RDAP sorting-and-paging, partial-response and reverse-search drafts

Mario Loffredo

IIT-CNR/Registro.it

Overview

- **draft-loffredo-regext-rdap-sorting-and-paging-01**

  Loffredo, M., Martinelli, M., and S. Hollenbeck, "*Registration Data Access Protocol (RDAP) Query Parameters for Result Sorting and Paging*", October 2017

- **draft-loffredo-regext-rdap-partial-response-00**

  Loffredo, M. and Martinelli, M., "*Registration Data Access Protocol (RDAP) Partial Response*", October 2017

- **draft-loffredo-regext-rdap-reverse-search-00**

  Loffredo, M. and Martinelli, M., "*Registration Data Access Protocol (RDAP) Reverse Search*", October 2017

sorting-and-paging & partial-response: REST BOP

- REST services should offer capabilities for result filtering, sorting, paging and subsetting in order to:
  - minimize the traffic (requests/responses) on the net
  - speed up the response time
  - improve the precision of the queries and, consequently, obtain more reliable results
  - decrease the load of the server in the query processing
  - spend less CPU time and memory on the client

sorting-and-paging & partial-response: RDAP status

- A search query can return a large result set
- The result set can be truncated according to the server limits
- RDAP lacks of result filtering, sorting and paging capabilities:
  - you cannot restrict the result set by adding further search conditions
  - you cannot obtain, in the response, the total number of the objects found in order to evaluate the query precision
  - you cannot specify possible sort criteria:
    - to have the most relevant objects at the beginning of the result set
    - to avoid the truncation of relevant results
  - you cannot scroll the result set when result set is truncated
- Servers can only provide full responses

sorting-and-paging: proposal

- New parameters:
  - **count**: a boolean value that allows to obtain in the response the number of objects found
  - **sortby**: a string value that allows to specify a sort order for the result set
  - **limit** & **offset**: numeric values that allows to specify what portion of the entire result set must be returned
- New properties:
  - **paging_count:** the number of objects found
  - **paging_links:** a ready-made reference to the next page of the result set
- RDAP conformance
  - Servers returning *paging_links* or *paging_count* properties MUST include "*paging_level_0*" in the *rdapConformance* array
- Alternative to offset
  - **cursor**: an opaque URL-safe string representing a logical pointer to the first result of the next page
    - Proposed by Brian Mountford from Google

sorting-and-paging: samples

- *paging_count* and *paging_links* based on offset:


```
{
"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_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"
    }
],
"paging_count":"73",
"domainSearchResults": [ ... ]
}
```


- *paging_links* based on cursor:


```
"paging_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"
    }
]
```

sorting-and-paging: offset vs. cursor

- Offset pagination
  - is supported natively by major RDBMSs and most popular NoSQL databases
  - provides maximum flexibility
  - does not scale well in case of huge result sets (over 100,000 records)
  - may return inconsistent pages when data are frequently updated
    - but this is not the case of registration data

- Cursor-based pagination (a.k.a. keyset pagination or seek-method)
  - scales well in case of huge result sets
  - is difficult to implement
    - is not natively supported by DBMSs
    - requires at least one key field
    - needs that all comparison and sort operations have to be reversed for backward pagination
    - raises further issues when objects aggregate information from different data structures (e.g. RDAP entities)
  - is not flexible
    - does not allow to sort by any field and paginate the results
    - does not allow to skip pages
  - could be considered impractical
    - time needed to build the current page could be much higher than the scrolling time
    - will my RDAP server usually deal with huge result sets?

partial-response: concepts

- Instead of returning responses with all data fields, only a subset is returned

- Two approaches:
  - fields:
    - is used by leading REST API providers (e.g. LinkedIn, Google, Facebook)
    - the client declares explicitly the data fields to obtain in the response
  - field set:
    - is used in digital libraries and bibliographic catalogues
    - the client declares a name identifying a server pre-defined set of data fields

partial-response: fields vs. field set

- fields:
  - provides maximum flexibility
    - clients can specify only the fields they need
  - is not easy to implement
    - fields have to be declared according to a given syntax
    - arrays and deep nested objects may complicate both syntax definition and server processing of the query
  - does not facilitate interoperability
    - clients should perfectly know the structure of returned objects to declare valid list of fields
  - raises additional issues according to server authorizations
    - clients could request unauthorized fields and servers should define a strategy for providing a response: to return an error or to return a response ignoring the unauthorized fields
- field set:
  - is less flexible
    - but, do RDAP users really need maximum flexibility?
  - can be easily implemented
  - facilitates interoperability
    - servers can define some basic field sets which, if known, increase the probability to get valid responses
  - fits better server authorizations
    - the list of fields for each set (except “id”) can be different according to the access levels
    - some field sets could be available only to some users

partial-response: proposal

- New parameter:
  - **fieldSet:** a string value identifying a pre-defined set of fields
- Required values are:
  - **id**: it contains only the “objectClassName” field and the field identifying the object
    - it can be used when the client wants to obtain a collection of object identifiers
  - **brief**: it contains the fields that can be included in a “short” response
    - it can be used when the client is asking for a set of properties which gives a basic knowledge of each object
  - **full**: it contains all the information the server can provide for a particular object
- Additional considerations:
  - *brief* and *full* field sets SHOULD be defined according to the access levels
  - servers MAY implement additional field sets
  - servers SHOULD also define a default field set

reverse-search: current state

- Reverse Whois is provided by many web applications
  - users can find domain names starting from the owner details
- Registries already perform reverse searches
  - registries adopt out-of-band solutions to provide registrars with domain names related to contacts, nameservers or DNSSEC keys due to:
    - the loss of synchronization between the registrar data and the registry data
    - the need of such data to perform massive EPP updates (e.g. changing the contacts in a list of domains, etc.)

reverse-search: possible objections

- Potential privacy risks:
  - ICANN, in its report about Next-Gen RDSs, points out that reverse Whois is allowed:
    - when it is driven by some permissible purposes
    - if it provides policies to enforce security as well as terms and conditions of use
  - RDAP relies on features available in other layers to provide security services (RFC 7481)
- Impact on server processing:
  - RDAP already supports searches
  - the impact of both standard and reverse searches can be mitigated by servers adopting ad hoc policies
    - sorting-and-paging & partial-response

reverse-search: proposal

- New paths:
  - **domains?entityHandle**=<entity handle search pattern>
  - **domains?entityFn**=<entity name search pattern>
    - Search patterns are the same as specified in section 3.2.3 of RFC 7482
- New parameter:
  - **entityRole**: a string value identifying a specific entity role to restrict results
    - Values are those detailed in section 10.2.4 of RFC 7483 (registrant, registrar, technical, etc.)
- In RDAP, entities are in relationship with all searchable objects
  - Evaluate the extension to the other paths (e.g. nameservers, entities)

Security considerations

- **sorting-and-paging & partial-response**
  - Search query requires more server resources than lookup query
    - this increases the risk of server resource exhaustion and subsequent denial of service due to abuse
  - Risks can be mitigated by:
    - restricting search functionality
    - limiting the rate of search requests
    - truncating the results in the response
    - providing partial responses
  - Truncation can result in a higher inefficiency if servers cannot:
    - return the truncated results
    - provide the most relevant results at the beginning of the result set
  - The capabilities presented in these drafts support security without reducing efficiency

- **reverse-search**
  - RDAP servers could provide reverse-search capabilities only to restricted communities
  - Two possible scenarios are:
    - servers provide reverse search only for registrars searching for their own domains
    - prevent unauthorized users to start a reverse search from a registrant detail

For discussion

- **sorting-and-paging**
  - How should sorting properties be defined? Is an IANA registry appropriate?
  - How might new parameters work without the use of an RDBMS? Would a server need to maintain state information across queries? If so, what are the implications?
  - 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?
- **reverse-search**
  - Should reverse search be based on other entity details like email, phone, country (code or name), city?
  - Should reverse search be extended to the other types of searches?

Thanks for your attention!

Q & A