[Note that this file is a concatenation of more than one RFC.]

Internet Engineering Task Force (IETF)
Request for Comments: 7480
Category: Standards Track
ISSN: 2070-1721

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

March 2015

HTTP Usage in the Registration Data Access Protocol (RDAP)

Abstract

This document is one of a collection that together describes 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. The purpose of this document is to clarify the use of standard HTTP mechanisms for this application.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7480.

Table of Contents

1. Introduction

This document describes the usage of the Hypertext Transfer Protocol (HTTP) [RFC7230] for the Registration Data Access Protocol (RDAP). 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 practices informed by the Representational State Transfer (REST) [REST] architectural style. By giving the various directory services common behavior, a single client is better able to retrieve data from directory services adhering to this behavior.

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 organization 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 determines an appropriate server to query along with the appropriate base Uniform Resource Locator (URL) to use in such queries. [RFC7484] describes one method to determine the server and the base URL. See Appendix C for more information.

2. A client issues an HTTP (or HTTPS) query using GET [RFC7231]. As an example, a query URL for the network registration 192.0.2.0 might be

   http://example.com/rdap/ip/192.0.2.0

   [RFC7482] details the various queries used in RDAP.

3. 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. [RFC7483] details a response in JavaScript Object Notation (JSON).

4. 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 pointing to the information or another server known to have knowledge of the location of the information. The client is expected to requery using that HTTP URL.

5. 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.

6. 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 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 [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 an 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 [RFC7482] and [RFC7483], while this document describes how RDAP clients and servers use HTTP to exchange queries and responses. [RFC7481] describes security considerations for RDAP.

3. Design Intents

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

First, each query is meant to require only one path of execution to obtain an answer. A response may contain an answer, no answer, or a redirect, and clients are not expected to fork multiple paths of execution to make a query.

Second, the semantics of the request/response allow for future and/or non-standard response formats. In this document, only a JSON [RFC7159] response media type is noted, with the response contents to be described separately (see [RFC7483]). 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, as well as widespread programmer understanding of client behaviors for web services styled after REST [REST], reducing the cost to deploy Registration Data Directory Services and clients. This protocol is forward compatible with HTTP 2.0.

4. Queries

4.1. HTTP Methods

Clients use the GET method to retrieve a response body and use the HEAD method to determine existence of data on the server. Clients SHOULD use either the HTTP GET or HEAD methods (see [RFC7231]). Servers are under no obligation to support other HTTP methods; therefore, clients using other methods will likely not interoperate properly.

Clients and servers MUST support HTTPS to support security services.

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, this section defines the minimal set of response codes in common use by servers that a client will need to understand. While some clients may be constructed with simple tooling that does not account for all of these response codes, a more robust client accounting for these codes will likely provide a better user experience. 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 (OK) response (see [RFC7231]).

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 (Moved Permanently) response code to indicate a permanent move or a 302 (Found), 303 (See Other), or 307 (Temporary Redirect) response code to indicate a non-permanent redirection, and it includes an HTTP(S) URL in the Location header field (see [RFC7231]). 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 [RFC7482].

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 uses

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 (that is, no data appropriately satisfying the query), it returns a 404 (Not Found) 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. A client MAY retry the query if that is appropriate for the respective response code.

5.4. Malformed Queries

If a server receives a query that it cannot interpret as an RDAP query, it returns a 400 (Bad Request) 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 (Too Many Requests) 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. Servers may place stricter limits upon clients that do not honor the Retry-After header. Optionally, the server MAY include additional information regarding the rate limiting in the HTTP entity body.

Note that this is not a defense against denial-of-service (DoS) 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 (CORS)

When responding to queries, it is RECOMMENDED that servers use the Access-Control-Allow-Origin header field, as specified by [W3C.REC-cors-20140116]. A value of "*" is suitable when RDAP is used for public resources.

This header (often called the CORS header) helps in-browser web applications by lifting the "same-origin" restriction (i.e., a browser may load RDAP client code from one web server but query others for RDAP data).

By default, browsers do not send cookies when cross origin requests are allowed. Setting the Access-Control-Allow-Credentials header field to "true" will send cookies. Use of the Access-Control-Allow-Credentials header field is NOT RECOMMENDED.

6. Extensibility

For extensibility purposes, this document defines an IANA registry for prefixes used in JSON [RFC7159] data serialization and URI path segments (see Section 8).

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

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

Figure 1: ABNF for JSON Names

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. It does require that RDAP clients and servers MUST support HTTPS.

This document makes recommendations for server implementations against DoS (Section 5.5) and interoperability with existing security mechanisms in HTTP clients (Section 5.6).

Additional security considerations to the RDAP protocol are covered in [RFC7481].

8. IANA Considerations

8.1. RDAP Extensions Registry

IANA has created a new category in the protocol registries labeled "Registration Data Access Protocol (RDAP)", and within that category, has established a URL-referenceable, stand-alone registry labeled "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 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 to contact regarding this registry entry

Intended usage: brief reasons for this registry entry (as defined by [RFC5226]).

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 Internationalized Resource Identifiers (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.

[RFC7483] provides such a mechanism.

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. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005, <http://www.rfc-editor.org/info/rfc3987>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC6585] Nottingham, M. and R. Fielding, "Additional HTTP Status Codes", RFC 6585, April 2012, <http://www.rfc-editor.org/info/rfc6585>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", RFC 7481, February 2015, <http://www.rfc-editor.org/info/rfc7481>.

[RFC7482] Newton, A. and S. Hollenbeck, "Registration Data Access Protocol (RDAP) Query Format", RFC 7482, February 2015, <http://www.rfc-editor.org/info/rfc7482>.

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

[RFC7484] Blanchet, M., "Finding the Authoritative Registration Data (RDAP) Service", RFC 7484, February 2015, <http://www.rfc-editor.org/info/rfc7484>.

[W3C.REC-cors-20140116] Kesteren, A., "Cross-Origin Resource Sharing", W3C Recommendation, REC-cors-20140116, January 2014, <http://www.w3.org/TR/2014/REC-cors-20140116/>.

10.2. Informative References

[REST] Fielding, R. and R. Taylor, "Principled Design of the Modern Web Architecture", ACM Transactions on Internet Technology, Vol. 2, No. 2, May 2002.

[RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, September 2004, <http://www.rfc-editor.org/info/rfc3912>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010, <http://www.rfc-editor.org/info/rfc5890>.

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

[SAC-051] Piscitello, D., Ed., "SSAC Report on Domain Name WHOIS Terminology and Structure", A report from the ICANN Security and Stability Advisory Committee (SSAC), September 2011.

[lacnic-joint-whois] LACNIC, "Joint Whois", December 2005, <ftp://anonymous@ftp.registro.br/pub/gter/gter20/02-jwhois-lacnic.pdf>.

Appendix A. Protocol Example

To demonstrate typical behavior 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 [RFC7483].

An example of an RDAP client and server exchange:

```
  Client:
      <TCP connect to rdap.example.com port 80>
      GET /rdap/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/rdap/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 /rdap/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 infrastructures do 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 ad hoc and improbably used query parameter with a random value of their choosing. As Section 4.3 instructs servers to ignore unknown parameters, this is compatible with the RDAP definition.

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. Bootstrapping and Redirection

The traditional deployment model of WHOIS [RFC3912] does not provide a mechanism for determining the authoritative source for information.

Some approaches have been implemented in the past, most notably the Joint WHOIS [lacnic-joint-whois] initiative. However, among other shortcomings, Joint WHOIS is implemented using proxies and server-side referrals.

These issues are solved in RDAP using HTTP redirects and bootstrapping. Bootstrapping is discussed in [RFC7484]. In constrained environments, the processes outlined in [RFC7484] may not be viable, and there may be the need for servers acting as a "redirector".

Redirector servers issue HTTP redirects to clients using a redirection table informed by [RFC7484]. Figure 2 diagrams a client using a redirector for bootstrapping.

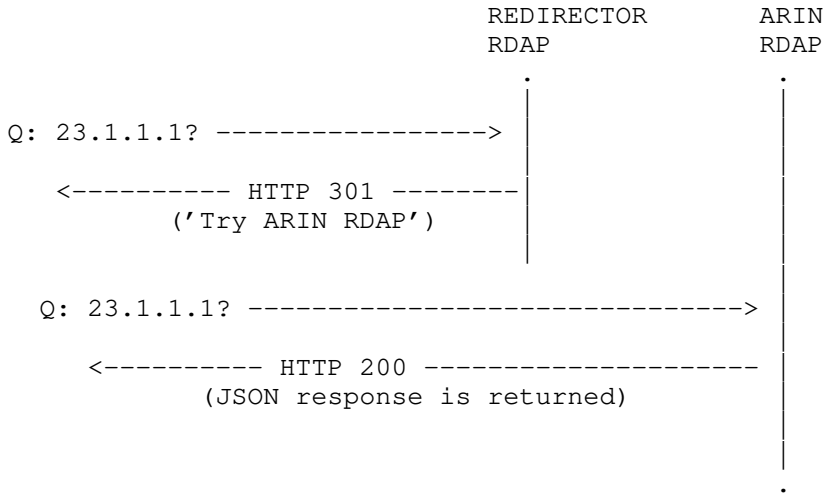

Figure 2: Querying RDAP Data for 23.1.1.1

In some cases, particularly sub-delegations made between Regional Internet Registries (RIRs) known as "ERX space" and transfers of networks, multiple HTTP redirects will be issued. Figure 3 shows such a scenario.

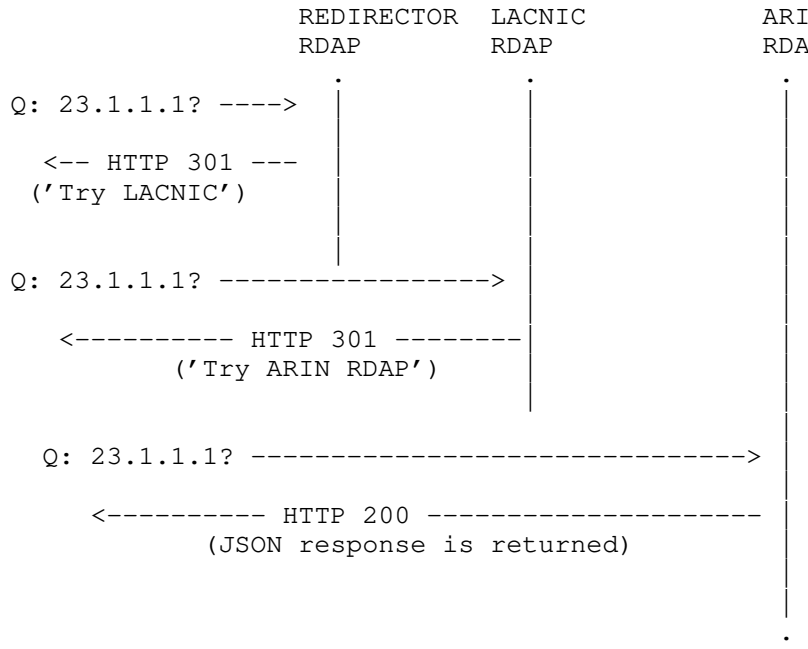

Figure 3: Querying RDAP Data for Data That Has Been Transferred

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 Working Group Last Call (WGLC).

Normative language reviews were provided by Murray S. Kucherawy, Andrew Sullivan, Tom Harrison, Ed Lewis, and Alexander Mayrhofer.

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

The concept of the redirector server informatively discussed in Appendix C was documented by Carlos M. Martinez and Gerardo Rada of

LACNIC and Linlin Zhou of CNNIC and subsequently incorporated into this document.

This document is the work product of the IETF's WEIRDS working group, of which Olaf Kolkman and Murray Kucherawy were chairs.

Authors' Addresses

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

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

Byron J. Ellacott
Asia Pacific Network Information Centre
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

Internet Engineering Task Force (IETF)
Request for Comments: 7481
Category: Standards Track
ISSN: 2070-1721

S. Hollenbeck
Verisign Labs
N. Kong
CNNIC

March 2015

Security Services for the Registration Data Access Protocol (RDAP)

Abstract

The Registration Data Access Protocol (RDAP) provides "RESTful" web services to retrieve registration metadata from Domain Name and Regional Internet Registries. This document describes information security services, including access control, authentication, authorization, availability, data confidentiality, and data integrity for RDAP.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7481.

Table of Contents

1. Introduction

The Registration Data Access Protocol (RDAP) is specified in multiple documents, including "Registration Data Access Protocol (RDAP) Query Format" [RFC7482], "JSON Responses for the Registration Data Access Protocol (RDAP)" [RFC7483], and "HTTP Usage in the Registration Data Access Protocol (RDAP)" [RFC7480].

One goal of RDAP is to provide security services that do not exist in the WHOIS [RFC3912] protocol, including access control, authentication, authorization, availability, data confidentiality, and data integrity. This document describes how each of these services is achieved by RDAP using features that are available in other protocol layers. Additional or alternative mechanisms can be added in the future. Where applicable, informative references to requirements for a WHOIS replacement service [RFC3707] are noted.

2. Conventions Used in This Document

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 [RFC2119].

2.1. Acronyms and Abbreviations

DNR: Domain Name Registry

HTTP: Hypertext Transfer Protocol

JSON: JavaScript Object Notation

RDAP: Registration Data Access Protocol

RIR: Regional Internet Registry

TLS: Transport Layer Security

3. Information Security Services and RDAP

RDAP itself does not include native security services. Instead, RDAP relies on features that are available in other protocol layers to provide needed security services, including access control, authentication, authorization, availability, data confidentiality, and data integrity. A description of each of these security services can be found in "Internet Security Glossary, Version 2" [RFC4949]. No requirements have been identified for other security services.

3.1. Access Control

WHOIS does not include specific features to control access to registration information. As described in the following sections, RDAP includes features to identify, authenticate, and authorize clients, allowing server operators to control access to information based on a client's identity and associated authorizations. Information returned to a client can be clearly marked with a status value (see Section 10.2.2 of [RFC7483]) that identifies the access granted to the client.

3.2. Authentication

This section describes security authentication mechanisms and the need for authorization policies to include them. It describes requirements for the implementations of clients and servers but does not dictate the policies of server operators. For example, a server operator with no policy regarding differentiated or tiered access to data will have no authorization mechanisms and will have no need for any type of authentication. A server operator with policies on differentiated access will have to construct an authorization scheme and will need to follow the specified authentication requirements.

WHOIS does not provide features to identify and authenticate clients. As noted in Section 3.1.4.2 of "Cross Registry Internet Service Protocol (CRISP) Requirements" [RFC3707], there is utility in allowing server operators to offer "varying degrees of access depending on policy and need." Clients have to be identified and authenticated to provide that utility.

RDAP's authentication framework needs to accommodate anonymous access as well as verification of identities using a range of authentication methods and credential services. To that end, RDAP clients and servers MUST implement the authentication framework specified in "Hypertext Transfer Protocol (HTTP/1.1): Authentication" [RFC7235]. The "basic" scheme can be used to send a client's user name and password to a server in plaintext, base64-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, HTTP over TLS [RFC2818] MUST be used to protect the client's credentials from disclosure while in transit (see Section 3.5).

Servers MUST support either Basic or Digest authentication; they are not required to support both. Clients MUST support both to interoperate with servers that support one or the other. Servers may provide a login page that triggers HTTP authentication. Clients should continue sending the HTTP authentication header once they receive an initial 401 (Unauthorized) response from the HTTP server as long as the scheme portion of the URL doesn't change.

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.

RDAP does not impose any unique server authentication requirements. The server authentication provided by TLS fully addresses the needs of RDAP. In general, transports for RDAP must either provide a TLS-protected transport (e.g., HTTPS) or a mechanism that provides an equivalent level of server authentication.

Work on HTTP authentication methods continues. RDAP is designed to be agile enough to support additional methods as they are defined.

3.2.1. Federated Authentication

The traditional client-server authentication model requires clients to maintain distinct credentials for every RDAP server. This situation can become unwieldy as the number of RDAP servers increases. Federated authentication mechanisms allow clients to use one credential to access multiple RDAP servers and reduce client

credential management complexity. RDAP MAY include a federated authentication mechanism that permits a client to access multiple RDAP servers in the same federation with one credential.

Federated authentication mechanisms used by RDAP MUST be fully supported by HTTP. OAuth, OpenID, Security Assertion Markup Language (SAML), and mechanisms based on Certification Authority (CA) are all possible approaches to provide federated authentication. At the time of this document's publication, negotiation or advertisement of federated authentication services is still an undefined mechanism by the noted federated authentication protocols. Developing this mechanism is beyond the scope of this document.

The OAuth authorization framework [RFC6749] describes a method for users to access protected web resources without having to hand out their credentials. Instead, clients are issued access tokens by authorization servers with the permission of the resource owners. Using OAuth, multiple RDAP servers can form a federation, and the clients can access any server in the same federation by providing one credential registered in any server in that federation. The OAuth authorization framework is designed for use with HTTP and thus can be used with RDAP.

OpenID [OpenID] is a decentralized single sign-on authentication system that allows users to log in at multiple web sites with one ID instead of having to create multiple unique accounts. An end user can freely choose which OpenID provider to use and can preserve their Identifier if they switch OpenID providers.

Note that OAuth and OpenID do not consistently require data confidentiality services to protect interactions between providers and consumers. HTTP over TLS [RFC2818] can be used as needed to provide protection against man-in-the-middle attacks.

SAML 2.0 [SAML] is an XML-based protocol that can be used to implement web-based authentication and authorization services, including single sign on. It uses security tokens containing assertions to exchange information about an end user between an identity provider and a service provider.

The Transport Layer Security protocol describes the specification of a client certificate in Section 7.4.6 of [RFC5246]. Clients who possess and present a valid X.509 digital certificate, issued by a CA, could be identified and authenticated by a server who trusts the corresponding CA. A certificate authentication method can be used to achieve federated authentication in which multiple RDAP servers all trust the same CAs, and then any client with a certificate issued by a trusted CA can access any RDAP server in the federation. This

certificate-based mechanism is supported by HTTPS and can be used with RDAP.

3.3. Authorization

WHOIS does not provide services to grant different levels of access to clients based on a client's authenticated identity. As noted in Section 3.1.4.2 of "Cross Registry Internet Service Protocol (CRISP) Requirements" [RFC3707], there is utility in allowing server operators to offer "varying degrees of access depending on policy and need." Access control decisions can be made once a client's identity has been established and authenticated (see Section 3.2).

Server operators MAY offer varying degrees of access depending on policy and need in conjunction with the authentication methods described in Section 3.2. If such varying degrees of access are supported, an RDAP server MUST provide granular access controls (that is, per registration data object) in order to implement authorization policies. Some 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.

The type of access allowed by a server will most likely vary from one operator to the next. A description of the response privacy considerations associated with different levels of authorization can be found in Section 13 of [RFC7483].

3.4. 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 "Internet Denial-of-Service Considerations" [RFC4732] is advised.

An RDAP service MAY use an HTTP throttling mechanism to limit the number of queries that a single client can send in a given period of time. If used, the server SHOULD return an HTTP 429 (Too Many Requests) response code as described in "Additional HTTP Status Codes" [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.

3.5. Data Confidentiality

WHOIS does not provide the ability to protect data from inadvertent disclosure while in transit. RDAP uses HTTP over TLS [RFC2818] to provide that protection by encrypting all traffic sent on the connection between client and server. HTTP over TLS MUST be used to protect all client-server exchanges unless operational constraints make it impossible to meet this requirement. It is also possible to encrypt discrete objects (such as command path segments and JSON-encoded response objects) at one endpoint, send them to the other endpoint via an unprotected transport protocol, and decrypt the object on receipt. Encryption algorithms as described in "Internet Security Glossary, Version 2" [RFC4949] are commonly used to provide data confidentiality at the object level.

There are no current requirements for object-level data confidentiality using encryption. Support for this feature could be added to RDAP in the future.

As noted in Section 3.2, the HTTP "basic" authentication scheme can be used to authenticate a client. When this scheme is used, HTTP over TLS MUST be used to protect the client's credentials from disclosure while in transit. If the policy of the server operator requires encryption to protect client-server data exchanges (such as to protect non-public data that cannot be accessed without client identification and authentication), HTTP over TLS MUST be used to protect those exchanges.

A description of privacy threats that can be addressed with confidentiality services can be found in Section 4. Section 10.2.2 of [RFC7483] describes status values that can be used to describe operator actions used to protect response data from disclosure to unauthorized clients.

3.6. Data Integrity

WHOIS does not provide the ability to protect data from modification while in transit. Web services such as RDAP commonly use HTTP over TLS [RFC2818] to provide that protection by using a keyed Message Authentication Code (MAC) to detect modifications. It is also possible to sign discrete objects (such as command path segments and JSON-encoded response objects) at one endpoint, send them to the

other endpoint via a transport protocol, and validate the signature of the object on receipt. Digital signature algorithms as described in "Internet Security Glossary, Version 2" [RFC4949] are commonly used to provide data integrity at the object level.

There are no current requirements for object-level data integrity using digital signatures. Support for this feature could be added to RDAP in the future.

The most specific need for this service is to provide assurance that HTTP 30x redirection hints [RFC7231] and response elements returned from the server are not modified while in transit. If the policy of the server operator requires message integrity for client-server data exchanges, HTTP over TLS MUST be used to protect those exchanges.

4. Privacy Threats Associated with Registration Data

Registration data has historically included personal data about registrants. WHOIS services have historically made this information available to the public, creating a privacy risk by revealing the personal details of registrants. WHOIS services have not had the benefit of authentication or access control mechanisms to gate access to registration data. As a result of this, proxy and privacy services have arisen to shield the identities of registrants.

The standardization of RDAP does not change or impact the data that operators may require to be collected from registrants, but it provides support for a number of mechanisms that may be used to mitigate privacy threats to registrants should operators choose to use them.

RDAP includes mechanisms that can be used to authenticate clients, allowing servers to support tiered access based on local policy. This means that all registration data need no longer be public, and personal data or data that may be considered more sensitive can have its access restricted to specifically privileged clients.

RDAP data structures allow servers to indicate via status values when data returned to clients has been made private, redacted, obscured, or registered by a proxy. "Private" means that the data is not designated for public consumption. "Redacted" means that some registration data fields are not being made available. "Obscured" means that data has been altered for the purposes of not readily revealing the actual registration information. One option that operators have available to them to reduce privacy risks to registrants is to adopt policies that make use of these status values to restrict the registrant data shared with any or all clients

according to the sensitivity of the data, the privileges of the clients, or some other heuristics.

RDAP uses the jCard [RFC7095] standard format for entity representation. Operators may find that many of the jCard fields are irrelevant for registry operation purposes or that they have no reason to collect information from registrants that would correspond to certain fields. Operators wishing to reduce privacy risks for registrants may restrict which information they collect and/or which fields they populate in responses.

In addition to privacy risks to registrants, there are also potential privacy risks for those who query registration data. For example, the fact that a registry employee performs a particular query may reveal information about the employee's activities that he or she would have preferred to keep private. RDAP supports the use of HTTP over TLS to provide privacy protection for those querying registrant data as well as registrants, unless operational constraints make it impossible to meet this requirement.

5. Security Considerations

One of the goals of RDAP is to provide security services that do not exist in the WHOIS protocol. This document describes the security services provided by RDAP and associated protocol layers, including authentication, authorization, availability, data confidentiality, and data integrity. Non-repudiation services were also considered and ultimately rejected due to a lack of requirements. There are, however, currently deployed WHOIS servers that can return signed responses that provide non-repudiation with proof of origin. RDAP might need to be extended to provide this service in the future.

As an HTTP-based protocol, RDAP is susceptible to code injection attacks. Code injection refers to adding code into a computer system or program to alter the course of execution. There are many types of code injection, including SQL injection, dynamic variable or function injection, include-file injection, shell injection, and HTML-script injection, among others. Data confidentiality and integrity services provide a measure of defense against man-in-the-middle injection attacks, but vulnerabilities in both client- and server-side software make it possible for injection attacks to succeed. Consistently checking and validating server credentials can help detect man-in-the-middle attacks.

As noted in Section 3.2.1, digital certificates can be used to implement federated authentication. There is a risk of too promiscuous, or even rogue, CAs being included in the list of acceptable CAs that the TLS server sends the client as part of the

TLS client-authentication handshake and lending the appearance of trust to certificates signed by those CAs. Periodic monitoring of the list of CAs that RDAP servers trust for client authentication can help reduce this risk.

The Transport Layer Security protocol [RFC5246] includes a null cipher suite that does not encrypt data and thus does not provide data confidentiality. This option MUST NOT be used when data confidentiality services are needed. Additional considerations for secure use of TLS are described in [SECURE-TLS-DTLS].

Data integrity services are sometimes mistakenly associated with directory service operational policy requirements focused on data accuracy. "Accuracy" refers to the truthful association of data elements (such as names, addresses, and telephone numbers) in the context of a particular directory object (such as a domain name). Accuracy requirements are out of scope for this protocol.

Additional security considerations are described in the specifications for HTTP [RFC7231], HTTP Basic and Digest access authentication [RFC7235], HTTP over TLS [RFC2818], and additional HTTP status codes [RFC6585]. Security considerations for federated authentication systems can be found in the OAuth [RFC6749] and OpenID [OpenID] specifications.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[RFC6585] Nottingham, M. and R. Fielding, "Additional HTTP Status Codes", RFC 6585, April 2012, <http://www.rfc-editor.org/info/rfc6585>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, June 2014, <http://www.rfc-editor.org/info/rfc7235>.

[RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", RFC 7480, March 2015, <http://www.rfc-editor.org/info/rfc7480>.

[RFC7482] Newton, A. and S. Hollenbeck, "Registration Data Access Protocol (RDAP) Query Format", RFC 7482, March 2015, <http://www.rfc-editor.org/info/rfc7482>.

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

6.2. Informative References

[OpenID] OpenID Foundation, "OpenID Authentication 2.0 - Final", December 2007, <http://specs.openid.net/auth/2.0>.

[RFC3707] Newton, A., "Cross Registry Internet Service Protocol (CRISP) Requirements", RFC 3707, February 2004, <http://www.rfc-editor.org/info/rfc3707>.

[RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, September 2004, <http://www.rfc-editor.org/info/rfc3912>.

[RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, December 2006, <http://www.rfc-editor.org/info/rfc4732>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, August 2007, <http://www.rfc-editor.org/info/rfc4949>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012, <http://www.rfc-editor.org/info/rfc6749>.

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

[SAML] OASIS, "Security Assertion Markup Language (SAML) v2.0", March 2005, <https://www.oasis-open.org/standards#samlv2.0>.

[SECURE-TLS-DTLS] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of TLS and DTLS", Work in Progress, draft-ietf-uta-tls-bcp-09, February 2015.

Acknowledgements

The authors would like to acknowledge the following individuals for their contributions to this document: Richard Barnes, Marc Blanchet, Alissa Cooper, Ernie Dainow, Spencer Dawkins, Jean-Philippe Dionne, Byron Ellacott, Stephen Farrell, Tony Hansen, Peter Koch, Murray Kucherawy, Barry Leiba, Andrew Newton, and Linlin Zhou.

Authors' Addresses

Scott Hollenbeck
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
United States

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

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