Registration Protocols Extensions M. Loffredo
Internet-Draft M. Martinelli
Intended status: Standards Track IIT-CNR/Registro.it
Expires: October 19, 2020 S. Hollenbeck
Verisign Labs
April 17, 2020

Registration Data Access Protocol (RDAP) Query Parameters for Result Sorting and Paging
draft-ietf-regext-rdap-sorting-and-paging-12

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 large responses. This document describes RDAP query extensions that allow clients to specify their preferences for sorting and paging result sets.

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:

- o reducing the server response bandwidth requirements;
- o improving server response time;
- o improving query precision and, consequently, obtaining more reliable results;

o decreasing server query processing load;
o 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 the 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 the use of query parameters.

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

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

While the lookup query does not raise issues in the response management, the search query can potentially generate a large result set that could be truncated according to the server limits. 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 might traverse the whole result set to find the relevant objects or, due to truncation, could not find them at all.

The specification described in this document 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 the new 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:

- o "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);

- o "sort": a string value that allows a client to request a specific sort order for the result set;

- o "cursor": a string value representing a pointer to a specific fixed size portion of the 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 as self-descriptive as possible.

Therefore, servers implementing the query parameters described in this specification SHOULD provide additional information in their responses about both the available sorting criteria and the possible pagination. Such information is collected in two OPTIONAL response elements named, respectively, "sorting_metadata" and "paging_metadata".

The "sorting_metadata" element contains the following properties:

- o "currentSort": "String" (OPTIONAL) either the value of sort "parameter" as specified in the query string or the sort applied by default, if any;

- o "availableSorts": "AvailableSort[]" (OPTIONAL) an array of objects each one describing an alternate available sorting criterion. Members are:

  * "property": "String" (REQUIRED) the name that can be used by the client to request the sorting criterion;
  * "default": "Boolean" (REQUIRED) whether the sorting criterion is applied by default;
  * "jsonPath": "String" (OPTIONAL) the JSONPath of the RDAP field corresponding to the property;
  * "links": "Link[]" (OPTIONAL) an array of links as described in RFC 8288 [RFC8288] containing the query string that applies the sorting criterion.

At least one between "currentSort" and "availableSorts" MUST be present.

The "paging_metadata" element contains the following fields:

- o "totalCount": "Numeric" (OPTIONAL) a numeric value representing the total number of objects found. It MUST be provided if the query string contains the "count" parameter;

- o "pageSize": "Numeric" (OPTIONAL) a numeric value representing the number of objects returned in the current page. It MUST be provided when the total number of objects exceeds the page size. This property is redundant for clients because the page size can be derived from the length of the search results array but it can be helpful if the end user interacts with the server through a web browser;

- o "pageNumber": "Numeric" (OPTIONAL) a numeric value representing the number of the current page in the result set. It MUST be provided when the total number of objects found exceeds the page size;

- o "links": "Link[]" (OPTIONAL) an array of links as described in RFC 8288 [RFC8288] containing the reference to the next page. In this specification, only the forward pagination is dealt because it is considered satisfactory in order to traverse the result set. Examples of additional references are to: the previous page, the first page, the last page.

2.1.1. RDAP Conformance

Servers returning the "paging_metadata" element in their response MUST include "paging" in the rdapConformance array as well as servers returning the "sorting_metadata" element MUST include "sorting".

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 objects matching the 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=" ( trueValue / falseValue )
trueValue = ("true" / "yes" / "1")
falseValue = ("false" / "no" / "0")
```

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

```
{
  "rdapConformance": [
        "rdap_level_0",
        "paging"
  ],
  ...
  "paging_metadata": {
    "totalCount": 43
  },
  "domainSearchResults": [
    ...
  ]
}
```


Figure 2: Example of RDAP response with "paging_metadata" element 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 features provided by the RDAP server could help avoid truncation of relevant results.

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=" sortItem *( "," sortItem )
sortItem = property-ref [":" ( "a" / "d" ) ]
property-ref = ALPHA *( ALPHA / DIGIT / "_" )
```

"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).

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

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

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

Figure 3: Examples of RDAP query reporting the "sort" parameter

With the only exception of the sort on IP addresses, servers MUST implement sorting according to the JSON value type of the RDAP field the sorting property refers to: JSON strings MUST be sorted lexicographically and JSON numbers MUST be sorted numerically. Even if IP addresses are represented as JSON strings, they MUST be sorted based on their numeric conversion.

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

2.3.1. Sorting Properties Declaration

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 JSONPath. The JSONPath in a JSON document [RFC8259] is equivalent to the XPath [W3C.CR-xpath-31-20161213] in a XML document. For example, the JSONPath to select the value of the ASCII name inside an RDAP domain object is "$.ldhName", whereby $ identifies the root of the document (DOM). Another way to select a value inside a JSON document is the JSON Pointer [RFC6901]. While JSONPath 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 JSONPath expression of the registration date in an 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 properties an RDAP server MAY implement is presented. The properties are divided into 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 a list of possible sorting properties, grouped by objects, is shown:

  * Domain: name
  * Nameserver: name, ipV4, ipV6.
  * Entity: fn, handle, org, email, voice, country, cc, city.

The correspondence between the sorting properties and the RDAP fields is shown in Table 1:

Object class	Sorting property	RDAP property	RFC 7483	RFC 6350	RFC 8605
Searchable objects	Common properties	eventAction values suffixed by "Date"	4.5.		
Domain	name	unicodeName/ldhName	5.3.		
Nameserver	name	unicodeName/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	
	cc	country code in vcard adr	5.1.		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 some cases:

- since the response to a search on either domains or nameservers might include both A-labels and U-labels ([RFC5890]) in general, a consistent sorting policy shall take unicodeName and ldhName as two formats of the same value rather than separately. Therefore, the unicodeName value MUST be taken while sorting, when unicodeName is missing, the value of ldhName MUST be considered instead;

- the jCard "sort-as" parameter MUST be ignored for the purpose of the sorting capability as described in this document;

- 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.

When more than one address per IP version is reported, sorting MUST be applied to the first value;

- o multiple events with a given action on an object might be returned. When it occurs, sorting MUST be applied to the most recent event;

- o 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 RFC 6350 [RFC6350]. When more than one value is reported, sorting MUST be applied to the preferred value identified by the parameter pref="1". If the pref parameter is missing, sorting MUST be applied to the first value.

Each RDAP provider MAY define other sorting properties than those shown in this document as well as it MAY map those sorting properties onto different locations.

The "jsonPath" field in the "sorting_metadata" element is used to clarify the RDAP field the sorting property refers to. The mapping between the sorting properties and the JSONPaths of the RDAP fields is shown in Table 2. The JSONPaths are provided according to the Goessner v.0.8.0 specification ([GOESSNER-JSON-PATH]). Further documentation about JSONPath operators used in Table 2 is included in Appendix A.

Object class	Sorting property	JSONPath
Searchable objects	registrationDate	$.domainSearchResults[*].events[?(@.eventAction=="registration")].eventDate
	reregistrationDate	$.domainSearchResults[*].events[?(@.eventAction=="reregistration")].eventDate
	lastChangedDate	$.domainSearchResults[*].events[?(@.eventAction=="last changed")].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[?(@.eventAc

		tion=="locked")].eventDate
	unlockedDate	$.domainSearchResults[*].events[?(@.eventAction=="unlocked")].eventDate
Domain	name	$.domainSearchResults[*].unicodeName
Nameserver	name	$.nameserverSearchResults[*].unicodeName
	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]
	cc	$.entitySearchResults[*].vcardArray[1][?(@[0]=="adr")][1].cc
	city	$.entitySearchResults[*].vcardArray[1][?(@[0]=="adr")][3][3]

Table 2: Sorting properties - JSONPath Mapping

Note about the JSONPaths of Table 2 that:

o those related to the event dates are defined only for the "domain" object. To obtain the equivalent JSONPaths for "entity" and "nameserver", the path segment "domainSearchResults" must be replaced with "entitySearchResults" and "nameserverSearchResults" respectively;

o those related to vCard elements are specified without taking into account the "pref" parameter. Servers always applying sorting to those values identified by the pref parameter SHOULD update a JSONPath by adding an appropriate filter. For example, if the email values identified by pref="1" are considered for sorting, the JSONPath of the "email" sorting property should be:

```
$.entitySearchResults[*].vcardArray[1][?(@[0]=="email" &&
@[1].pref=="1")][3]
```

2.3.2. Representing Sorting Links

An RDAP server MAY use the "links" array of the "sorting_metadata" element 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"
  ],
  ...
  "sorting_metadata": {
     "currentSort": "name",
     "availableSorts": [
       {
       "property": "registrationDate",
       "jsonPath": "$.domainSearchResults[*]
          .events[?(@.eventAction==\"registration\")].eventDate",
       "default": false,
       "links": [
         {
         "value": "https://example.com/rdap/domains?name=*nr.com
                   &sort=name",
         "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=name",
         "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. "cursor" Parameter

The cursor parameter defined in this specification can be used to encode information about any pagination method. For example, in the case of a simple implementation of the cursor parameter to represent offset pagination information, the cursor value "b2Zmc2V0PTEwMCxsaW1pdD01MAo=" is the mere Base64 encoding of "offset=100,limit=50". Likewise, in a simple implementation to represent keyset pagination information, the cursor value "a2V5PXRoZWxhc3Rkb21haW5vZnRoZXBhZ2UuY29t=" represents the mere Base64 encoding of "key=thelastdomainofthepage.com" whereby the key value identifies the last row of the current page.

This solution lets RDAP providers to implement a pagination method according to their needs, the user access levels, the submitted queries. In addition, servers can change the method over time without announcing anything to the clients. The considerations that has led to this solution are reported in more detail in Appendix B.

The ABNF syntax of the cursor paramter is the following:

```
cursor = "cursor=" 1*( ALPHA / DIGIT / "/" / "=" / "-" / "_" )
```

```
https://example.com/rdap/domains?name=*nr.com
&cursor=wJlCDLIl6KTWypN7T6vc6nWEmEYe99Hjf1XY1xmqV-M=
```

Figure 5: An example of RDAP query reporting the "cursor" parameter

2.4.1. Representing Paging Links

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

```
{
  "rdapConformance": [
    "rdap_level_0",
    "paging"
  ],
  ...
  "notices": [
    {
      "title": "Search query limits",
      "type": "result set truncated due to excessive load",
      "description": [
      "search results for domains are limited to 50"
      ]
    }
  ],
  "paging_metadata": {
    "totalCount": 73,
    "pageSize": 50,
    "pageNumber": 1,
    "links": [
      {
      "value": "https://example.com/rdap/domains?name=*nr.com",
      "rel": "next",
      "href": "https://example.com/rdap/domains?name=*nr.com
              &cursor=wJlCDLIl6KTWypN7T6vc6nWEmEYe99Hjf1XY1xmqV-M=",
      "title": "Result Pagination Link",
      "type": "application/rdap+json"
      }
    ]
  },
  "domainSearchResults": [
    ...
  ]
}
```


Figure 6: Example of a "paging_metadata" instance to implement cursor pagination

3. Negative Answers

The value constraints for the parameters are defined by their ABNF syntax. Therefore, each request including an invalid value for a parameter SHOULD obtain an HTTP 400 (Bad Request) response code. The same response SHOULD be returned in the following cases:

- if in both single and multi sort the client provides an unsupported value for the "sort" parameter as well as a value related to an object property not included in the response;

o if the client submits an invalid value for the "cursor" parameter.

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

4. 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.

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. 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 .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

5.2. APNIC

Responsible Organization: Asia-Pacific Network Information Centre
Location: https://github.com/APNIC-net/rdap-rmp-demo/tree/sorting-and-paging
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 the specification except for nameserver sorting and unicodeName sorting.
Contact Information: Tom Harrison, tomh@apnic.net

6. IANA Considerations

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

Extension identifier: paging
Registry operator: Any
Published specification: This document.
Contact: IESG <iesg@ietf.org>
Intended usage: This extension describes a best practice for result set paging.

Extension identifier: sorting
Registry operator: Any
Published specification: This document.
Contact: IESG <iesg@ietf.org>
Intended usage: This extension describes a best practice for result set sorting.

7. Security Considerations

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

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. This risk can be mitigated by either restricting search functionality or 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, returns 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 "cursor" parameter enables the user to scroll the result set by submitting a sequence of sustainable queries according to the server limits.

8. Acknowledgements

The authors would like to acknowledge Brian Mountford, Tom Harrison, Karl Heinz Wolf and Jasdip Singh for their contribution to the development of 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>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC8605] Hollenbeck, S. and R. Carney, "vCard Format Extensions: ICANN Extensions for the Registration Data Access Protocol (RDAP)", RFC 8605, DOI 10.17487/RFC8605, May 2019, <https://www.rfc-editor.org/info/rfc8605>.

[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>.

9.2. 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/>.

[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>.

[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>.

[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/>.

Appendix A. JSONPath operators

A JSONPath expression represents a path to find an element (or a set of elements) in a JSON content.

The base JSONPath specification requires that implementations support a set of "basic operators". These operators are used to access the elements of a JSON structure like objects and arrays, and their subelements, respectively, object members and array items. No operations are defined for retrieving parent or sibling elements of a given element. The root element is always referred to as $ regardless if it is an object or array.

Additionally, the specification permits implementations to support arbitrary script expressions: these can be used to index into an object or an array, or to filter elements from an array. While script expression behaviour is implementation-defined, most implementations support the basic relational and logical operators, as well as both object member and array item access, sufficiently similarly for the purposes of this document. Commonly-supported operators/functions divided into "top-level operators" and "filter operators" are documented in Table 3 and Table 4 respectively.

Operator	Descritpion
$	Root element
.<name>	Object member access (dot-notation)
['<name>']	Object member access (bracket-notation)
[<number>]	Array item access
*	All elements within the specified scope
[?(<expression>)]	Filter expression

Table 3: JSONPath Top-Level Operators

Operator	Descritpion
@	Current element being processed
.<name>	Object member access
[<number>]	Array item access
==	Left is equal to right
!=	Left is not equal to right
<	Left is less than right
<=	Left is less than or equal to right
>	Left is greater than right
>=	Left is greater than or equal to right
&&	Logical conjunction
||	Logical disjunction

Table 4: JSONPath Filter Operators

Appendix B. Approaches to Result Pagination

An RDAP query could return a response with hundreds, even thousands, of objects, especially when partial matching is used. For that reason, the cursor parameter addressing result pagination is defined to make responses easier to handle.

Presently, the most popular methods to implement pagination in REST API are: offset pagination and keyset pagination. Both two pagination methods don't require the server to handle the result set in a storage area across the requests since a new result set is generated each time a request is submitted. Therefore, they are preferred in comparison to any other method requiring the management of a REST session.

Using limit and offset operators represents the traditionally used method to implement results pagination. Both of them can be used individually:

- o "limit": means that the server must return the first N objects of the result set;

- o "offset": means that the server must skip the first N objects and must return objects starting from position N+1.

When limit and offset are used together, they allow to identify a specific portion of the result set. For example, the pair "offset=100,limit=50" returns first 50 objects starting from position 101 of the result set.

Despite its easiness of implementation, offset pagination raises some well known drawbacks:

- o when offset has a very high value, scrolling the result set could take some time;

- o it always requires to fetch all the rows before dropping as many rows as specified by offset;

- o it may return inconsistent pages when data are frequently updated (i.e. real-time data) but this doesn't seem the case of registration data.

The keyset pagination [SEEK] consists in adding a query condition that enables the selection of the only data not yet returned. 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 is an opaque URL-safe string representing a logical pointer to the first result of the next page (Figure 5).

Nevertheless, even keyset pagination can be troublesome:

- o it needs at least one key field;

- o it does not allow to sort just by any field because the sorting criterion must contain a key;

- o it works best with full composite values support by DBMS (i.e. [x,y]>[a,b]), emulation is possible but ugly and less performant;

- o it does not allow to directly navigate to arbitrary pages because the result set must be scrolled in sequential order starting from the initial page;

- o implementing the bi-directional navigation is tedious because all comparison and sort operations have to be reversed.

B.1. Specific Issues Raised by RDAP

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

- o an RDAP object is a conceptual aggregation of information generally collected from more than one data structure (e.g. table) and this makes even harder for the developers the implementation of the keyset pagination that is already quite difficult. 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 (i.e. arrays) and, therefore, data selection might be a time consuming process. This situation occurs even though the selection is supported by 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.

Ultimately, both pagination methods have benefits and drawbacks.

Appendix C. Change Log

00: Initial working group version ported from draft-loffredo-regext-rdap-sorting-and-paging-05

01: Removed both "offset" and "nextOffset" to keep "paging_metadata" consistent between the pagination methods. Renamed "Considerations about Paging Implementation" section in ""cursor" Parameter". Removed "FOR DISCUSSION" items. Provided a more detailed description of both "sorting_metadata" and "paging_metadata" objects.

02: Removed both "offset" and "limit" parameters. Added ABNF syntax of cursor parameter. Rearranged the layout of some sections. Removed some items from "Informative References" section. Changed "IANA Considerations" section.

03: Added "cc" to the list of sorting properties in "Sorting Properties Declaration" section. Added RFC8605 to the list of "Informative References".

04: Replaced "ldhName" with "name" in the "Sorting Properties Declaration" section. Clarified the sorting logic with respect to the JSON value types and the sorting policy for multivalued fields.

05: Clarified the logic of sorting on IP addresses. Clarified the mapping between the sorting properties and the RDAP fields. Updated "Acknowledgements" section.

06: Renamed "pageCount" to "pageSize" and added "pageNumber" in the "paging_metadata" object.

07: Added "Paging Responses to POST Requests" section.

08: Added "Approaches to Result Pagination" section to appendix. Added the case of requesting a sort on a property not included in the response to the errors listed in the "Negative Answers" section.

09: Updated the "Implementation Status" section to include APNIC implementation. Moved the "RDAP Conformance" section up in the document. Removed the "Paging Responses to POST Requests" section. Updated the "Acknowledgements" section. Removed unused references. In the "Sorting Properties Declaration" section:

* clarified the logic of sorting on events;
* corrected the JSONPath of the "lastChanged" sorting property;
* provided a JSONPath example taking into account the vCard "pref" parameter.

10: Corrected the JSONPaths of both "fn" and "org" sorting properties in Table 2. Corrected JSON content in Figure 4. Moved [W3C.CR-xpath-31-20161213] and [RFC7942] to the "Normative References". Changed the rdapConformance tags "sorting_level_0" and "paging_level_0" to "sorting" and "paging" respectively.

11: Added the "JSONPath operators" section to appendix.

12: Changed the content of "JSONPath operators" section.

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/