Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: December 19, 2020

M. Loffredo
M. Martinelli
IIT-CNR/Registro.it

June 17, 2020

Registration Data Access Protocol (RDAP) Partial Response draft-ietf-regext-rdap-partial-response-12

Abstract

The Registration Data Access Protocol (RDAP) does not include capabilities to request partial responses. Servers will only return full responses that include all of the information that a client is authorized to receive. A partial response capability that limits the amount of information returned, especially in the case of search queries, could bring benefits to both clients and servers. This document describes an RDAP query extension that allows clients to specify their preference for obtaining a partial response.

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 https://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 December 19, 2020.

Copyright Notice

Copyright (c) 2020 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 (https://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

The use of partial responses in RESTful API [REST] design is very common. The rationale is quite simple: instead of returning objects in API responses with all data fields, only a subset of the fields in each result object is returned. The benefit is obvious: fewer data transferred over the network means less bandwidth usage, faster server responses, less CPU time spent both on the server and the client, and less memory usage on the client.

Several leading API providers [LINKEDIN] [FACEBOOK] [GOOGLE] implement partial response features by providing an optional query parameter through which clients identify the fields they wish to receive. Support for partial responses is also considered a leading principle by many best practice guidelines in REST API implementation [REST-API1] [REST-API2] in order to improve performance, save on bandwidth and possibly accelerate the overall interaction. In other contexts, for example in digital libraries and bibliographic catalogues, servers can respond according to different element sets

(i.e. "brief" to obtain a short response and "full" to obtain the complete response).

Currently, RDAP does not provide a client with any way to request a partial response. Servers can only provide the client with a full response [RFC7483]. Servers cannot limit the amount of information returned in a response based on a client's preferences, and this creates inefficiencies.

The protocol described in this specification extends RDAP search capabilities to enable partial responses through the provisioning of pre-defined sets of fields that clients can submit to an RDAP service by adding a new query parameter. The service is implemented using the Hypertext Transfer Protocol (HTTP) [RFC7230] and the conventions described in [RFC7480].

1.1. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. RDAP Path Segment Specification

The path segment defined in this section is an OPTIONAL extension of search path segments defined in [RFC7482]. This document defines an RDAP query parameter, "fieldSet", whose value is a string identifying a server-defined set of supported fields (Figure 1).

```
https://example.com/rdap/domains?name=example*.com&fieldSet=afieldset
```

Figure 1: Example of RDAP search query reporting the "fieldSet" parameter

This solution can be implemented by RDAP providers with less effort than field selection and is easily requested by clients. The considerations that have led to this solution are described in more detail in Appendix A.

2.1. Subsetting Metadata

According to most advanced principles in REST design, collectively known as HATEOAS (Hypermedia as the Engine of Application State) [HATEOAS], a client entering a REST application through an initial URI should use server-provided links to dynamically discover

available actions and access the resources it needs. In this way, the client is not required to have prior knowledge of the service and, consequently, to hard code the URIs of different resources. This allows the server to make URI changes as the API evolves without breaking clients. Definitively, a REST service should be as self-descriptive as possible.

Therefore, servers implementing the query parameter described in this specification SHOULD provide additional information in their responses about the available field sets. Such information is collected in a new data structure named "subsetting_metadata" containing the following properties:

- "currentFieldSet": "String" (REQUIRED) either the value of the "fieldSet" parameter as specified in the query string, or the field set applied by default;

- "availableFieldSets": "AvailableFieldSet[]" (OPTIONAL) an array of objects, with each element describing an available field set. Members are:

  * "name": "String" (REQUIRED) the field set name;
  * "default": "Boolean" (REQUIRED) whether the field set is applied by default;
  * "description": "String" (OPTIONAL) a human-readable description of the field set;
  * "links": "Link[]" (OPTIONAL) an array of links as described in [RFC8288] containing the query string that applies the field set.

2.1.1. RDAP Conformance

Servers returning the "subsetting_metadata" section in their responses MUST include "subsetting" in the rdapConformance array.

2.1.2. Representing Subsetting Links

An RDAP server MAY use the "links" array of the "subsetting_metadata" element to provide ready-made references [RFC8288] to the available field sets (Figure 2). The target URI in each link is the reference to an alternative to the current view of results identified by the context URI.

```
{
  "rdapConformance": [
    "rdap_level_0",
    "subsetting"
  ],
  ...
  "subsetting_metadata": {
    "currentFieldSet": "afieldset",
    "availableFieldSets": [
      {
      "name": "anotherfieldset",
      "description": "Contains some fields",
      "default": false,
      "links": [
        {
        "value": "https://example.com/rdap/domains?name=*nr.com
                  &fieldSet=afieldset",
        "rel": "alternate",
        "href": "https://example.com/rdap/domains?name=*nr.com
                 &fieldSet=anotherfieldset",
        "title": "Result Subset Link",
        "type": "application/rdap+json"
        }
      ]
      },
    ...
    ]
  },
  ...
  "domainSearchResults": [
    ...
  ]
}
```

Figure 2: Example of a "subsetting_metadata" instance

3. Dealing with Relationships

Representation of second level objects within a field set produces additional considerations. Since the topmost objects could be returned according to different field sets, the same field sets could be applied to their related objects. As a consequence, the response could contain either no relationship or associated objects which are in turn provided according to a field set.

4. Basic Field Sets

This section defines three basic field sets which servers MAY implement to facilitate their interaction with clients:

- "id": the server provides only the key field, respectively: "handle" for entities, "ldhName" for domains and nameservers. If a returned domain or nameserver is an Internationalized Domain Name (IDN) [RFC5890], then the "unicodeName" field MUST be included in the response. This field set could be used when the client wants to obtain a collection of object identifiers (Figure 3);

- "brief": the field set contains the fields that can be included in a "short" response. This field set could be used when the client is asking for a subset of the full response which provides only basic knowledge of each object;

- "full": the field set contains all of the information the server can provide for a particular object.

The "objectClassName" field is implicitly included in each of the above field sets. RDAP providers are RECOMMENDED to include a "self" link in each field set. RDAP providers MAY also add any property providing service information.

Fields included in the "brief" and "full" field sets MUST be returned according to the user's access and authorization levels.

```
{
  "rdapConformance": [
    "rdap_level_0",
    "subsetting"
  ],
  ...
  "domainSearchResults": [
    {
      "objectClassName": "domain",
      "ldhName": "example1.com",
      "links": [
        {
        "value": "https://example.com/rdap/domain/example1.com",
        "rel": "self",
        "href": "https://example.com/rdap/domain/example1.com",
        "type": "application/rdap+json"
        }
      ]
    },
    {
      "objectClassName": "domain",
      "ldhName": "example2.com",
      "links": [
        {
        "value": "https://example.com/rdap/domain/example2.com",
        "rel": "self",
        "href": "https://example.com/rdap/domain/example2.com",
        "type": "application/rdap+json"
        }
      ]
    },
    ...
  ]
}
```

Figure 3: Example of RDAP response according to the "id" field set

5. Negative Answers

Each request including an unsupported field set SHOULD produce an HTTP 400 (Bad Request) response code. Optionally, the response MAY include additional information regarding the negative answer in the HTTP entity body.

6. IANA Considerations

IANA is requested to register the following value in the RDAP Extensions Registry:

Extension identifier: subsetting
Registry operator: Any
Published specification: This document.
Contact: IESG <iesg@ietf.org>
Intended usage: This extension describes best practice for partial response provisioning.

7. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

7.1. IIT-CNR/Registro.it

Responsible Organization: Institute of Informatics and Telematics of the National Research Council (IIT-CNR)/Registro.it
Location: https://rdap.pubtest.nic.it/
Description: This implementation includes support for RDAP queries using data from .it public test environment.
Level of Maturity: This is an "alpha" test implementation.
Coverage: This implementation includes all of the features described in this specification.
Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

7.2. APNIC

Responsible Organization: Asia-Pacific Network Information Centre
Location: https://github.com/APNIC-net/rdap-rmp-demo/tree/partial-response
Description: A proof-of-concept for RDAP mirroring.
Level of Maturity: This is a proof-of-concept implementation.
Coverage: This implementation includes all of the features described in this specification.
Contact Information: Tom Harrison, tomh@apnic.net

8. Security Considerations

A search query typically requires more server resources (such as memory, CPU cycles, and network bandwidth) when compared to a lookup query. This increases the risk of server resource exhaustion and subsequent denial of service due to abuse. This risk can be mitigated by supporting the return of partial responses combined with other strategies (e.g. restricting search functionality, limiting the rate of search requests, and truncating and paging results).

Support for partial responses gives RDAP operators the ability to implement data access control policies based on the HTTP authentication mechanisms described in [RFC7481]. RDAP operators can vary the information returned in RDAP responses based on a client's access and authorization levels. For example:

- the list of fields for each set can differ based on the client's access and authorization levels;
- the set of available field sets could be restricted based on the client's access and authorization levels.

Servers can also define different result limits according to the available field sets, so a more flexible truncation strategy can be implemented. The new query parameter presented in this document provides RDAP operators with a way to implement a server that reduces inefficiency risks.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", RFC 7480, DOI 10.17487/RFC7480, March 2015, <https://www.rfc-editor.org/info/rfc7480>.

[RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", RFC 7481, DOI 10.17487/RFC7481, March 2015, <https://www.rfc-editor.org/info/rfc7481>.

[RFC7482] Newton, A. and S. Hollenbeck, "Registration Data Access Protocol (RDAP) Query Format", RFC 7482, DOI 10.17487/RFC7482, March 2015, <https://www.rfc-editor.org/info/rfc7482>.

[RFC7483] Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", RFC 7483, DOI 10.17487/RFC7483, March 2015, <https://www.rfc-editor.org/info/rfc7483>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/info/rfc8288>.

9.2. Informative References

[CQL] Whitaker, G., "Catnap Query Language Reference", September 2017, <https://github.com/gregwhitaker/catnap/wiki/Catnap-Query-Language-Reference>.

[FACEBOOK]
facebook.com, "facebook for developers - Using the Graph API", July 2017, <https://developers.facebook.com/docs/graph-api/using-graph-api>.

[GOOGLE] google.com, "Making APIs Faster: Introducing Partial Response and Partial Update", March 2010, <http://googlecode.blogspot.it/2010/03/making-apis-faster-introducing-partial.html>.

[HATEOAS] Jedrzejewski, B., "HATEOAS - a simple explanation", 2018, <https://www.e4developer.com/2018/02/16/hateoas-simple-explanation/>.

[LINKEDIN]
linkedin.com, "Java One 2009: Building Consistent RESTful APIs in a High Performance Environment", July 2009, <https://blog.linkedin.com/2009/07/08/brandon-duncan-java-one-building-consistent-restful-apis-in-a-high-performance-environment>.

[REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000, <http://www.restapitutorial.com/media/RESTful_Best_Practices-v1_1.pdf>.

[REST-API1]
Jobinesh, P., "RESTful Java Web Services - Second Edition", September 2015.

[REST-API2]
Masse, M., "REST API Design Rulebook", October 2011.

Appendix A. Approaches to Partial Response Implementation

Looking at the implementation experiences described in Section 1, two approaches to the implementation of partial response are observed:

- The client explicitly describes the data fields to be returned;
- The client describes a name identifying a server-defined set of data fields.

The former is more flexible than the latter because clients can specify all the data fields they need. However, it has some drawbacks:

- Fields have to be declared according to a given syntax. This is a simple task when the data structure of the object is flat, but it is much more difficult when the object has a tree structure like that of a JSON object. The presence of arrays and deep nested objects complicate both the syntax definition of the query and, consequently, the processing required on the server side;

- Clients need to recognize the returned data structure to avoid cases when the requested fields are invalid;

- The request of some fields might not match the client's access and authorization levels. Clients might request unauthorized fields and servers should define a strategy for responding, such as always returning an error response or returning a response that ignores the unauthorized fields.

A.1. Specific Issues Raised by RDAP

In addition to those listed above, RDAP responses raise some specific issues:

- Relevant entity object information is included in a jCard, but such information cannot be easily selected because it is split into the items of a jagged array;

- RDAP responses contain some properties providing service information (e.g. rdapConformance, links, notices, remarks, etc.) which are not normally selected but they are just as important. They could be returned anyway but, in this case, the server would provide unrequested data.

It is possible to address these issues. For example, the Catnap Query Language [CQL] is a comprehensive expression language that can be used to customize the JSON response of a RESTful web service. Application of CQL to RDAP responses would explicitly identify the output fields that would be acceptable when a few fields are requested but it would become very complicated when processing a larger number of fields. In the following, two CQL expressions for a domain search query are shown (Figure 4). In the first, only objectClassName and ldhName are requested. In the second, the fields of a possible WHOIS-like response are listed.

```
https://example.com/rdap/domains?name=example*.com
        &fields=domainSearchResults(objectClassName,ldhName)

https://example.com/rdap/domains?name=example*.com
        &fields=domainSearchResults(objectClassName,ldhName,
                unicodeName,
                status,
                events(eventAction,eventDate),
                entities(objectClassName,handle,roles),
                nameservers(objectClassName,ldhName))
```

Figure 4: Examples of CQL expressions for a domain search query

The latter approach seems to facilitate RDAP interoperability. Servers can define basic field sets which, if known to clients, can increase the probability of obtaining a valid response. The usage of field sets makes the query string be less complex. Moreover, the definition of pre-defined sets of fields makes it easier to establish result limits.

Finally, considering that there is no real need for RDAP users to have the maximum flexibility in defining all the possible sets of logically connected fields (e.g. users interested in domains usually need to know the status, the creation date, and the expiry date of each domain), the latter approach is preferred.

Acknowledgements

The authors would like to acknowledge Scott Hollenbeck, Tom Harrison, Karl Heinz Wolf, Jasdip Singh and Patrick Mevzek for their contribution to this document.

Change Log

00: Initial working group version ported from draft-loffredo-regext-rdap-partial-response-03

01: Removed "FOR DISCUSSION" items. Changed the basic field sets from REQUIRED to OPTIONAL. Removed the definition of fields included in "brief" field set. Provided a more detailed description of "subsetting_metadata" structure. Removed some references.

02: Added the "Negative Answers" section. Changed "IANA Considerations" section.

03: Added the "unicodeName" field in the id fieldSet when a returned domain or nameserver is an IDN. Added RFC5890 to "Normative References" section.

04: Recommended the RDAP providers to include a "self" link in any field set other than "full". Updated "Acknowledgements" section.

05: Moved "Approaches to Partial Response Implementation" section to the appendix.
06: Clarified the use of self links in "Basic Field Sets" section. Added APNIC to the implementations of the "Implementation Status" section.
07: Changed "only a subset is returned" to "only a subset of fields in each result object is returned" in the "Introduction" section. Moved the "RDAP Conformance" section up in the document. Updated the "Acknowledgements" section.
08: Changed the rdapConformance tag "subsetting_level_0" to "subsetting". Moved [RFC7942] to the "Normative References".
09: Corrected the "rdapConformance" content in Figure 3.
10: Corrected the JSON content in Figure 2. Clarified the meaning of both context and target URIs in a result subset link defined in Section 2.1.2. Updated the "Acknowledgements" section.
11: Minor pre-AD review edits.
12: Additional minor pre-AD review edits.

Authors' Addresses

Mario Loffredo
IIT-CNR/Registro.it
Via Moruzzi,1
Pisa 56124
IT

Email: mario.loffredo@iit.cnr.it
URI: http://www.iit.cnr.it

Maurizio Martinelli
IIT-CNR/Registro.it
Via Moruzzi,1
Pisa 56124
IT

Email: maurizio.martinelli@iit.cnr.it
URI: http://www.iit.cnr.it