Network Working Group A. Newton
Internet-Draft ARIN
Intended status: Standards Track B. Ellacott
Expires: February 23, 2015 APNIC
N. Kong
CNNIC
August 22, 2014

HTTP usage in the Registration Data Access Protocol (RDAP)
draft-ietf-weirds-using-http-10

Abstract

This document is one of a collection that together describe the Registration Data Access Protocol (RDAP). It describes how RDAP is transported using the Hypertext Transfer Protocol (HTTP). RDAP is a successor protocol to the very old WHOIS protocol.

Status of This Memo

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

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

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

This Internet-Draft will expire on February 23, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

This document describes the usage of HTTP[RFC7230] for Registration Data Directory Services. The goal of this document is to tie together usage patterns of HTTP into a common profile applicable to the various types of Directory Services serving Registration Data using RESTful practices. By giving the various Directory Services common behavior, a single client is better able to retrieve data from Directory Services adhering to this behavior.

The registration data expected to be presented by this service is Internet resource registration data - registration of domain names

and Internet number resources. This data is typically provided by WHOIS [RFC3912] services, but the WHOIS protocol is insufficient to modern registration data service requirements. A replacement protocol is expected to retain the simple transactional nature of WHOIS, while providing a specification for queries and responses, redirection to authoritative sources, support for Internationalized Domain Names (IDNs, [RFC5890]), and support for localized registration data such as addresses and organisation or person names.

In designing these common usage patterns, this document introduces considerations for a simple use of HTTP. Where complexity may reside, it is the goal of this document to place it upon the server and to keep the client as simple as possible. A client implementation should be possible using common operating system scripting tools (e.g. bash and wget).

This is the basic usage pattern for this protocol:

1. A client issues an HTTP query using GET[RFC7231]. As an example, a query for the network registration 192.0.2.0 might be http://example.com/ip/192.0.2.0. [I-D.ietf-weirds-rdap-query] details the various queries used in RDAP.

2. 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. [I-D.ietf-weirds-json-response] details a response in JSON standardized by the IETF. Other formats may be used as well.

3. If the receiving server does not have the information for the query but does have knowledge of where the information can be found, it will return a redirection response (3xx) with the Location: header field containing an HTTP(S) URL (Uniform Resource Locator) pointing to the information or another server known to have knowledge of the location of the information. The client is expected to re-query using that HTTP URL.

4. If the receiving server does not have the information being requested and does not have knowledge of where the information can be found, it returns a 404 response.

5. If the receiving server will not answer a request for policy reasons, it will return an error response (4xx) indicating the reason for giving no answer.

It is important to note that it is not the intent of this document to redefine the meaning and semantics of HTTP. The purpose of this

document is to clarify the use of standard HTTP mechanisms for this application.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

As is noted in Security and Stability Advisory Committee (SSAC) Report on WHOIS Terminology and Structure [SAC-051], the term "WHOIS" is overloaded, often referring to a protocol, a service and data. In accordance with [SAC-051], this document describes the base behavior for a Registration Data Access Protocol (RDAP). [SAC-051] describes a protocol profile of RDAP for Domain Name Registries (DNRs), the Domain Name Registration Data Access Protocol (DNRD-AP).

In this document, an RDAP client is an HTTP User Agent performing an RDAP query, and an RDAP server is an HTTP server providing an RDAP response. RDAP query and response formats are described in [I-D.ietf-weirds-rdap-query] and [I-D.ietf-weirds-json-response], while this document describes how RDAP clients and servers use HTTP to exchange queries and responses. [I-D.ietf-weirds-rdap-sec] describes security consideration for RDAP.

3. Design Intents

There are a few design criteria this document attempts to meet.

First, each query is meant to return either zero or one result. With the maximum upper bound being set to one, the issuance of redirects is simplified to the traditional query/response model used by HTTP [RFC7230]. Should an entity contain more than one result, some of which are better served by other servers, the redirection model becomes much more complicated.

Second, the semantics of the request/response allow for future and/or non-standard response formats. In this document, only a JSON [RFC4627] response media type is noted, with the response contents to be described separately (see [I-D.ietf-weirds-json-response]). This document only describes how RDAP is transported using HTTP with this format.

Third, this protocol is intended to be able to make use of the range of mechanisms available for use with HTTP. HTTP offers a number of mechanisms not described further in this document. Operators are able to make use of these mechanisms according to their local policy, including cache control, authorization, compression, and redirection.

HTTP also benefits from widespread investment in scalability, reliability, and performance, and widespread programmer understanding of client behaviours for RESTful web services, reducing the cost to deploy Registration Data Directory Services and clients.

4. Queries

4.1. HTTP Methods

Clients SHOULD use either the HTTP GET or HEAD methods (see [RFC7231]). The GET method should be used to retreive a response body. The HEAD method can be used to determine existence of data on the server. Servers are under no obligation to support other HTTP methods.

4.2. Accept Header

To indicate to servers that an RDAP response is desired, clients include an Accept: header field with an RDAP specific JSON media type, the generic JSON media type, or both. Servers receiving an RDAP request return an entity with a Content-Type: header containing the RDAP specific JSON media type.

This specification does not define the responses a server returns to a request with any other media types in the Accept: header field, or with no Accept: header field. One possibility would be to return a response in a media type suitable for rendering in a web browser.

4.3. Query Parameters

Servers MUST ignore unknown query parameters. Use of unknown query parameters for cache-busting is described in Appendix B.

5. Types of HTTP Response

This section describes the various types of responses a server may send to a client. While no standard HTTP response code is forbidden in usage, at a minimum clients SHOULD understand the response codes described in this section as they will be in common use by servers. It is expected that usage of response codes and types for this application not defined here will be described in subsequent documents.

5.1. Positive Answers

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 returns that answer in the body of a 200 response.

5.2. Redirects

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, or a 302, 303 or 307 response code to indicate a non-permanent redirection, 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. Servers are under no obligation to return a URL conformant to [I-D.ietf-weirds-rdap-query].

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.

5.3. Negative Answers

If a server wishes to respond that it has an empty result set, it returns a 404 response code. Optionally, it MAY include additional information regarding the negative answer in the HTTP entity body.

If a server wishes to inform the client that information about the query is available, but cannot include the information in the response to the client for policy reasons, the server MUST respond with an appropriate response code out of HTTP's 4xx range. Clients MAY retry the query based on the respective response code.

5.4. Malformed Queries

If a server receives a query which it cannot interpret as an RDAP query, it returns a 400 response code. Optionally, it MAY include additional information regarding this negative answer in the HTTP entity body.

5.5. Rate Limits

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.

Note that this is not a defense against denial-of-service attacks, since a malicious client could ignore the code and continue to send queries at a high rate. A server might use another response code if it did not wish to reveal to a client that rate limiting is the reason for the denial of a reply.

5.6. Cross-Origin Resource Sharing

When responding to queries, it is RECOMMENDED that servers use the Access-Control-Allow-Origin header field, as specified by [W3C.CR-cors-20130129]. As the use of RDAP is for public resources, a value of "*" is suitable for most cases.

This header (often called the CORS header), helps in-browser web applications by lifting the "same-origin" restriction.

6. Extensibility

For extensibility purposes, this document defines an IANA registry for prefixes used in JSON [RFC4627] data serialization and URI path segments (see Section 8).

Prefixes and identifiers SHOULD only consist of the alphabetic ASCII characters A through Z in both uppercase and lowercase, the numerical digits 0 through 9, underscore characters, and SHOULD NOT begin with an underscore character, numerical digit or the characters "xml". The following describes the production of JSON names in ABNF [RFC5234].

ABNF for JSON names

```
  name = ALPHA *( ALPHA / DIGIT / "_" )
```

Figure 1

This restriction is a union of the Ruby programming language identifier syntax and the XML element name syntax and has two purposes. First, client implementers using modern programming

languages such as Ruby or Java can use libraries that automatically promote JSON names to first order object attributes or members. Second, a clean mapping between JSON and XML is easy to accomplish using these rules.

7. Security Considerations

This document does not pose strong security requirements to the RDAP protocol. However, it does not restrict against the use of security mechanisms offered by the HTTP protocol.

This document made recommendations for server implementations against denial-of-service (Section 5.5) and interoperability with existing security mechanism in HTTP clients (Section 5.6).

Additional security considerations to the RDAP protocol are covered in [I-D.ietf-weirds-rdap-sec].

8. IANA Considerations

8.1. RDAP Extensions Registry

This specification proposes an IANA registry for RDAP extensions. The purpose of this registry is to ensure uniqueness of extension identifiers. The extension identifier is used as a prefix in JSON names and as a prefix of path segments in RDAP URLs.

The production rule for these identifiers is specified in Section 6.

In accordance with RFC5226, the IANA policy for assigning new values shall be Specification Required: values and their meanings must be documented in an RFC or in some other permanent and readily available reference, in sufficient detail that interoperability between independent implementations is possible.

The following is a preliminary template for an RDAP extension registration:

Extension identifier: the identifier of the extension

Registry operator: the name of the registry operator

Published specification: RFC number, bibliographical reference or URL to a permanent and readily available specification

Person & email address to contact for further information: The names and email addresses of individuals for contact regarding this registry entry

Intended usage: brief reasons for this registry entry

The following is an example of a registration in the RDAP extension registry:

Extension identifier: lunarNic

Registry operator: The Registry of the Moon, LLC

Published specification: http://www.example/moon_apis/rdap

Person & email address to contact for further information:
Professor Bernardo de la Paz <berny@moon.example>

Intended usage: COMMON

9. Internationalization Considerations

9.1. URIs and IRIs

Clients can use IRIs [RFC3987] for internal use as they see fit, but MUST transform them to URIs [RFC3986] for interaction with RDAP servers. RDAP servers MUST use URIs in all responses, and again clients can transform these URIs to IRIs for internal use as they see fit.

9.2. Language Identifiers in Queries and Responses

Under most scenarios, clients requesting data will not signal that the data be returned in a particular language or script. On the other hand, when servers return data and have knowledge that the data is in a language or script, the data SHOULD be annotated with language identifiers whenever they are available, thus allowing clients to process and display the data accordingly.

The mechanism for including a language identifier in a response will be defined in subsequent documents describing specific response formats.

9.3. Language Identifiers in HTTP Headers

Given the description of the use of language identifiers in Section 9.2, unless otherwise specified servers SHOULD ignore the HTTP [RFC7231] Accept-Language header field when formulating HTTP entity responses, so that clients do not conflate the Accept-Language header with the 'lang' values in the entity body.

However, servers MAY return language identifiers in the Content-Language header field so as to inform clients of the intended language of HTTP layer messages.

10. Contributing Authors and Acknowledgements

John Levine provided text to tighten up the Accept header field usage and the text for the section on 429 responses.

Marc Blanchet provided some clarifying text regarding the use of URLs with redirects, as well as very useful feedback during WGLC.

Normative language reviews were provided by Murray S. Kucherawy and Andrew Sullivan.

Jean-Phillipe Dionne provided text for the Security Considerations section.

11. References

11.1. Normative References

[I-D.ietf-weirds-json-response]
Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", draft-ietf-weirds-json-response-08 (work in progress), August 2014.

[I-D.ietf-weirds-rdap-query]
Newton, A. and S. Hollenbeck, "Registration Data Access Protocol Query Format", draft-ietf-weirds-rdap-query-11 (work in progress), July 2014.

[I-D.ietf-weirds-rdap-sec]
Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol", draft-ietf-weirds-rdap-sec-07 (work in progress), August 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

[RFC6585] Nottingham, M. and R. Fielding, "Additional HTTP Status Codes", RFC 6585, April 2012.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[W3C.CR-cors-20130129]
Kesteren, A., "Cross-Origin Resource Sharing", World Wide Web Consortium Candidate Recommendation CR-cors-20130129, January 2013, <http://www.w3.org/TR/2013/CR-cors-20130129>.

11.2. Informative References

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

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

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

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

[SAC-051] Piscitello, D., Ed., "SSAC Report on Domain Name WHOIS Terminology and Structure", September 2011.

Appendix A. Protocol Example

To demonstrate typical behaviour of an RDAP client and server, the following is an example of an exchange, including a redirect. The data in the response has been elided for brevity, as the data format is not described in this document. The media type used here is described in [I-D.ietf-weirds-json-response].

An example of an RDAP client and server exchange:

```
Client:
    <TCP connect to rdap.example.com port 80>
    GET /ip/203.0.113.0/24 HTTP/1.1
    Host: rdap.example.com
    Accept: application/rdap+json

rdap.example.com:
    HTTP/1.1 301 Moved Permanently
    Location: http://rdap-ip.example.com/ip/203.0.113.0/24
    Content-Length: 0
    Content-Type: application/rdap+json
    <TCP disconnect>

Client:
    <TCP connect to rdap-ip.example.com port 80>
    GET /ip/203.0.113.0/24 HTTP/1.1
    Host:  rdap-ip.example.com
    Accept: application/rdap+json

rdap-ip.example.com:
    HTTP/1.1 200 OK
    Content-Type: application/rdap+json
    Content-Length: 9001

    { ... }
    <TCP disconnect>
```

Appendix B. Cache Busting

Some HTTP [RFC7230] cache infrastructure does not adhere to caching standards adequately, and could cache responses longer than is intended by the server. To overcome these issues, clients can use an adhoc and improbably used query parameter with a random value of their choosing. As Section 4.3 instructs servers to ignore unknown parameters, this is unlikely to have any known side effects.

An example of using an unknown query parameter to bust caches:

```
http://example.com/ip/192.0.2.0?__fuhgetaboutit=xyz123
```

Use of an unknown parameter to overcome misbehaving caches is not part of any specification and is offered here for informational purposes.

Appendix C. Changelog

RFC Editor: Please remove this section.

Initial WG -00: Updated to working group document 2012-September-20

-01

* Updated for the sections moved to the JSON responses draft.

* Simplified media type, removed "level" parameter.

* Updated 2119 language and added boilerplate.

* In section 1, noted that redirects can go to redirect servers as well.

* Added Section 9.2 and Section 9.3.

-02

* Added a section on 429 response codes.

* Changed Accept header language in section 4.1

* Removed reference to the now dead requirements draft.

* Added contributing authors and acknowledgements section.

* Added some clarifying text regarding complete URLs in the redirect section.

* Changed media type to application/rdap+json

* Added media type registration

-03

* Removed forward reference to draft-ietf-weirds-json-response.

* Added reference and recommended usage of CORS

-04

- Revised introduction and abstract.

- Added negative responses other than 404.

- Added security considerations.

- Added and corrected references: CORS, RFC3912, RFC3987, RFC5890.

- Expanded on first use several acronyms.

- Updated 2119 language.

-05

- Update the media type registration.

- Further explained the SHOULD in section 5.

- Split the references into normative and informative.

- Other minor fixes.

-06

- Rewritten the third paragraph in Section 3 to avoid contradictions

- Simplified the wording in Seciton 5.1.

- Removed some RFC 2119 words in Section 5.2, 5.3, 5.4 and 5.5.

- Corrected RFC 6839 as an informative reference.

- Replaced MAYs with cans in Seciton 9.1.

- Replaced MAY with can in Appendix B.

- Added a note in in Appendix C for the RFC Editor to remove this section.

-07

- Dropped reference to MUST with application/rdap+json

- Dropped IANA registration of application/rdap+json

-08

- Keep alive version.

-09

- Changed status lines in example to include http version number.

- Removed charset from media types in examples.

- Changed wording of 404 negative response to specifically say "empty result set".

- Changed references to HTTP.

-10

- Corrected references to HTTP.

- Added a reference to draft-ietf-weirds-json-response (discuss item from Barry Leiba)

- Added a reference to draft-ietf-weirds-rdap-query (discuss item from Barry Leiba)

- Noted that redirect URLs do not have to conform to draft-ietf-weirds-rdap-query (comment by Richard Barnes)

- Noted that CORS header is most likely to be "*" (comment by Richard Barnes)

- Added reference to draft-ietf-weirds-rdap-sec (comment by Richard Barnes)

- Added a sentence to the abstract explaining the purpose of RDAP (comment by Stephen Farrell)

- Added further references to draft-ietf-weirds-rdap-query and draft-ietf-weirds-json-response (comment by Stephen Farrell)

- Added comment regarding the use of the CORS header (comment by Stephen Farrell)

- Explanded SSAC (comment by Sean Turner)

- Added text about HEAD and GET.

Authors' Addresses

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

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

Byron J. Ellacott
Asia Pacific Network Information Center
6 Cordelia Street
South Brisbane QLD 4101
Australia

Email: bje@apnic.net
URI: http://www.apnic.net

Ning Kong
China Internet Network Information Center
4 South 4th Street, Zhongguancun, Haidian District
Beijing 100190
China

Phone: +86 10 5881 3147
Email: nkong@cnnic.cn