Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 26, 2018

A. Newton
ARIN

January 22, 2018

A Description of RDAP JSON Messages Using JSON Content Rules
draft-newton-rdap-jcr-06

Abstract

This document describes the JSON responses in the Registration Data Access Protocol with the formal notation of JSON Content Rules.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 26, 2018.

Table of Contents

1. Introduction

The JSON [RFC7159] responses of the Registration Data Access Protocol [RFC7483] are officially defined with English prose. Those definitions contain imprecise or ambiguous JSON structures and require lengthy, tedious examples in the attempt to offer clarification. The English prose can be difficult for non-native English readers, and the examples create their own confusion.

This document describes the JSON found in RDAP with JSON Content Rules [I-D.newton-json-content-rules] (JCR).

JCR overcomes some of the obstacles of describing JSON with English prose, reducing the tediousness of the prose and accompanying lengthy examples to understandable data structures. Additionally, JCR has mechanisms which can be used by software developers to create test harnesses and technology compatibility kits.

Though this document describes all of the JSON found in [RFC7483], it presents the structures in a different order. The rules defined here

use the JCR mixin style of specification, where common structures are defined in group rules instead of separately, distinct objects.

2. Response

[RFC7483] describes ten distinct JSON response: five entity class response, an error response, a help response, and three search responses.

```
@{root} $entity_response = {
    $response_mixin,
    $entity_mixin
}

@{root} $nameserver_response = {
    $response_mixin,
    $nameserver_mixin
}

@{root} $domain_response = {
    $response_mixin,
    $domain_mixin
}

@{root} $network_response = {
    $response_mixin,
    $network_mixin
}

@{root} $autnum_response = {
    $response_mixin,
    $autnum_mixin
}

@{root} $error_response = {
    $response_mixin,
    $error_mixin
}

@{root} $help_response = {
    $response_mixin,
    $lang ?
}

@{root} $domainSearch_response = {
    $response_mixin,
    $lang ?,
    $domainSearchResult
```

```
}

@{root} $nameserverSearch_response = {
    $response_mixin,
    $lang ?,
    $nameserverSearchResult
}

@{root} $entitySearch_response = {
    $response_mixin,
    $lang ?,
    $entitySearchResult
}
```

Figure 1

All of the responses have a common set of object members described by response_mixin.

```
$response_mixin = (
   $rdapConformance ?,
   "notices" : $notices ?
)
```

Figure 2

3. Object Classes

The primary data structures in RDAP are called object classes. These are first order object instances with identifiers. They are JSON objects which contain other JSON data types.

3.1. Entity Object Class

The Entity object class represents persons or organizations.

```
$entities = "entities" : [ $entity_oc * ]

$entity_oc = {
   $entity_mixin
}

$entity_mixin = (
   "objectClassName" : "entity",
   $common_mixin,
   "vcardArray"      : [ "vcard", $vcard * ] ?,
   "asEventActor"    : [ $noActorEvent * ] ?,
   $roles ?,
   $publicIds ?,
   $entities ?,
   "networks"        : [ $network_oc * ] ?,
   "autnums"         : [ $autnum_oc * ] ?
)

$roles = "roles" : [ string * ]
```

Figure 3

The Entity object class incorporates jCard [RFC7095] (vCard in JSON) for contact information.

Each jCard is made up of vCard properties in an array. The very first property must be "version", and the "fn" property must appear once and only once.

```
$vcard = [
   [ "version", {}, "text", "4.0" ],
   $vcard_not_fn_properties * ,
   [ "fn", { $vcard_type_param?, $vcard_language_param? ,
             $vcard_altid_param? , $vcard_pid_param? ,
             $vcard_pref_param?, $vcard_any_param? },
             "text", string ],
   $vcard_not_fn_properties *
]

$vcard_not_fn_properties =
   (
     $vcard_source |
     $vcard_kind |
     $vcard_n |
     $vcard_nicname |
     $vcard_photo |
     $vcard_bday |
     $vcard_anniversary |
     $vcard_gender |
     $vcard_adr |
     $vcard_tel |
     $vcard_email |
     $vcard_impp |
     $vcard_lang |
     $vcard_tz |
     $vcard_geo |
     $vcard_title |
     $vcard_role |
     $vcard_logo |
     $vcard_org |
     $vcard_member |
     $vcard_related |
     $vcard_categories |
     $vcard_note |
     $vcard_prodid |
     $vcard_rev |
     $vcard_sound |
     $vcard_uid |
     $vcard_clientpidmap |
     $vcard_url |
     $vcard_key |
     $vcard_fburl |
     $vcard_caladruri |
     $vcard_caluri |
     $vcard_x10
   )
```

Figure 4

Each vCard property is composed of a property name, a set of parameters, and data values.

```
$vcard_source = [ "source", $vcard_source_prop ]
$vcard_source_prop = (
  { $vcard_pid_param? , $vcard_pref_param?, $vcard_altid_param? ,
    $vcard_mediatype_param?, $vcard_any_param? } ,
  "uri", uri
)

$vcard_kind = [ "kind", $vcard_kind_prop ]
$vcard_kind_prop = (
  { $vcard_any_param? },
  "text", $vcard_kind_type
)

$vcard_n = [ "n", $vcard_n_prop ]
$vcard_n_prop = (
  { $vcard_sort_as_param? , $vcard_language_param? ,
    $vcard_altid_param? , $vcard_any_param? },
  "text", $vcard_string_type_array
)

$vcard_nicname = [ "nicname", $vcard_nicname_prop ]
$vcard_nicname_prop = (
  { $vcard_type_param? , $vcard_language_param? ,
    $vcard_altid_param? , $vcard_pid_param? ,
    $vcard_pref_param? , $vcard_any_param ? },
  "text", $vcard_string_type_array
)

$vcard_photo = [ "photo", $vcard_photo_prop ]
$vcard_photo_prop = (
  { $vcard_altid_param? , $vcard_type_param? ,
    $vcard_mediatype_param? , $vcard_pref_param? ,
    $vcard_pid_param? , $vcard_any_param? },
  "uri", uri
)

$vcard_bday = [ "bday", $vcard_bday_prop ]
$vcard_bday_prop = (
  ( { $vcard_language_param?, $vcard_altid_param?,
      $vcard_calscale_param?, $vcard_any_param? },
    "text", string
  ) |
  ( { $vcard_altid_param?, $vcard_calscale_param?,
```

```
        $vcard_any_param? },
      "date-and-or-time", $vcard_date_and_or_time_type
  )
)

$vcard_anniversary = [ "anniversary", $vcard_anniversary_prop ]
$vcard_anniversary_prop = (
  { $vcard_altid_param?, $vcard_calscale_param?, $vcard_any_param? },
  "date-and-or-time", $vcard_date_and_or_time_type
)

$vcard_gender = [ "gender", $vcard_gender_prop ]
$vcard_gender_prop = (
  { $vcard_any_param? },
  "text",
  ( $vcard_gender_type | [ $vcard_gender_type, string + ] )
)

$vcard_adr = [ "adr", $vcard_adr_prop ]
$vcard_adr_prop = (
  { $vcard_label_param? , $vcard_language_param? ,
    $vcard_geo_param ? , $vcard_tz_param? ,
    $vcard_altid_param? , $vcard_pid_param? ,
    $vcard_pref_param? , $vcard_type_param? ,
    $vcard_any_param? },
  "text", $vcard_string_type_array
)

$vcard_tel = [ "tel", $vcard_tel_prop ]
$vcard_tel_prop = (
  ( { $vcard_type_param? , $vcard_pid_param? , $vcard_pref_param? ,
      $vcard_altid_param?, $vcard_any_param? }, "text", string
  ) |
  ( { $vcard_type_param? , $vcard_pid_param? , $vcard_pref_param? ,
      $vcard_altid_param?, $vcard_mediatype_param?,
      $vcard_any_param? },
    "uri", uri..tel
  )
)

$vcard_email = [ "email", $vcard_email_prop ]
$vcard_email_prop = (
  { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param? ,
    $vcard_altid_param? , $vcard_any_param ? },
  "text", string
)

$vcard_impp = [ "impp", $vcard_impp_prop ]
```

```
   $vcard_impp_prop = (
     { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param? ,
       $vcard_mediatype_param? , $vcard_altid_param? ,
       $vcard_any_param },
     "uri", uri..impp
   )

   $vcard_lang = [ "lang", $vcard_lang_prop ]
   $vcard_lang_prop = (
     { $vcard_pid_param? , $vcard_pref_param? , $vcard_altid_param? ,
       $vcard_type_param? , $vcard_any_param? },
     "language-tag", $vcard_language_tag_type
   )

   $vcard_tz = [ "tz", $vcard_tz_prop ]
   $vcard_tz_prop = (
     { $vcard_altid_param?, $vcard_pid_param? , $vcard_pref_param? ,
       $vcard_type_param? , $vcard_mediatype_param?,
       $vcard_any_param? },
     ( ( "text", string ) |
     ( "uri" , uri ) |
     ( "utc-offset" , $vcard_utc_offset_type ) )
   )

   $vcard_geo = [ "geo", $vcard_geo_prop ]
   $vcard_geo_prop = (
     { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param? ,
       $vcard_mediatype_param? , $vcard_altid_param? ,
       $vcard_any_param? },
     "uri", uri..geo
   )

   $vcard_title = [ "title", $vcard_title_prop ]
   $vcard_title_prop = (
     { $vcard_language_param? , $vcard_pid_param? , $vcard_pref_param? ,
       $vcard_altid_param? , $vcard_type_param? , $vcard_any_param? },
     "text", string
   )

   $vcard_role = [ "role", $vcard_role_prop ]
   $vcard_role_prop = (
     { $vcard_language_param? , $vcard_pid_param? , $vcard_pref_param? ,
       $vcard_altid_param? , $vcard_type_param? , $vcard_any_param? },
     "text", string
   )

   $vcard_logo = [ "logo", $vcard_logo_prop ]
   $vcard_logo_prop = (
```

```
    { $vcard_altid_param? , $vcard_type_param? ,
      $vcard_mediatype_param? , $vcard_pref_param? ,
      $vcard_pid_param? , $vcard_language_param? ,
      $vcard_any_param? },
    "uri", uri
  )

  $vcard_org = [ "org", $vcard_org_prop ]
  $vcard_org_prop = (
    { $vcard_sort_as_param? , $vcard_language_param? ,
      $vcard_pid_param? , $vcard_pref_param? , $vcard_altid_param? ,
      $vcard_type_param? , $vcard_any_param? },
    "text", $vcard_string_type_array
    )

  $vcard_member = [ "member", $vcard_member_prop ]
  $vcard_member_prop = (
    { $vcard_altid_param? , $vcard_mediatype_param? ,
      $vcard_pref_param? , $vcard_pid_param? ,  $vcard_any_param? },
    "uri", uri
  )

  $vcard_related = [ "related", $vcard_related_prop ]
  $vcard_related_prop = (
    ( { $vcard_mediatype_param? , $vcard_pref_param? ,
        $vcard_altid_param? , $vcard_type_param? ,
        $vcard_any_param? },
      "uri", uri
    ) |
    ( { $vcard_language_param? , $vcard_pref_param? ,
        $vcard_altid_param? , $vcard_type_param? ,
        $vcard_any_param? },
      "text", $vcard_related_type_array
    )
  )

  $vcard_categories = [ "categories", $vcard_categories_prop ]
  $vcard_categories_prop = (
    { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param ?,
      $vcard_altid_param? , $vcard_any_param? },
    "text", $vcard_string_type_array
  )

  $vcard_note = [ "note", $vcard_note_prop ]
  $vcard_note_prop = (
    { $vcard_language_param? , $vcard_pid_param? ,
      $vcard_pref_param? , $vcard_type_param? ,
      $vcard_altid_param? , $vcard_any_param ? },
```

```
    "text", string
  )

  $vcard_prodid = [ "prodid", $vcard_prodid_prop ]
  $vcard_prodid_prop = ( { $vcard_any_param ? }, "text", string )

  $vcard_rev = [ "rev", $vcard_rev_prop ]
  $vcard_rev_prop = (
    { $vcard_any_param ? }, "timestamp", datetime
  )

  $vcard_sound = [ "sound", $vcard_sound_prop ]
  $vcard_sound_prop = (
    { $vcard_altid_param? , $vcard_mediatype_param? ,
      $vcard_language_param? , $vcard_pref_param? ,
      $vcard_pid_param? ,  $vcard_any_param? },
    "uri", uri
  )

  $vcard_uid = [ "uid", $vcard_uid_prop ]
  $vcard_uid_prop = (
    ( { $vcard_any_param? }, "uri", uri ) |
    ( { $vcard_any_param? }, "text", string )
  )

  $vcard_clientpidmap = [ "clientpidmap", $vcard_clientpidmap_prop ]
  $vcard_clientpidmap_prop = ( {}, "text", [ /^[0-9]+$/, uri ] )

  $vcard_url = [ "url", $vcard_url_prop ]
  $vcard_url_prop = (
    { $vcard_altid_param? , $vcard_mediatype_param? ,
      $vcard_pref_param? , $vcard_pid_param? ,  $vcard_any_param? },
    "uri", uri
  )

  $vcard_key = [ "key", $vcard_key_prop ]
  $vcard_key_prop = (
    ( { $vcard_mediatype_param? , $vcard_pref_param? ,
        $vcard_altid_param? , $vcard_type_param? ,
        $vcard_any_param? },
      "uri", uri
    ) |
    ( { $vcard_pref_param? , $vcard_altid_param? ,
        $vcard_type_param? , $vcard_any_param? },
      "text", string
    )
  )
```

```
$vcard_fburl = [ "fburl", $vcard_fburl_prop ]
$vcard_fburl_prop = (
  { $vcard_altid_param? , $vcard_mediatype_param? ,
    $vcard_altid_param? , $vcard_pref_param? ,
    $vcard_pid_param? ,  $vcard_any_param? },
  "uri", uri
)

$vcard_caladruri = [ "caladruri", $vcard_caladruri_prop ]
$vcard_caladruri_prop = (
  { $vcard_altid_param? , $vcard_mediatype_param? ,
    $vcard_altid_param? , $vcard_pref_param? ,
    $vcard_pid_param? ,  $vcard_any_param? },
  "uri", uri
)

$vcard_caluri = [ "caluri", $vcard_caluri_prop ]
$vcard_caluri_prop = (
  { $vcard_altid_param? , $vcard_mediatype_param? ,
    $vcard_altid_param? , $vcard_pref_param? ,
    $vcard_pid_param? ,  $vcard_any_param? },
  "uri", uri
)

$vcard_x10 = [ $vcard_x_name_type, $vcard_x10_prop ]
$vcard_x10_prop = (
  { $vcard_language_param? , $vcard_pref_param? ,
    $vcard_altid_param? , $vcard_pid_param? ,
    $vcard_type_param? , $vcard_mediatype_param? ,
    $vcard_calscale_param? , $vcard_sort_as_param? ,
    $vcard_geo_param? , $vcard_tz_param? ,
    $vcard_label_param? , $vcard_any_param? },
  $vcard_x_name_type, $vcard_string_type_array
)
```

Figure 5

vCard parameters are grouped into a JSON object, where each member of the object is a parameter.

```
 $vcard_language_param = "language" : $vcard_language_tag_type
 $vcard_pref_param = "pref" : $vcard_pref_type
 $vcard_altid_param = "altid" : string
 $vcard_pid_param = "pid" : $vcard_pid_type_array
 $vcard_type_param = "type" : $vcard_type_type_array
 $vcard_mediatype_param = "mediatype" : $vcard_mediatype_type
 $vcard_calscale_param = "calscale" : $vcard_calscale_type
 $vcard_sort_as_param = "sort-as" : $vcard_string_type_array
 $vcard_geo_param = "geo" : uri..geo
 $vcard_tz_param = "tz" : $vcard_utc_offset_type
 $vcard_label_param = "label" : $vcard_string_type_array
 $vcard_any_param = /^x\-[A-Za-z0-9\-]*$/ : $vcard_string_type_array
```

Figure 6

vCard values are determined by a string denoting the value type, followed by the value or an array of those values.

```
 $vcard_string_type_array =
   ( string | [ ( string | [ string * ] ) * ] )
 $vcard_language_tag_type =:
   /[a-z]{2}(-[A-Z][a-zA-Z]*(\-[A-Z]{2})?)?/
 $vcard_mediatype_type =: /^\w+\/[\-.\w]+(?:\+[\-.\w]+)?$/
 $vcard_utc_offset_type =: /^(([\-\+]\d{1,2}):?(\d{2})?)?$/
 $vcard_date_or_time =: (
   /^(\d{4})-?(\d{2})-?(\d{2})$/ |
   /^(\d{4})$/ |
   /^--(\d{2})-?(\d{2})$/ |
   /^--(\d{2})$/ |
   /^---(\d{2})$/ |
   /^(\d{2})(([-+]\d{1,2}):?(\d{2})?)?$/ |
   /^(\d{2}):?(\d{2})(([\-\+]\d{1,2}):?(\d{2})?)?$/ |
   /^(\d{2}):?(\d{2}):?(\d{2})(Z|([\-\+]\d{1,2}):?(\d{2})?)?$/ |
   /^-(\d{2}):?(\d{2})(Z|([\-\+]\d{1,2}):?(\d{2})?)?$/ |
   /^-(\d{2})(([\-\+]\d{1,2}):?(\d{2})?)?$/ |
   /^--(\d{2})(([\-\+]\d{1,2}):?(\d{2})?)?$/
 )
 $vcard_date_and_or_time_type =: (
   datetime | $vcard_date_or_time
 )
 $vcard_group_type =: /^[a-zA-Z0-9\-]+$/
 $vcard_type_type =: (
   "home" | "work" | $vcard_tel_type | $vcard_related_type |
   $vcard_x_name_type
```

```
    )
    $vcard_type_type_array = (
      $vcard_type_type | [ $vcard_type_type * ]
    )
    $vcard_tel_type =: (
      "text" | "voice" | "fax" | "cell" | "video" |
      "pager" | "textphone" | "main-number"
    )
    $vcard_tel_type_array = ( $vcard_tel_type | [ $vcard_tel_type * ] )
    $vcard_related_type =: (
      "contact" | "acquaintance" | "friend" | "met" |
      "co-worker" | "colleague" | "co-resident" |
      "neighbor" | "child" | "parent" | "sibling" |
      "spouse" | "kin" | "muse" | "crush" | "date" |
      "sweetheart" | "me" | "agent" | "emergency"
    )
    $vcard_related_type_array = (
      $vcard_related_type | [ $vcard_related_type * ]
    )
    $vcard_index_type =: /^[1-9]?[0-9]*$/
    $vcard_expertise_level_type =: (
      "beginner" | "average" | "expert"
    )
    $vcard_hobby_type =: ( "high" | "medium" | "low" )
    $vcard_pref_type =: /^[1-9]?[0-9]{1}$|^100$/
    $vcard_pid_type =: /^[0-9]+(\.[0-9]+)?$/
    $vcard_pid_type_array = (
      $vcard_pid_type | [ $vcard_pid_type * ]
    )
    $vcard_gender_type =: ( "M" | "F" | "O" | "N" | "U" )
    $vcard_gender_type_array = (
      $vcard_gender_type | [ $vcard_gender_type * ]
    )
    $vcard_kind_type =: (
      "individual" | "group" | "org" | "location" |
      "application" | "device"
    )
    $vcard_iana_token_type =: /^[A-Za-z0-9\-]*$/
    $vcard_x_name_type =: /^x\-[A-Za-z0-9\-]*$/
    $vcard_calscale_type =: (
      "gregorian" | $vcard_iana_token_type | $vcard_x_name_type
    )
```

Figure 7

3.2. Nameserver Object Class

The nameserver object class represents DNS nameservers in registries.

```
$nameservers = "nameservers" : [ $nameserver_oc * ]

$nameserver_oc = {
   $nameserver_mixin
}

$nameserver_mixin = (
   "objectClassName" : "nameserver",
   $common_mixin,
   "ldhName"         : fqdn,
   "unicodeName"     : idn ?,
   "ipAddresses"     : {
       "v4" : [ ipv4 + ] ?,
       "v6" : [ ipv6 + ] ?
   } ?,
   $entities ?
)
```

Figure 8

3.3. Domain Object Class

The Domain object class is the most complex of all the object classes defined in RDAP. It represents both forward and reverse DNS delegations. It's complexity is mostly due to the DNSSEC provisions of the object class.

```
$domain_oc = {
   $domain_mixin
}

$domain_mixin = (
   "objectClassName" : "domain",
   $common_mixin,
   "ldhName"         : fqdn,
   "unicodeName"     : idn ?,
   "variants"        : [ $variant * ] ?,
   $nameservers ?,
   $secureDNS ?,
   $entities ?,
   $publicIds ?,
   "network"          : $network_oc ?
)
```

```
$variant = {
   $variantRelation ?,
   "relation"     : [ string * ] ?,
   "idnTable"     : string ?,
   "variantNames" : [
       { "ldhName" : fqdn, "unicodeName" : idn } *
   ]
}

$variantRelation = "relation" : [ string * ]

$secureDNS = "secureDNS" : {
   "zoneSigned"       : boolean ?,
   "delegationSigned" : boolean ?,
   "maxSigLife"       : integer ?,
   "dsData"           : [ $dsData_obj * ] ?,
   "keyData"          : [ $keyData_obj * ] ?
}

$dsData_obj = {
   "keyTag"     : integer,
   "algorithm"  : integer,
   "digest"     : string,
   "digestType" : integer,
   $events ?,
   $links ?
}

$keyData_obj = {
   "flags"     : integer,
   "protocol"  : integer,
   "publicKey" : string,
   "algorithm" : integer,
   $events ?,
   $links ?
}
```

Figure 9

3.4. IP Network Object Class

The IP Network object class represents IP network registrations in RIRs.

```
$network_oc = {
   $network_mixin
}

$network_mixin = (
   "objectClassName" : "ip network",
   $common_mixin,
   "startAddress"    : ( ipv4 | ipv6 ) ?,
   "endAddres"       : ( ipv4 | ipv6 ) ?,
   "ipVersion"       : ( "v4" | "v6" ) ?,
   "name"            : string ?,
   "type"            : string ?,
   "country"         : /[A-Z]{2}/ ?,
   "parentHandle"    : string ?,
   $entities ?
)
```

Figure 10

3.5. Autnum Object Class

The Autnum object class represents an autonomous system number or blocks of autonomous system numbers in an RIR.

```
$autnum_oc = {
   $autnum_mixin
}

$autnum_mixin = (
   "objectClassName" : "autnum",
   $common_mixin,
   "startAutnum"     : int32 ?,
   "endAutnum"       : int32 ?,
   "name"            : string ?,
   "type"            : string ?,
   "country"         : string ?,
   $entities ?
)
```

Figure 11

4. Search Results

Search results in RDAP are merely arrays of object classes.

```
$domainSearchResult =
    "domainSearchResults"     : [ $domain_oc + ]

$nameserverSearchResult =
    "nameserverSearchResults" : [ $nameserver_oc + ]

$entitySearchResult =
    "entitySearchResults"     : [ $entity_oc + ]
```

Figure 12

5. Error Response

Section 6 of [RFC7483] describes RDAP error responses.

```
$error_mixin = (
   "errorCode"   : integer,
   "title"       : string ?,
   "description" : [ string * ] ?
)
```

Figure 13

6. Common Structures

Section 4 of [RFC7483] describes eight common structures used throughout the JSON in RDAP.

Most of these common structures are grouped together in a rule called common_mixin.

```
$common_mixin = (
   "handle"  : string ?,
   "remarks" : [ $notice * ] ?,
   $links ?,
   $events ?,
   $status ?,
   $port43 ?,
   $lang ?
)
```

Figure 14

6.1. RDAP Conformance

The rdapConformance array is the versioning and capabilities negotiation mechanism of RDAP.

```
$rdapConformance = "rdapConformance" : [ string * ]
```

Figure 15

6.2. Links

Structures in RDAP may link to information in other data systems using links. Additionally, RDAP uses "self" links to identify instances of RDAP object classes. The data found in each link is described by [RFC5988].

RDAP links are an array of distinct objects, each representing a separate link.

```
$links = ( "links" : [ $link * ] )

; see RFC 5988
$link = {
   "value"    : uri ?,
   "rel"      : string ?,
   "href"     : uri,
   "hreflang" : [ $lang_value * ] ?,
   "title"    : string ?,
   "media"    : string ?,
   "type"     :
       /[a-zA-Z][a-zA-Z0-9]*\/[a-zA-Z][a-zA-Z0-9]*/ ?
}
```

Figure 16

6.3. Notices And Remarks

In RDAP, notices and remarks share the same structure. The difference is that notices are meta-data regarding the entirety of a response whereas remarks are meta-data covering a specific instance of an object class.

```
$notices = [ $notice * ]

$notice = {
    "title"       : string ?,
    "description" : [ string * ],
    $noticeRemarkType ?,
    $links ?,
    $lang ?
}

$noticeRemarkType = "type" : string
```

Figure 17

6.4. Language Identifier

The "lang" member occurs many RDAP data structures. And the same construct is used in the links structures.

```
; the language value as defined in RFC 5646
$lang_value =: /[a-z]{2}(\-[A-Z][a-zA-Z]*(\-[A-Z]{2})?)?/

$lang = ( "lang" : $lang_value )
```

Figure 18

6.5. Events

RDAP events note when a specific action has occured on an object instance, and by whom. The same structure appears in all object classes, as well as being re-used by entities embedded by other objects.

```
$events = "events" : [ $event * ]

$noActorEvent_mixin = (
    $eventAction,
    "eventDate"   : datetime,
    $links ?,
    $lang ?
)

$eventAction = "eventAction" : string

$noActorEvent = {
    $noActorEvent_mixin
}

$event = {
    $noActorEvent_mixin,
    "eventActor" : string ?
}
```

Figure 19

6.6. Status

The status of RDAP object instances is indicated by an array of strings, where the value of the strings are registered in an IANA registry.

```
$status = "status" : [ string * ]
```

Figure 20

6.7. Port 43

RDAP object classes reference their corresponding Whois representation using the "port43" object member. This is simply a string holding the hostname of the Whois service.

```
$port43 = "port43" : string
```

Figure 21

6.8. Public IDs

Some RDAP services are required to identify entities and domains by public identifiers, such as ICANN Registrar IDs. The publicIds object member is an array of objects to represent these identifiers.

```
$publicIds = "publicIds" : [ $publicId * ]

$publicId = {
  "type"       : string,
  "identifier" : string
}
```

Figure 22

7. Validating Responses

Many JSON roots for RDAP are defined in Figure 1. For applications where an RDAP query must yield a specific response, the appropriate root must be used for validating the response.

For example, if the RDAP query https://example.com/rdap/ip/2001:db8::0 is expected to yield an IP network response, then the validation must only use the $network_response root.

8. Stricter Validation

RDAP has a very lenient JSON model where information and structures not strictly forbidden are allowed. However, this lenient model allows information from RDAP help and error responses to be found in other responses, and responses for single objects to be found in responses from searches.

For applications where these structures cannot be mixed and validation is desired, override rules may be used.

For readability, two new rules are created to group the single objects and the search results.

The responses with a single object returned represented by mixins. This rule groups those mixins.

```
$object_class = (
    $entity_mixin |
    $nameserver_mixin |
    $domain_mixin |
    $network_mixin |
    $autnum_mixin
)
```

Figure 23

The responses to searches are objects containing a single search member which is an array containing the object class mixins. This rule groups the search members.

```
$search_results = (
    $domainSearchResult |
    $nameserverSearchResult |
    $entitySearchResult
)
```

Figure 24

Using the new rules in Figure 23 and Figure 24, override rules (rules rewriting existing rules) can be written for greater strictness.

These rules for single object responses prevent search and error response elements to be present.

```
@{root} $entity_response = {
    $response_mixin,
    $entity_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}

@{root} $nameserver_response = {
    $response_mixin,
    $nameserver_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}

@{root} $domain_response = {
    $response_mixin,
    $domain_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}

@{root} $network_response = {
    $response_mixin,
    $network_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}

@{root} $autnum_response = {
    $response_mixin,
    $autnum_mixin,
    @{not} $error_mixin,
    @{not} $search_results
}
```

Figure 25

These rules for help and error responses prevent single object and search arrays.

```
@{root} $error_response =  {
    $response_mixin,
    $error_mixin,
    @{not} $object_class,
    @{not} $search_results
}

@{root} $help_response = {
    $response_mixin,
    $lang ?,
    @{not} $error_mixin,
    @{not} $object_class,
    @{not} $search_results
}
```

Figure 26

These rules for search responses prevent single object and error data structures.

```
@{root} $domainSearch_response = {
    $response_mixin,
    $lang ?,
    $domainSearchResult,
    @{not} $error_mixin,
    @{not} $object_class
}

@{root} $nameserverSearch_response = {
    $response_mixin,
    $lang ?,
    $nameserverSearchResult,
    @{not} $error_mixin,
    @{not} $object_class
}

@{root} $entitySearch_response = {
    $response_mixin,
    $lang ?,
    $entitySearchResult,
    @{not} $error_mixin,
    @{not} $object_class
}
```

Figure 27

Stricter validation is also possible by overriding rules that take generic strings and overriding them with enumerated lists of values from the RDAP JSON values registry.

These rules override the generic domain variant relations in RDAP with values found in the IANA registry.

```
    $variantRelation = "relation" : [ $variantRelation_values * ]

    $variantRelation_values = (
        "registered" |
        "unregistered" |
        "registration restricted" |
        "open registration" |
        "conjoined"
    )
```

Figure 28

These rules override the generic event actions in RDAP with values found in the IANA registry.

```
         $eventAction = "eventAction" : $eventAction_values

         $eventAction_values = (
             "registration" |
             "reregistration" |
             "last changed" |
             "expiration" |
             "deletion" |
             "reinstantiation" |
             "transfer" |
             "locked" |
             "unlocked" |
             "last update of RDAP database" |
             "registrar expiration" |
             "enum validation expiration"
         )
```

Figure 29

These rules override the generic notice and remark types in RDAP with values found in the IANA registry.

```
$noticeRemarkType = "type" : $noticeRemarkType_values

$noticeRemarkType_values = (
    "result set truncated due to authorization" |
    "result set truncated due to excessive load" |
    "result set truncated due to unexplainable reasons" |
    "object truncated due to authorization" |
    "object truncated due to excessive load" |
    "object truncated due to unexplainable reasons"
)
```

Figure 30

These rules override the generic entity roles in RDAP with values found in the IANA registry.

```
$roles = "roles" : [ $role_values * ]

$role_values = (
    "registrant" |
    "technical" |
    "administrative" |
    "abuse" |
    "billing" |
    "registrar" |
    "reseller" |
    "sponsor" |
    "proxy" |
    "notifications" |
    "noc"
)
```

Figure 31

These rules override the generic status values in RDAP with values found in the IANA registry.

```
$status = "status" : [ $status_values * ]

$status_values = (
    "validated" |
    "renew prohibited" |
    "update prohibited" |
    "transfer prohibited" |
    "delete prohibited" |
    "proxy" |
    "private" |
    "removed" |
    "obscured" |
    "associated" |
    "active" |
    "inactive" |
    "locked" |
    "pending create" |
    "pending renew" |
    "pending transfer" |
    "pending update" |
    "pending delete" |
    "add period" |
    "auto renew period" |
    "client delete prohibited" |
    "client hold" |
    "client renew prohibited" |
    "client transfer prohibited" |
    "client update prohibited" |
    "pending restore" |
    "redemption period" |
    "renew period" |
    "server delete prohibited" |
    "server renew prohibited" |
    "server transfer prohibited" |
    "server update prohibited" |
    "server hold" |
    "transfer period"
)
```

Figure 32

9. Complete Rulesets for RDAP

The following rulesets, along with a test framework and examples of good and bad RDAP JSON instances, may be found at <https://github.com/arineng/draft-rdap-jcr>.

The following is the complete ruleset of JSON Content Rules for RDAP.

```
;
; JSON Content Rules (JCR) ruleset
; for the Registry Data Access Protocol (RDAP)
;
; Specified in RFC 7483
;

# ruleset-id rdap_level_0

;
; The various types of responses
;

@{root} $entity_response = {
    $response_mixin,
    $entity_mixin
}

@{root} $nameserver_response = {
    $response_mixin,
    $nameserver_mixin
}

@{root} $domain_response = {
    $response_mixin,
    $domain_mixin
}

@{root} $network_response = {
    $response_mixin,
    $network_mixin
}

@{root} $autnum_response = {
    $response_mixin,
    $autnum_mixin
}

@{root} $error_response = {
    $response_mixin,
```

```
      $error_mixin
   }

   @{root} $help_response = {
      $response_mixin,
      $lang ?
   }

   @{root} $domainSearch_response = {
      $response_mixin,
      $lang ?,
      $domainSearchResult
   }

   @{root} $nameserverSearch_response = {
      $response_mixin,
      $lang ?,
      $nameserverSearchResult
   }

   @{root} $entitySearch_response = {
      $response_mixin,
      $lang ?,
      $entitySearchResult
   }

   $response_mixin = (
      $rdapConformance ?,
      "notices" : $notices ?
   )

   ;
   ; RFC 7483 Section 4.1 - RDAP Conformance
   ;

   $rdapConformance = "rdapConformance" : [ string * ]

   ;
   ; RFC 7483 Section 4.2 - Links
   ;

   $links = ( "links" : [ $link * ] )

   ; see RFC 5988
   $link = {
      "value"    : uri ?,
      "rel"      : string ?,
      "href"     : uri,
```

```
      "hreflang" : [ $lang_value * ] ?,
      "title"    : string ?,
      "media"    : string ?,
      "type"     :
          /[a-zA-Z][a-zA-Z0-9]*\/[a-zA-Z][a-zA-Z0-9]*/ ?
   }

   ;
   ; RFC 7483 Section 4.3 - Notices
   ;

   $notices = [ $notice * ]

   $notice = {
       "title"       : string ?,
       "description" : [ string * ],
       $noticeRemarkType ?,
       $links ?,
       $lang ?
   }

   $noticeRemarkType = "type" : string

   ;
   ; RFC 7483 Section 4.4 - Language Identifier
   ;

   ; the language value as defined in RFC 5646
   $lang_value =: /[a-z]{2}(\-[A-Z][a-zA-Z]*(\-[A-Z]{2})?)?/

   $lang = ( "lang" : $lang_value )

   ;
   ; RFC 7483 Section 4.5 - Events
   ;

   $events = "events" : [ $event * ]

   $noActorEvent_mixin = (
       $eventAction,
       "eventDate"   : datetime,
       $links ?,
       $lang ?
   )

   $eventAction = "eventAction" : string

   $noActorEvent = {
```

```
      $noActorEvent_mixin
   }

   $event = {
      $noActorEvent_mixin,
      "eventActor" : string ?
   }


   ;
   ; RFC 7483 Section 4.6 - Status
   ;

   $status = "status" : [ string * ]

   ;
   ; RFC 7483 Section 4.7 - Port43
   ;

   $port43 = "port43" : string

   ;
   ; RFC 7482 Section 4.8 - Public Ids
   ;

   $publicIds = "publicIds" : [ $publicId * ]

   $publicId = {
     "type"       : string,
     "identifier" : string
   }

   ;
   ; Common Object Class Structures
   ;

   $common_mixin = (
      "handle"  : string ?,
      "remarks" : [ $notice * ] ?,
      $links ?,
      $events ?,
      $status ?,
      $port43 ?,
      $lang ?
   )

   ;
   ; RFC 7483 Section 5.1 - Entity Object Class
```

```
   ;

   $entities = "entities" : [ $entity_oc * ]

   $entity_oc = {
      $entity_mixin
   }

   $entity_mixin = (
      "objectClassName" : "entity",
      $common_mixin,
      "vcardArray"      : [ "vcard", $vcard * ] ?,
      "asEventActor"    : [ $noActorEvent * ] ?,
      $roles ?,
      $publicIds ?,
      $entities ?,
      "networks"        : [ $network_oc * ] ?,
      "autnums"         : [ $autnum_oc * ] ?
   )

   $roles = "roles" : [ string * ]

   ; jCard (see RFC 7095) is an algorithm for converting
   ; vCard to JSON. Each jCard is represented by an array of
   ; vCard properties. The "version" property must be the
   ; first property, and there must be one and only one "fn"
   ; property in some part of the array.
   ;
   ; Here we represent this by an ordered array, with "version"
   ; being the first property, then listing all the properties
   ; that match by property name, then matching
   ; the "fn" property, then matching any other property again.
   $vcard = [
      [ "version", {}, "text", "4.0" ],
      $vcard_not_fn_properties * ,
      [ "fn", { $vcard_type_param?, $vcard_language_param? ,
                $vcard_altid_param? , $vcard_pid_param? ,
                $vcard_pref_param?, $vcard_any_param? },
                "text", string ],
      $vcard_not_fn_properties *
   ]

   $vcard_not_fn_properties =
      (
        $vcard_source |
        $vcard_kind |
        $vcard_n |
        $vcard_nicname |
```

```
        $vcard_photo |
        $vcard_bday |
        $vcard_anniversary |
        $vcard_gender |
        $vcard_adr |
        $vcard_tel |
        $vcard_email |
        $vcard_impp |
        $vcard_lang |
        $vcard_tz |
        $vcard_geo |
        $vcard_title |
        $vcard_role |
        $vcard_logo |
        $vcard_org |
        $vcard_member |
        $vcard_related |
        $vcard_categories |
        $vcard_note |
        $vcard_prodid |
        $vcard_rev |
        $vcard_sound |
        $vcard_uid |
        $vcard_clientpidmap |
        $vcard_url |
        $vcard_key |
        $vcard_fburl |
        $vcard_caladruri |
        $vcard_caluri |
        $vcard_x10
      )

   ; vCard properties. Each one is defined with a common
   ; definition, and then a group that matches the property name.
   $vcard_source = [ "source", $vcard_source_prop ]
   $vcard_source_prop = (
     { $vcard_pid_param? , $vcard_pref_param?, $vcard_altid_param? ,
       $vcard_mediatype_param?, $vcard_any_param? } ,
     "uri", uri
   )

   $vcard_kind = [ "kind", $vcard_kind_prop ]
   $vcard_kind_prop = (
     { $vcard_any_param? },
     "text", $vcard_kind_type
   )

   $vcard_n = [ "n", $vcard_n_prop ]
```

```
   $vcard_n_prop = (
     { $vcard_sort_as_param? , $vcard_language_param? ,
       $vcard_altid_param? , $vcard_any_param? },
     "text", $vcard_string_type_array
   )

   $vcard_nicname = [ "nicname", $vcard_nicname_prop ]
   $vcard_nicname_prop = (
     { $vcard_type_param? , $vcard_language_param? ,
       $vcard_altid_param? , $vcard_pid_param? ,
       $vcard_pref_param? , $vcard_any_param ? },
     "text", $vcard_string_type_array
   )

   $vcard_photo = [ "photo", $vcard_photo_prop ]
   $vcard_photo_prop = (
     { $vcard_altid_param? , $vcard_type_param? ,
       $vcard_mediatype_param? , $vcard_pref_param? ,
       $vcard_pid_param? , $vcard_any_param? },
     "uri", uri
   )

   $vcard_bday = [ "bday", $vcard_bday_prop ]
   $vcard_bday_prop = (
     ( { $vcard_language_param?, $vcard_altid_param?,
         $vcard_calscale_param?, $vcard_any_param? },
       "text", string
     ) |
     ( { $vcard_altid_param?, $vcard_calscale_param?,
         $vcard_any_param? },
       "date-and-or-time", $vcard_date_and_or_time_type
     )
   )

   $vcard_anniversary = [ "anniversary", $vcard_anniversary_prop ]
   $vcard_anniversary_prop = (
     { $vcard_altid_param?, $vcard_calscale_param?, $vcard_any_param? },
     "date-and-or-time", $vcard_date_and_or_time_type
   )

   $vcard_gender = [ "gender", $vcard_gender_prop ]
   $vcard_gender_prop = (
     { $vcard_any_param? },
     "text",
     ( $vcard_gender_type | [ $vcard_gender_type, string + ] )
   )

   $vcard_adr = [ "adr", $vcard_adr_prop ]
```

```
   $vcard_adr_prop = (
     { $vcard_label_param? , $vcard_language_param? ,
       $vcard_geo_param ? , $vcard_tz_param? ,
       $vcard_altid_param? , $vcard_pid_param? ,
       $vcard_pref_param? , $vcard_type_param? ,
       $vcard_any_param? },
     "text", $vcard_string_type_array
   )

   $vcard_tel = [ "tel", $vcard_tel_prop ]
   $vcard_tel_prop = (
     ( { $vcard_type_param? , $vcard_pid_param? , $vcard_pref_param? ,
         $vcard_altid_param?, $vcard_any_param? }, "text", string
     ) |
     ( { $vcard_type_param? , $vcard_pid_param? , $vcard_pref_param? ,
         $vcard_altid_param?, $vcard_mediatype_param?,
         $vcard_any_param? },
       "uri", uri..tel
     )
   )

   $vcard_email = [ "email", $vcard_email_prop ]
   $vcard_email_prop = (
     { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param? ,
       $vcard_altid_param? , $vcard_any_param ? },
     "text", string
   )

   $vcard_impp = [ "impp", $vcard_impp_prop ]
   $vcard_impp_prop = (
     { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param? ,
       $vcard_mediatype_param? , $vcard_altid_param? ,
       $vcard_any_param },
     "uri", uri..impp
   )

   $vcard_lang = [ "lang", $vcard_lang_prop ]
   $vcard_lang_prop = (
     { $vcard_pid_param? , $vcard_pref_param? , $vcard_altid_param? ,
       $vcard_type_param? , $vcard_any_param? },
     "language-tag", $vcard_language_tag_type
   )

   $vcard_tz = [ "tz", $vcard_tz_prop ]
   $vcard_tz_prop = (
     { $vcard_altid_param?, $vcard_pid_param? , $vcard_pref_param? ,
       $vcard_type_param? , $vcard_mediatype_param?,
       $vcard_any_param? },
```

```
    ( ( "text", string ) |
    ( "uri" , uri ) |
    ( "utc-offset" , $vcard_utc_offset_type ) )
  )

  $vcard_geo = [ "geo", $vcard_geo_prop ]
  $vcard_geo_prop = (
    { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param? ,
      $vcard_mediatype_param? , $vcard_altid_param? ,
      $vcard_any_param? },
    "uri", uri..geo
  )

  $vcard_title = [ "title", $vcard_title_prop ]
  $vcard_title_prop = (
    { $vcard_language_param? , $vcard_pid_param? , $vcard_pref_param? ,
      $vcard_altid_param? , $vcard_type_param? , $vcard_any_param? },
    "text", string
  )

  $vcard_role = [ "role", $vcard_role_prop ]
  $vcard_role_prop = (
    { $vcard_language_param? , $vcard_pid_param? , $vcard_pref_param? ,
      $vcard_altid_param? , $vcard_type_param? , $vcard_any_param? },
    "text", string
  )

  $vcard_logo = [ "logo", $vcard_logo_prop ]
  $vcard_logo_prop = (
    { $vcard_altid_param? , $vcard_type_param? ,
      $vcard_mediatype_param? , $vcard_pref_param? ,
      $vcard_pid_param? , $vcard_language_param? ,
      $vcard_any_param? },
    "uri", uri
  )

  $vcard_org = [ "org", $vcard_org_prop ]
  $vcard_org_prop = (
    { $vcard_sort_as_param? , $vcard_language_param? ,
      $vcard_pid_param? , $vcard_pref_param? , $vcard_altid_param? ,
      $vcard_type_param? , $vcard_any_param? },
    "text", $vcard_string_type_array
    )

  $vcard_member = [ "member", $vcard_member_prop ]
  $vcard_member_prop = (
    { $vcard_altid_param? , $vcard_mediatype_param? ,
      $vcard_pref_param? , $vcard_pid_param? ,  $vcard_any_param? },
```

```
    "uri", uri
  )

  $vcard_related = [ "related", $vcard_related_prop ]
  $vcard_related_prop = (
    ( { $vcard_mediatype_param? , $vcard_pref_param? ,
        $vcard_altid_param? , $vcard_type_param? ,
        $vcard_any_param? },
      "uri", uri
    ) |
    ( { $vcard_language_param? , $vcard_pref_param? ,
        $vcard_altid_param? , $vcard_type_param? ,
        $vcard_any_param? },
      "text", $vcard_related_type_array
    )
  )

  $vcard_categories = [ "categories", $vcard_categories_prop ]
  $vcard_categories_prop = (
    { $vcard_pid_param? , $vcard_pref_param? , $vcard_type_param ?,
      $vcard_altid_param? , $vcard_any_param? },
    "text", $vcard_string_type_array
  )

  $vcard_note = [ "note", $vcard_note_prop ]
  $vcard_note_prop = (
    { $vcard_language_param? , $vcard_pid_param? ,
      $vcard_pref_param? , $vcard_type_param? ,
      $vcard_altid_param? , $vcard_any_param ? },
    "text", string
  )

  $vcard_prodid = [ "prodid", $vcard_prodid_prop ]
  $vcard_prodid_prop = ( { $vcard_any_param ? }, "text", string )

  $vcard_rev = [ "rev", $vcard_rev_prop ]
  $vcard_rev_prop = (
    { $vcard_any_param ? }, "timestamp", datetime
  )

  $vcard_sound = [ "sound", $vcard_sound_prop ]
  $vcard_sound_prop = (
    { $vcard_altid_param? , $vcard_mediatype_param? ,
      $vcard_language_param? , $vcard_pref_param? ,
      $vcard_pid_param? ,  $vcard_any_param? },
    "uri", uri
  )
```

```
   $vcard_uid = [ "uid", $vcard_uid_prop ]
   $vcard_uid_prop = (
     ( { $vcard_any_param? }, "uri", uri ) |
     ( { $vcard_any_param? }, "text", string )
   )

   $vcard_clientpidmap = [ "clientpidmap", $vcard_clientpidmap_prop ]
   $vcard_clientpidmap_prop = ( {}, "text", [ /^[0-9]+$/, uri ] )

   $vcard_url = [ "url", $vcard_url_prop ]
   $vcard_url_prop = (
     { $vcard_altid_param? , $vcard_mediatype_param? ,
       $vcard_pref_param? , $vcard_pid_param? ,  $vcard_any_param? },
     "uri", uri
   )

   $vcard_key = [ "key", $vcard_key_prop ]
   $vcard_key_prop = (
     ( { $vcard_mediatype_param? , $vcard_pref_param? ,
         $vcard_altid_param? , $vcard_type_param? ,
         $vcard_any_param? },
       "uri", uri
     ) |
     ( { $vcard_pref_param? , $vcard_altid_param? ,
         $vcard_type_param? , $vcard_any_param? },
       "text", string
     )
   )

   $vcard_fburl = [ "fburl", $vcard_fburl_prop ]
   $vcard_fburl_prop = (
     { $vcard_altid_param? , $vcard_mediatype_param? ,
       $vcard_altid_param? , $vcard_pref_param? ,
       $vcard_pid_param? ,  $vcard_any_param? },
     "uri", uri
   )

   $vcard_caladruri = [ "caladruri", $vcard_caladruri_prop ]
   $vcard_caladruri_prop = (
     { $vcard_altid_param? , $vcard_mediatype_param? ,
       $vcard_altid_param? , $vcard_pref_param? ,
       $vcard_pid_param? ,  $vcard_any_param? },
     "uri", uri
   )

   $vcard_caluri = [ "caluri", $vcard_caluri_prop ]
   $vcard_caluri_prop = (
     { $vcard_altid_param? , $vcard_mediatype_param? ,
```

```
      $vcard_altid_param? , $vcard_pref_param? ,
      $vcard_pid_param? ,  $vcard_any_param? },
    "uri", uri
  )

  $vcard_x10 = [ $vcard_x_name_type, $vcard_x10_prop ]
  $vcard_x10_prop = (
    { $vcard_language_param? , $vcard_pref_param? ,
      $vcard_altid_param? , $vcard_pid_param? ,
      $vcard_type_param? , $vcard_mediatype_param? ,
      $vcard_calscale_param? , $vcard_sort_as_param? ,
      $vcard_geo_param? , $vcard_tz_param? ,
      $vcard_label_param? , $vcard_any_param? },
    $vcard_x_name_type, $vcard_string_type_array
  )

  ; Each vCard property can have parameters.
  $vcard_language_param = "language" : $vcard_language_tag_type
  $vcard_pref_param = "pref" : $vcard_pref_type
  $vcard_altid_param = "altid" : string
  $vcard_pid_param = "pid" : $vcard_pid_type_array
  $vcard_type_param = "type" : $vcard_type_type_array
  $vcard_mediatype_param = "mediatype" : $vcard_mediatype_type
  $vcard_calscale_param = "calscale" : $vcard_calscale_type
  $vcard_sort_as_param = "sort-as" : $vcard_string_type_array
  $vcard_geo_param = "geo" : uri..geo
  $vcard_tz_param = "tz" : $vcard_utc_offset_type
  $vcard_label_param = "label" : $vcard_string_type_array
  $vcard_any_param = /^x\-[A-Za-z0-9\-]*$/ : $vcard_string_type_array

  ; Each vCard property has a value type.
  $vcard_string_type_array =
    ( string | [ ( string | [ string * ] ) * ] )
  $vcard_language_tag_type =:
    /[a-z]{2}(-[A-Z][a-zA-Z]*(\-[A-Z]{2})?)?/
  $vcard_mediatype_type =: /^\w+\/[\-.\w]+(?:\+[\-.\w]+)?$/
  $vcard_utc_offset_type =: /^(([\-\+]\d{1,2}):?(\d{2})?)?$/
  $vcard_date_or_time =: (
    /^(\d{4})-?(\d{2})-?(\d{2})$/ |
    /^(\d{4})$/ |
    /^--(\d{2})-?(\d{2})$/ |
    /^--(\d{2})$/ |
    /^---(\d{2})$/ |
    /^(\d{2})(([-+]\d{1,2}):?(\d{2})?)?$/ |
    /^(\d{2}):?(\d{2})(([\-\+]\d{1,2}):?(\d{2})?)?$/ |
    /^(\d{2}):?(\d{2}):?(\d{2})(Z|([\-\+]\d{1,2}):?(\d{2})?)?$/ |
    /^-(\d{2}):?(\d{2})(Z|([\-\+]\d{1,2}):?(\d{2})?)?$/ |
    /^-(\d{2})(([\-\+]\d{1,2}):?(\d{2})?)?$/ |
```

```
    /^--(\d{2})(([\-\+]\d{1,2}):?(\d{2})?)?$/
  )
  $vcard_date_and_or_time_type =: (
    datetime | $vcard_date_or_time
  )
  $vcard_group_type =: /^[a-zA-Z0-9\-]+$/
  $vcard_type_type =: (
    "home" | "work" | $vcard_tel_type | $vcard_related_type |
    $vcard_x_name_type
  )
  $vcard_type_type_array = (
    $vcard_type_type | [ $vcard_type_type * ]
  )
  $vcard_tel_type =: (
    "text" | "voice" | "fax" | "cell" | "video" |
    "pager" | "textphone" | "main-number"
  )
  $vcard_tel_type_array = ( $vcard_tel_type | [ $vcard_tel_type * ] )
  $vcard_related_type =: (
    "contact" | "acquaintance" | "friend" | "met" |
    "co-worker" | "colleague" | "co-resident" |
    "neighbor" | "child" | "parent" | "sibling" |
    "spouse" | "kin" | "muse" | "crush" | "date" |
    "sweetheart" | "me" | "agent" | "emergency"
  )
  $vcard_related_type_array = (
    $vcard_related_type | [ $vcard_related_type * ]
  )
  $vcard_index_type =: /^[1-9]?[0-9]*$/
  $vcard_expertise_level_type =: (
    "beginner" | "average" | "expert"
  )
  $vcard_hobby_type =: ( "high" | "medium" | "low" )
  $vcard_pref_type =: /^[1-9]?[0-9]{1}$|^100$/
  $vcard_pid_type =: /^[0-9]+(\.[0-9]+)?$/
  $vcard_pid_type_array = (
    $vcard_pid_type | [ $vcard_pid_type * ]
  )
  $vcard_gender_type =: ( "M" | "F" | "O" | "N" | "U" )
  $vcard_gender_type_array = (
    $vcard_gender_type | [ $vcard_gender_type * ]
  )
  $vcard_kind_type =: (
    "individual" | "group" | "org" | "location" |
    "application" | "device"
  )
  $vcard_iana_token_type =: /^[A-Za-z0-9\-]*$/
  $vcard_x_name_type =: /^x\-[A-Za-z0-9\-]*$/
```

```
   $vcard_calscale_type =: (
     "gregorian" | $vcard_iana_token_type | $vcard_x_name_type
   )

   ;
   ; RFC 7483 Section 5.2 - Nameserver Object Class
   ;

   $nameservers = "nameservers" : [ $nameserver_oc * ]

   $nameserver_oc = {
      $nameserver_mixin
   }

   $nameserver_mixin = (
      "objectClassName" : "nameserver",
      $common_mixin,
      "ldhName"         : fqdn,
      "unicodeName"     : idn ?,
      "ipAddresses"     : {
          "v4" : [ ipv4 + ] ?,
          "v6" : [ ipv6 + ] ?
      } ?,
      $entities ?
   )

   ;
   ; RFC 7483 Section 5.3 - Domain Object Class
   ;

   $domain_oc = {
      $domain_mixin
   }

   $domain_mixin = (
      "objectClassName" : "domain",
      $common_mixin,
      "ldhName"         : fqdn,
      "unicodeName"     : idn ?,
      "variants"        : [ $variant * ] ?,
      $nameservers ?,
      $secureDNS ?,
      $entities ?,
      $publicIds ?,
      "network"          : $network_oc ?
   )

   $variant = {
```

```
      $variantRelation ?,
      "relation"     : [ string * ] ?,
      "idnTable"     : string ?,
      "variantNames" : [
          { "ldhName" : fqdn, "unicodeName" : idn } *
      ]
   }

   $variantRelation = "relation" : [ string * ]

   $secureDNS = "secureDNS" : {
      "zoneSigned"       : boolean ?,
      "delegationSigned" : boolean ?,
      "maxSigLife"       : integer ?,
      "dsData"           : [ $dsData_obj * ] ?,
      "keyData"          : [ $keyData_obj * ] ?
   }

   $dsData_obj = {
      "keyTag"     : integer,
      "algorithm"  : integer,
      "digest"     : string,
      "digestType" : integer,
      $events ?,
      $links ?
   }

   $keyData_obj = {
      "flags"     : integer,
      "protocol"  : integer,
      "publicKey" : string,
      "algorithm" : integer,
      $events ?,
      $links ?
   }

   ;
   ; RFC 7483 Section 5.4 - IP Network Object Class
   ;

   $network_oc = {
      $network_mixin
   }

   $network_mixin = (
      "objectClassName" : "ip network",
      $common_mixin,
      "startAddress"    : ( ipv4 | ipv6 ) ?,
```

```
      "endAddres"       : ( ipv4 | ipv6 ) ?,
      "ipVersion"       : ( "v4" | "v6" ) ?,
      "name"            : string ?,
      "type"            : string ?,
      "country"         : /[A-Z]{2}/ ?,
      "parentHandle"    : string ?,
      $entities ?
   )

   ;
   ; RFC 7483 Section 5.5 - Autnum Object Class
   ;

   $autnum_oc = {
      $autnum_mixin
   }

   $autnum_mixin = (
      "objectClassName" : "autnum",
      $common_mixin,
      "startAutnum"     : int32 ?,
      "endAutnum"       : int32 ?,
      "name"            : string ?,
      "type"            : string ?,
      "country"         : string ?,
      $entities ?
   )

   ;
   ; RFC 7483 Section 6 - Error
   ;

   $error_mixin = (
      "errorCode"   : integer,
      "title"       : string ?,
      "description" : [ string * ] ?
   )

   ;
   ; RFC 7483 Section 8 - Search Results
   ;

   $domainSearchResult =
       "domainSearchResults"     : [ $domain_oc + ]

   $nameserverSearchResult =
       "nameserverSearchResults" : [ $nameserver_oc + ]
```

```
$entitySearchResult =
    "entitySearchResults"     : [ $entity_oc + ]
```

Figure 33: Complete Ruleset for RDAP

The following is the complete ruleset of override rules for stricter validation of RDAP.

```
    ;
    ; Override rules for strict RDAP checking.
    ;

    ;
    ; Object class response
    ;

    @{root} $entity_response = {
        $response_mixin,
        $entity_mixin,
        @{not} $error_mixin,
        @{not} $search_results
    }

    @{root} $nameserver_response = {
        $response_mixin,
        $nameserver_mixin,
        @{not} $error_mixin,
        @{not} $search_results
    }

    @{root} $domain_response = {
        $response_mixin,
        $domain_mixin,
        @{not} $error_mixin,
        @{not} $search_results
    }

    @{root} $network_response = {
        $response_mixin,
        $network_mixin,
        @{not} $error_mixin,
        @{not} $search_results
    }

    @{root} $autnum_response = {
        $response_mixin,
        $autnum_mixin,
```

```
        @{not} $error_mixin,
        @{not} $search_results
    }
    ;
    ; Help and error response
    ;

    @{root} $error_response =  {
        $response_mixin,
        $error_mixin,
        @{not} $object_class,
        @{not} $search_results
    }

    @{root} $help_response = {
        $response_mixin,
        $lang ?,
        @{not} $error_mixin,
        @{not} $object_class,
        @{not} $search_results
    }
    ;
    ; Search responses
    ;

    @{root} $domainSearch_response = {
        $response_mixin,
        $lang ?,
        $domainSearchResult,
        @{not} $error_mixin,
        @{not} $object_class
    }

    @{root} $nameserverSearch_response = {
        $response_mixin,
        $lang ?,
        $nameserverSearchResult,
        @{not} $error_mixin,
        @{not} $object_class
    }

    @{root} $entitySearch_response = {
        $response_mixin,
        $lang ?,
        $entitySearchResult,
        @{not} $error_mixin,
        @{not} $object_class
    }
```

```
;
; Object class mixins
;

$object_class = (
    $entity_mixin |
    $nameserver_mixin |
    $domain_mixin |
    $network_mixin |
    $autnum_mixin
)
;
; All search results
;

$search_results = (
    $domainSearchResult |
    $nameserverSearchResult |
    $entitySearchResult
)
;
; IANA Status Values
;

$status = "status" : [ $status_values * ]

$status_values = (
    "validated" |
    "renew prohibited" |
    "update prohibited" |
    "transfer prohibited" |
    "delete prohibited" |
    "proxy" |
    "private" |
    "removed" |
    "obscured" |
    "associated" |
    "active" |
    "inactive" |
    "locked" |
    "pending create" |
    "pending renew" |
    "pending transfer" |
    "pending update" |
    "pending delete" |
    "add period" |
    "auto renew period" |
    "client delete prohibited" |
```

```
            "client hold" |
            "client renew prohibited" |
            "client transfer prohibited" |
            "client update prohibited" |
            "pending restore" |
            "redemption period" |
            "renew period" |
            "server delete prohibited" |
            "server renew prohibited" |
            "server transfer prohibited" |
            "server update prohibited" |
            "server hold" |
            "transfer period"
        )
        ;
        ; IANA Notice and Remark Types
        ;

        $noticeRemarkType = "type" : $noticeRemarkType_values

        $noticeRemarkType_values = (
            "result set truncated due to authorization" |
            "result set truncated due to excessive load" |
            "result set truncated due to unexplainable reasons" |
            "object truncated due to authorization" |
            "object truncated due to excessive load" |
            "object truncated due to unexplainable reasons"
        )
        ;
        ; IANA Roles
        ;

        $roles = "roles" : [ $role_values * ]

        $role_values = (
            "registrant" |
            "technical" |
            "administrative" |
            "abuse" |
            "billing" |
            "registrar" |
            "reseller" |
            "sponsor" |
            "proxy" |
            "notifications" |
            "noc"
        )
```

```
;
; IANA Domain Variant Relations
;

$variantRelation = "relation" : [ $variantRelation_values * ]

$variantRelation_values = (
    "registered" |
    "unregistered" |
    "registration restricted" |
    "open registration" |
    "conjoined"
)
;
; IANA Event Actions
;

$eventAction = "eventAction" : $eventAction_values

$eventAction_values = (
    "registration" |
    "reregistration" |
    "last changed" |
    "expiration" |
    "deletion" |
    "reinstantiation" |
    "transfer" |
    "locked" |
    "unlocked" |
    "last update of RDAP database" |
    "registrar expiration" |
    "enum validation expiration"
)
```

Figure 34: Override Ruleset for RDAP

10. Normative References

[I-D.newton-json-content-rules]
Newton, A. and P. Cordell, "A Language for Rules Describing JSON Content", draft-newton-json-content-rules-09 (work in progress), September 2017.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <https://www.rfc-editor.org/info/rfc5988>.

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

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <https://www.rfc-editor.org/info/rfc7159>.

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

Appendix A. Acknowledgements

Bernhard Reutner-Fischer and participants of the IETF REGEXT mailing list contributed to the JCR found in this document.

Mario Laffredo provided significant feedback on the rulesets in this document, especially with regard to jCard validation, and provided helpful guidance on validation based on his own experience with the implementation of an RDAP validator.

Author’s Address

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

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