IETF 103, Bangkok, November 6, 2018

RDAP
sorting-and-paging and reverse-search drafts

Request for RegExt WG adoption

Mario Loffredo

IIT-CNR/.it Registry

Current versions

- **draft-loffredo-regext-rdap-sorting-and-paging-05**

  Loffredo, M., Martinelli, M., and S. Hollenbeck, "*Registration Data Access Protocol (RDAP) Query Parameters for Result Sorting and Paging*", September 2018

- **draft-loffredo-regext-rdap-reverse-search-03**

  Loffredo, M., and M. Martinelli, "*Registration Data Access Protocol (RDAP) Reverse Search*", October 2018

sorting-and-paging: reasons

- A search query can return a large result set that can be truncated due to server limits

- RDAP lacks of capabilities for:
  - restricting the result set by search refinement
  - returning the total number of the objects found in order to evaluate the accuracy of the query
  - specifying possible sort criteria:
    - to have the most relevant objects at the beginning of the result set
    - to avoid the truncation of relevant results
  - scrolling the result set when it is truncated

sorting-and-paging: proposal

- New parameters:
  - **count**: allows the user to obtain the total number of results
  - **sort**: allows the user to sort the results
  - **limit** & **offset**: allow the user to scroll the results

- New properties:
  - **sorting_metadata:** includes information about both current and available sort criteria;
  - **paging_metadata:** includes the total number of results and paging information.

- RDAP conformance
  - *sorting_level_0*
  - *paging_level_0*

- Alternative to offset
  - **cursor**: an opaque string representing a pointer to the first result of the next page

sorting-and-paging: offset vs. cursor

	Offset	Cursor
Pros	Is natively supported	Scales well
	Provides maximum flexibility	
Cons	Does not scale well over 100,000 records	Is difficult to implement
	Is not suitable for real-time data	Is not flexible
		Could be considered impractical

sorting_metadata: sample

```
{
  "rdapConformance": [ "rdap_level_0", "sorting_level_0" ],
  ...
  "sorting_metadata": {
     "currentSort": "ldhName",
     "availableSorts": [
       {
       "property": "registrationDate",
       "jsonPath": "$.domainSearchResults[*].events[?(@.eventAction==\"registration\")].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": [
    ...
  ]
}
```

paging_metadata: samples

Offset

```
{
  "rdapConformance": [ "rdap_level_0", "paging_level_0" ],
  ...
  "notices": [
    {
      "title": "Search query limits",
      "type": "result set truncated due to excessive load",
      "description": [ "search results 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": [
    ...
  ]
}
```

Cursor

```
{
  "rdapConformance": [ "rdap_level_0", "paging_level_0" ],
  ...
  "notices": [
    {
      "title": "Search query limits",
      "type": "result set truncated due to excessive load",
      "description": [ "search results 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": [
    ...
  ]
}
```

For discussion

- Should the described metadata be part of a more general “metadata” section, including other contents (e.g. rate limits, information about the server, information about the response, etc.)?
- Should the RDAP specification report both offset and cursor parameters and let operators implement pagination, according to their needs, the user access levels, and the submitted queries?

reverse-search: reasons

- Reverse Whois is provided by many web applications
  - users can find domain names starting from the owner details

- Registries already perform reverse searches
  - registrars use out-of-band solutions to obtain the domain names related to other registration objects (e.g. contacts, nameservers)

- Requirements from ICANN
  - ANNEX E of “A Next-Generation Registration Directory Service (RDS)” (June 2014)
    - *Example #3 – Step 2 The RDS User may then perform a Reverse Query on values already known about the subject, searching the RDS for a list of domain names that include given values as:*
      - *Registrant Name/Organization*
      - *Registrant Phone/Alt Phone*
      - *Registrant Postal addresses, or*
      - *Registrant Email/Alt Email*
  - Specification 4 of “Registry Agreement“ (July, 2017)
    - *“1.10.2 Registry Operator will offer partial match capabilities, at least, on the following fields: domain name, contacts and registrant’s name, and contact and registrant’s postal address, including all the sub-fields described in EPP (e.g., street, city, state or province, etc.).”*
    - *“1.10.3 Registry Operator will offer exact-match capabilities, at least, on the following fields: Registrar ID, …”*

reverse-search: possible objections

- Potential privacy risks:
  - ICANN points out that reverse search is allowed:
    - when it is driven by some permissible purposes
    - if it is allowed under certain conditions involving security as well as terms of use
  - RDAP relies on features of other protocol layers to enforce security (RFC 7481)

- Impact on server processing:
  - RDAP already supports searches
  - the impact can be mitigated by implementing other capabilities
    - sorting and paging, partial response, filtering

reverse-search: proposal

- New paths:
  - **domains?entityHandle**=<reverse search pattern>
  - **domains?entityFn**=<reverse search pattern>
  - **domains?entityEmail**=<reverse search pattern>
  - **domains?entityAddr**=<reverse search pattern>

- <reverse search pattern> is a JSON object including two members:
  - **value**: it represents the search pattern to be matched by the corresponding entity property. It can be:
    - for the first three paths, a string;
    - for the fourth path, a JSON object, in turn, containing the information described in Section 2.4 of RFC 5733.
  - **role**: it is a string whose possible values are those detailed in Section 10.2.4 of RFC 7483

reverse-search: samples

```
entityHandle={"value":"CID-40*","role":"registrant"}

entityFn={"value":"Bobby*","role":"registrant"}

entityEmail={"value":"loffredo@example.com","role":"registrant"}

entityAddr={"value":{"cc":"CA","city":"Sydney"},"role":"registrant"}
```

reverse-search: search pattern

- Why should we represent the reverse search pattern as a JSON object ?
  - the entity role is a search parameter
    - reverse search is almost always executed by specifying an entity role
  - a compact notation would be recommendable when dealing with complex queries
    - find domains whose registrant’s email matches “…” AND tech’s address matches {….}
  - JSON is the best candidate to pass objects in a string

- JSON in URLs:
  - JSON contains some characters that are invalid in a URL
  - Solutions:
    - **encoding URL**
    - converting JSON to binary (Base64, CBOR);
    - using a JSON variation that complies with URL specifications (Rison, URLON, JSURL)

For discussion

- Should reverse search be based on other entity properties?
- Should reverse search be extended to other objects?

Thanks for your attention!

Q & A