Software Requirements Specification

For

RESTful Whois

Requirements for Version 1.5

CNNIC

Table of Contents

Document Revision History

Name	Date	Reason for Changes	Version
Linlin Zhou	2012-11-19	Creation	0.1
Linlin Zhou	2012-11-28	Requirements review	0.2
Linlin Zhou	2012-12-10	Adjustments according to query and response 01 version	0.3
Linlin Zhou	2013-1-25	Query and response format modification version 02	0.4
Linlin Zhou	2013-3-25	Query supports u-label	0.5
Linlin Zhou	2013-5-7	Response format modification according to version 03	0.6
Linlin Zhou	2013-7-28	Vcard object	0.7
Linlin Zhou	2013-10-12	Query and response format modification	0.8
Linlin Zhou	2013-12-31	Well-known URI and IDN requirements	0.9
Linlin Zhou	2014-1-27	IDN search, nameserver search by IP	1.0
Linlin Zhou	2014-4-4	Response http code examples and case-insensitive	1.1
Linlin Zhou	2014-4-23	Text clarification and add todo appendix	1.2
Linlin Zhou	2014-5-14	Refine details discussed in meetings and response 07 update	1.3
Linlin Zhou	2014-5-29	Reverse domain, authorization update and resultsTruncated requirements clarification	1.4
Linlin Zhou	2014-6-16	Search response order Domain search only supports DNR domain vCard information has no validation process Entity query adjustment	1.5

1. Introduction

1.1 Purpose

This document includes software requirements for RESTful Whois, release number 1.5. RESTful Whois Reference Implementation is open source software distributed under the terms of the BSD public license.

The Internet Engineering Task Force (IETF) is now chartering work to standardize a RESTful-based Registration Data Access Protocol (RDAP) for Domain Registries and Regional Internet Registires that will address many of the deficiencies in the original WHOIS protocol. The implementation of this software is designed to facilitate the evaluation and adoption of the replacement protocol.

To date there are 6 drafts in the IETF Weirds Working Group that will be the basis of this document. They are:

- Using the Registration Data Access Protocol (RDAP) with HTTP (draft-ietf-weirds-using-http-08)
- Unified Registration Data Access Protocol Query Format (draft-ietf-weirds-rdap-query-10)
- JSON Responses for the Registy Data Access Protocol (RDAP) (draft-ietf-weirds-json-response-07)
- Security Services for the Registration Data Access Protocol (draft-ietf-weirds-rdap-sec-06)
- Redirection Service for Registration Data Access Protocol (draft-ietf-weirds-redirects-03)
- Domain Name Registration Data Access Protocol Object Inventory Analysis (draft-ietf-weirds-object-inventory-02)
- Finding the Authoritative Registration Data (RDAP) Service (draft-ietf-weirds-bootstrap-01)

1.2 Document Conventions

In general this document prioritizes in writing the requirements specified in the above IETF drafts. Every requirement statement is assumed to have its own priority as to define in most appropriate way the system behavior. In addition there are various other documents that represent the described system, where it is needed, and serve only for better understanding of the deployment. Please refer to the official documentation of the program at *http://restfulwhois.org* if you have specific questions based on your system.

1.3 Intended Audience and Reading Suggestions

This requirement document contains general information about RESTful Whois, main classes and use cases, functions, features and special technologies. It describes in detail all that RESTful Whois needs to work properly and with safety.

This document is intended for

Developers: in order to be sure they are developing the right project that fulfills requirements provided in this document.

Testers: in order to have an exact list of the features and functions that have to respond according to requirements and provided diagrams.

Users: in order to get familiar with the idea of the project and suggest other features that would make it even more functional.

For each one of the reader types to better understand this document, here is a suggestion of the chapters to read in this document:

- Developers: 1.1, 1.3, 2.2, 2.3, 2.5, 2.7, 3, 4, 5
- Testers: 1.1, 2.1, 2.4, 2.5, 2.7, 3, 4, 5
- Users: 1, 2.1, 2.2,2.3, 2.6, 4.1

1.4 Project Scope

RESTful Whois system is the new implementation of Registration Data Access Protocol. As indicated in the charter of IETF Weirds Working Group, the software shall be for data to be delivered via a RESTful data service using HTTP (optionally using TLS), and use standard features of HTTP to support differential service levels to different classes of user. The data shall have one mandatory format, though the working group may consider other optional formats. The overall effort will be broadly aligned with a subset of the Cross Registry Internet Service Protocol (CRISP) Requirements (RFC 3707), but with the explicit additional goals of producing a simple, easy-to-implement protocol, supporting internationalized registration data and, specifically for name registries, capturing the needs of internationalized domain names in the data model.

1.5 References

More about RESTful Whois can be found at

- *http://restfulwhois.org/*

  This is the project Trac website. Trac is an enhanced wiki and issue tracking system for software development projects. It provides an interface to Git, an integrated wiki and convenient reporting facilities. This web-based software is used for project management. You can find all the documentation, source code and release news here.

- *https://github.com/cnnic/restfulwhois*

  In this website you can find out more about the project and discuss any questions in the forums. You can go back and look at previous releases, code and problems that have been solved. There you can also find information about the developers as well as the project's main characteristics such as programming language and algorithms.

2. Overall Description

2.1 Product Perspective

The major components of this system are clients (browser or command line clients), RESTful Whois server which will also support port 43 proxy function to translate port 43 WHOIS queries into RESTful queries. The other optional component is redirect server that maps the original query URI to the URI format expected by the receiving server. Please refer to Appendix C for URI mapping methods. In addition, a data importing interface is designed in this system for importing data from different registrars. This interface also provides administration functions of user authorization and object extensions.

Bellow in figure 2.1 you can see the schema of the basic client-server relationship of the system.

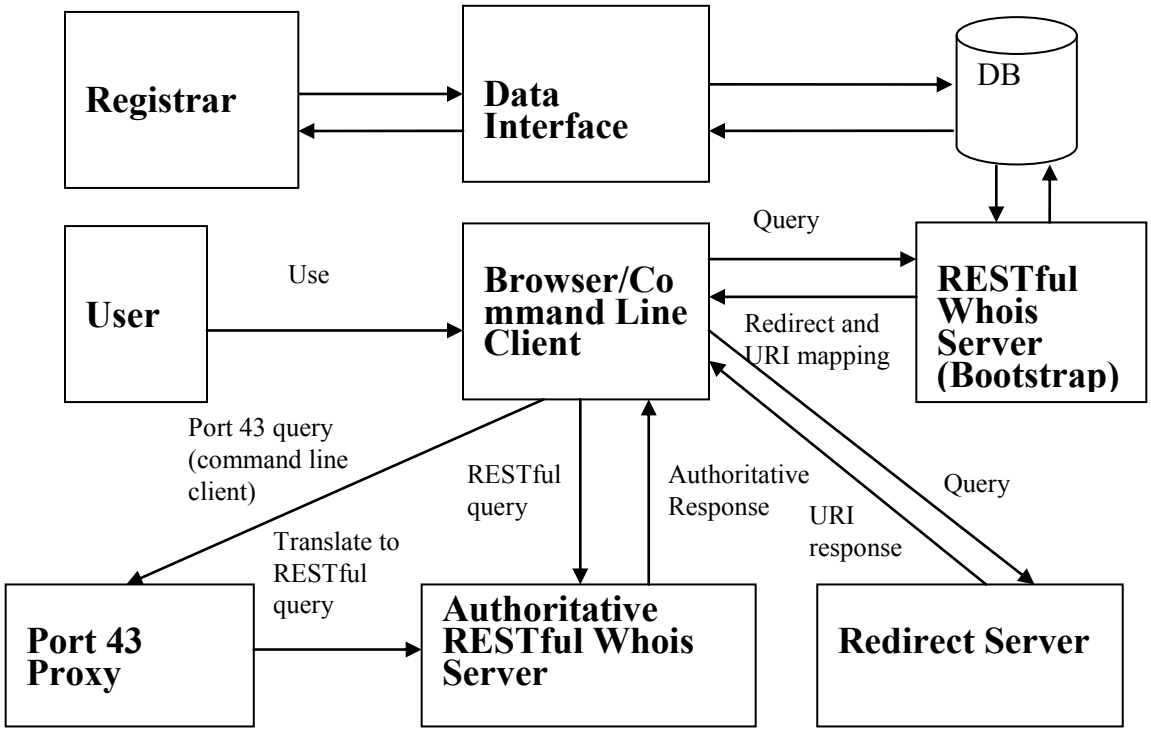


Figure 2.1 System basic schema

2.2 Product Features

The major features this program contains are the following:

- Offer operating support for most of the known and commercial operating systems.
- The default text encoding for JSON responses in RDAP is UTF-8, and all servers and clients must support UTF-8. Servers and clients may optionally support other character encodings.

- For queries, a client issues an HTTP query using GET. The system shall adopt the query URL format according to the working group drafts.
- If the receiving server has the information for the query, it examines the Accept header field of the query and returns a 200 response with a response entity appropriate for the requested format. The system shall apply a data format that is well-established according to the working group drafts.
- An authentication mechanism should be basic authentication method defined in RFC2617 over TLS.
- The system is able to deliver a reply that is effectively a referral or redirect to another server.
- The system should support http caching to decrease network loads and improve overall reliability of system.
- The system should allow for parameter configuration of policy options, i.e., it should not dictate specific policies, but allow deployments to configure their own.
- A uniform error code sets should be defined for system internal exceptions.
- IDN will be considered to be implemented. The basic idea is the client sends the IDN query to the server in A-label, the server will respond all the information it has in different languages.
- The system should also include a PORT 43 WHOIS server proxy.

2.3 User Classes and Characteristics

Physical Actors:

- User: The user is the one that sends queries to the RESTful Whois server with port 43 queries or RESTful queries.

System Actors:

- Client: The client is the system that connects to the server and handles the query.
- RESTful Whois Server: The server is the system that accepts multiple connections from clients and responds the results to the client.
- Redirect Server: The server is used to redirect the queries to the authoritative RESTful Whois server.

2.4 Operating Environment

This program will operate in the following operating environment for the client and the server:

- Apple Mac OS X
- Linux/Unix
- Microsoft Windows

2.5 Design and Implementation Constraints

This program is developed in Java 7. Mysql is the selected database.

2.6 User Documentation

Here are the official links of the project where you can retrieve more information about it and download the latest version:

Online Documentation

https://github.com/cnnic/restfulwhois

http://restfulwhois.org/trac/RestfulWhois/wiki/Documentation

3. System Features

System features are organized by use cases and functional hierarchy so that the main functions of the system will be understandable.

3.1 Accept Header

3.1.1 Description

The HTTP header specified in the data transition process.

3.1.2 Stimulus/Response Sequences

1. Clients should put the media type of the format they desire in the Accept header field.

   Accept: application/json

   Accept: application/rdap+json

   Accept: application/rdap+json; appliaction/json

2. Servers should respond with an appropriate media type in the Content-Type header in accordance with the preference rules for the Accept header in HTTP [RFC2616].

3. Clients may use a generic media type for the desired data format of the response (e.g. "application/json"), but servers should respond with the most appropriate media type (e.g. "application/rdap"). In other words, a client may use "application/json" to express that it desires JSON or "application/rdap" to express that it desires RDAP specific JSON, but the server would respond with "application/rdap".

3.2 Query Parameter

3.2.1 Description

This feature gives the specification of how to handle URL parameters.

3.2.2 Stimulus/Response Sequences

1. For cache-busting, clients may use the adhoc and improbably used query parameter with a random value of their choosing.

2. Servers should ignore unknown query parameters.

3.3 JSON Formats

3.3.1 Description

The server can provide response in JSON format according to the query header.

3.3.2 Stimulus/Response Sequences

1. Clients query the response structured in JSON.

3.4 Response Type

3.4.1 Description

Typically, there are 4 types of server answers: positive answers, redirect answers, negative answers and malformed answers.

3.4.2 Stimulus/Response Sequences

1. If a server has the information requested by the client and wishes to respond to the client with the information according to its policies, it should encode the answer in the format most appropriate according to the standard and defined rules for processing the HTTP Accept header, and return that answer in the body of a 200 response.

2. If a server wishes to inform a client that the answer to a given query can be found elsewhere, it returns either a 301 response code to indicate a permanent move, and it includes an HTTP(s) URL in the Location: header field. The client is expected to issue a subsequent request to satisfy the original query using the given URL without any processing of the URL. In other words, the server is to hand back a complete URL and the client should not have to transform the URL to follow it.

   For this application, such an example of a permanent move might be a Top Level Domain (TLD) operator informing a client the information being sought can be found with another TLD operator (i.e. a query for the domain bar in foo.example is found at http://foo.example/domain/bar).

   For example, if the client sends http://serv1.example.com/weirds/domain/example.com, the server redirecting to https://serv2.example.net/weirds2/ would set the Location: field to the value: https://serv2.example.net/weirds2/domain/example.com.

3. If a server wishes to respond that it has no information regarding the query, it should return a 404 response code. Optionally, it may include additional information regarding the negative answer in the HTTP entity body.

4. If a server receives a query which it cannot understand, it should return a 400 response code. Optionally, it may include additional information regarding this negative answer in the HTTP entity body.

5. 422 http code

   If a server receives a search request but cannot process the request because it does not support a particular style of partial match searching, it SHOULD return an HTTP 422 [RFC4918] error. When returning a 422 error, the server MAY also return an error response body as specified in Section 7 of [I-D.ietf-weirds-json-response] if the requested media type is one that is specified in [I-D.ietf-weirds-using-http].

   If a query string starts with * or is *, server will respond 422.

6. Some servers apply rate limits to deter address scraping and other abuses. When a server declines to answer a query due to rate limits, it returns a 429 response code as described in [RFC6585]. A client that receives a 429 response SHOULD decrease its query rate, and honor the Retry-After header field if one is present.

3.4.3 Some Specific Examples

1. An RDAP server treats each query string as Unicode in UTF-8 encoding. If a string is not valid UTF-8, the server can immediately stop processing the query and return an HTTP 400 error response code.

   http://rdap.restfulwhois.org:80/.well-known/rdap/domain/σειράτάξησυπουργείωνΣύνθεσηυπουργικούσυμβουλίουουουο.bnnhg

   return 400 where Σ is not a valid IDN unicode

2. When a query or search string is a valid punycode, such as xn--hxaajaoebldbselhkqsqmapxidccaaahjrgk3chhdip9bclcgddbb4ooioa.bnnhg, if it exists, return 200 and response body, if it does not exist, return 404 and error message.

3. A search for "/domains?name=xn--*" returns 200 and search results.

4. A search for "/nameservers?name=ns.xn--*", if exists, returns 200 Ok, otherwise returns 404.

5. Query or search whether a record exists

   If not exists, server returns 404, the same way as query.

3.5 Cache Control

3.5.1 Description

Simple cache control requirements are specified below.

3.5.2 Stimulus/Response Sequences

The protocol should support the notion of including in the reply a suggested time-to-live period during which the client is expected to cache the reply and not query for it again.

3.6 Extensibility

3.6.1 Description

Extensible objects can be included in the JSON response.

3.6.2 Stimulus/Response Sequences

1. Extensible JSON names: name = ALPHA *(ALPHA / DIGIT / "_")

2. Example: lunarNic_beforeOneSmallStep

3.7 Common Data Structure

3.7.1 Description

The following are four common data structures to be used by DNRD-AP, NRRD-AP, and other RD-AP protocols.

3.7.2 Stimulus/Response Sequences

1. "rdapConformance" is simply an array of strings, each providing a hint as to the specifications used in the construction of the response. An example rdapConformance data structure.

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

Example rdapConformance structure with custom extensions noted.

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

2. Notices and remarks. "notices" is an array of "notice" objects. Each "notice" object contains a "title" string representing the title of the notice object, an array of strings named "description" for the purposes of conveying any descriptive text about the notice, and an "uri" string holding a URI referencing a service that may provide additional information about the notice. 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"
        }
      ]
    }
  ]

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

3. Language Identifier

The third 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"
```

4. Link Object

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.

```
{
      "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"
}
```

A simple example:

```
"links" :
```

```
[
   {
    "value" : "http://example.com/ip/2001:db8::123",
    "rel" : "self",
    "href" : "http://example.com/ip/2001:db8::123"
   },
   {
    "value" : "http://example.com/ip/2001:db8::123",
    "rel" : "up",
    "href" : "http://example.com/ip/2001:db8::/48"
   }
]
```

rel: alternate/self/up

5. Events

This data structure represents events that have occurred on an instance of an object class.

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

6. Status

This data structure, named 'status', is an array of strings indicating the state of a registered object. See Appendix A 6.1.

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.

8. Public IDs

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

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

The following is an example of a 'publicIds' structure.

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

9. Example

This is an example response with rdapConformance, notices, lang and links 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",
  "description" : [ "A network used for example documentation" ],
  "parentHandle" : "YYYY-RIR",
  "remarks" :
  [
    {
```

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

3.8 Error Respond

3.8.1 Description

Some non-answer responses may return entity bodies with information that could be more descriptive. This feature describes the structure of error respond.

3.8.2 Stimulus/Response Sequences

1. The basic structure of that response is a data 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".

2. Example:

```
{
  "errorCode": 418,
  "title": "Your beverage choice is not available",
  "description":
  [
    "I know coffee has more ummpppphhh.",
    "But I cannot provide."
  ]
}
```

3. 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. The media type for the JSON structure maybe "application/rdap_error+json".

3.9 well-known prefix

To access per-resource metadata, ".well-known/rdap" specifies the rdap resource location.

http://rdap.restfulwhois.org/.well-known/rdap is the URI of CNNIC implementation. The prefix URI must be present.

3.10 Modification of 6 types of query URIs

1. IP examples:

http://rdap.restfulwhois.org/.well-known/rdap/ip/192.0.2.0

http://rdap.restfulwhois.org/.well-known/rdap/ip/192.0.2.0/24

http://rdap.restfulwhois.org/.well-known/rdap/ip/2001:db8::0

2. AS number examples:

2 bytes AS number

http://rdap.restfulwhois.org/.well-known/rdap/autnum/12

4 bytes AS number

http://rdap.restfulwhois.org/.well-known/rdap/autnum/65538

3. Domain examples:

http://rdap.restfulwhois.org/.well-known/rdap/domain/2.0.192.in-addr.arpa

http://rdap.restfulwhois.org/.well-known/rdap/domain/1.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa

http://rdap.restfulwhois.org/.well-known/rdap/domain/xn--fo-5ja.example

4. Nameserver examples:

http://rdap.restfulwhois.org/.well-known/rdap/nameserver/ns1.example.com

http://rdap.restfulwhois.org/.well-known/rdap/nameserver/ns1.xn--fo-5ja.example

5. Entity examples:

http://rdap.restfulwhois.org/.well-known/rdap/entity/XXXX

6. Help examples:

http://rdap.restfulwhois.org/.well-known/rdap/help

3.11 IP Query

3.11.1 Description

This feature allows the user to query IP Whois information.

3.11.2 Stimulus/Response Sequences

1. Query syntax: ip/<IP address> or ip/<CIDR prefix>/<CIDR length>

2. User query http://example.com/ip/XXX/ or http://example.com/ip/XXX/YY/. XXX is either an IPv4 or IPv6 address. XXX/YY is either an IPv4 or IPv6 CIDR specified in RFC4632.

3. Query parameters validation

IPv4:

XXX: 0.0.0.0-255.255.255.255

YY: integer between 0 and 32

0.0.0.0 and 0.*.*.* (* is not 0) are all validated parameters.

IPv6:

XXX: 0:0:0:0:0:0:0:0-ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

YY: integer between 0 and 128

0:0:0:0:0:0:0:0 and 0:*:*:*:*:*:*:* (* is not 0) are all validated parameters.

4. High level structure of response.

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

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

```
{
   "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"
      }
    ]
  }
 ]
}
```

5. Response objects

   Objects include rdapConformance, notices, handle, startAddress, endAddress, ipVersion, name, type, country, parentHandle, status, entities, remarks, links, port43, events and roles. rdapConformance, handle, startAddress, endAddress and ipVersion objects are **required.**

3.12 Autonomous System Query

3.12.1 Description

This feature allows the user to query autonomous system Whois information.

3.12.2 Stimulus/Response Sequences

1. Query symtax: Syntax: autnum/<autonomous system number>

2. User query http://example.com/autnum/65551. In some registries, registration of autonomous system numbers is done on an individual number basis, while other registries may register blocks of autonomous system numbers. The semantics of this query is such that if a number falls within a range of registered blocks, the target of the query is the block registration, and that individual number registrations are considered a block of numbers with a size of 1.

3. Query parameters validation

   Asplain format ASN is supported.

   2-byte ASN: 0-65536

   4-byte only ASN: 65536 – 4294967295

   4 byte AS numbers range from 0 - 4294967295

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

```
{
 "handle" : "XXXX-RIR",
 "startAutnum" : "10",
"endAutnum" : "15",
 "name": "AS-RTR-1",
 "description" : [ "AS for Exchange" ],
 "type" : "DIRECT ALLOCATION",
 "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"
    }
  ],
  "events" :
  [
    {
      "eventAction" : "registration",
      "eventDate" : "1990-12-31T23:59:60Z"
    },
    {
      "eventAction" : "last changed",
      "eventDate" : "1991-12-31T23:59:60Z"
    }
  ],
  "entities" :
  [
    {
      "handle" : "XXXX",
      "vCard" :
      [
        [ "version", {}, "text", "4.0" ],
        [ "fn", {}, "text", "Joe Bob, Inc." ],
        [ "fn", {}, "text", "Bobby Joe Shopping" ],
        [ "label", {}, "text", "123 Maple Ave\n",
                               "Suite 90001\n",
                               "Vancouver\n",
                               "BC\n",
                               "1239\n" ],
                             [ "email", {}, "text", "joe at bob.com" ],
        [ "email", {}, "text", "bob at joe.com" ],
        [ "tel", { "type": "work" }, "uri", "tel:+1-958-555-4321" ],
        [ "tel", { "type": "work" }, "uri", "tel:+1-958-555-4322" ],
        [ "tel", { "type": "fax" }, "uri", "tel:+1-958-555-4323" ],
        [ "tel", { "type": "cell" }, "uri", "tel:+1-958-555-4324" ],
      ],
      "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"
        }
      ],
      "events" :
      [
        {
          "eventAction" : "registration",
          "eventDate" : "1990-12-31T23:59:60Z"
        },
        {
          "eventAction" : "last changed",
          "eventDate" : "1991-12-31T23:59:60Z"
        }
      ]
    }
  ]
}
```

5. Response objects

Response objects include rdapConformance, notices, handle, startAutnum, endAutnum, name, type, status, country, entities, remarks, links, port43 and events. rdapConformance, handle, startAutnum, and endAutnum are Must objects.

3.13 Domain Query

3.13.1 Description

This feature allows the user to query domain Whois information.

3.13.2 Stimulus/Response Sequences

1. Query Syntax: domain/<domain name>

2. User query http://example.com/domain/XXXX/. XXXX is a fully-qualified domain name [RFC4343] in either the in-addr.arpa or ip6.arpa zones (for RIRs) or a fully-qualified domain name in a zone administered by the server operator (for DNRs). Internationalized domain names represented in either A-label or U-label format [RFC5890] are also valid domain names. IDNs SHOULD NOT be represented as a mixture of A-labels and U-labels; that is, any IDN SHOULD use only A-labels or only U-labels.

   If the client sends the server an IDN in U-label format, servers that support IDNs MUST convert the IDN into A-label format and perform IDNA processing as specified in RFC 5891 [RFC5891]. The server should perform an exact match lookup using the A-label.

   Example: http://example.com /domain/2.0.192.in-addr.arpa or http://example.com /domain/example.com

3. Query parameters validation

in-addr.arpa and ip6.arpa zones are supported, but domains in format of \[xy...y/zz].ip6.arpa or \[xy...y/zz].ip6.arpa are not supported. The reverse domain should match the corresponding IP segments.

IDN validation, U-labels with any of the following characteristics MUST be rejected:

1) Labels that are not in NFC.

2) Labels containing "--" (two consecutive hyphens) in the third and fourth character positions.

3) Labels whose first character is a combining mark (see The Unicode Standard, Section 2.11 [Unicode]).

4) Labels containing prohibited code points, i.e., those that are assigned to the "DISALLOWED" category of the Tables document [RFC5892].

5) Labels containing code points that are identified in the Tables document as "CONTEXTJ", i.e., requiring exceptional contextual rule processing on lookup, but that do not conform to those rules. Note that this implies that a rule must be defined, not null: a character that requires a contextual rule but for which the rule is null is treated in this step as having failed to conform to the rule.

6) Labels containing code points that are identified in the Tables document as "CONTEXTO", but for which no such rule appears in the table of rules. Applications resolving DNS names or carrying out equivalent operations are not required to test contextual rules for "CONTEXTO" characters, only to verify that a rule is defined (although they MAY make such tests to provide better protection or give better information to the user).

7) Labels containing code points that are unassigned in the version of Unicode being used by the application, i.e., in the UNASSIGNED category of the Tables document.

The above 4)-7) items are not supported currently.

4. High level structure of response.

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

The RIR Domain Object Class:

```
{
  "handle" : "XXXX",
  "ldhName" : "192.in-addr.arpa",
  "nameServers" :
  [
    { "ldhName" : "ns1.rir.example" },
    { "ldhName" : "ns2.rir.example" }
  ],
  "delegationKeys" :
  [
    {
      "algorithm": 7,
      "digest" : "E68C017BD813B9AE2F4DD28E61AD014F859ED44C",
      "digestType" : 1,
      "keyTag" : 53814
    }
  ],
  "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"
    }
  ],
  "events" :
  [
    {
      "eventAction" : "registration",
      "eventDate" : "1990-12-31T23:59:60Z"
    },
    {
                        "eventAction" : "last changed",
      "eventDate" : "1991-12-31T23:59:60Z",
      "eventActor" : "joe@bob.com"
    }
```

```
],
"entities" :
[
  {
   "handle" : "XXXX",
   "vCard" :
   [
     [ "version", {}, "text", "4.0" ],
     [ "fn", {}, "text", "Joe Bob, Inc." ],
     [ "fn", {}, "text", "Bobby Joe Shopping" ],
     [ "label", {}, "text", "123 Maple Ave\n",
                            "Suite 90001\n",
                            "Vancouver\n",
                            "BC\n",
                            "1239\n" ],
     [ "email", {}, "text", "joe at bob.com" ],
     [ "email", {}, "text", "bob at joe.com" ],
     [ "tel", { "type": "work" }, "uri", "tel:+1-958-555-4321" ],
     [ "tel", { "type": "work" }, "uri", "tel:+1-958-555-4322" ],
     [ "tel", { "type": "fax" }, "uri", "tel:+1-958-555-4323" ],
     [ "tel", { "type": "cell" }, "uri", "tel:+1-958-555-4324" ],
   ],
   "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"
     }
   ],
   "events" :
   [
     {
          "eventAction" : "registration",
      "eventDate" : "1990-12-31T23:59:60Z"
```

```
            },
            {
             "eventAction" : "last changed",
             "eventDate" : "1991-12-31T23:59:60Z",
             "eventActor" : "joe@bob.com"
            }
           ]
          }
         ],
          "network" :
            {
             "handle" : "xxx",
             "startAddress" : "xxx",
             "endAddress" : "xxx",
             "ipVersion" : "xxx",
             "name": "xxx",
             "type" : "xxx",
             "country" : "xxx",
             "parentHandle" : "xxx",
             "status" : [ "xxx" ]
            }
  }
```

The DNR Domain Object Class:

```
{
   "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" ],
      "variantNames" :
      [
        {
         "ldhName": "xn--fo-8ja.example",
```

```
          "unicodeName" : "foo.example"
        }
      ]
                  }
  ],
  "status" : [ "locked", "transferProhibited" ],
  "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"
       }
     ],
     "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"
      }
    ],
    "events" :
    [
      {
       "eventAction" : "registration",
       "eventDate" : "1990-12-31T23:59:60Z"
      },
      {
       "eventAction" : "last changed",
       "eventDate" : "1991-12-31T23:59:60Z"
      }
    ]
  }
],
"delegationKeys" :
[
  {
   "algorithm": 7,
   "digest" : "E68C017BD813B9AE2F4DD28E61AD014F859ED44C",
   "digestType" : 1,
   "keyTag" : 53814
  }
],
"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"
   }
  ],
  "port43" : "whois.example.net",
  "events" :
  [
   {
    "eventAction" : "registration",
    "eventDate" : "1990-12-31T23:59:60Z"
   },
   {
    "eventAction" : "last changed",
    "eventDate" : "1991-12-31T23:59:60Z",
    "eventActor" : "joe@bob.com"
   },
   {
    "eventAction" : "transfer",
    "eventDate" : "1991-12-31T23:59:60Z",
    "eventActor" : "joe@bob.com"
   },
   {
    "eventAction" : "expiration",
    "eventDate" : "2016-12-31T23:59:60Z",
    "eventActor" : "joe@bob.com"
   }
  ],
  "entities" :
  [
   {
    "handle" : "XXXX",
    "vCard" :
    [
      [ "version", {}, "text", "4.0" ],
      [ "fn", {}, "text", "Joe Bob, Inc." ],
      [ "fn", {}, "text", "Bobby Joe Shopping" ],
      [ "label", {}, "text", "123 Maple Ave\n",
                                  "Suite 90001\n",
                        "Vancouver\n",
                        "BC\n",
                        "1239\n" ],
      [ "email", {}, "text", "joe at bob.com" ],
      [ "email", {}, "text", "bob at joe.com" ],
      [ "tel", { "type": "work" }, "uri", "tel:+1-958-555-4321" ],
      [ "tel", { "type": "work" }, "uri", "tel:+1-958-555-4322" ],
      [ "tel", { "type": "fax" }, "uri", "tel:+1-958-555-4323" ],
      [ "tel", { "type": "cell" }, "uri", "tel:+1-958-555-4324" ],
    ],
    "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"
      }
    ],
    "events" :
    [
      {
        "eventAction" : "registration",
        "eventDate" : "1990-12-31T23:59:60Z"
      },
      {
        "eventAction" : "last changed",
        "eventDate" : "1991-12-31T23:59:60Z"
      }
    ]
  }
 ]
}
```

5. Response objects

Objects include rdapConformance, notices, handle, ldhName, unicodeName, variants (relation, idnTable and variantNames), nameServers, secureDNS (zoneSigned, delegationSigned, maxSigLife, dsData (keyTag, algorithm, digest, digestType, events and links), keyData (flagsprotocol, publicKey, algorithm, events and links)), entities, status, publicIds, remarks, links, port43, events and network. rdapConformance, handle and ldhName are **required**.

3.14 Name Server Query

3.14.1 Description

This feature allows the user to query name server Whois information.

3.14.2 Stimulus/Response Sequences

1. Query syntax: nameserver/<name server name>

2. User query http://example.com/nameserver/<name server name>. The <name server name> parameter represents a fully qualified name as specified in RFC 952 [RFC0952] and RFC 1123 [RFC1123]. Internationalized names represented in either A-label or U-label format [RFC5890] are also valid name server names. IDN labels SHOULD NOT be represented as a mixture of A-labels and U-labels.

Example: http://example.com/nameserver/ns1.example.com

3. The following are 3 examples of a nameserver object.

Example with all values given:

```
{
 "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"
  }
 ],
 "port43" : "whois.example.net",
 "events" :
 [
  {
   "eventAction" : "registration",
   "eventDate" : "1990-12-31T23:59:60Z"
  },
  {
   "eventAction" : "last changed",
   "eventDate" : "1991-12-31T23:59:60Z",
   "eventActor" : "joe@bob.com"
  }
 ]
}
```

The following is an example of the simplest nameserver object:

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

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" ] }
}
```

4. Response objects

Response objects include rdapConformance, notices, handle, ldhName, unicodeName, ipAddresses (v6 and v4), entities, status, remarks, links, port43 and events. rdapConformance and ldhName are **required**.

3.15 Entity Query

3.15.1 Description

This feature allows the user to query entity Whois information.

3.15.2 Stimulus/Response Sequences

1. Query syntax: entity/<handle>

2. An entity usually represents a contact, registrant or registrar. User query http://example.com/ entity/<handle>.

   Example: http://example.com/ entity/CID-4005

3. RIR Entity Object Class:

```
{
      "handle" : "XXXX",
      "vCard" :
      [
        [ "version", {}, "text", "4.0" ],
        [ "fn", {}, "text", "Joe Bob, Inc." ],
        [ "fn", {}, "text", "Bobby Joe Shopping" ],
        [ "label", {}, "text", "123 Maple Ave\n",
                               "Suite 90001\n",
                               "Vancouver\n",
                               "BC\n",
                               "1239\n" ],
        [ "email", {}, "text", "joe at bob.com" ],
        [ "email", {}, "text", "bob at joe.com" ],
        [ "tel", { "type": "work" }, "uri", "tel:+1-958-555-4321" ],
        [ "tel", { "type": "work" }, "uri", "tel:+1-958-555-4322" ],
        [ "tel", { "type": "fax" }, "uri", "tel:+1-958-555-4323" ],
        [ "tel", { "type": "cell" }, "uri", "tel:+1-958-555-4324" ],
      ],
      "roles" : [ "registrant" ],
```

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


The DNR Entity Object Class:


```
{
    "handle" : "XXXX",
    "vCard" :
    [
      [ "version", {}, "text", "4.0" ],
      [ "fn", {}, "text", "Joe Bob, Inc." ],
      [ "fn", {}, "text", "Bobby Joe Shopping" ],
      [ "label", {}, "text", "123 Maple Ave\n",
                             "Suite 90001\n",
                             "Vancouver\n",
                             "BC\n",
                             "1239\n" ],
      [ "email", {}, "text", "joe at bob.com" ],
      [ "email", {}, "text", "bob at joe.com" ],
      [ "tel", { "type": "work" }, "uri", "tel:+1-958-555-4321" ],
```

```
      [ "tel", { "type": "work" }, "uri", "tel:+1-958-555-4322" ],
      [ "tel", { "type": "fax" }, "uri", "tel:+1-958-555-4323" ],
      [ "tel", { "type": "cell" }, "uri", "tel:+1-958-555-4324" ],
    ],
    "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"
      }
    ],
    "port43" : "whois.example.net",
    "events" :
    [
      {
        "eventAction" : "registration",
        "eventDate" : "1990-12-31T23:59:60Z"
      },
      {
        "eventAction" : "last changed",
        "eventDate" : "1991-12-31T23:59:60Z",
        "eventActor" : "joe@bob.com"
      }
    ]
  }
```

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. If the nested entity is the same with the top level entity object, the nested entity will not show in the response.

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

```
      },
      ...
    ],
    ...
  }
```

4. vcardArray

The following example shows the vcardArray required objects and their appearance times.

```
"vcardArray":[
      "vcard",
      [
        ["version", {}, "text", "4.0"],//required and in the top most level
        ["fn", {}, "text", "xxx"],//appear once at most
        ["kind", {}, "text", "xxx"],//appear once at most
        ["org", {
          "type":"xxx"
        }, "text", "xxx"],// appear once at most
        ["title", {}, "text", "xxx"],// appear once at most
        ["adr",
          { "type":"xxx" },
          "text",
          [
            "xxx",
            "xxx",
            "xxx",
            "xxx",
            "xxx",
            "xxx",
            "xxx"
          ]
        ], // can appear multiple objects
        ["tel",
          { "type":["xxx", "xxx"] },
          "uri", "xxx"
        ], // can appear multiple objects
        ["tel",
          {
            "type":["xxx", "xxx", "xxx", "xxx", "xxx"]
          },
          "uri",
          "xxx"
        ], // can appear multiple objects
        ["email",
          { "type":"xxx" },
          "text", "xxx"
        ], // can appear multiple objects
        ["url", { "type":"xxx" },
          "uri", "xxx"]
      ] // can appear multiple objects
    ]
```

5. Response objects

Response objects include rdapConformance, notices, handle, vcardArray, roles, publicIds, entities, remarks, links, events, status, port43, asEventActor, networks and autnums. rdapConformance、handle、vcardArray are **required**.

6. There's no validation process for vCard information in entity object which will show the information stored in database.

3.16 Help Query

1. Query syntax: help

2. Response example

```
{
  "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"
        }
      ]
    }
  ]
}
```

3.17 Authentication

3.17.1 Description

RDAP must include an authentication framework that can accommodate anonymous access as well as verification of identities using a range of authentication methods and credential services.

3.17.2 Stimulus/Response Sequences

Authentication:

1. The RDAP authentication framework must use authentication methods that are fully specified and available to existing HTTP clients and servers, and be capable of supporting future authentication methods defined for use with HTTP.

2. Clients and servers must implement the authentication framework specified in RFC2617. The "basic" scheme can be used to send a client's user name and password to a server in plaintext, based64-encoded form. The "digest" scheme can be used to authenticate a client without exposing the client's plaintext password. If the "basic" scheme is used another protocol (such as HTTP Over TLS [RFC2818]) must be used to protect the client's credentials from disclosure while in transit. Basic plus https mechanism is adopted in this system.

3. The Transport Layer Security Protocol [RFC5246] includes an optional feature to identify and authenticate clients who possess and present a valid X.509 digital certificate [RFC5280]. Support for this feature is OPTIONAL.

Federated Authentication:

1. Including a federated authentication mechanism that permits a client to access multiple RDAP servers in the same federation with one credential.

2. There are 3 possible approaches to solve federated authentication. One solution is using OAuth [RFC6749], the other is OpenID, and the third one is using Certification Authority (CA) specified in Transport Layer Security Protocol [RFC5246].

3.18 Authorization

3.18.1 Description

This feature allows the server operators to offer varying degrees of access depending on policy and need.

3.18.2 Stimulus/Response Sequences

1. Including an authorization framework that is capable of providing granular (per registration data object) access controls according to the policies of the operator.

2. Server operators will offer varying degrees of access depending on policy and need in conjunction with the authentication methods.

   Examples:

   - Clients will be allowed access only to data for which they have a relationship.
   - Unauthenticated or anonymous access status may not yield any contact information.
   - Full access may be granted to a special group of authenticated clients.

This system could allow user to configure object fields displayed in the response according to different level of authorization, and could configure displayed information according to query parameter. The response information will show all the nested objects. For example, if a user has the authorization of a domain, the response information will show all the domain, ns, entity etc. objects. It is possible that this user does not

have the authorization of the ns in the domain response, so the response information will not support this kind of query.

3.19 Redirection

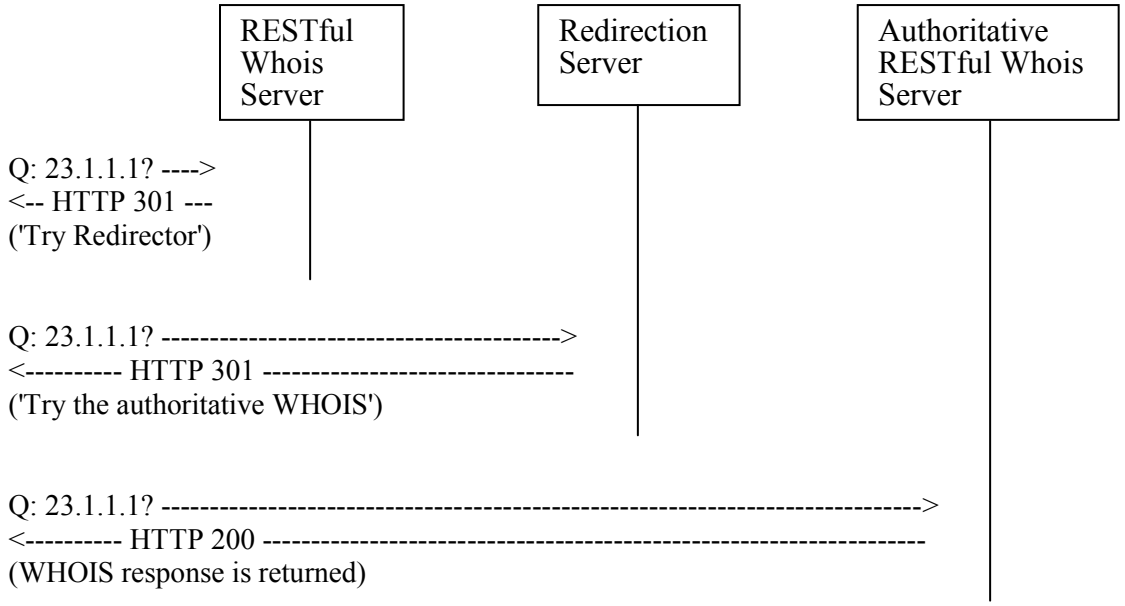
3.19.1 Description

A RESTful WHOIS redirection service requirement is specified in this section.

3.19.2 Stimulus/Response Sequences

1. A RESTful WHOIS server could provide a 301 MOVED PERMANENTLY Redirect answer pointing to an URL hosted on a different RESTful WHOIS server.

2. Redirection query



3. Avoid loops in redirection. There's one solution in the current draft. When using an URI to indicate redirection, the URI have the structure like /redirect/[step]. The step parameter is a counter of redirection times. The step can not be greater than the locally set threshold.

3.20 Modification of Search URI

Domain search:
http://rdap.restfulwhois.org/.well-known/rdap/domains?name=example*.com

Parameter name could be either A-label or U-label format.

Nameserver search:
http://rdap.restfulwhois.org/.well-known/rdap/nameservers?name=ns1.example*.com
http://rdap.restfulwhois.org/.well-known/rdap /nameservers?ip=YYYY

Entity search:
http://rdap.restfulwhois.org/.well-known/rdap/entities?fn=Bobby%20Joe*

http://rdap.restfulwhois.org/.well-known/rdap/entities?handle=CID-40*

3.21 Search

Partial string searching uses the asterisk ('*', ASCII value 0x002A) character to match zero or more trailing characters. A character string representing multiple domain name labels MAY be concatenated to the end of the search pattern to limit the scope of the search.

If a server receives a search request but cannot process the request because it does not support a particular style of partial match searching, it SHOULD return an HTTP 422 [RFC4918] error.

For the purposes of comparison, Normalization Form KC (NFKC, [Unicode-UAX15]) with case folding is used to maximize predictability and the number of matches.

1. Domain Search

Syntax: /domains?name=XXXX

The following path would be used to find **DNR** information for domain names matching the "example*.com" pattern: /domains?name=example*.com

Internationalized Domain Names (IDNs) in U-label format [RFC5890] can also be used as search patterns. Searches for these names are of the form /domains?name=XXXX, where XXXX is a search pattern representing a domain name in U-label format [RFC5890].

Validation specification, refer to section 3.13.2.

The response is sorted in ascending order of ldh_name.

2. Nameserver Search

Syntax: nameservers?name=<nameserver search pattern>

The following path would be used to find DNR information for name server names matching the "ns1.example*.com" pattern: /nameservers?name=ns1.example*.com

Internationalized name server names in U-label format [RFC5890] can also be used as search patterns. Searches for these names are of the form //nameservers?name=XXXX, where XXXX is a search pattern representing a name server name in U-label format [RFC5890].

Syntax: /nameservers?ip=YYYY

YYYY is a search pattern representing an IPv4 [RFC1166] or IPv6 [RFC5952] address. The following URL would be used to search for name server names that resolve to the "192.0.2.0" address:

http://example.com/.well-known/rdap/nameservers?ip=192.0.2.0

The response is sorted in ascending order of ldh_name.

3. Entity Search

Syntax: entities?fn=<entity name search pattern>

Syntax: entities?handle=<entity handle search pattern>

Searches for entity information are of the form /entities?fn=XXXX, where XXXX is a search pattern representing an entity name as specified in Section 7.1 of [I-D.ietf-weirds-json-response]. The following path would be used to find information for entity names matching the "Bobby Joe*" pattern.

/entities?fn=Bobby%20Joe*

where XXXX is a search pattern representing an entity handle as specified in Section 6.1 of [I-D.ietf-weirds-json-response]. The following path would be used to find information for entity names matching the "CID-40*" pattern.

/entities?handle=CID-40*

URLs MUST be properly encoded according to the rules of [RFC3986]. In the example above, "Bobby Joe*" is encoded to "Bobby%20Joe*".

The response is sorted in ascending order of handle.

4. Search Response

The names of the arrays are as follows:

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

```
{
  "rdapConformance" :
  [
    "rdap_level_0"
  ],
  ...
  "resultsTruncated" : true
  "domainSearchResults" :
  [
    {
```

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


3.22 Unicode normalization forms

Reference: http://unicode.org/reports/tr15/.

According to RFC5890, NFC is the selected normalization form.

For example, our database is set to be NFC or NFKC. If a user searches sé* will match the “sémaphore” record in database.

Another example, if database is set to be NFD or NFKD and a user searched se*, “separate” and “sémaphore” records will be returned. Because In NFD or NFKD, é is "Decomposed" as e followed by acute accent: Unicode U+0045 (e) U+00C9 (‘).

3.23 Fullwidth and Halfwidth

Servers map fullwidth and halfwidth characters to their decomposition equivalents.

3.24 Case-insensitive

If the pattern consists entirely of ASCII characters of domain and ns search, servers should use case-insensitive prefix matching against ASCII labels to determine partial matches.

3.25 Truncated Responses

"resultsTruncated" can be used in search response and also can be used where a single object has been returned and data in that object has been truncated. The objects of “networks” and “autnums” can be truncated in the entity lookup response in this system.

4. External Interface Requirements

4.1 Port 43 Proxy Interface

The proxy interface should be able accept client queries that conforms to RFC 3912, with the parameters as dictated by the gTLD registry/registrar policies. The proxy will translate port 43 WHOIS queries into RESTful queries, query the RESTful RDAP Service, and return the plain text results via PORT 43 back to clients.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The RESTful Whois system should at least follow the performance specification defined in the new gTLD guide book.

Availability ≤ 864 min of downtime (≈ 98%)

Query RTT ≤ 2000 ms, for at least 95% of the queries

Update time ≤ 60 min, for at least 95% of the probes

5.2 Safety Requirements

The security requirements can be found in the Security Services for the Registration Data Access Protocol (draft-ietf-weirds-rdap-sec-06) for more details. Authentication and authorization parts are included in the section 3.13 and 3.14. Other security requirements such as availability and data confidentiality are described in the above draft. The following are some important points of these security requirements.

- **Availability**

An RDAP service has to be available to be useful. There are no RDAP-unique requirements to provide availability, but as a general security consideration a service operator needs to be aware of the issues associated with denial of service. A thorough reading of [RFC4732] is recommended.

- **Data Confidentiality**

    - RDAP or a protocol layer used by RDAP must include features to protect plaintext client credentials used for client authentication.

- Being fully specified and available to existing HTTP clients and servers.

- Being capable of supporting future data confidentiality methods defined for use with HTTP.

- The HTTP "basic" authentication scheme can be used to authenticate a client. When this scheme is used HTTP over TLS [RFC2818] must be used to protect the client's credentials from disclosure while in transit. HTTP Over TLS MAY also be used to protect client-server data exchanges if the policy of the server operator requires encryption.

5.3 Software Quality Attributes

- RESTful Whois is developed under BSD distribution license terms and conditions and can be distributed under those terms.

- RESTful Whois can be found and downloaded from *https://github.com/cnnic/restfulwhois* and the project's official website at *http://restfulwhois.org/* for free.

6. Appendix A. Response Reference

6.1 Status

Many of the object classes have a member named 'status'. This member is an array of strings, with each string denoting a status associated with the containing object. The following is a list of suggested values to use in the 'status' array:

1. 'validated' -- 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.

2. 'renew prohibited' --Renewal or reregistration of the object instance is forbidden.

3. 'update prohibited' -- Updates to the object instance are forbidden.

4. 'transfer prohibited' -- Transfers of the registration from one registrar to another are forbidden. This type of status normally applies to DNR domain names.

5. 'delete prohibited' -- Deletion of the registration of the object instance is forbidden. This type of status normally applies to DNR domain names.

6. 'proxy' -- The registration of the object instance has been performed by a third party. This is most commonly applied to entities.

7. 'private' -- The information of the object instance is not designated for public consumption. This is most commonly applied to entities.

8. 'redacted' -- The information of the object instance is not designated for public consumption. This is most commonly applied to entities.

9. 'obscured' -- 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.

10. 'associated' -- 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.

11. 'active' -- 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.

12. 'inactive' -- The object instance is not in use.

13. 'locked' -- Changes to the object instance cannot be made, including the assocation of other object instances.

14. 'pending create' -- A request has been received for the creation of the object instance but this action is not yet complete.

15. 'pending renew' -- A request has been received for the renewal of the object instance but this action is not yet complete.

16. 'pending transfer' -- A request has been received for the transfer of the object instance but this action is not yet complete.

17. 'pending update' -- A request has been received for the update or modification of the object instance but this action is not yet complete.

18. 'pending delete' -- 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 nolonger published in DNS but has not yet been purged from the registry database.

6.2 Roles

Entity object classes have a member named 'roles'. This member is an array of strings, with each string indicating the role or relationship the entity object instance has with a containing object, such as a domain name or IP network. An entity object instance can have more than one type of relationship with a containing object.

The following is a list of suggested values to use in the 'roles' array:

1. 'registrant' -- The entity object instance is the registrant of the registration.

2. 'technical' -- The entity object instance is a technical contact for the registration.

3. 'administrative' -- The entity object instance is an administrative contact for the registration.

4. 'abuse' -- The entity object instance handles network abuse issues on behalf of the registrant of the registration.

5. 'billing' -- The entity object instance handles payment and billing issues on behalf of the registrant of the registration.

6. 'registrar' -- The entity object instance represents the authority responsible for the registration in the registry.

7. 'reseller' -- The entity object instance represents a third party through which the registration was conducted (i.e. not the registry or registrar).
8. 'sponsor' -- The entity object instance represents a domain policy sponsor, such as an ICANN approved sponsor.
9. 'proxy' -- The entity object instance represents a proxy for another entity object, such as a registrant.
10. 'notifications' -- An entity object instance designated to receive notifications about association object instances.
11. 'noc' -- The entity object instance handles communications related to a network operations center (NOC).

6.3 Variant Relations

A structure for noting variants of domain names and the relationship those variants have with a registered domain name. The following is a list of suggested values to use as the variant relation values:

1. 'registered' -- the variant names are registered in the registry.
2. 'unregistered' -- the variant names are not found in the registry.
3. 'registration restricted' -- registration of the variant names is restricted to certain parties or within certain rules.
4. 'open registration' -- registration of the variant names is available to generally qualified registrants.
5. 'conjoined' -- registration of the variant names is conjoined with the registration of the containing domain registration.

7. Appendix B. Authentication Reference

7.1 HTTP Authentication Mechanisms Defined in RFC2617

1. HTTP Basic Authentication

Server side:

When the server wants the user agent to authenticate itself towards the server, it can send an authentication request.

This request should be sent using the HTTP 401 Not Authorized response code containing a WWW-Authenticate HTTP header.

The WWW-Authenticate header for basic authentication (used most often) is constructed as following:

WWW-Authenticate: Basic realm="insert realm"

Client side

When the user agent wants to send the server authentication credentials it may use the *Authorization* header.

The *Authorization* header is constructed as follows:

1. Username and password are combined into a string "username:password"
2. The resulting string literal is then encoded using Base64
3. The authorization method and a space i.e. "Basic" is then put before the encoded string.

For example, if the user agent uses 'Aladdin' as the username and 'open sesame' as the password then the header is formed as follows:

Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

2. HTTP Digest Authentication

RFC2617 (to be added)

7.2 Certification Authority Defined in RFC5246

RFC5246 (to be added)

8. Appendix C. Redirection Reference

8.1 Loop Control Used in DNS

RFC 1034 warns against such situations on P35, "Bound the amount of work (packets sent, parallel processes started) so that a request can't get into an infinite loop or start off a chain reaction of requests or queries with other implementations EVEN IF SOMEONE HAS INCORRECTLY CONFIGURED SOME DATA." Bind defines a loop number 16 in recursive or authoritative servers. If refferal number exceeds the threshold, DNS will not respond.

8.2 Bootstrap Approaches

At the first, there are several options for bootstrap. They are shown below.

1. Master Server

   A well known master redirect server strores all the mapping information. All queries start from this server whose query volume will be very high, usually above 200q/s.

2. Static Table

   Build a static table or file into the clients. This solution may be suitable for RIRs, but not suitable for DNRs. Because there are only 5 RIRs in the world, the IP addresses don’t change a lot. But domain name Whois servers usually change too often.

3. Random

Current RIR redirect model: user contacts with the one of the registry and the Whois server sends a query and return the response to the user.

4. DNS

- SRV

_weirds._tcp.TLD SRV 0 5 80 weirds.server.net.

_weirds._tcp.TLD SRV 0 5 443 weirds.server.net.

- NAPTR

TLD NAPTR 100 10 "U" "weirds:http" "!.*!http://weirds.server.net!"

TLD NAPTR 100 10 "U" "weirds:https" "!.*!https://weirds.server.net!"

- CNAME or A

TLD.weirds.arpa. CNAME XXX

weirds.server.net CNAME XXX

After a long discussion about the bootstrap solution, please refer to the draft-ietf-weirds-bootstrap-01 (http://datatracker.ietf.org/doc/draft-ietf-weirds-bootstrap/). This document proposes an IANA registry method to get the URL of different object authoritative URL.

9. Appendix D. Todo List

This year's first finalized version will be released in July, 2014. There are 3 functions we plan to realize after July. The priority is from high to low.

1. Html website update
2. Openid
3. OAuth