Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 8, 2013

A. Newton
ARIN
B. Ellacott
APNIC
N. Kong
CNNIC

December 5, 2012

Using the Registration Data Access Protocol (RDAP) with HTTP
draft-ietf-weirds-using-http-01

Abstract

This document describes the usage of the Registration Data Access Protocol (RDAP) using HTTP.

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 June 8, 2013.

Copyright Notice

Table of Contents

1. Introduction

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

In designing these common usage patterns, this draft endeavours to satisfy requirements for a Registration Data Access Protocol (RDAP) that is documented in [draft-kucherawy-weirds-requirements]. This draft also introduces an additional design consideration to define a simple use of HTTP. Where complexity may reside, it is the goal of this specification 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.

This is the basic usage pattern for this protocol:

1. A client issues an HTTP query using GET. As an example, a query for the network registration 192.0.2.0 might be http://example.com/ip/192.0.2.0.

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.

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 containing an HTTP URL 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 should return a 404 response.

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 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 Doman Name Registries (DNRs), DNRD-AP. This document and others from the IETF WEIRDS working group describe a single protocol, RDAP, for access to the data of both DNRs and Regional Internet Registries (RIRs). RIRs are also often referred to as number resource registries and are responsible for the registration of IP address networks and autonomous system numbers.

3. Design Intents

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

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 known query/respone model used by HTTP [RFC2616]. Should a result contain more than one result, some of which are better served by other servers, the redirection model becomes much more complicated.

Second, multiple response formats are supported by this protocol. At present the IETF WEIRDS working group is defining only a JSON [RFC4627] response format, but server operators may use other data formats when those formats are requested.

Third, HTTP offers a number of transport protocol 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. Accept Header

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

```
Accept: application/rdap
```

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].

```
Content-Type: application/rdap
```

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".

4.2. Query Parameters

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

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. 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 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.

5.2. Redirects

If a server wishes to inform a client that the answer to a given query can be found elsewhere, it SHOULD return either a 301 or a 307 response code and an HTTP URL in the Location: header. The client is expected to issue a subsequent 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.

A server SHOULD use a 301 response to inform the client of a permanent move and a 307 response otherwise. 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).

5.3. Negative Answers

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.

5.4. Malformed Queries

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.

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 7).

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 may 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. IANA Considerations

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 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

8. Internationalization Considerations

8.1. URIs and IRIs

Clients MAY use IRIs 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 clients MAY transform these URIs to IRIs.

8.2. Language Identifiers in Queries and Responses

Depending on the data format of the response, servers MAY include data in character sets other than ASCII and languages other than English (the data format will most likely be in Unicode and almost certainly languages other than English will be encountered). 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 thus allowing clients to process and display the data accordingly.

A language identifier in the response is specified in section 5.3 of [draft-ietf-weirds-json-response]. It is used to indicate the language/script of the response data. It is possible that registration data is stored in several different languages and returned in a single response. Data portion of different language types SHOULD be tagged with its corresponding identifier if known.

8.3. Language Identifiers in HTTP Headers

Given the description of the use of language identifiers in Section 8.2, unless otherwise specified servers SHOULD ignore the HTTP [RFC2616] Accept-Language header when formulating responses.

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

9. Normative References

[draft-kucherawy-weirds-requirements]
Kucherawy, M., "Requirements For Internet Registry Services", Work in progress: Internet Drafts draft-kucherawy-weirds-requirements-04.txt, April 2011.

[draft-ietf-weirds-json-response]
Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", Work in progress: Internet Drafts draft-ietf-weirds-json-response-01.txt, December 2012.

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

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

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

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

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

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

Appendix A. Cache Busting

To overcome issues with misbehaving HTTP [RFC2616] cache infrastructure, clients MAY use an adhoc and improbably used query parameter with a random value of their choosing. As Section 4.2 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 B. Changelog

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 8.2 and Section 8.3.

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