- Suite 1234

3. street address -- 4321 Rue Somewhere

4. locality (e.g., city) -- Quebec

5. region (e.g., state or province) -- QC

6. postal code -- G1V 2M2

7. country name (full name) -- Canada

The second example is an unstructured address. It uses the label attribute, which is a string containing a newline (\n) character to separate address components in an unordered, unspecified manner. Note that in this example, the structured address array is still given but that each string is an empty string.

Appendix D. Secure DNS

Section 5.3 defines the "secureDNS" member to represent secure DNS information about domain names.

DNSSEC provides data integrity for DNS through the digital signing of resource records. To enable DNSSEC, the zone is signed by one or more private keys and the signatures are stored as RRSIG records. To complete the chain of trust in the DNS zone hierarchy, a digest of each DNSKEY record (which contains the public key) must be loaded into the parent zone, stored as DS records, and signed by the parent's private key (RRSIG DS record), as indicated in "Resource Records for the DNS Security Extensions" [RFC4034]. Creating the DS records in the parent zone can be done by the registration authority "Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)" [RFC5910].

Only DS-related information is provided by RDAP, since other information is not generally stored in the registration database. Other DNSSEC-related information can be retrieved with other DNS tools such as dig.

The domain object class (Section 5.3) can represent this information using either the "dsData" or "keyData" object arrays. Client implementers should be aware that some registries do not collect or do not publish all of the secure DNS meta-information.

Appendix E. Motivations for Using JSON

This section addresses a common question regarding the use of JSON over other data formats, most notably XML.

It is often pointed out that many DNRs and one RIR support the EPP [RFC5730] standard, which is an XML serialized protocol. The logic is that since EPP is a common protocol in the industry, it follows that XML would be a more natural choice. While EPP does influence this specification quite a bit, EPP serves a different purpose, which is the provisioning of Internet resources between registries and accredited registrars and serving a much narrower audience than that envisioned for RDAP.

By contrast, RDAP has a broader audience and is designed for public consumption of data. Experience from RIRs with first generation RESTful web services for WHOIS indicate that a large percentage of clients operate within browsers and other platforms where full-blown XML stacks are not readily available and where JSON is a better fit.

Additionally, while EPP is used in much of the DNR community it is not a universal constant in that industry. And finally, EPP’s use of XML predates the specification of JSON. If EPP had been defined today, it may very well have used JSON instead of XML.

Beyond the specific DNR and RIR communities, the trend in the broader Internet industry is also switching to JSON over XML, especially in the area of RESTful web services (see [JSON_ascendancy]). Studies have also found that JSON is generally less bulky and consequently faster to parse (see [JSON_performance_study]).

Acknowledgements

This document is derived from original work on RIR responses in JSON by Byron J. Ellacott, Arturo L. Servin, Kaveh Ranjbar, and Andrew L. Newton. Additionally, this document incorporates work on DNR responses in JSON by Ning Kong, Linlin Zhou, Jiagui Xie, and Sean Shen.

The components of the DNR object classes are derived from a categorization of WHOIS response formats created by Ning Kong, Linlin Zhou, Guangqing Deng, Steve Sheng, Francisco Arias, Ray Bellis, and Frederico Neves.

Tom Harrison, Murray Kucherawy, Ed Lewis, Audric Schiltknecht, Naoki Kambe, and Maarten Bosteels contributed significant review comments and provided clarifying text. James Mitchell provided text regarding the processing of unknown JSON attributes and identified issues

leading to the remodeling of events. Ernie Dainow and Francisco Obispo provided concrete suggestions that led to a better variant model for domain names.

Ernie Dainow provided the background information on the secure DNS attributes and objects for domains, informative text on DNSSEC, and many other attributes that appear throughout the object classes of this document.

The switch to and incorporation of jCard was performed by Simon Perreault.

Olaf Kolkman and Murray Kucherawy chaired the IETF's WEIRDS working group from which this document has been created.

Authors' Addresses

Andrew Lee Newton
American Registry for Internet Numbers
3635 Concorde Parkway
Chantilly, VA 20151
United States

EMail: andy@arin.net
URI: http://www.arin.net

Scott Hollenbeck
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
United States

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