RDAP implementation experience at .it

Mario Loffredo, Maurizio Martinelli
IIT-CNR/.it Registry
mario.loffredo, maurizio.martinelli@iit.cnr.it
RDAP applications:

- Validator
- Crawler
- Server
- Client

Future activities
RDAP validator

- Verifies the response compliance with both RDAP and jCard specifications

- Based on JSON Schema draft-07
  - https://json-schema.org/

- Developed in Java
  - https://github.com/everit-org/json-schema

- Takes in consideration so many RFCs and standards:
  - RDAP: 7480, 7481, 7482, 7483, 7484, 8056
  - jCard: 6350, 6473, 6474, 6715, 6969, 7095, 8605
  - And then: ISO.3166.1988, ISO.8601.2000, ISO.8601.2004, CCITT.X520.1988, 3282, 3339, 3986, 4034, 5396, 5545, 5646, 5910, 5952, 5980, 5988, ...
Based on the RDAP validator

Checks the responses from the servers included in IANA Bootstrap Service Registries

Validation in three steps:
• Parsing
• Validation against the standard profile
• Validation against the gTLD profile (in progress)
  • RDAP Technical Implementation Guide
  • RDAP Response Profile
So far the following issues have been discovered:

- **about jCard:**
  - required fn element is not returned
  - only the version element is returned
  - tel element including uri type returns an invalid URI value
  - address returned as the value of the label parameter in adr element but the adr value is null instead of an array of empty strings (i.e. ['"", ...])
  - lang element value returned in uppercase instead lowercase
  - country code parameter (RFC8605) named “CC” instead of “cc”
  - kind element value is “organization” instead of “org”

- **about the standard profile:**
  - coded values (e.g. role, status, event action) are unregistered
  - errorCode in error response is returned as String instead of Number
  - IP network start/endAddress is formatted as a network instead as an address
  - rdapConformance is missing
  - server sets Content-type to “text” instead of “application/rdap+json”
• **about the gTLD Profile:**
  • IANA Registrar ID is unregistered
  • domain registrar abuse contact is missing
  • some coded values are misspelled (e.g. domain status notice and RDDS Inaccuracy notice)

• **general:**
  • server doesn’t return an answer
  • server doesn’t return a valid content
RDAP server (1)

- A challenging mapping between .it data model and RDAP data model has been required
- Only authenticated users are allowed to submit search queries
- Different contents according to users’ profile
- Bootstrapping support
- Based on .it public test environment registration data
- Available at [https://rdap.pubtest.nic.it](https://rdap.pubtest.nic.it)
Several extensions have been implemented:

- **counting, sorting and paging**
  - draft-ietf-regext-rdap-sorting-and-paging-03

- **partial response**
  - draft-ietf-regext-rdap-partial-response-02

- **reverse search**
  - draft-ietf-regext-rdap-reverse-search-01

- **advanced searching and filtering**

- **new contact representation**
  - draft-stepanek-jscontact-01

- **domain suggestion**

- **specification**

- ...
Counting, sorting and paging

- **New parameters:**
  - `count`: allows the user to obtain the total number of results
  - `sort`: allows the user to sort the results
  - `cursor`: an opaque string representing a pointer to a specific fixed size portion of the result set
    - The pagination information is encoded (e.g. offset/limit, keyset)

- **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`
sorting_metadata: sample

{  
    "rdapConformance": [ "rdap_level_0", "sorting_level_0" ],
    ...
    "sorting_metadata": {
        "currentSort": "ldhName",
        "availableSorts": [  
            {  
                "property": "registrationDate",
                "jsonPath": "$.domainSearchResults[*].events[?(0.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"
                    }  
                ],
            },
            ...
        ],
    },
    "domainSearchResults": [  
        ...
    ]
}

- **REQUIRED:** property
- **OPTIONAL:** currentSort, availableSorts (at least one must be present)
- **RECOMMENDED:** jsonPath, default, links
```json
{
  "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&count=true",
      "rel": "next",
      "href": "https://example.com/rdap/domains?name=*nr.com&cursor=wJ1CDLI16KTWypN7T6vc6nWEmEYe99Hjf1XY1xmqV-M=",
      "title": "Result Pagination Link",
      "type": "application/rdap+json"
    } ]
  },
  "domainSearchResults": [ 
    ...
  ]
}
```

- **OPTIONAL**: `totalCount`, `links` (at least one must be present)
- **RECOMMENDED**: `pageCount`
The client declares a server pre-defined set of data fields instead of declaring explicitly the data fields.

New parameter:
- **fieldSet**: is a string identifying a server pre-defined set of fields

Recommended field sets:
- **id**: contains only the key field (i.e. "handle" or "ldhName")
- **brief**: identifies a set of fields conveying a basic knowledge of each object
- **full**: contains all the information the server can provide for a particular object

**NOTE:**
- Field sets might be provided according to users access levels
- Server **MAY** add any service information (e.g. notices) and implement additional field sets
- Servers **SHOULD** also define a "default" field set

New properties:
- **subsetting_metadata**: includes information about both current and available field sets

RDAP conformance
- **subsetting_level_0**
subsetting_metadata: sample

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

- REQUIRED: name
- OPTIONAL: currentFieldSet, availableFieldSets (at least one must be present)
- RECOMMENDED: description, default, links
```
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**: 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 RFC 5733
- **role**: is a string whose possible values are those detailed in RFC 7483

NOTE: value is REQUIRED, role is OPTIONAL
entityHandle = {"value": "CID-40*", "role": "administrative"}

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

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

entityAddr = {"value": {"cc": "CA"}, "role": "registrar"}
Privacy considerations

- The use of this capability **MUST** be compliant with the rules about privacy protection each RDAP provider is subject to.

- Sensitive registration data **MUST** be protected and accessible for permissible purposes only.

- RDAP servers **MUST** provide reverse search only to those requestors who are authorized according to a lawful basis.

**Scenarios:**

- Registrars searching for their own domains

- Operators in the exercise of an official authority or performing a specific task in the public interest that is set out in law

- Reverse searches only on those contacts that have previously given the explicit consent for publishing and processing their personal data
Advanced searching and filtering

- **New parameters:**
  - **query**: allows the user to submit a complex search
    - Must be used in place of a RDAP search path (e.g. domains?name)
  - **filter**: allows the user to filter the results according to the values of those RDAP properties that are not used as search path segments (e.g. status)
    - Can be used in addition to either a search path or the query path

- **New properties:**
  - **filtering_metadata**: includes information about the available filters

- **RDAP conformance**
  - **filtering_level_0**
query & filter samples

domains?name=we*.it&filter=["registrationDate","ge","2018-01-20"]

domains?name=we*.it&filter="or":[["registrationDate","ge","2018-01-20"],["expirationDate","le","2019-01-20"]]

name=we*.it&filter="not":{"or":[["registrationDate","ge","2018-01-20"],["expirationDate","le","2019-01-20"]]

domains?name=wu*it&filter="transferDate","isnull"

domains?query=[{"name","eq","test-*.it"},{"nsLdhName","eq","wnsl.rtr-dev.com"}]

domains?query=[["name","eq","test-*.it"],
{"entityAddr","eq","value":{"cc":"be"},"role":"registrant"}],
&filter="or":[["registrationDate","ge","2018-01-20"],["expirationDate","le","2019-01-20"]]}
filtering_metadata: sample

```json
{
    "rdapConformance": [ "rdap_level_0", "filtering_level_0" ],
    "filtering_metadata": {
        "availableFilters": [
            {
                "property": "registrationDate",
                "jsonPath": "$.domainSearchResults[*].events[?(@.eventAction=="registration")].eventDate"
            },
            {
                "property": "lastChangedDate",
                "jsonPath": "$.domainSearchResults[*].events[?(@.eventAction=="last changed")].eventDate"
            },
            {
                "property": "expirationDate",
                "jsonPath": "$.domainSearchResults[*].events[?(@.eventAction=="expiration")].eventDate"
            },
            {
                "property": "status",
                "jsonPath": "$.domainSearchResults[*].status"
            }
        ],
        "domainSearchResults": [
            ...
        ]
    }
}
```

- **REQUIRED:** `property`
- **OPTIONAL:** `currentFilter`, `availableFilters` (at least one must be present)
- **RECOMMENDED:** `jsonPath`
New parameter:
- **jscontact**: allows the user to obtain a more efficient contact representation than jCard. Default is jscontact=false

New properties:
- **jscontact**: replaces the vcardArray element

RDAP conformance
- **jscontact_level_0**


```json
{
"rdapConformance": [ "rdap_level_0" ],
...
"vcardArray": [
"vcard",
[
"version": {}, "text": "4.0" ],
[ "fn", {}, "text", "ccTLD '.it' Registry - IIT/CNR" ],
[ "kind", {}, "text", "org" ],
[ "org", {}, "text", "ccTLD '.it' Registry - IIT/CNR" ],
[ "adr",
  { "cc": "it" },
  "text",
  [ "", "", "Via Giuseppe Moruzzi 1", "Pisa", "PI", "56124", "Italy" ]
],
[ "tel", { "type": "voice" }, "uri", "tel:+39.0503139811" ],
[ "email", {}, "text", "hostmaster@nic.it" ]
]
],
...
}
```
"rdapConformance": [ "rdap_level_0", "jscontact_level_0" ],
...
"jscontact": {
  "kind": "org"
  "fullName": "ccTLD '.it' Registry - IIT/CNR",
  "organization": "ccTLD '.it' Registry - IIT/CNR",
  "addresses": [
  {
  "type": "work",
  "fullAddress": "Via Giuseppe Moruzzi 1 Pisa PI 56124 Italy IT ",
  "street": "Via Giuseppe Moruzzi, 1",
  "locality": "Pisa",
  "region": "PI",
  "postcode": "56124",
  "country": "Italy",
  "countryCode": "it"
  }
  ],
  "phones": [
  {
  "type": "work",
  "value": "+39.0503139811"
  }
  ],
  "emails": [
  {
  "type": "work",
  "value": "hostmaster@nic.it"
  }
  ],
...
}
Domain suggestion

- New parameter:
  - `searchtype`: “suggestion”

- NOTE:
  - This search is allowed only for the “domains?name” path
  - The search pattern MUST be a domain name in LDH or U-label format
  - Partial matching is not allowed

- Additional parameters:
  - `language`: one of the values described in RFC 5646. Each RDAP provider can define a default value
  - `maxLength`: the maximum length of the domain without considering TLD suffix. Range [1-63]
  - `useHypens`: if hyphens will appear in resulting domain suggestions. Default is false
  - `useNumbers`: if digits 0-9 will appear in resulting domain suggestions. Default is false
  - `useIdns`: if IDNs will appear in resulting domain suggestions. Default is false
  - `showRegistered`: if registered domains will appear in resulting domain suggestions. Default is false
  - `showCensurable`: if all objectionable domain will be included in the response. Default is false

- Sample:
  
  domains?name=carwash.com&searchtype=suggestion&language=en

- The response is provided according to the “id“ field set
A REST service should provide clients with a machine-processable specification to describe:

- the requests in terms of available paths, parameters and bodies
- the responses in terms of returned properties and values
- the authentication methods

New endpoint:
- `specification`

Bootstrapping is implemented through the method as described in RFC8521 (i.e. `specification/{RDAP-provider-tag}`)

Specifications can be provided according to different REST API specification languages:
- OpenAPI
- RAML
- APIBlueprint
- JSON Schema
- ...

Each specification language has its own:
- format
- media type for its delivery as a REST response
- set of tools covering every phase of the API life cycle (design, build, test, documentation and sharing)
Server:

- provides a machine-processable specification of:
  - the URI templates of non-standard path segments
  - the description and the formal constraints for each property or value extending the response
  - the supported authentication methods

- can announce to clients any change about its capabilities and make it suddenly available

Client:

- can configure itself, according to any server specification and user access level

- enables the user to submit only valid requests

- displays and validates the responses more efficiently

- can adopt open source software dedicated to validation, data parsing, requests handling and user interface generation
{  
"rdapConformance" : [ "rdap_level_0" ]  
"notices" : {  
  "title" : "Server specification",  
  "description" : [ "The list of specifications available for this RDAP server according to different formats"],  
  "links" : [  
    {  
      "value" : "http://example.com/rdap/specification",  
      "rel" : "describedby",  
      "title" : "OpenAPI-JSON",  
      "type" : "application/vnd.oai.openapi+json",  
      "href" : "http://example.com/rdap/specification/openapi.json"  
    },  
    ...  
  ]  
}  
}
RDAP servers:
- can be pretty different in both requests and responses
- can’t provide a machine-processable description of their own features

Current RDAP clients:
- are based on RFC7482
- provide users with fixed capabilities

As a consequence:
- users might waste time submitting requests that can’t be accepted because they are not implemented by the server or because they are not allowed, according to the user access level
- users/clients must know the features of all the servers they interact with
- if a server changes its features, such a change is not immediately recognized by clients and, normally, it requires an additional effort by client implementers
- if the standard response is extended with some additional properties or values, the client can’t provide users with their on-line description
- responses cannot be formally validated according to a specification (as in EPP by using XML schemas)
How about implementing a client able to configure itself according to a server specification?

- It would be based on server “ specification” extension
- Specifications could be automatically converted
- Client UI would be automatically generated

Processing steps:

user selects the target server;

the specification is requested to the server;

if (no specification is available)
    RFC7482 is loaded
else
    if (no specification format is OpenAPI)
        the specification is converted in in OpenAPI;
    the client UI is generated by the Swagger-UI library;

Development still in progress
Future activities

- Moving forward current IETF drafts
- Evaluating the submission of new IETF drafts
- Contributing to fix/replace jCard
- Completing the crawler validation against the RDAP gTLD profile
- Completing the client
- Migrating the server on live environment
thank you