       JSON Responses for the Registration Data Access Protocol (RDAP)
                    draft-ietf-weirds-json-response-06

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

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

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

   This Internet-Draft will expire on April 06, 2014.

Copyright Notice

Table of Contents

1. Introduction

   This document describes responses in the JSON [RFC4627] format for
   the RESTful web queries as defined by the Registration Data Access
   Protocol Lookup Format [I-D.ietf-weirds-rdap-query].

   The data model for JSON responses is specified in four 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
       given query

   4. the response to an error

   The object classes represent responses for two major categories of
   data: responses returned by Regional Internet Registries (RIRs) for
   registrations data related to IP addresses, reverse DNS names, and
   Autonomous System numbers; and responses returned by Domain Name
   Registries (DNRs) for registration data related to forward DNS names.
   The following object classes are served 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 MUST understand to function correctly,
however some deployments may want to include additional object
classes to suit individual needs. Anticipating this need for
extension, Section 3.2 of this document defines a mechanism for
extending the JSON objects that are described in this document.

2. Terminology and Definitions

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

DNR: "Domain Name Registry".

LDH: "Letters, Digits, Hyphen".

member: data found with in an object as defined by JSON
                  [RFC4627].

object: a data structure as defined by JSON [RFC4627].

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".

3. Use of JSON

3.1. Signaling

Media type signaling for the JSON data specified in this document is
specified in [I-D.ietf-weirds-using-http].

3.2. Naming

Clients processing JSON [RFC4627] responses are under no obligation
to process unrecognized JSON attributes but SHOULD NOT treat them as
an error. Servers MAY insert values signified by names into the JSON
responses which are not specified in this document. Insertion of
unspecified values into JSON responses SHOULD have names prefixed
with a short identifier followed by an underscore followed by a
meaningful name. 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
[I-D.ietf-weirds-using-http].

Consider the following JSON response with JSON names, 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 name
"lunarNic_beforeOneSmallStep" to signify registrations occurring
before the first moon landing and the name
"lunarNic_harshMistressNotes" containing 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 names ignored by clients may also be used
   for future revisions to this specification.

   Clients processing JSON responses MUST be prepared for values
   specified in this document to be absent from a response as no JSON
   value listed is required to appear in a response. In other words,
   servers MAY remove values as is needed by the policies of the server
   operator.

   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.

4. **Common Data Types**

   JSON [RFC4627] defines the data types of a number, character string,
   boolean, array, object and null. This section describes the
   semantics and/or syntax reference for data types used in this
   document derived from the JSON character string.

   '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 is 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 described
in [RFC1166]. 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 2 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 JSON vCards as described in
[I-D.ietf-jcardcal-jcard]

5. Common Data Structures

This section defines common data structures used commonly in object
classes.

5.1. RDAP Conformance

   The first data structure is named "rdapConformance" and is simply 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 top most 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 should use a string
   prefixed with the appropriate identifier from the IANA prefix
   identifier registry specified in [I-D.ietf-weirds-using-http]. 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".

   Example rdapConformance structure with custom extensions noted:

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

                               Figure 4

5.2. 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" : [ "title1", "title2" ],
    "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
   "value", "rel", and "href" JSON values 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"
        }

    ]
```

5.3. Notices And Remarks

   The "notices" and "remarks" data structures take the same form. The
   "notices" structure denotes information about the service providing
   RDAP information, whereas the "remarks" structure denotes information
   about the object class it is contained within (see Section 6
   regarding object classes).

   Both are an array of objects. Each object contains an optional
   "title" string representing the title of the object, an array of
   strings named "description" for the purposes of conveying any
   descriptive text, and an optional "links" array as described in
   Section 5.2.

   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 6

   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. Servers SHOULD NOT split sentences across
   multiple strings of this array. Each string is to represent a
   semantic division in the descriptive text.

   An example of the remarks data structure:

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

                             Figure 7

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

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

5.4. Language Identifier

   This data structure is a simple JSON name/value of "lang" with a
   string containing a language identifier as described by [RFC5646].

   "lang" : "mn-Cyrl-MN"

                             Figure 8

   The 'lang' attribute may appear anywhere in an object class or data
   structure.

5.5. Events

   This data structure represents events that have occurred on an
   instance of an object class (see Section 6 regarding object classes).

   This is an example of an "events" array.

```
   "events" :
   [
     {
       "eventAction" : "registration",
       "eventActor" : "SOMEID-LUNARNIC",
       "eventDate" : "1990-12-31T23:59:60Z"
     },
     {
       "eventAction" : "last changed",
       "eventActor" : "OTHERID-LUNARNIC",
       "eventDate" : "1991-12-31T23:59:60Z"
     }
   ]
```

                               Figure 9

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

   o 'eventAction' -- a string denoting the reason for the event

   o 'eventActor' -- an optional identifier denoting the actor
      responsible for the event

   o 'eventDate' -- a string containing the time and date the event
      occurred.

   o 'links' -- see Section 5.2.

   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. If the event actor being referenced is an RDAP
   entity, the link reference SHOULD be made with "rel" of "related" and
   "type" of "application/rdap+json".

   See Section 10.2.2 for a list of values for the 'eventAction' string.
   See Appendix B regarding the various ways events can be modeled.

5.6. Status

   This data structure, named 'status', is an array of strings
   indicating the state of a registered object (see Section 10.2.1 for a
   list of values).

5.7. Port 43 Whois Server

This data structure, named 'port43', is a simple string containing
the fully-qualified host name 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.

5.8. 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:

o type - a string denoting the type of public identifier

o 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 10

5.9. An Example

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

```
   {
     "rdapConformance" :
     [
       "rdap_level_0"
     ],
     "notices" :
     [
       {
         "title" : "Content Redacted",
         "description" :
         [
           "Without full authorization, content has been redacted.",
           "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",
     "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 11

6. Object Classes

Object classes represent structures appropriate for a response from
the queries specified in [I-D.ietf-weirds-rdap-query].

Each object class contains a "links" array as specified in
Section 5.2. 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, servers SHOULD
provide a link representing a URI for that object class instance
using the "self" relationship as specified in the IANA registry
specified by [RFC5988]. As explained in Section 6.2, this may be not
always be possible for name server data. Clients MUST be able to
process object instances without a "self" link. When present,
clients MAY use the self link for caching data. Servers MAY provide
more than one "self" link for any given object instance.

Self links SHOULD contain a "type" element containing the
"application/rdap+json" media type when referencing RDAP object
instances as defined by this document. Clients SHOULD NOT assume
self links without this media type are represented as RDAP objects 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"
    }
 ]
```

6.1. 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 Lookup Format
[I-D.ietf-weirds-rdap-query]. 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 [RFC4627] with one construct, the entity object class,
to aid in the re-use of code by implementers.

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. For
illustrative purposes, it does not include rdapConformance or notices
data structures.


```
{
  "handle":"XXXX",
  "vcardArray":[
    "vcard",
    [
      ["version", {}, "text", "4.0"],
      ["fn", {}, "text", "Joe User"],
      ["n", {}, "text",
        ["User", "Joe", "", "", ["ing. jr", "M.Sc."]]
      ],
      ["bday", {}, "date-and-or-time", "--02-03"],
      ["anniversary",
        {}, "date-and-or-time", "2009-08-08T14:30:00-05:00"
      ],
      ["gender", {}, "text", "M"],
      ["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":"1990-12-31T23:59:60Z"
      }
    ],
    "asEventActor":[
      {
        "eventAction":"last changed",
        "eventDate":"1991-12-31T23:59:60Z"
      }
    ]
  }
```

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.

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

                              Figure 12

   This object has the following members:

   o handle -- a string representing an registry unique identifier of
      the entity

   o vcardArray -- a JSON vCard with the entity's contact information

   o roles -- an array of strings, each signifying the relationship an
      object would have with its closest containing object (see
      Section 10.2.3 for a list of values)

   o publicIds - see Section 5.8

   o entities - an array of entity objects as defined by this section.

   o remarks -- see Section 5.3

   o links -- see Section 5.2

   o events -- see Section 5.5

   o asEventActor -- this data structure takes the same form as the
      'events' data structure (see Section 5.5), 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.

   o status -- see Section 5.6

   o   port43 -- see Section 5.7

   The following is an example of a entity that might be served by a
   DNR. For illustrative purposes, it does not include rdapConformance
   or notices data structures.


```
{
  "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":"1990-12-31T23:59:60Z"
        },
        {
          "eventAction":"last changed",
          "eventDate":"1991-12-31T23:59:60Z",
          "eventActor":"joe@example.com"
        }
      ]
    }
```

   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.

6.2. The Nameserver Object Class

   The nameserver object class represents information regarding DNS name
   servers 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. For illustrative
purposes, it does not include rdapConformance or notices data
structures.

```
     {
       "handle" : "XXXX",
       "ldhName" : "ns1.xn--fo-5ja.example",
       "unicodeName" : "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" : "1990-12-31T23:59:60Z"
         },
         {
           "eventAction" : "last changed",
           "eventDate" : "1991-12-31T23:59:60Z",
           "eventActor" : "joe@example.com"
         }
       ]
     }
```

Figure 13

Figure 13 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 14 is an example of a nameserver object as would be found in
RIRs and some DNRs, while Figure 15 is an example of a nameserver
object as would be found in other DNRs.

The following is an example of the simplest nameserver object:

```
  {
    "ldhName" : "ns1.example.com"
  }
```

                          Figure 14

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

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

                          Figure 15

As nameservers can be modeled by some registries to be first-class
objects, they may also have an array of entities (Section 6.1)
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.

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

Figure 16

The nameserver object class has the following members:

o handle -- a string representing an registry unique identifier of
   the nameserver

o ldhName -- a string containing the LDH name of the nameserver (see
   Section 4)

o unicodeName -- a string containing a DNS Unicode name of the
   nameserver (see Section 4)

o 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

o entities -- an array of entity objects as defined by Section 6.1.

o status - see Section 5.6

o remarks - see Section 5.3

o links - see Section 5.2

o port43 - see Section 5.7

o events - see Section 5.5

6.3. 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:

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

The following is a description of the members of this object:

o handle -- a string representing a registry unique identifier of
   the domain object instance

o ldhName -- a string describing a domain name in LDH form as
   described in Section 4

o unicodeName -- a string containing a domain name with U-labels as
   described in Section 4

o 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 10.2.4 for a list of suggested variant
      relations).

   * idnTable -- the name of the 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 4).

   o nameservers -- an array of nameserver objects as defined by
      Section 6.2

   o secureDNS -- an object with the following members:

      * zoneSigned -- boolean 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 life time
         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 RFC 4034 RFC 4034 [RFC4034] in
            presentation format

         + algorithm -- an integer as specified by the algorithm field
            of a DNS DS record as specified 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 5.5

         + links - see Section 5.2

      * 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 RFC 4034 [RFC4034] in
            presentation format

         + events - see Section 5.5

         + links - see Section 5.2

      See Appendix D for background information on these objects.

   o entities -- an array of entity objects as defined by Section 6.1.

   o status - see Section 5.6

   o publicIds - see Section 5.8

   o remarks - see Section 5.3

   o links - see Section 5.2

   o port43 - see Section 5.7

   o events - see Section 5.5

   The following is an example of a JSON domain object representing a
   reverse DNS delegation point that might be served by an RIR. For
   illustrative purposes, it does not include rdapConformance or notices
   data structures.

   {
     "handle" : "XXXX",
     "ldhName" : "192.in-addr.arpa",
     "nameServers" :
     [
       { "ldhName" : "ns1.rir.example" },
       { "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" : "1990-12-31T23:59:60Z"
        },
        {
          "eventAction" : "last changed",
          "eventDate" : "1991-12-31T23:59:60Z",
          "eventActor" : "joe@example.com"
        }
      ],
      "entities" :
      [
        {
```

```
            "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" : "1990-12-31T23:59:60Z"
          },
          {
            "eventAction" : "last changed",
            "eventDate" : "1991-12-31T23:59:60Z",
            "eventActor" : "joe@example.com"
          }
        ]
      }
    ]
  }
```

   The following is an example of a JSON domain object representing a
   forward DNS delegation point that might be served by a DNR. For
   illustrative purposes, it does not include rdapConformance or notices
   data structures.

```
  {
    "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" : "foeo.example"
          }
        ]
      },
      {
        "relation" : [ "unregistered", "restricted registration" ],
        "idnTable": ".EXAMPLE Swedish",
        "variantNames" :
        [
          {
            "ldhName": "xn--fo-8ja.example",
            "unicodeName" : "foo.example"
          }
        ]
      }
    ],
    "status" : [ "locked", "transfer prohibited" ],
    "publicIds":[
      {
        "type":"ENS_Auth ID",
        "identifier":"1234567890"
      }
    ],
    "nameServers" :
    [
      {
        "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" : "1990-12-31T23:59:60Z"
            },
            {
              "eventAction" : "last changed",
              "eventDate" : "1991-12-31T23:59:60Z"
            }
          ]
        },
        {
          "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" : "1990-12-31T23:59:60Z"
      },
      {
        "eventAction" : "last changed",
        "eventDate" : "1991-12-31T23:59:60Z"
      }
    ]
  }
],
"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" : "1990-12-31T23:59:60Z"
     },
     {
       "eventAction" : "last changed",
       "eventDate" : "1991-12-31T23:59:60Z",
       "eventActor" : "joe@example.com"
     },
     {
       "eventAction" : "transfer",
       "eventDate" : "1991-12-31T23:59:60Z",
       "eventActor" : "joe@example.com"
     },
     {
       "eventAction" : "expiration",
       "eventDate" : "2016-12-31T23:59:60Z",
       "eventActor" : "joe@example.com"
     }
   ],
   "entities" :
   [
     {
       "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" : "1990-12-31T23:59:60Z"
        },
        {
          "eventAction" : "last changed",
          "eventDate" : "1991-12-31T23:59:60Z"
        }
      ]
    }
  ]
}
```

6.4. 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
   [I-D.ietf-weirds-rdap-query]. 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:


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


   The following is an example of the JSON object for the network
   registration information. For illustrative purposes, it does not
   include rdapConformance or notices data structures.


```
{
```

```
      "handle" : "XXXX-RIR",
      "startAddress" : "2001:db8::0",
      "endAddress" : "2001:db8::0:FFFF:FFFF:FFFF:FFFF:FFFF",
      "ipVersion" : "v6",
      "name": "NET-RTR-1",
      "type" : "DIRECT ALLOCATION",
      "country" : "AU",
      "parentHandle" : "YYYY-RIR",
      "status" : [ "allocated" ],
      "remarks" :
      [
        {
          "description" :
          [
            "She sells sea shells down by the sea shore.",
            "Originally written by Terry Sullivan."
          ]
        }
      ],
      "links" :
      [
        {
          "value" : "http://example.ent/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" : "1990-12-31T23:59:60Z"
        },
        {
          "eventAction" : "last changed",
          "eventDate" : "1991-12-31T23:59:60Z"
        }
      ],
      "entities" :
      [
        {
```

```
            "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" : "1990-12-31T23:59:60Z"
          },
          {
            "eventAction" : "last changed",
            "eventDate" : "1991-12-31T23:59:60Z"
          }
        ]
      }
    ]
  }
```

The following is a description of the members of this object:

o handle -- a string representing an RIR unique identifier of the
   network registration

o startAddress -- the starting IP address of the network, either
   IPv4 or IPv6

o endAddress -- the ending IP address of the network, either IPv4 or
   IPv6

o ipVersion -- a string signifying the IP protocol version of the
   network: "v4" signifying an IPv4 network, "v6" signifying an IPv6
   network

o name -- an identifier assigned to the network registration by the
   registration holder

o type -- a string containing an RIR-specific classification of the
   network

o country -- a string containing the name of the 2 character country
   code of the network

o parentHandle -- a string containing an RIR-unique identifier of
   the parent network of this network registration

o status -- an array of strings indicating the state of the IP
   network

o entities -- an array of entity objects as defined by Section 6.1.

o remarks - see Section 5.3

o links - see Section 5.2

o port43 - see Section 5.7

o events - see Section 5.5

6.5. 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
[I-D.ietf-weirds-rdap-query]. 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.
For illustrative purposes, it does not include rdapConformance or
notices data structures.

```
{
  "handle" : "XXXX-RIR",
  "startAutnum" : 10,
  "endAutnum" : 15,
  "name": "AS-RTR-1",
  "type" : "DIRECT ALLOCATION",
  "status" : [ "allocated" ],
  "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" : "1990-12-31T23:59:60Z"
        },
        {
          "eventAction" : "last changed",
          "eventDate" : "1991-12-31T23:59:60Z"
        }
      ],
      "entities" :
      [
        {
          "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" : "1990-12-31T23:59:60Z"
         },
         {
           "eventAction" : "last changed",
           "eventDate" : "1991-12-31T23:59:60Z"
         }
       ]
     }
   ]
 }
```

The following is a description of the members of this object:

o handle -- a string representing an RIR-unique identifier of the
   autnum registration

o startAutnum -- a number representing the starting number [RFC5396]
   in the block of autonomous system numbers

o endAutnum -- a number representing the ending number [RFC5396] in
   the block of autonomous system numbers

o name -- an identifier assigned to the autnum registration by the
   registration holder

o type -- a string containing an RIR-specific classification of the
   autnum

o status -- an array of strings indicating the state of the autnum

o country -- a string containing the name of the 2 character country
   code of the autnum

o entities -- an array of entity objects as defined by Section 6.1.

o remarks - see Section 5.3

o links - see Section 5.2

o port43 - see Section 5.7

o events - see Section 5.5

7. 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" followed by an array of strings named
"description".

This is an example of the common response body. For illustrative
purposes, it does not include rdapConformance or notices data
structures.

```
{
  "errorCode": 418,
  "title": "Your beverage choice is not available",
  "description":
  [
    "I know coffee has more ummppphhh.",
    "Sorry, dude!"
  ]
}
```

                               Figure 17

A client MAY simply use the HTTP response code as the server is not
required to include error data in the response body. However, if a
client wishes to parse the error data, it SHOULD first check that the
Content-Type header contains the appropriate media type.

This is an example of the common response body with and
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 18

8. Responding to Help Queries

   The appropriate response to /help queries as defined by
   [I-D.ietf-weirds-rdap-query] is to use the notices structure as
   defined in Section 5.3.

   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 19

9. Responding To Searches

   [I-D.ietf-weirds-rdap-query] 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:

   o for /domains searches, the array is "domainSearchResults"

   o for /nameservers searches, the array is "nameserverSearchResults"

   o 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" :
     [
       {
         "handle" : "1-XXXX",
         "ldhName" : "1.example.com",
         ...
       },
       {
         "handle" : "2-XXXX",
         "ldhName" : "2.example.com",
         ...
       }
     ]
   }
```

                         search_response_example

   In addition to the arrays, the response object may contain a member
   named "searchResultsTruncated" which is a boolean. When this value
   is set to true, it indicates the accompanying array did not contain
   all of the data that matched the search criteria. When this value is
   true, it is recommended that servers include a textual description
   for the truncation in a notices structure.

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

```
{
  "rdapConformance" :
  [
    "rdap_level_0"
  ],
  "notices" :
  [
    {
      "title" : "Search Policy",
      "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"
        }
      ]
    }
  ],
  "searchResultsTruncated" : true
  "domainSearchResults" :
  [
    ...
  ]
}
```

                    search_response_truncated_example

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 [RFC4627]

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

Published specification: [[this document]]

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/ [1]

Person & email address to contact for further information: Andy Newton <andy@hxr.us>

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

This section requests that the IANA establish an RDAP JSON Values Registry for use in the status (Section 5.6), role (Section 6.1), event action (Section 5.5), and domain variant relation (Section 6.3) fields specified in RDAP.

Each entry in the registry should contain 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:

    * 'status' - denotes a value for the 'status' object member as defined by Section 5.6.

     * 'role' - denotes a value for the 'role' array as defined in
        Section 6.1.

     * 'event action' - denotes a value for an event action as
        defined in Section 5.5.

     * 'domain variant relation' - denotes a relationship between a
        domain and a domain variant as defined in Section 6.3.

   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 should be governed by a designated expert or multiple
   designated experts. Registration of values into this registry should
   be accomplished by providing the information above to the designated
   expert(s).

   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 can 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.

10.2.1. Status

   This section registers the following values into the RDAP JSON Values
   Registry:

   1.

          * 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: Andrew Newton

          * Registrant Contact Information: andy@hxr.us

   2.

          * Value: renew prohibited

          * Type: status

          * Description: Renewal or reregistration of the object instance
             is forbidden.

          * Registrant Name: Andrew Newton

          * Registrant Contact Information: andy@hxr.us

   3.

          * Value: update prohibited

          * Type: status

          * Description: Updates to the object instance are forbidden.

          * Registrant Name: Andrew Newton

          * Registrant Contact Information: andy@hxr.us

4.

* 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: Andrew Newton

* Registrant Contact Information: andy@hxr.us

5.

* 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: Andrew Newton

* Registrant Contact Information: andy@hxr.us

6.

* 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: Andrew Newton

* Registrant Contact Information: andy@hxr.us

7.

* 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: Andrew Newton

* Registrant Contact Information: andy@hxr.us

8.

* Value: redacted

* Type: status

* Description: Some of the information of the object instance
  has not been made available. This is most commonly applied
  to entities.

* Registrant Name: Andrew Newton

* Registrant Contact Information: andy@hxr.us

9.

* 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: Andrew Newton

* Registrant Contact Information: andy@hxr.us

10.

* 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: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

   11.

      * 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 EPP 'OK' status.

      * Registrant Name: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

   12.

      * Value: inactive

      * Type: status

      * Description: The object instance is not in use. See
         'active'.

      * Registrant Name: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

   13.

      * Value: locked

      * Type: status

      * Description: Changes to the object instance cannot be made,
         including the assocation of other object instances.

      * Registrant Name: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

   14.

* 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: Andrew Newton

* Registrant Contact Information: andy@hxr.us

15.

* 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: Andrew Newton

* Registrant Contact Information: andy@hxr.us

16.

* 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: Andrew Newton

* Registrant Contact Information: andy@hxr.us

17.

* 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: Andrew Newton

* Registrant Contact Information: andy@hxr.us

18.

* 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 in no
  longer published in DNS but has not yet been purged from the
  registry database.

* Registrant Name: Andrew Newton

* Registrant Contact Information: andy@hxr.us

10.2.2. Event Actions

This section registers the following values into the RDAP JSON Values
Registry:

1.

* Value: registration

* Type: event action

* Description: The object instance was initially registered.

* Registrant Name: Andrew Newton

* Registrant Contact Information: andy@hxr.us

2.

* Value: reregistration

* Type: event action

* Description: The object instance was registered subsequently
  to initial registration.

* Registrant Name: Andrew Newton

* Registrant Contact Information: andy@hxr.us

3.

* Value: last changed

* Type: event action

* Description: An action noting when the information in the object instance was last changed.

* Registrant Name: Andrew Newton

* Registrant Contact Information: andy@hxr.us

4.

* Value: expiration

* Type: event action

* Description: The object instance has been removed or will be removed at a pre-determined date and time from the registry.

* Registrant Name: Andrew Newton

* Registrant Contact Information: andy@hxr.us

5.

* Value: deletion

* Type: event action

* Description: The object instance was removed from the registry at a point in time that was not pre-determined.

* Registrant Name: Andrew Newton

* Registrant Contact Information: andy@hxr.us

6.

* Value: reinstantiation

* Type: event action

* Description: The object instance was reregistered after having been removed from the registry.

   * Registrant Name: Andrew Newton

   * Registrant Contact Information: andy@hxr.us

   7.

      * Value: transfer

      * Type: event action

      * Description: The object instance was transferred from one
         registrant to another.

      * Registrant Name: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

   8.

      * Value: locked

      * Type: event action

      * Description: The object instance was locked (see the 'locked'
         status).

      * Registrant Name: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

   9.

      * Value: unlocked

      * Type: event action

      * Description: The object instance was unlocked (see the
         'locked' status).

      * Registrant Name: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

10.2.3. **Roles**

   This section registers the following values into the RDAP JSON Values
   Registry:

   1.

         * 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: Andrew Newton

         * Registrant Contact Information: andy@hxr.us

   2.

         * Value: technical

         * Type: role

         * Description: The entity object instance is a technical
            contact for the registration.

         * Registrant Name: Andrew Newton

         * Registrant Contact Information: andy@hxr.us

   3.

         * Value: administrative

         * Type: role

         * Description: The entity object instance is an administrative
            contact for the registration.

         * Registrant Name: Andrew Newton

         * Registrant Contact Information: andy@hxr.us

   4.

         * Value: abuse

        * Type: role

        * Description: The entity object instance handles network abuse
           issues on behalf of the registrant of the registration.

        * Registrant Name: Andrew Newton

        * Registrant Contact Information: andy@hxr.us

   5.

        * Value: billing

        * Type: role

        * Description: The entity object instance handles payment and
           billing issues on behalf of the registrant of the
           registration.

        * Registrant Name: Andrew Newton

        * Registrant Contact Information: andy@hxr.us

   6.

        * Value: registrar

        * Type: role

        * Description: The entity object instance represents the
           authority responsible for the registration in the registry.

        * Registrant Name: Andrew Newton

        * Registrant Contact Information: andy@hxr.us

   7.

        * 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: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

   8.

      * Value: sponsor

      * Type: role

      * Description: The entity object instance represents a domain
         policy sponsor, such as an ICANN approved sponsor.

      * Registrant Name: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

   9.

      * Value: proxy

      * Type: role

      * Description: The entity object instance represents a proxy
         for another entity object, such as a registrant.

      * Registrant Name: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

   10.

      * Value: notifications

      * Type: role

      * Description: An entity object instance designated to receive
         notifications about association object instances.

      * Registrant Name: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

   11.

      * Value: noc

      * Type: role

   * Description: The entity object instance handles
     communications related to a network operations center (NOC).

   * Registrant Name: Andrew Newton

   * Registrant Contact Information: andy@hxr.us

[10.2.4](#). Variant Relations

   This section registers the following values into the RDAP JSON Values
   Registry:

   1.

      * Value: registered

      * Type: domain variant relation

      * Description: The variant names are registered in the registry.

      * Registrant Name: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

   2.

      * Value: unregistered

      * Type: domain variant relation

      * Description: The variant names are not found in the registry.

      * Registrant Name: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

   3.

      * Value: registration restricted

      * Type: domain variant relation

      * Description: Registration of the variant names is restricted
        to certain parties or within certain rules.

      * Registrant Name: Andrew Newton

      * Registrant Contact Information: andy@hxr.us

4.

   * Value: open registration

   * Type: domain variant relation

   * Description: Registration of the variant names is available to
     generally qualified registrants.

   * Registrant Name: Andrew Newton

   * Registrant Contact Information: andy@hxr.us

5.

   * Value: conjoined

   * Type: domain variant relation

   * Description: Registration of the variant names occurs
     automatically with the registration of the containing domain
     registration.

   * Registrant Name: Andrew Newton

   * Registrant Contact Information: andy@hxr.us

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 [RFC4627] to
   prevent code injection.

12. Internationalization Considerations

12.1. Character Encoding

   The default text encoding for JSON and XML responses in RDAP is UTF-8
   [RFC3629], and all servers and clients MUST support UTF-8. Servers
   and clients MAY optionally support other character encodings.

12.2. URIs and IRIs

   [I-D.ietf-weirds-using-http] defines the use of URIs and IRIs in
   RDAP.

12.3. **Language Tags**

   Section 5.4 defines the use of language tags in the JSON responses
   defined in this document.

12.4. **Internationalized Domain Names**

   Internationalized Domain Names (IDNs) are denoted in this
   specification by the separation of DNS names in LDH form and Unicode
   form (see Section 4). Representation of IDNs in registries is
   described by the "variants" object in Section 6.3 and the suggested
   values listed in Section 10.2.4.

13. **Privacy Considerations**

   This specification suggests status values to denote contact and
   registrant information that has been marked as private and/or has
   been redacted or obscured. See Section 10.2.1 for the list of status
   values. See Appendix A.1 on guidance to apply those values to
   contacts and registrants.

14. **Contributing Authors and 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 word 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, and Guangqing Deng, Steve Sheng and Francisco Arias, Ray
   Bellis, and Frederico Neves.

   Ed Lewis 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 DNSSEC
   attributes and objects for domains, informative text on DNSSEC, and
   many other attributes that appear throughout the object classes of
   this draft.

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

15. References

15.1. Normative References

[RFC0791]   Postel, J., "Internet Protocol", STD 5, RFC 791, September
            1981.

[RFC1166]   Kirkpatrick, S., Stahl, M., and M. Recker, "Internet
            numbers", RFC 1166, July 1990.

[RFC2616]   Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
            Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
            Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3339]   Klyne, G., Ed. and C. Newman, "Date and Time on the
            Internet: Timestamps", RFC 3339, July 2002.

[RFC3629]   Yergeau, F., "UTF-8, a transformation format of ISO
            10646", STD 63, RFC 3629, November 2003.

[RFC3986]   Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
            Resource Identifier (URI): Generic Syntax", STD 66, RFC
            3986, January 2005.

[RFC4034]   Arends, R., Austein, R., Larson, M., Massey, D., and S.
            Rose, "Resource Records for the DNS Security Extensions",
            RFC 4034, March 2005.

[RFC4343]   Eastlake, D., "Domain Name System (DNS) Case Insensitivity
            Clarification", RFC 4343, January 2006.

[RFC4627]   Crockford, D., "The application/json Media Type for
            JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC5322]   Resnick, P., Ed., "Internet Message Format", RFC 5322,
            October 2008.

[RFC5396]   Huston, G. and G. Michaelson, "Textual Representation of
            Autonomous System (AS) Numbers", RFC 5396, December 2008.

[RFC5646]   Phillips, A. and M. Davis, "Tags for Identifying
            Languages", BCP 47, RFC 5646, September 2009.

[RFC5890]   Klensin, J., "Internationalized Domain Names for
            Applications (IDNA): Definitions and Document Framework",
            RFC 5890, August 2010.

   [RFC5952]   Kawamura, S. and M. Kawashima, "A Recommendation for IPv6
               Address Text Representation", RFC 5952, August 2010.

   [RFC5988]   Nottingham, M., "Web Linking", RFC 5988, October 2010.

   [I-D.ietf-jcardcal-jcard]
               Kewisch, P., "jCard: The JSON format for vCard", draft-
               ietf-jcardcal-jcard-06 (work in progress), September 2013.

   [I-D.ietf-weirds-using-http]
               Newton, A., Ellacott, B., and N. Kong, "HTTP usage in the
               Registration Data Access Protocol (RDAP)", draft-ietf-
               weirds-using-http-07 (work in progress), July 2013.

   [I-D.ietf-weirds-rdap-query]
               Newton, A. and S. Hollenbeck, "Registration Data Access
               Protocol Query Format", draft-ietf-weirds-rdap-query-07
               (work in progress), October 2013.

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

15.2. Informative References

   [RFC3912]   Daigle, L., "WHOIS Protocol Specification", RFC 3912,
               September 2004.

   [RFC5730]   Hollenbeck, S., "Extensible Provisioning Protocol (EPP)",
               STD 69, RFC 5730, August 2009.

   [RFC5910]   Gould, J. and S. Hollenbeck, "Domain Name System (DNS)
               Security Extensions Mapping for the Extensible
               Provisioning Protocol (EPP)", RFC 5910, May 2010.

   [RFC6350]   Perreault, S., "vCard Format Specification", RFC 6350,
               August 2011.

   [RFC6839]   Hansen, T. and A. Melnikov, "Additional Media Type
               Structured Syntax Suffixes", RFC 6839, January 2013.

   [JSON_acendancy]
               MacVittie, ., "The Stealthy Ascendancy of JSON", 04 2011.

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

   [JSON_performance_study]
                Montana State University - Bozeman, Montana State
                University - Bozeman, Montana State University - Bozeman,
                Montana State University - Bozeman, "Comparison of JSON
                and XML Data Interchange Formats: A Case Study", 2009.

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 10.2.3
   be used.

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

   {
     ...
     "entities" :
     [
       {
         "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", "admin" ],
        "remarks" :
        [
          {
            "description" :
            [
              "She sells sea shells down by the sea shore.",
              "Originally written by Terry Sullivan."
            ]
          }
        ],
        "events" :
        [
          {
            "eventAction" : "registration",
            "eventDate" : "1990-12-31T23:59:60Z"
          },
          {
            "eventAction" : "last changed",
            "eventDate" : "1991-12-31T23:59:60Z"
          }
        ]
      }
    ]
  }
```

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 10.2.1).

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

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

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 5.8) represents that information.

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

```
{
  ...
  "entities":[
    {
      "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"
          }
        ]
      }
    ]
  }
```

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 5.5) is
   used without the 'eventActor' object member.

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

```
   "events" :
   [
     {
       "eventAction" : "registration",
       "eventDate" : "1990-12-31T23:59:60Z"
     }
   ]
```

                               Figure 20

   For the second use case, the 'events' data structure (Section 5.5) 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" : "1990-12-31T23:59:60Z"
  }
]
```

                           Figure 21

For the third use case, the 'asEventActor' array is used when an
entity (Section 6.1) 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.

```
{
  "handle" : "XXXX",
  "ldhName" : "foo.example",
  "status" : [ "locked", "transfer Prohibited" ],
  ...
  "entities" :
  [
    {
      "handle" : "XXXX",
      ...
      "asEventActor" :
      [
        {
          "eventAction" : "last changed",
          "eventDate" : "1990-12-31T23:59:60Z"
        }
      ]
    }
  ]
}
```

Appendix C. Structured vs Unstructured Addresses

The entity (Section 6.1) object class uses jCard
[I-D.ietf-jcardcal-jcard] 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."]]
      ],
      ["bday", {}, "date-and-or-time", "--02-03"],
      ["anniversary",
        {}, "date-and-or-time", "2009-08-08T14:30:00-05:00"
      ],
      ["gender", {}, "text", "M"],
      ["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 22

    The arrays in Figure 22 with the first member of "adr" represent
    postal addresses. In the first example, the postal address is given
    as a 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 6.3 defines the "secureDNS" member to represent secure DNS
   information about domain names.

   DNSSEC provides data integrity for DNS through digital signing of
   resource records. To enable DNSSEC, the zone is signed by one or
   more private keys and the signatures 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 Delegation Signer (DS) records and
   signed by the parent's private key (RRSIG DS record), "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 6.3](#)) 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 serves 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 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_acendancy]). Studies
have also found that JSON is generally less bulky and consequently
faster to parse (see [JSON_performance_study]).

Appendix F. Changelog

Initial -00 Adopted as working group document 2012-September-18.

-01

    Minor spelling corrections. Changed "Registry Data" to
    "Registration Data" for the sake of consistency.

    Transitioned to RFC 5988 links and relationship types from our
    own custom "uris" structure.

Some examples had 'status' as a string. Those have been
corrected as 'status' is always an array of strings.

Domain variants can now have a multi-valued relationship with
domain registrations.

"names" in the entity object class was changed to
"entityNames".

Some IP address examples change to IPv6.

Change phone number examples and added reference to E.164.

Added section on motivations for using JSON.

Added error response body section.

Added JSON naming section.

Added common data structures section.

Added the IANA Considerations section and the media type
registration.

Added 'lang' name/value.

Added internationalization considerations section.

-02

Removed level from media type registration.

Textual changes as given by Ed Lewis.

Fixed object class linking example noted by Francisco Obispo

Fixed a lot of other examples called out by Alex Sergeyev

Added a note that JSON names are case sensitive

Added 'status' to IP networks as suggested by Alex Sergeyev

-03

Added jCard verbiage and examples and deleted overlapping
contact information and the appendix on postal addresses

Removed the IANA considerations as they have been moved to
another document

Changed the remarks structure to be like notices

Reordering and rewording some of the sections so they flow
better

Added note about object class "self" links

Changed ipAddresses in nameserver object class to separate out
v6 from v4

Changed IP network version identifier from integer to string to
be more consistent with ipAddresses identifier in nameserver
object classes

Changed DNS names to LDH names and Unicode names

Modified the definition of 'conjoined' variant relationship so
it was circular

Added 'proxy', 'private', 'redacted', and 'obscured' status
values (most useful for entities).

Added a privacy considerations section

Added a security considerations section

Added 'reseller' and 'sponsor' to the list of entity roles

Added the 'events' common data structure

Added 'asEventActor' to entities

Added appendix on event modeling

Removed the subclasses/superclassing between RIRs/DNRs for
entity and domain object classes

Change suggested status/relation/etc values to be case/spacing
consistent

Normalized some of the definitions of object class members

Modifying the JSON signaling section to reference the guidance
in draft-ietf-weirds-using-http

Changed the text regarding the process of unknown JSON
attributes

-04

'description' removed from IP network and autnum because it is
redundant with the remarks structure.

Added 'entities' array to nameservers.

Added 'status' to autnum.

Added 'publicIds' to entity and domain.

Added embedded entities to the entity object class.

Added 'idnTable' to variants objects in domain object class.

Changed the numbers for startNum and endNum in autnum to
numbers instead of strings.

Added an example for error response with full rdapConformance
and notices.

Added a section discussing help.

Changed entities to use vcardArray and changed the examples to
be current with jCard.

Added a section on structured vs unstructured addresses.

Added associated to the list of status values.

Added a secure DNS section changed the 'delegationKey' object
into the 'secureDNS' object.

Changed the suggested values to an IANA registry.

Added 'proxy' to the list of entity roles.

-05

Added IANA registration for RDAP JSON Media Type

Added 'associated' status type. This was done earlier but got
dropped during a reorganization of the document.

Added the following status types:

active

inactive

locked

pending create

pending renew

pending update

pending transfer

pending delete

renew prohibited

Added the following event actions:

locked

unlocked

Added the following roles:

notifications

noc

Changed the 'tech' role to 'technical'

Many document reference changes.

Many examples have been fixed.

Added links to dsData and keyData.

Changed flags and protocols to integers in keyData.

Added 'entities' to the specified list for autnum.

Added SHOULD/SHOULD NOT language about using "type": "application/rdap+json" for self links.

Added 'port43' to ip networks and autnum.

-06

Fix search response example.

Change the returned search arrays to 'domainSearchResults',
'entitySearchResults', and 'nameserverSearchResults'.

Authors' Addresses

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

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

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

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