    Registration Data Access Protocol (RDAP) Partial Response
              draft-ietf-regext-rdap-partial-response-08

Abstract

   The Registration Data Access Protocol (RDAP) does not include
   capabilities to request partial responses. In fact, according to the
   user authorization, the server can only return full responses. A
   partial response capability, 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 October 16, 2020.

Copyright Notice

Table of Contents

1. Introduction

   The use of partial response 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 fields in
   each result object is returned. The benefit is obvious: less data
   transferred over the network means less bandwidth usage, faster
   server response, less CPU time spent both on the server and the
   client, as well as less memory usage on the client.

   Several leading APIs providers (e.g. LinkedIn [LINKEDIN], Facebook
   [FACEBOOK], Google [GOOGLE]) implement the partial response feature
   by providing an optional query parameter by which users require the
   fields they wish to receive. Partial response is also considered a
   leading principle by many best practices guidelines in REST APIs
   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 provide responses according to

   different element sets (i.e. "brief" to get back a short response and
   "full" to get back the complete response)

   Currently, RDAP does not provide a client with any way to request a
   partial response: the server can only provide the client with the
   full response ([RFC7483]). Furthermore, servers cannot define the
   limits of the results according to partial responses and this causes
   strong inefficiencies.

   The protocol described in this specification extends RDAP search
   capabilities to enable partial responses through the provisioning of
   pre-defined sets of fields the user can request 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 RFC 7480 ([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 RFC 2119 ([RFC2119]).

2. RDAP Path Segment Specification

   The path segment defined in this section is an OPTIONAL extension of
   search path segments defined in RFC 7482 ([RFC7482]). This document
   defines an RDAP query parameter, "fieldSet", whose value is a string
   identifying a server pre-defined set of fields (Figure 1).

   This solution can be implemented by the RDAP providers with less
   effort than fields selection and easily requested by consumers. The
   considerations that has led to opt for this solution are reported in
   more detail in Appendix A.

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

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

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 the server-provided links to dynamically discover
   available actions and access the resources it needs. In this way,
   the client is not requested to have prior knowledge of the service

and, consequently, to hard code the URIs of different resources.
This would allow the server to make URI changes as the API evolves
without breaking the 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:

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

o "availableFieldSets": "AvailableFieldSet[]" (OPTIONAL) an array of
   objects each one describing an alternate 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
      RFC 8288 ([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). Each link represents a reference to an
alternate view of the results.

```
{
  "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

   Some additional considerations can be made about how second level
   objects could be represented within a field set. In fact, since the
   topmost objects could be returned according to different field sets,
   the same thing could go for 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

   In order to improve interoperability between clients and servers, the
   name, as well as the list of fields for each field set, should be
   shared by most of RDAP providers. This section defines three basic

field sets which servers MAY implement to facilitate their
interaction with clients:

o "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 IDN ([RFC5890]), then the
   "unicodeName" field MUST be included in the response. This field
   set could be used when the client wants to simply obtain a
   collection of object identifiers (Figure 3);

o "brief": it 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 gives a basic knowledge of
   each object;

o "full": it contains all 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 "brief" and "full" field sets could be returned
according to the user access levels.

```
   {
     "rdapConformance": [
           "rdap_level_0",
     ],
     ...
     "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 obtain an HTTP
   400 (Bad Request) response code.

   Optionally, the response MAY include additional information regarding
   the negative answer in the HTTP entity body.

6. 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 RFC 7942
   ([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".

6.1. IIT-CNR/Registro.it

   Responsible Organization: Institute of Informatics and Telematics
   of 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

6.2. 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

7. 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 a best practice for
      partial response provisioning.

8. Security Considerations

   The search query typically requires more server resources (such as
   memory, CPU cycles, and network bandwidth) when compared to the
   lookup query. This increases the risk of server resource exhaustion
   and subsequent denial of service due to abuse. Partial response can
   contribute together with other strategies (e.g. restricting search
   functionality, limiting the rate of search requests, truncating and
   paging results) to mitigate this risk.

   Furthermore, partial response can support RDAP operators to implement
   a versatile access control policy through the HTTP authentication
   mechanisms as described in RFC 7481 ([RFC7481]). In fact, RDAP
   operators can follow different, not alternative, approaches to the
   building of responses according to the user access levels:

   o the list of fields for each set can be different according to the
      user access levels;

   o some field sets could be available only to some users.

   Servers can also define different results limits according to the
   available field sets, so a more flexible truncation strategy can be
   realized.

   Therefore, the new query parameter presented in this document
   provides the RDAP operators with a way to implement a secure server
   without penalizing its efficiency.

9. Acknowledgements

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

10. References

10.1. 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>.

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

10.2. 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 can be detected:

   o the client declares explicitly the data fields to get back;

   o the client declares a name identifying a server pre-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:

o 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
   the one of a JSON object. The presence of arrays and deep nested
   objects contributes to complicate both the syntax definition of
   the query and, consequently, the processing phase on the server
   side;

o clients should perfectly know the returned data structure to avoid
   cases when the requested fields are invalid;

o the request of some fields might not match the user access levels.
   Clients might put unauthorized fields in their requests and
   servers should define a strategy for providing a response:
   returning always 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:

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

o 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.

As an example compliant to the first approach, the Catnap Query
Language ([CQL]) is a comprehensive expression language that can be
used to customize the JSON response of a RESTful web service. The
practical application of CQL to RDAP responses points out that
declaring explicitly the output fields would still be acceptable when
a few fields are requested but it would become very complicated if
the fields should be more. In the following, two CQL expressions for
a search domain 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 search domain query

The latter approach seems to facilitate RDAP interoperability. In
fact, servers can define some basic field sets which, if known to the
clients, can increase the probability to get a valid response. The
usage of field sets lets the query string be less complex. In
addition, the definition of pre-defined sets of fields makes easier
to establish the results limits.

Finally, considering that there is not a 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, the expiry date of each
domain), the latter approach is preferred.

Appendix B. 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".

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