         Registration Data Access Protocol (RDAP) Partial Response
              draft-loffredo-regext-rdap-partial-response-03

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.
   Partial responses capability, especially in the case of search
   queries, could bring benefits to both clients and servers. This
   document describes a 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 August 5, 2019.

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 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 is returned.
   The benefit is obvious: less data transferred over the network mean
   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, by adding a new query
parameter and using a RESTful web service. The service is
implemented using the Hypertext Transfer Protocol (HTTP) ([RFC7230])
and the conventions described in RFC 7480 ([RFC7480]).

Impact on the current state of RDAP implementation is low.

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. Approaches to Partial Response Implementation

Looking at the implementation experiences described above, 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. Anyway, 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 contribute 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 object to avoid cases
   when the required fields are not compliant with the object data
   structure.

o The request of some fields cannot match the user access levels.
   Clients could put unauthorized fields in their requests and
   servers should define a strategy for providing a response: to

   return always an error response or to return a response ignoring
   the unauthorized fields.

   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 1): 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 1: 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 (for example, users interested in domains usually need to know the status, the creation date, the expire date of each domain), the latter approach is preferred.

3. RDAP Path Segment Specification

The new query parameter is an OPTIONAL extension of search path segments defined in RFC 7482 ([RFC7482]). The query parameter is "fieldSet" whose value is a string identifying a server pre-defined set of fields (Figure 2). Values REQUIRED to be implemented are:

o id: the server provides only the "objectClassName" field and the key field ("handle" for entities, "ldhName" for domains and nameservers). This field set can be used when the client wants to 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 can be used when the client is asking for a subset of the full response which gives a basic knowledge of each object. The fields are those defined in Section 3.1;

o full: it contains all the information the server can provide for a particular object. Additional considerations are reported in Section 3.2.

Fields belonging to brief and full field sets should be provided according to users access levels. RDAP providers MAY add any property providing service information. Servers MAY implement additional field sets not included in the list above. Servers SHOULD also define a "default" field set.

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

Figure 2: Example of RDAP search query reporting the fieldSet
                          parameter

```
   {
     "rdapConformance": [
           "rdap_level_0",
     ],
     ...
     "domainSearchResults": [
       {
         "objectClassName": "domain",
         "ldhName": "example1.com"
       },
       {
         "objectClassName": "domain",
         "ldhName": "example2.com"
       },
       ...
     ]
   }
```

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

3.1. Brief Field Set

   In order to ensure the highest degree of interoperability, the brief
   field set should contain the most commonly used data elements. Based
   on the assumption that an RDAP server will return almost the same
   data as those replied by the corresponding Whois service, the
   elements included in the brief field set could be those identified in
   RFC 7485 ([RFC7485]) as mostly supported (i.e. by more than one third
   of contacted services).

   Therefore, RDAP servers are RECOMMENDED to return the following
   elements in the brief field set (Table 1):

```
+-----------+----------------+---------------------------------+
| Object    | Whois property | RDAP property                   |
| class     |                |                                 |
+-----------+----------------+---------------------------------+
| Domain    | Domain Name    | ldhName                         |
|           | Domain Status  | status                          |
|           | Creation Date  | event whose eventAction type is |
|           |                | "registration"                  |
|           | Expiration Date| event whose eventAction type is |
|           |                | "expiration"                    |
|           | Update Date    | event whose eventAction type is |
|           |                | "last update"                   |
|           |                |                                 |
| Nameserver| Name Server    | ldhName                         |
|           |                |                                 |
| Entity    | Entity ID      | handle                          |
|           | Entity Name    | vcard fn                        |
|           | Entity         | vcard org                       |
|           | Organization   |                                 |
|           | Entity Email   | vcard email                     |
|           | Entity Phone   | vcard tel with type="voice"     |
|           | Entity Fax     | vcard tel with type="fax"       |
|           | Entity Country | country name in vcard adr       |
|           | Entity City    | locality in vcard adr           |
|           | Entity Postal  | postal code in vcard adr        |
|           | Code           |                                 |
+-----------+----------------+---------------------------------+
```

            Table 1: Elements included in brief field set

   The term "Entity" refers to any kind of contact.

3.2. Full Field Set

   With regards to the full field set, some additional considerations
   can be made about how second level objects could be represented. 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 full response could range from the
   one containing no relationship up to a response where each related
   object is in turn in full format.

   FOR DISCUSSION: Should this specification furtherly detail the full
   field set according to the different representations of the related
   objects?

3.3. 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
   self-descriptive as much as possible.

   Therefore, the implementation of the query parameter described in
   this specification recommends servers to provide additional
   information in their responses about the available field sets. Such
   information is collected in a new data structures named
   "subsetting_metadata" containing the following fields:

   o "currentFieldSet": the value of fieldSet parameter as specified in
      the query string;

   o "availableFieldSets": an array of objects each one describing an
      available field set:

      * "name": the field set name;
      * "description": a human-readable description of the field set;
      * "default": whether the field set is applied by default;
      * "links": an array of links as described in RFC 8288 ([RFC8288])
         containing the query string that applies the field set.

   Both "currentFieldSet" and "availableFieldSets" are OPTIONAL fields
   of the "subsetting_metadata" structure. In particular, the
   "currentFieldSet" field is provided when the query string contains a
   valid value for fieldSet parameter, while the "availableFieldSets"
   field SHOULD be provided when the fieldSet parameter is missing in
   the query string or when it is present and the server implements more
   than a field set for the RDAP object. At least the "name" field is
   REQUIRED in each item of the "availableFieldSets" array while the
   other fields are RECOMMENDED.

3.3.1. Representing Subsetting Links

   An RDAP server MAY use the "links" array of the "subsetting_metadata"
   section to provide ready-made references ([RFC8288]) to the available
   field set (Figure 4). Each link represents a reference to an
   alternate view of the results.

```
{
  "rdapConformance": [
    "rdap_level_0",
    "subsetting_level_0"
  ],
  ...
  "subsetting_metadata": {
      "currentFieldSet": "brief",
      "availableFieldSets": [
      {
      "name": "id",
      "description": "Contains "objectClassName" and the key field",
      "default": false,
      "links": [
          {
          "value": "https://example.com/rdap/domains?name=*nr.com
                    &fieldSet=brief",
          "rel": "alternate",
          "href": "https://example.com/rdap/domains?name=*nr.com
                    &fieldSet=id",
          "title": "Result Subset Link",
          "type": "application/rdap+json"
          },
          ...
      ]
    },
    "domainSearchResults": [
      ...
    ]
  }
}
```

             Figure 4: Example of a "subsetting_metadata" instance

4. RDAP Conformance

   Servers returning the "subsetting_metadata" section in their
   responses MUST include "subsetting_level_0" in the rdapConformance
   array.

5. 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".

5.1. 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 the public test environment of .it ccTLD.
Level of Maturity: This is a "proof of concept" research
implementation.
Coverage: This implementation includes all of the features
described in this specification.
Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

6. Security Considerations

Search query typically requires more server resources (such as
memory, CPU cycles, and network bandwidth) when compared to 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 help RDAP operators to regulate
access control based on client identification, implemented by HTTP
basic or digest authentication as described in RFC 7481 ([RFC7481])
or by a federated authentication system
([I-D.hollenbeck-regext-rdap-openid]). 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 (except "id") can be different
   according to the user access levels. At present, this is already
   implemented for the full response, but it could be done also for
   the other defined field sets. In some cases, it might happen that
   brief and full field sets are exactly the same;

o some field sets could be available only to some users. In this
   case, servers could define additional field sets to those
   indicated above ("id", "brief", "full"), making them available
   only to users with specific access levels.

Servers can also define different results limits according to the
available field sets, so a more flexible truncation strategy can be
realized and users can take advantage of a more efficient results
paging implementation
([I-D.loffredo-regext-rdap-sorting-and-paging]).

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

7. IANA Considerations

   This document has no actions for IANA.

8. Acknowledgements

   The authors would like to acknowledge Scott Hollenbeck for his
   contribution to this document.

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>.

   [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>.

   [RFC7485] Zhou, L., Kong, N., Shen, S., Sheng, S., and A. Servin,
              "Inventory and Analysis of WHOIS Registration Objects",
              RFC 7485, DOI 10.17487/RFC7485, March 2015,
              <https://www.rfc-editor.org/info/rfc7485>.

   [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/>.

   [I-D.hollenbeck-regext-rdap-openid]
             Hollenbeck, S., "Federated Authentication for the
             Registration Data Access Protocol (RDAP) using OpenID
             Connect", draft-hollenbeck-regext-rdap-openid-10 (work in
             progress), August 2018.

   [I-D.loffredo-regext-rdap-sorting-and-paging]
             Loffredo, M., Martinelli, M., and S. Hollenbeck,
             "Registration Data Access Protocol (RDAP) Query Parameters
             for Result Sorting and Paging", draft-loffredo-regext-
             rdap-sorting-and-paging-05 (work in progress), September
             2018.

   [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.

   [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>.

Appendix A. Change Log

   00: Initial version.
   01: Added Catnap Query Language as an example of language that can
        be used to declare explicitly the output fields of RDAP responses.
        Revised some sentences and references.
   02: Added "Subsetting Metadata" and "RDAP Conformance" sections.
   03: Added "Brief Field Set" and "Full Field Set" sections.

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