Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2013

A. Newton
ARIN
K. Ranjbar
RIPE NCC
A. Servin
LACNIC
B. Ellacott
APNIC
S. Hollenbeck
Verisign
S. Sheng
F. Arias
ICANN
N. Kong
CNNIC
F. Obispo
ISC

July 12, 2012

Using HTTP for RESTful Whois Services by Internet Registries
draft-designteam-weirds-using-http-01

Abstract

This document describes the use of HTTP in Whois services using RESTful web methodologies.

Status of this Memo

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 January 13, 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 Registration Data Access Protocols that are 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 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 192.168.0.0 might be http://example.com/ip/192.168.0.0.

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 Redirect header containing an HTTP URL pointing to 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

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 (RD-AP). At present, there are two known types of RD-AP, a Domain Name Registration Data Access Protocol (DNRD-AP) and a Number Resource Registration Data Access Protocol (NRRD-AP). Both the DNRD-AP and NRRD-AP are to be built upon this base behavior, the RD-AP.

Note that other types of RD-AP may exist in the future.

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. This document outlines the base usage of JSON and XML, but server operators may support other formats as they desire if appropriate.

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, and SHOULD use the Accept header parameter "level" to indicate the version of the format acceptable [RFC2616].

```
Accept: applicaiton/weirds_blah+json;level=0
```

Figure 1

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]. Servers SHOULD affix a media type parameter of "level" appropriate to the version of the format being sent.

```
Content-Type: application/weirds_blah+json;level=0
```

Figure 2

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 and corresponding level (e.g. "application/weirds+json;level=0"). In other words, a client may use "application/json" to express that it desires JSON or "application/weirds_blah+json" to express that it desires WEIRDS BLAH in JSON. The server MUST respond with "application/weirds_blah+json;level=0".

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 Redirect 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 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 lack of information as defined by Section 8.

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 about why it does not understand the query as defined by Section 8.

6. Use of JSON

6.1. Signaling

Clients may signal their desire for JSON using the "application/json" media type or a more application specific JSON media type.

6.2. Naming

Clients processing JSON [RFC4627] responses SHOULD ignore values associated with unrecognized names. Servers MAY insert values signified by names into the JSON responses which are not specified in this document. Insertion of unspecified values into JSON responses SHOULD have names prefixed with a short identifier followed by an underscore followed by a meaningful name.

For example, a JSON object may have "handle" and "remarks" formally documented in a specification. Clients adhering to that specification will have appropriate knowledge of the meaning of "handle" and "remarks".

Consider the following JSON response with JSON names.

```
{
  "handle" : "ABC123",
  "remarks" : [
    "she sells seas shells",
    "down by the seashore"
  ]
}
```

Figure 3

If The Registry of the Moon desires to express information not found in the specification, it might select "lunarNic" as its identifying prefix and insert, as an example, the name "lunarNic_beforeOneSmallStep" to signify registrations occuring before the first moon landing and the name "lunarNic_harshMistressNotes" containing other descriptive text.

Consider the following JSON response with JSON names, some of which should be ignored by clients without knowledge of their meaning.

```
{
  "handle" : "ABC123",
  "lunarNic_beforeOneSmallStep" : "TRUE THAT!",
  "remarks" : [
    "she sells seas shells",
    "down by the seashore"
  ],
  "lunarNic_harshMistressNotes" : [
    "In space,",
    "nobody can hear you scream."
  ]
}
```

Figure 4

Insertion of unrecognized names ignored by clients may also be used for future revisions to specifications and specifications deriving extensions from a base specification.

JSON names 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 produciton of JSON names in ABNF [RFC5234].

ABNF for JSON names

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

Figure 5

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 (e.g. using the example above, the values may be referenced as network.handle or network.lunarNic_beforeOneSmallStep). Second, a clean mapping between JSON and XML is easy to accomplish using the JSON representation.

Clients processing JSON responses MUST be prepared for values specified in the registry response documents to be absent from a

response as no JSON value listed is required to appear in the response. In other words, servers MAY remove values as is needed by the policies of the server operator.

7. Use of XML

7.1. Signaling

Clients may signal their desire for XML using the "application/xml" media type or a more application specific XML media type.

7.2. Naming and Structure

Well-formed XML may be programmatically produced using the JSON encodings due to the JSON naming rules outlined in Section 6.2 and the following simple rules:

1. Where a JSON name is given, the corresponding XML element has the same name.

2. Where a JSON value is found, it is the content of the corresponding XML element.

3. Where a JSON value is an array, the XML element is to be repeated for each element of the array.

4. The root tag of the XML document is to be "response".

Consider the following JSON response.


```
{
  "startAddress" : "10.0.0.0",
  "endAddress" : "10.0.0.255",
  "remarks" : [
    "she sells seas shells",
    "down by the seashore"
  ],
  "uris" : [
    {
      "type" : "source",
      "uri" : "http://whois-rws.net/network/xxxx"
    },
    {
      "type" : "parent",
      "uri" : "http://whois-rws.net/network/yyyy"
    }
  ]
}
```


Figure 6

The corresponding XML would look like this:


```
<response>
    <startAddress>10.0.0.0</startAddress>
    <endAddress>10.0.0.255</endAddress>
    <remarks>She sells sea shells</remarks>
    <remarks>down by the seashore</remarks>
    <uris>
        <type>source</type>
        <uri>http://whois-rws.net/network/xxxx</uri>
    </uris>
    <uris>
        <type>parent</type>
        <uri>http://whois-rws.net/network/yyyy</uri>
    </uris>
</response>
```


JSON values converted to XML element content MUST be properly escaped. XML offers various means for escaping data, but such escaping MUST account for the '<', '>', and '&' characters and MUST redact all C0 control characters except tab, carriage return, and new-line. (Redaction of disallowed control characters is a protocol requirement, though in practice most Internet registries do not allow this data in their data stores and therefore do not need to account for this rule.)

The rules for clients processing XML responses are the same as those with JSON: clients SHOULD ignore unrecognized XML elements, and servers MAY insert XML elements with tag names according to the naming rules in Section 6.2. And as with JSON, clients MUST be prepared for XML elements specified in the registry response documents to be absent from a response as no XML element listed is required to appear in the response.

8. Common Error Response Body

As specified in Section 5, some non-answer responses may return entity bodies with information that could be more descriptive.

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".

This is an example of the JSON version of the common response body.


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


Figure 7

This is an example of the XML version of the common response body.


```
<response>
    <errorCode>418</errorCode>
    <title>Your beverage choice is not available</title>
    <description>I know coffee has more ummppphhh.</description>
    <description>But I cannot provide.</description>
</response>
```


Figure 8

The media type for the JSON structure is "application/rdap_error+json" and the media type for the XML document is "application/rdap_error+xml". Conformance to this specification is considered to be level 0 for both media types.

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.

9. Common Data Structures

This section defines two common data structures to be used by DNRD-AP, NRRD-AP, and other RD-AP protocols. As such, the names identifying these data structures are not to be redefined by any registry specific RD-AP specifications. Each of these datatypes MAY appear within any other data object of a response, but the intended purpose is that they will be mostly used in the top-most data object of a response.

The first data structure is named "rdapConformance" and is simply an array of strings, each providing a hint as to the specifications used in the construction of the response.

An example rdapConformance data structure.

```
"rdapConformance" : [
  "nrrdap_level_0"
]
```

Figure 9

The second data structure is named "notices" and 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 a "uri" string holding a URI referencing a service that may provide additional information about the notice.

An exmaple of the notices data structure.

```
"notices" : [
  "notice" : {
    "title" : "Terms of Use",
    "description" : [
      "This service is subject to The Registry of the Moons",
      "terms of service."
    ],
    "uri" : "http://example.com/our-terms-of-use"
  }
]
```

Figure 10

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


```
{
  "rdapConformance" : [
    "nrrdap_level_0"
  ]
  "notices" : [
    "notice" : {
      "title" : "Content Redacted",
      "description" : [
        "Without full authorization, content has been redacted.",
        "Sorry, dude!"
      ],
      "uri" : "http://example.com/our-redaction-policies"
    }
  ]
  "startAddress" : "10.0.0.0",
  "endAddress" : "10.0.0.255",
  "remarks" : [
    "she sells seas shells",
    "down by the seashore"
  ],
  "uris" : [
    {
      "type" : "source",
      "uri" : "http://whois-rws.net/network/xxxx"
    },
    {
      "type" : "parent",
      "uri" : "http://whois-rws.net/network/yyyy"
    }
  ]
}
```


Figure 11

10. Common Datatypes

This section describes common data types found in Internet registries, the purpose being a common and normalized list of normative references to other specifications to be used by multiple RD-AP applications. Unless otherwise stated by the response specification of an Internet registry using this specification as a basis, the data types can assume to be as follows:

1. IPv4 addresses - [RFC0791]

2. IPv6 addresses - [RFC5952]

3. country code - [ISO.3166.1988]

4. domain name - [RFC4343]

5. email address - [RFC5322]

6. date and time strings - [RFC3339]

11. IANA Considerations

11.1. Registration of RDAP Error Media Type for JSON

This specification registers the "application/rdap_error+json" media type.

Type name: application

Subtype name: rdap_error+json

Required parameters: n/a

Optional parameters: level

Encoding considerations: n/a

Security considerations: n/a

Interoperability considerations: n/a

Published specification: [[this document]]

Applications that use this media type: RESTful Whois applications

Additional information: n/a

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

Intended usage: COMMON

Restrictions on usage: none

Author: Andy Newton

Change controller: IETF

11.2. Registration of RDAP Error Media Type for XML

This specification registers the "application/rdap_error+xml" media type.

Type name: application

Subtype name: rdap_error+xml

Required parameters: n/a

Optional parameters: level

Encoding considerations: n/a

Security considerations: n/a

Interoperability considerations: n/a

Published specification: [[this document]]

Applications that use this media type: RESTful Whois applications

Additional information: n/a

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

Intended usage: COMMON

Restrictions on usage: none

Author: Andy Newton

Change controller: IETF

12. Internationalization Considerations

12.1. URIs vs IRIs

Clients MAY use IRIs as they see fit, but MUST transform them to URIs [RFC3986] for interaction with RD-AP servers. RD-AP servers MUST use URIs in all responses, and clients MAY transform these URIs to IRIs.

12.2. Character Encoding

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

13. 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.

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

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

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

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

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

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

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

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

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

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

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

[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 the 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. Areas of Improvement

Things that need to be done to this draft.

1. authentication what?

2. clean up must should, ref 2119?

3. better language on data formats... it was just a rough start

4. Security considerations?

5. Is there a privacy considerations things we have to do now?

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

Kaveh Ranjbar
RIPE Network Coordination Centre
Singel 258
Amsterdam 1016AB
NL

Email: kranjbar@ripe.net
URI: http://www.ripe.net

Arturo L. Servin
Latin American and Caribbean Internet Address Registry
Rambla Republica de Mexico 6125
Montevideo 11300
UY

Email: aservin@lacnic.net
URI: http://www.lacnic.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

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

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

Steve Sheng
Internet Corporation for Assigned Names and Numbers
4676 Admiralty Way, Suite 330
Marina del Rey, CA 90292
United States of America

Phone: +1.310.823.9358
Email: steve.sheng@icann.org

Francisco Arias
Internet Corporation for Assigned Names and Numbers
4676 Admiralty Way, Suite 330
Marina del Rey, CA 90292
United States of America

Phone: +1.310.823.9358
Email: francisco.arias@icann.org

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

Francisco Obispo
Internet Systems Consortium
950 Charter St
Redwood City, CA 94063
United States of America

Phone: +1.650.423.1374
Email: fobispo@isc.org