Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: March 25, 2019

M. Loffredo
M. Martinelli
IIT-CNR/Registro.it
S. Hollenbeck
Verisign Labs

September 21, 2018

Registration Data Access Protocol (RDAP) Query Parameters for Result Sorting and Paging
draft-loffredo-regext-rdap-sorting-and-paging-05

Abstract

The Registration Data Access Protocol (RDAP) does not include core functionality for clients to provide sorting and paging parameters for control of large result sets. This omission can lead to unpredictable server processing of queries and client processing of responses. This unpredictability can be greatly reduced if clients can provide servers with their preferences for managing response values. This document describes RDAP query extensions that allow clients to specify their preferences for sorting and paging result sets.

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 March 25, 2019.

Copyright Notice

Table of Contents

1. Introduction

The availability of functionality for result sorting and paging provides benefits to both clients and servers in the implementation of RESTful services [REST]. These benefits include:

- reducing the server response bandwidth requirements;
- improving server response time;
- improving query precision and, consequently, obtaining more reliable results;
- decreasing server query processing load;
- reducing client response processing time.

Approaches to implementing features for result sorting and paging can be grouped into two main categories:

1. Sorting and paging are implemented through the introduction of additional parameters in the query string (i.e. ODATA protocol [OData-Part1]);

2. Information related to the number of results and the specific portion of the result set to be returned, in addition to a set of ready-made links for the result set scrolling, are inserted in the HTTP header of the request/response.

However, there are some drawbacks associated with use of the HTTP header. First, the header properties cannot be set directly from a web browser. Moreover, in an HTTP session, the information on the status (i.e. the session identifier) is usually inserted in the header or in the cookies, while the information on the resource identification or the search type is included in the query string. The second approach is therefore not compliant with the HTTP standard [RFC7230]. As a result, this document describes a specification based on use of query parameters.

Currently the RDAP protocol [RFC7482] defines two query types:

- lookup: the server returns only one object;
- search: the server returns a collection of objects.

While the lookup query does not raise issues in the management of large result sets, the search query can potentially generate a large result set that could be truncated according to the limits of the server. In addition, it is not possible to obtain the total number of the objects found that might be returned in a search query response [RFC7483]. Lastly, there is no way to specify sort criteria to return the most relevant objects at the beginning of the result set. Therefore, the client could traverse the whole result set to find the relevant objects or, due to truncation, could not find them at all.

The protocol described in this specification extends RDAP query capabilities to enable result sorting and paging, by adding new query parameters that can be applied to RDAP search path segments. The service is implemented using the Hypertext Transfer Protocol (HTTP) [RFC7230] and the conventions described in RFC 7480 [RFC7480].

The implementation of these parameters is technically feasible, as operators for counting, sorting and paging rows are currently supported by the major RDBMSs.

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 [RFC2119].

2. RDAP Query Parameter Specification

The new query parameters are OPTIONAL extensions of path segments defined in RFC 7482 [RFC7482]. They are as follows:

- "count": a boolean value that allows a client to request the total number of objects found (that due to truncation can be different from the number of returned objects);
- "sort": a string value that allows a client to request a specific sort order for the result set;
- "limit" and "offset": numeric values that allow a client to request a specific portion of the entire result set.

Augmented Backus-Naur Form (ABNF) [RFC5234] is used in the following sections to describe the formal syntax of these new parameters.

2.1. Sorting and Paging 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 parameters described in this specification recommends servers to provide additional information in their responses about both the available sorting criteria and the possible pagination. Such information is collected in two new data structures named, respectively, "sorting_metadata" and "paging_metadata".

Obviously, both the new data structures are OPTIONAL because their presence in the response not only depends on the implementation of sorting and paging query capabilities but also on some situations related to the results. For example, it is quite natural to expect

that the "paging_metadata" section will not be present at the last result page when the server implements only the forward pagination.

The "sorting_metadata" structure contains the following fields:

- "currentSort": the value of sort parameter as specified in the query string;
- "availableSorts": an array of objects each one describing an available sorting criterion:
  - "property": the name that can be used by the client to request the sorting criterion;
  - "jsonPath": the JSON Path of the RDAP field corresponding to the property;
  - "default": whether the sorting criterion is applied by default;
  - "links": an array of links as described in RFC 8288 [RFC8288] containing the query string that applies the sorting criterion.

Both "currentSort" and "availableSorts" are OPTIONAL fields of the "sorting_metadata" structure. In particular, the "currentSort" field is provided when the query string contains a valid value for sort parameter, while the "availableSorts" field SHOULD be provided when the sort parameter is missing in the query string or when it is present and the server implements more than a sorting criterion for the RDAP object. At least the "property" field is REQUIRED in each item of the "availableSorts" array while the other fields are RECOMMENDED.

The "paging_metadata" structure contains the following fields:

- "totalCount": a numeric value representing the total number of objects found;
- "pageCount": a numeric value representing the number of objects returned in the current page;
- "offset": a numeric value identifying the start of current page in the result set;
- "nextOffset": a numeric value identifying the start of the next page in the result set or null if the result set has been completely scrolled;
- "links": an array of links as described in RFC 8288 [RFC8288] containing the reference to next page.

Only the "pageCount" field is REQUIRED in the "paging_metadata" structure. The other fields appear when pagination occurs. In this specification, only the forward pagination is dealt because it is considered satisfactory in order to traverse the result set. If a server should also implement backward pagination, an appropriate field (e.g. "prevOffset") identifying the start of the previous page is RECOMMENDED. Finally, the "totalCount" field is provided if the query string contains the count parameter.

FOR DISCUSSION: Should the metadata described in this specification be part of a more general "metadata" section including other contents (e.g rate limits, information about the server, information about the response, metadata information related to other parameters)?

2.2. "count" Parameter

Currently the RDAP protocol does not allow a client to determine the total number of the results in a query response when the result set is truncated. This is rather inefficient because the user cannot evaluate the query precision and, at the same time, cannot receive information that could be relevant.

The count parameter provides additional functionality (Figure 1) that allows a client to request information from the server that specifies the total number of elements matching a particular search pattern.

```
https://example.com/rdap/domains?name=*nr.com&count=true
```

Figure 1: Example of RDAP query reporting the count parameter

The ABNF syntax is the following:

```
count = "count" EQ ( trueValue / falseValue )
trueValue = ("true" / "yes" / "1")
falseValue = ("false" / "no" / "0")
EQ = "="
```

A trueValue means that the server MUST provide the total number of the objects in the "totalCount" field of the "paging_metadata" section (Figure 2). A falseValue means that the server MUST NOT provide this number.

```
{
  "rdapConformance": [
        "rdap_level_0",
        "paging_level_0"
  ],
  ...
  "paging_metadata": {
    "totalCount": 73
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 2: Example of RDAP response with "paging_metadata" section containing the "totalCount" field

2.3. "sort" Parameter

The RDAP protocol does not provide any capability to specify results sort criteria. A server could implement a default sorting scheme according to the object class, but this feature is not mandatory and might not meet user requirements. Sorting can be addressed by the client, but this solution is rather inefficient. Sorting and paging features provided by the RDAP server could help avoid truncation of relevant results and allow for scrolling the result set using subsequent queries.

The sort parameter allows the client to ask the server to sort the results according to the values of one or more properties and according to the sort direction of each property. The ABNF syntax is the following:

```
sort = "sort" EQ sortItem *( "," sortItem )
sortItem = property-ref [":" ( "a" / "d" ) ]
```

"a" means that the ascending sort MUST be applied, "d" means that the descending sort MUST be applied. If the sort direction is absent, an ascending sort MUST be applied (Figure 3).

In the sort parameter ABNF syntax, property-ref represents a reference to a property of an RDAP object. Such a reference could be expressed by using a JSON Path. The JSON Path in a JSON document [RFC8259] is equivalent to the XPath [W3C.CR-xpath-31-20161213] in a XML document. For example, the JSON Path to select the value of the ASCII name inside an RDAP domain object is "$.ldhName", where $ identifies the root of the document (DOM). Another way to select a value inside a JSON document is the JSON Pointer [RFC6901]. While

JSON Path or JSON Pointer are both standard ways to select any value inside JSON data, neither is particularly easy to use (e.g. "$.events[?(@.eventAction='registration')].eventDate" is the JSON Path expression of the registration date in a RDAP domain object).

Therefore, this specification provides a definition of property-ref in terms of RDAP properties. However, not all the RDAP properties are suitable to be used in sort criteria, such as:

- properties providing service information (e.g. links, notices, remarks, etc.);
- multivalued properties (e.g. status, roles, variants, etc.);
- properties modeling relationships to other objects (e.g. entities).

On the contrary, some properties expressed as values of other properties (e.g. registration date) could be used in such a context.

In the following, a list of the proposed properties for sort criteria is presented. The properties are divided in two groups: object common properties and object specific properties.

- Object common properties. Object common properties are derived from the merge of the "eventAction" and the "eventDate" properties. The following values of the sort parameter are defined:
  - registrationDate
  - reregistrationDate
  - lastChangedDate
  - expirationDate
  - deletionDate
  - reinstantiationDate
  - transferDate
  - lockedDate
  - unlockedDate
- Object specific properties. With regard to the specific properties, some of them are already defined among the query paths. In the following the list of the proposed sorting properties, grouped by objects, is shown:
  - Domain: ldhName
  - Nameserver: ldhName, ipV4, ipV6.
  - Entity: fn, handle, org, email, voice, country, city.

In the following, the correspondence between the sorting properties and the RDAP fields is shown (Table 1):

Object class	Sorting property	RDAP property	Reference in RFC 7483	Reference in RFC 6350
Searchable objects	Common properties	eventAction values suffixed by "Date"	4.5.	
Domain	ldhName	ldhName	5.3.	
Nameserver	ldhName	ldhName	5.2.	
	ipV4	v4 ipAddress	5.2.	
	ipV6	v6 ipAddress	5.2.	
Entity	handle	handle	5.1.	
	fn	vcard fn	5.1.	6.2.1
	org	vcard org	5.1.	6.6.4
	voice	vcard tel with type="voice"	5.1.	6.4.1
	email	vcard email	5.1.	6.4.2
	country	country name in vcard adr	5.1.	6.3.1
	city	locality in vcard adr	5.1.	6.3.1

Table 1: Sorting properties definition

With regard to the definitions in Table 1, some further considerations must be made to disambiguate cases where the RDAP property is multivalued:

- Even if a nameserver can have multiple IPv4 and IPv6 addresses, the most common configuration includes one address for each IP version. Therefore, the assumption of having a single IPv4 and/or IPv6 value for a nameserver cannot be considered too stringent.

- With the exception of handle values, all the sorting properties defined for entity objects can be multivalued according to the definition of vCard as given in RFC6350 [RFC6350]. When more than a value is reported, sorting can be applied to the preferred value identified by the parameter pref="1".

Each RDAP provider MAY define other sorting properties than those shown in this document.

The "jsonPath" field in the "sorting_metadata" section is used to clarify the RDAP field the sorting property refers to. In the following, the mapping between the sorting properties and the JSON Paths of the RDAP fields is shown (Table 2). The JSON Paths are provided according to the Goessner v.0.8.0 specification ([GOESSNER-JSON-PATH]):

Object class	Sorting property	JSON Path
Searchable objects	registrationDate	"$.domainSearchResults[*].events[?(@.eventAction=="registration")].eventDate
	reregistrationDate	"$.domainSearchResults[*].events[?(@.eventAction=="reregistration")].eventDate
	lastChangedDate	"$.domainSearchResults[*].events[?(@.eventAction=="lastChanged")].eventDate
	expirationDate	"$.domainSearchResults[*].events[?(@.eventAction=="expiration")].eventDate
	deletionDate	"$.domainSearchResults[*].events[?(@.eventAction=="deletion")].eventDate
	reinstantiationDate	"$.domainSearchResults[*].events[?(@.eventAction=="reinstantiation")].eventDate
	transferDate	"$.domainSearchResults[*].events[?(@.eventAction=="transfer")].eventDate
	lockedDate	"$.domainSearchResults[*].events[?(@.eventAction=="locked")].eventDate
	unlockedDate	"$.domainSearchResults[*].events[?(@.eventAction=="unlocked")].eventDate
Domain	ldhName	$.domainSearchResults[*].ldhName
Nameserver	ldhName	$.nameserverSearchResults[*].ldhName
	ipV4	$.nameserverSearchResults[*].ipAddresses.v4[0]
	ipV6	$.nameserverSearchResults[*].ipAddresses.v6[0]
Entity	handle	$.entitySearchResults[*].handle
	fn	$.entitySearchResults[*].vcardArray[1][?(@[0]="fn")][3]
	org	$.entitySearchResults[*].vcardArray[1][?(@[

		0]="org")][3]	
	voice	$.entitySearchResults[*].vcardArray[1][?(@[	
		0]=="tel" && @[1].type=="voice")][3]	
	email	$.entitySearchResults[*].vcardArray[1][?(@[	
		0]=="email")][3]	
	country	$.entitySearchResults[*].vcardArray[1][?(@[	
		0]=="adr")][3][6]	
	city	$.entitySearchResults[*].vcardArray[1][?(@[	
		0]=="adr")][3][3]	

Table 2: Sorting properties - JSON Path Mapping

If the sort parameter reports an allowed sorting property, it MUST be provided in the "currentSort" field of the "sorting_metadata" structure.

```
https://example.com/rdap/domains?name=*nr.com&sort=ldhName

https://example.com/rdap/domains?name=*nr.com&sort=registrationDate:d

https://example.com/rdap/domains?name=*nr.com&sort=lockedDate,ldhName
```

Figure 3: Examples of RDAP query reporting the sort parameter

2.3.1. Representing Sorting Links

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

```
{
  "rdapConformance": [
    "rdap_level_0",
    "sorting_level_0"
  ],
  ...
  "sorting_metadata": {
     "currentSort": "ldhName",
     "availableSorts": [
     {
     "property": "registrationDate",
     "jsonPath": "$.domainSearchResults[*].events[?(@.eventAction==\"registratio
n\")].eventDate",
     "default": false,
     "links": [
        {
        "value": "https://example.com/rdap/domains?name=*nr.com
                  &sort=ldhName",
        "rel": "alternate",
        "href": "https://example.com/rdap/domains?name=*nr.com
                 &sort=registrationDate",
        "title": "Result Ascending Sort Link",
        "type": "application/rdap+json"
        },
        {
        "value": "https://example.com/rdap/domains?name=*nr.com
                  &sort=ldhName",
        "rel": "alternate",
        "href": "https://example.com/rdap/domains?name=*nr.com
                 &sort=registrationDate:d",
        "title": "Result Descending Sort Link",
        "type": "application/rdap+json"
        }
     ]
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 4: Example of a "sorting_metadata" instance to implement result sorting

2.4. "limit" and "offset" Parameters

An RDAP query could return a response with hundreds of objects, especially when partial matching is used. For that reason, two parameters addressing result pagination are defined to make responses easier to handle:

- "limit": means that the server MUST return the first N objects of the result set in the response;
- "offset": means that the server MUST skip the first N objects and MUST return objects starting from position N+1.

The ABNF syntax is the following:

```
EQ = "="
limit = "limit" EQ positive-number
offset = "offset" EQ positive-number
positive-number = non-zero-digit *digit
non-zero-digit = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" /
"9"
digit = "0" / non-zero-digit
```

When limit and offset are used together, they allow implementation of result pagination. The following examples illustrate requests to return, respectively, the first 5 objects, the set of objects starting from position 6, and first 5 objects starting from position 11 of the result set (Figure 5).

```
https://example.com/rdap/domains?name=*nr.com&limit=5

https://example.com/rdap/domains?name=*nr.com&offset=5

https://example.com/rdap/domains?name=*nr.com&limit=5&offset=10
```

Figure 5: Examples of RDAP query reporting the limit and offset parameters

2.4.1. Representing Paging Links

An RDAP server MAY use the "links" array of the "paging_metadata" section to provide a ready-made reference [RFC8288] to the next page of the result set (Figure 6). Examples of additional "rel" values are "first", "last", "prev".

```
{
  "rdapConformance": [
    "rdap_level_0",
    "paging_level_0"
  ],
  ...
  "notices": [
    {
      "title": "Search query limits",
      "type": "result set truncated due to excessive load",
      "description": [
      "search results for domains are limited to 10"
      ]
    }
  ],
  "paging_metadata": {
    "totalCount": 73,
    "pageCount": 10,
    "offset": 10,
    "nextOffset": 20,
    "links": [
      {
      "value": "https://example.com/rdap/domains?name=*nr.com",
      "rel": "next",
      "href": "https://example.com/rdap/domains?name=*nr.com&limit=10
               &offset=10",
      "title": "Result Pagination Link",
      "type": "application/rdap+json"
      }
    ]
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 6: Example of a "paging_metadata" instance to implement result pagination based on offset and limit

3. Negative Answers

The value constraints for the parameters are defined by their ABNF syntax. Therefore, each request providing an invalid value for a parameter SHOULD obtain an HTTP 400 (Bad Request) response code. The same response SHOULD be returned if the client provides an unsupported value for the sort parameter in both single and multi sort.

The server can provide a different response when it supports the limit and/or offset parameters and the client submits values that are out of the valid ranges. The possible cases are:

- If the client submits a value for the limit parameter that is greater than the number of objects to be processed, it is RECOMMENDED that server returns a response including only the processed objects.

- If the client submits a value for the offset parameter that is greater than the number of objects to be processed, it is RECOMMENDED that server returns an HTTP 404 (Not Found) response code.

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

4. RDAP Conformance

Servers returning the "paging_metadata" section in their responses MUST include "paging_level_0" in the rdapConformance array as well as servers returning the "sorting_metadata" section MUST include "sorting_level_0".

5. Implementation Considerations

The implementation of the new parameters is technically feasible, as operators for counting, sorting and paging are currently supported by the major RDBMSs.

In the following, the match between the new defined parameters and the SQL operators is shown (Table 3):

New query parameter	SQL operator
count	count(*) query without offset, limit and order by [MYSQL-COUNT],[POSTGRES-COUNT],[ORACLE-COUNT]
sort	order by [MYSQL-SORT],[POSTGRES-SORT],[ORACLE-SORT]
limit	limit n (in MySql [MYSQL-LIMIT] and Postgres [POSTGRES-LIMIT]) FETCH FIRST n ROWS ONLY (in Oracle [ORACLE-LIMIT])
offset	offset m (in Postgres) OFFSET m ROWS (in Oracle)
limit + offset	limit n offset m (in MySql and Postgres) OFFSET m ROWS FETCH NEXT n ROWS ONLY (in Oracle)

Table 3: New query parameters vs. SQL operators

With regard to Oracle, Table 3 reports only one of the three methods that can be used to implement limit and offset parameters. The others are described in [ORACLE-ROWNUM] and [ORACLE-ROW-NUMBER].

In addition, similar operators are completely or partially supported by the most known NoSQL databases (MongoDB, CouchDB, HBase, Cassandra, Hadoop) so the implementation of the new parameters seems to be practicable by servers working without the use of an RDBMS.

5.1. Considerations about Paging Implementation

The use of limit and offset operators represents the most common way to implement results pagination. However, when offset has a high value, scrolling the result set could take some time. In addition, offset pagination may return inconsistent pages when data are frequently updated (i.e. real-time data) but this is not the case of registration data. An alternative approach to offset pagination is the keyset pagination, a.k.a. seek-method [SEEK] or cursor based pagination. This method has been taken as the basis for the implementation of a cursor parameter [CURSOR] by some REST API providers (e.g. [CURSOR-API1],[CURSOR-API2]). The cursor parameter is an opaque URL-safe string representing a logical pointer to the first result of the next page (Figure 7).

```
{
  "rdapConformance": [
    "rdap_level_0",
    "paging_level_0"
  ],
  ...
  "notices": [
    {
      "title": "Search query limits",
      "type": "result set truncated due to excessive load",
      "description": [
      "search results for domains are limited to 10"
      ]
    }
  ],
  "paging_metadata": {
    "totalCount": 73,
    "pageCount": 10,
    "links": [
      {
      "value": "https://example.com/rdap/domains?name=*nr.com",
      "rel": "next",
      "href": "https://example.com/rdap/domains?name=*nr.com&limit=10
              &cursor=wJlCDLIl6KTWypN7T6vc6nWEmEYe99Hjf1XY1xmqV-M=",
      "title": "Result Pagination Link",
      "type": "application/rdap+json"
      }
    ]
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 7: Example of a "paging_metadata" instance to implement keyset pagination

But keyset pagination raises some drawbacks with respect to offset pagination:

- it needs at least one key field;
- it does not allow to sort by any field and paginate the results because sorting has to be made on the key field;
- it does not allow to skip pages because they have to be scrolled in sequential order starting from the initial page;
- it makes very hard the navigation of the result set in both directions because all comparison and sort operations have to be reversed.

Furthermore, in the RDAP context, some additional considerations can be made:

- an RDAP object is a conceptual aggregation of information collected from more than one data structure (e.g. table) and this makes even harder for the developers the implementation of the seek-method that is already quite difficult. In fact, for example, the entity object can gather information from different data structures (registrars, registrants, contacts, resellers, and so on), each one with its own key field mapping the RDAP entity handle;

- depending on the number of the page results as well as the number and the complexity of the properties of each RDAP object in the response, the time required by offset pagination to skip the previous pages could be much faster than the processing time needed to build the current page. In fact, RDAP objects are usually formed by information belonging to multiple data structures and containing multivalued properties (e.g. arrays) and, therefore, data selection is a time consuming process. This situation occurs even though the data selection process makes use of indexes;

- depending on the access levels defined by each RDAP operator, the increase of complexity and the decrease of flexibility of keyset pagination with respect to the offset pagination could be considered impractical.

Finally, the keyset pagination is not fully compliant with the additional RDAP capabilities proposed by this document. In fact, the presence of a possible cursor parameter does not seem to be consistent with both the sorting capability and the possibility to implement additional ready-made links besides the classic "next page" link. But, while the provisioning of more paging links can be superfluous, dropping the sorting capability seems quite unreasonable.

If pagination is implemented by using a cursor, both "offset" and "nextOffset" fields MUST not be included in the "paging_metadata" section.

FOR DISCUSSION: Should RDAP specification reports both offset and cursor parameters and let operators to implement pagination according to their needs, the user access levels, the submitted queries?

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 the public test environment of .it ccTLD. The RDAP server does not implement any security policy because data returned by this server are only for experimental testing purposes. The RDAP server implements both offset and cursor based pagination (the latter only when sort and offset parameters are not present in the query string).
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

6.2. Google Registry

Responsible Organization: Google Registry
Location: https://www.registry.google/rdap/

Description: This implementation includes support for RDAP queries for TLDs such as .GOOGLE, .HOW, .SOY, and .xn--q9jyb4c . The RDAP server implements cursor based pagination (the number of objects per page is fixed so the limit parameter is not available). The link used to request the next page is included in the notice section of the response.
Level of Maturity: Production.
Coverage: This implementation includes the cursor parameter described in this specification.
Contact Information: Brian Mountford, mountford@google.com

7. Security Considerations

Security services for the operations specified in this document are described in RFC 7481 [RFC7481].

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. This risk can be mitigated by either restricting search functionality and limiting the rate of search requests. Servers can also reduce their load by truncating the results in the response. However, this last security policy can result in a higher inefficiency if the RDAP server does not provide any functionality to return the truncated results.

The new parameters presented in this document provide the RDAP operators with a way to implement a secure server without penalizing its efficiency. The "count" parameter gives the user a measure to evaluate the query precision and, at the same time, return a significant information. The sort parameter allows the user to obtain the most relevant information at the beginning of the result set. In both cases, the user doesn’t need to submit further unnecessary search requests. Finally, the limit and offset parameters enable the user to scroll the result set by submitting a sequence of sustainable queries according to the server limits.

8. IANA Considerations

This document has no actions for IANA.

9. Acknowledgements

The authors would like to acknowledge Brian Mountford for his contribution to the development of this document.

10. References

10.1. Normative References

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

[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>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <https://www.rfc-editor.org/info/rfc6350>.

[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>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

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

10.2. Informative References

[CURSOR] Nimesh, R., "Paginating Real-Time Data with Keyset Pagination", July 2014, <https://www.sitepoint.com/paginating-real-time-data-cursor-based-pagination/>.

[CURSOR-API1] facebook.com, "facebook for developers - Using the Graph API", July 2017, <https://developers.facebook.com/docs/graph-api/using-graph-api>.

[CURSOR-API2] twitter.com, "Pagination", 2017, <https://developer.twitter.com/en/docs/ads/general/guides/pagination.html>.

[GOESSNER-JSON-PATH] Goessner, S., "JSONPath - XPath for JSON", 2007, <http://goessner.net/articles/JsonPath/>.

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

[MYSQL-COUNT] mysql.com, "MySQL 5.7 Reference Manual, Counting Rows", October 2015, <https://dev.mysql.com/doc/refman/5.7/en/counting-rows.html>.

[MYSQL-LIMIT] mysql.com, "MySQL 5.7 Reference Manual, SELECT Syntax", October 2015, <https://dev.mysql.com/doc/refman/5.7/en/select.html>.

[MYSQL-SORT]
mysql.com, "MySQL 5.7 Reference Manual, Sorting Rows", October 2015, <https://dev.mysql.com/doc/refman/5.7/en/sorting-rows.html>.

[OData-Part1]
Pizzo, M., Handl, R., and M. Zurmuehl, "OData Version 4.0. Part 1: Protocol Plus Errata 03", June 2016, <http://docs.oasis-open.org/odata/odata/v4.0/errata03/os/complete/part1-protocol/odata-v4.0-errata03-os-part1-protocol-complete.pdf>.

[ORACLE-COUNT]
Oracle Corporation, "Database SQL Language Reference, COUNT", March 2016, <http://docs.oracle.com/database/122/SQLRF/COUNT.htm>.

[ORACLE-LIMIT]
Oracle Corporation, "Database SQL Language Reference, SELECT, Row limiting clause", March 2016, <http://docs.oracle.com/database/122/SQLRF/SELECT.htm>.

[ORACLE-ROW-NUMBER]
Oracle Corporation, "Database SQL Language Reference, SELECT, ROW_NUMBER", March 2016, <http://docs.oracle.com/database/122/SQLRF/ROW_NUMBER.htm#SQLRF06100>.

[ORACLE-ROWNUM]
Oracle Corporation, "Database SQL Language Reference, SELECT, ROWNUM Pseudocolumn", March 2016, <http://docs.oracle.com/database/122/SQLRF/ROWNUM-Pseudocolumn.htm#SQLRF00255>.

[ORACLE-SORT]
Oracle Corporation, "Database SQL Language Reference, SELECT, Order by clause", March 2016, <http://docs.oracle.com/database/122/SQLRF/SELECT.htm>.

[POSTGRES-COUNT]
postgresql.org, "PostgresSQL, Aggregate Functions", September 2016, <https://www.postgresql.org/docs/9.6/static/functions-aggregate.html>.

[POSTGRES-LIMIT]
postgresql.org, "PostgresSQL, LIMIT and OFFSET", September 2016, <https://www.postgresql.org/docs/9.6/static/queries-limit.html>.

[POSTGRES-SORT]
postgresql.org, "PostgresSQL, Sorting Rows", September 2016, <https://www.postgresql.org/docs/9.6/static/queries-order.html>.

[REST] Fredrich, T., "RESTful Service Best Practices, Recommendations for Creating Web Services", April 2012, <http://www.restapitutorial.com/media/RESTful_Best_Practices-v1_1.pdf>.

[RFC6901] Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Pointer", RFC 6901, DOI 10.17487/RFC6901, April 2013, <https://www.rfc-editor.org/info/rfc6901>.

[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>.

[SEEK] EverSQL.com, "Faster Pagination in Mysql - Why Order By With Limit and Offset is Slow?", July 2017, <https://www.eversql.com/faster-pagination-in-mysql-why-order-by-with-limit-and-offset-is-slow/>.

[W3C.CR-xpath-31-20161213]
Robie, J., Dyck, M., and J. Spiegel, "XML Path Language (XPath) 3.1", World Wide Web Consortium CR CR-xpath-31-20161213, December 2016, <https://www.w3.org/TR/2016/CR-xpath-31-20161213>.

Appendix A. Change Log

00: Initial version.
01: Added the paragraph "Considerations about Paging Implementation" to "Implementation Considerations" section. Added "Implementation Status" section. Added acknowledgements. Renamed the property reporting the paging links.
02: Corrected the value of "title" field in "paging_links" property. Updated references to RFC5988 (obsoleted by RFC 8288) and RFC7159 (obsoleted by RFC 8259). Revised some sentences.
03: Added the paragraph "Google Registry" to "Implementation Status" section.

04: Rearranged the information about pagination included in RDAP responses. Added the section "Paging Metadata". Replaced the wrong reference to RFC 5266 with the correct reference to RFC 5226.
05: Renamed "sortby" parameter in "sort". Removed "country" from the list of sorting properties. Added "sorting_level_0" into the "rdapConformance" array. Changed the title of section "Paging Metadata" in "Sorting and Paging Metadata". Changed the "IANA Considerations" section. Added "Representing Sorting Links" section. Changed the name of some sorting properties to be compliant with EPP.

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

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

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