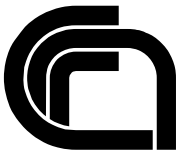

Consiglio Nazionale delle Ricerche

A JSON Schema for RDAP

M. Loffredo, M. Martinelli

IIT TR-04/2018

Technical Report

Marzo 2018

Istituto di Informatica e Telematica

A JSON Schema for RDAP

Loffredo M., Martinelli M. – IIT CNR

Summary

Abstract

This document presents a comprehensive JSON Schema for RDAP. The proposed schema has been used to validate, in some JUnit tests, the responses of the servers listed in IANA RDAP Bootstrap Registries.

1. Introduction

There is a broad consensus that modern REST APIs should be self-descriptive. A REST service should provide clients with a human readable, but most importantly, machine processable specification to explain:

- the requests in terms of available endpoints and query parameters;
- the responses in terms of formats, values, constraints and possible extensions.

RDAP servers can implement different capabilities in both requests and responses, but, currently, they are unable to provide a formal description of their own features. In fact:

- the help query returns a human readable but not machine processable information;
- information included in the IANA Registries about extensions can support the documentation of the responses but it cannot be used for formal validation and online description;
- servers cannot provide a formal information about the RDAP standard and extended capabilities they implement;
- clients cannot configure themselves according to servers features.

As a consequence:

- users might waste time on submitting requests that cannot be accepted because they are not implemented by servers or because they are not allowed according to users access levels;
- clients are recommended to know beforehand the features of the servers;
- any change on servers capabilities results in a change on clients features and, therefore, in an additional work by clients implementers;
- if responses are extended, clients cannot provide their users with an online description of such extensions;
- responses cannot be formally validated according to a schema (as it happens in EPP by using an XML Schema); this could generate situations where incorrect responses, that have not been previously validated in some way, are provided.

JSON Schema is a vocabulary allowing the annotation and validation of JSON documents. Together with JSON Hyper-Schema, it allows to define formally the requests and the related responses of a REST service. A lot of open source software, written in different languages, is available on the web. It involves JSON Schema creation and validation, data parsing, hyper-schema handling and user interface generation.

The advantages of providing an online JSON Schema would be:

- servers could provide a machine readable description of their own features about available queries and responses;
- clients could use such information to configure themselves either to enable users to submit correct requests and to build user interfaces validating the responses.

2. JSON Schema

JSON ([RFC8259]) became very popular because it is lightweight, designed to be language independent and providing an easy way to read and write data between server and client. Anyway, since JSON data don't need a schema to be parsed (JSON is not a document markup language so it is not necessary to define new tags or attributes to represent data), a further validation process should take place to determine whether complex and structured data are valid or not.

JSON Schema is a draft standard ([JSON-SCHEMA], [JSON-SCHEMA-VALIDATION], [JSON-HYPER-SCHEMA], [JSON-SCHEMA-POINTER]) providing a coherent schema through which validating JSON data responses and related REST API requests.

A JSON Schema is written in JSON as well.

3. RDAP

RDAP definition is composed by a set of RFCs each one covering a specific aspect of the protocol. In order to specify a JSON Schema for RDAP, two documents should be considered:

- RFC7482 that reports the query format ([RFC7482]);
- RFC7483 that reports the JSON responses ([RFC7483]).

In particular, the query format represents the fundamentals for the definition of the JSON Hyper-Schemas while the responses should be described according to the JSON Schema rules.

Another factor to consider is that, in RDAP, objects can reference other objects. This can happen in the following cases:

- an object can contain a relationship to another object; for example, a domain object can include a set of related entity objects described in the "*entities*" property;
- RDAP allows two types of queries: the lookup query, returning a single object, and the search query, returning an array of objects. Therefore, a search response includes a set of objects.

Finally, some properties are common to different objects and some properties can appear in different responses. For example, the "*events*" property is common to all the objects because each one can have a set of related events (creation, update, transfer, etc.) and all the responses can contain the "*notices*" property.

4. A JSON Schema for RDAP

At the time of writing this document, a comprehensive JSON Schema for jCard is missing. Since the authors had to create a JSON Schema for RDAP, they had previously to write a schema for jCard. Anyway, jCard is not used only in RDAP so it seemed logical to describe two distinct JSON Schemas, one for jCard and one for RDAP, in two distinct documents. The former is a standalone JSON Schema, while the latter refers the former.

Therefore, in this document, the JSON Schema for jCard (*jcard.json*) will not be reported entirely but it will be referenced only by the JSON Schema for the RDAP entity object (*rdap_entity_object.json*).

The entire JSON Schema for RDAP is composed by the following files:

- ***jcard.json***: contains the definition of the jCard object;
- ***rdap_common.json***: contains the definitions of types and objects referenced by the other schemas;
- ***rdap_help.json***: contains the definitions of the help request and response;

- ***rdap_error_object.json***: contains the definition of the error object;
- ***rdap_error.json***: contains the definition of the error response;
- ***rdap_autnum_object.json***: contains the definition of the autnum object;
- ***rdap_autnum.json***: contains the definitions of autnum lookup request and response;
- ***rdap_ipnetwork_object.json***: contains the definition of the ip network object;
- ***rdap_ipnetwork.json***: contains the definitions of ip network lookup request and response;
- ***rdap_entity_object.json***: contains the definition of the entity object;
- ***rdap_entity.json***: contains the definitions of entity lookup request and response;
- ***rdap_nameserver_object.json***: contains the definition of the nameserver object;
- ***rdap_nameserver.json***: contains the definitions of nameserver lookup request and response;
- ***rdap_domain_object.json***: contains the definitions of the domain object;
- ***rdap_domain:json***: contains the definition of domain lookup request and response;
- ***rdap_entities.json***: contains the definition of entities search request and response;
- ***rdap_nameservers.json***: contains the definition of nameservers search request and response;
- ***rdap_domains.json***: contains the definition of domains search request and response;
- ***rdap_response_common.json***: contains the definition the properties shared by all the responses;
- ***rdap_response.json***: contains the definition of a generic response.

4.1. rdap_common.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_common.json",
  "definitions": {
    "positiveInteger": {
      "type": "integer",
      "minimum": 0
    },
    "uniqueStringArray": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "uniqueItems": true
    },
    "stringArray": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "datetime": {
      "type": "string",
      "format": "date-time"
    },
    "uri": {
      "type": "string",
      "format": "uri"
    },
    "ipv4": {
      "type": "string",
      "format": "ipv4"
    },
    "ipv6": {
      "type": "string",
      "format": "ipv6"
    },
    "ip": {
      "oneOf": [
        {
          "$ref": "#/definitions/ipv4"
        },
```

```
    {
      "$ref": "#/definitions/ipv6"
    }
  ]
},
  "rirBasicType": {
        "type": "string",
        "enum": [
          "DIRECT ALLOCATION",
          "DIRECT ASSIGNMENT",
          "REALLOCATION",
          "REASSIGNMENT",
          "ALLOCATED UNSPECIFIED",
          "ALLOCATED-BY-RIR",
          "ALLOCATED-BY-LIR"
         ]
       },
"rirType": {
    "anyOf":[
      {
      "$ref": "#/definitions/rirBasicType"
        },
        {
         "type" : "string",
         "description": "non standard rir type"
        }
       ]
 },
"ldhName": {
  "type": "string",
  "format": "hostname"
},
"unicodeName": {
  "type": "string",
  "format": "idn-hostname"
},
"domainName": {
  "oneOf": [
    {
      "$ref": "#/definitions/ldhName"
    },
    {
      "$ref": "#/definitions/unicodeName"
    }
  ]
},
"eventAction": {
  "type": "string",
  "enum": [
    "registration",
    "reregistration",
    "last changed",
    "expiration",
    "deletion",
    "reinstantiation",
    "transfer",
    "locked",
    "unlocked",
    "last update of RDAP database",
    "registrar expiration",
    "enum validation expiration"
  ]
},
"noticeType": {
  "type": "string",
  "enum": [
    "result set truncated due to authorization",
    "result set truncated due to excessive load",
    "result set truncated due to unexplainable reasons",
```

```
      "object truncated due to authorization",
      "object truncated due to excessive load",
      "object truncated due to unexplainable reasons"
    ]
  },
  "statusValue": {
    "type": "string",
    "enum": [
      "validated",
      "renew prohibited",
      "update prohibited",
      "transfer prohibited",
      "delete prohibited",
      "client renew prohibited",
      "server renew prohibited",
      "client update prohibited",
      "server update prohibited",
      "client transfer prohibited",
      "server transfer prohibited",
      "client delete prohibited",
      "server delete prohibited",
      "client hold",
      "server hold",
      "proxy",
      "private",
      "removed",
      "obscured",
      "associated",
      "active",
      "inactive",
      "locked",
      "pending create",
      "pending renew",
      "pending transfer",
      "pending update",
      "pending delete",
      "pending restore",
      "auto renew period",
      "redemption period",
      "renew period",
      "transfer period",
      "add period"
    ]
  },
  "status": {
    "type": "array",
    "items": {
      "anyOf":[
        {
            "$ref": "#/definitions/statusValue"
        },
        {
         "type" : "string",
         "description": "non standard status"
        }
       ]
    },
    "uniqueItems": true
  },
  "port43": {
    "oneOf": [
      {
        "$ref": "#/definitions/ip"
      },
      {
        "type": "string",
        "format": "hostname"
      }
    ]
```

```
},
"lang": {
  "type": "string",
  "pattern": "[a-z]{2}(-[A-Z][a-zA-Z]*(\\-[A-Z]{2})?)?"
},
"country": {
  "type": "string",
  "patter": "^[A-Z]{2}$"
},
"link": {
  "type": "object",
  "properties": {
    "value": {
      "$ref": "#/definitions/uri"
    },
    "rel": {
      "type": "string",
      "enum": [
        "about",
        "alternate",
        "appendix",
        "archives",
        "author",
        "blocked-by",
        "bookmark",
        "canonical",
        "chapter",
        "cite-as",
        "collection",
        "contents",
        "convertedFrom",
        "copyright",
        "create-form",
        "current",
        "describedby",
        "describe",
        "disclosure",
        "dns-prefetch",
        "duplicate",
        "edit",
        "edit-form",
        "edit-media",
        "enclosure",
        "first",
        "glossary",
        "help",
        "hosts",
        "hub",
        "icon",
        "index",
        "item",
        "last",
        "latest-version",
        "license",
        "lrdd",
        "memento",
        "monitor",
        "monitor-group",
        "next",
        "next-archive",
        "nofollow",
        "noreferrer",
        "original",
        "payment",
        "pingback",
        "preconnect",
        "predecessor-version",
        "prefecth",
        "preload",
```

```
          "prerender",
          "prev",
          "preview",
          "previous",
          "prev-archive",
          "profile",
          "related",
          "replies",
          "restconf",
          "search",
          "section",
          "self",
          "service",
          "start",
          "stylesheet",
          "subsection",
          "successor-version",
          "tag",
          "terms-of-service",
          "timegate",
          "timemap",
          "type",
          "up",
          "version-history",
          "via",
          "webmention",
          "working-copy",
          "working-copy-of"
        ]
      },
      "href": {
        "$ref": "#/definitions/uri"
      },
      "hreflang": {
        "type": "array",
        "uniqueItems": true,
        "items": {
          "$ref": "#/definitions/lang"
        }
      },
      "title": {
        "type": "string"
      },
      "media": {
        "type": "string"
      },
      "type": {
        "type": "string",
        "pattern": "[a-zA-Z][a-zA-Z0-9]*/[a-zA-Z][a-zA-Z0-9]*"
      }
    }
  },
  "links": {
    "type": "array",
    "items": {
      "$ref": "#/definitions/link"
    }
  },
  "eventWithoutActor": {
    "type": "object",
    "properties": {
      "eventAction": {
      "anyOf":[
        {
        "$ref": "#/definitions/eventAction"
          },
          {
           "type" : "string",
           "description": "non standard event action"
```

```
        }
      ]
    },
    "eventDate": {
      "$ref": "#/definitions/datetime"
    },
    "links": {
      "$ref": "#/definitions/links"
    }
  }
},
"event": {
  "allOf": [
    {
      "$ref": "#/definitions/eventWithoutActor"
    },
    {
      "properties": {
        "eventActor": {
          "type": "string"
        }
      }
    }
  ]
},
"events": {
  "type": "array",
  "items": {
    "$ref": "#/definitions/event"
  }
},
"notice": {
  "type": "object",
  "properties": {
    "title": {
      "type": "string"
    },
    "type": {
          "anyOf":[
            {
            "$ref": "#/definitions/noticeType"
              },
              {
               "type" : "string",
               "description": "non standard notice type"
              }
            ]
    },
    "description": {
      "$ref": "#/definitions/stringArray"
    },
    "links": {
      "$ref": "#/definitions/links"
    }
  }
},
"notices": {
  "type": "array",
  "items": {
    "$ref": "#/definitions/notice"
  }
},
"remarks": {
  "type": "array",
  "items": {
    "$ref": "#/definitions/notice"
  }
},
"publicId": {
```

```
      "type": "object",
      "properties": {
        "type": {
          "type": "string"
        },
        "identifier": {
          "type": "string"
        }
      },
      "required": [
        "type",
        "identifier"
      ]
    },
    "publicIds": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/publicId"
      }
    }
  },
  "type": "object",
  "properties": {
    "handle": {
      "type": "string"
    },
    "status": {
      "$ref": "#/definitions/uniqueStringArray"
    },
    "port43": {
      "$ref": "#/definitions/port43"
    },
    "publicIds": {
      "$ref": "#/definitions/publicIds"
    },
    "events": {
      "$ref": "#/definitions/events"
    },
    "remarks": {
      "$ref": "#/definitions/remarks"
    },
    "links" : {
      "$ref": "#/definitions/links"
    }
  }
}
```

4.2. rdap_help.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_help.json",
    "type": "object",
    "properties": {
        "rdapConformance": {
          "$ref": "rdap_common.json#/definitions/uniqueStringArray"
        },
        "lang" : {
            "$ref": "rdap_common.json#/definitions/lang"
        },
        "notices" : {
           "$ref": "rdap_common.json#/definitions/notices"
        }
    },
  "additionalProperties" : false,
  "links": [
    {
      "description": "Get help",
      "href": "/help",
```

```
      "method": "GET",
      "rel": "self",
      "title": "help",
      "targetSchema" : {"$ref":"#"}
    }
  ]
}
```

4.3. rdap_error_object.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_error_object.json",
      "properties": {
        "errorCode": {
          "$ref": "rdap_common.json#/definitions/positiveInteger"
        },
        "title": {
          "type": "string"
        },
        "description": {
          "$ref": "rdap_common.json#/definitions/stringArray"
        }
      },
         "required": [
           "errorCode"
         ],
          "not": {
           "required": [
             "domainSearchResults",
             "nameserverSearchResults",
             "entitySearchResults"
           ]
         }
}
```

4.4. rdap_error.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_error.json",
  "oneOf": [
          {
         "allOf": [
           {
             "$ref": "rdap_response_common.json#"
           },
           {
             "$ref": "rdap_error_object.json#"
           }
         ]
         },
         {
         "allOf": [
           { "not" : {
                   "$ref": "rdap_response_common.json#"
                }
           },
           {
             "$ref": "rdap_error_object.json#"
           }
           ]
         }
       ]
}
```

4.5. rdap_autnum_object.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_autnum_object.json",
  "definitions" : {
        "autnumClass" : {"const": "autnum" },
      "autnum": {
            "type": "integer",
            "minimum": 0,
            "maximum": 4294967295
          }
        },
        "allOf": [
          {
            "$ref": "rdap_common.json#"
          },
          {
            "properties": {
              "objectClassName": {
                "$ref": "#/definitions/autnumClass"
              },
              "startAutnum": {
                "$ref": "#/definitions/autnum"
              },
              "endAutnum": {
                "$ref": "#/definitions/autnum"
              },
              "name": {
                "type": "string"
              },
              "type": {
                "$ref": "rdap_common.json#/definitions/rirType"
              },
              "country": {
                "$ref": "rdap_common.json#/definitions/country"
              },
              "entities": {
                "type": "array",
                "items": {
                  "$ref": "rdap_entity_object.json#"
                }
              }
            },
              "required": [
                "objectClassName"
              ],
              "not": {
                "required": [
                  "domainSearchResults",
                  "nameserverSearchResults",
                  "entitySearchResults"
                ]
              }
          }
       ]
}
```

4.6. rdap_autnum.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_autnum.json",
  "allOf": [
          {
              "$ref": "rdap_response_common.json#"
          },
              {
```

```
            "$ref": "rdap_autnum_object.json#"
        }
  ],
  "links": [
    {
      "description": "Get autnum",
      "href": "/autnum/{autnum}",
      "hrefSchema": {
        "properties": {
          "autnum": {
            "$ref": "#/definitions/autnum"
          }
        }
      },
      "method": "GET",
      "rel": "self",
      "title": "autnum",
      "targetSchema" : {"$ref":"#"}
    }
  ]
}
```

4.7. rdap_ipnetwork_object.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_ipnetwork_object.json",
  "definitions" : {
     "ipnetworkClass" : {"const": "ip network" }
    },
        "allOf": [
          {
            "$ref": "rdap_common.json#"
          },
          {
            "properties": {
              "objectClassName": {
                "$ref": "#/definitions/ipnetworkClass"
              },
              "stratAddress": {
                "$ref": "rdap_common.json#/definitions/ip"
              },
              "endAddress": {
                "$ref": "rdap_common.json#/definitions/ip"
              },
              "ipVersion": {
                "type": "string",
                "enum": [
                  "v4",
                  "v6"
                ]
              },
              "name": {
                "type": "string"
              },
              "type": {
                "$ref": "rdap_common.json#/definitions/rirType"
              },
              "country": {
                "$ref": "rdap_common.json#/definitions/country"
              },
              "parentHandle": {
                "type": "string"
              },
              "entities": {
                "type": "array",
                "items": {
                  "$ref": "rdap_entity_object.json#"
                }
```

```
            }
          },
             "required": [
               "objectClassName"
             ],
              "not": {
               "required": [
                 "domainSearchResults",
                 "nameserverSearchResults",
                 "entitySearchResults"
               ]
             }
      }
  ]
}
```

4.8. rdap_ipnetwork.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_ipnetwork.json",
  "allOf": [
           {
               "$ref": "rdap_response_common.json#"
           },
              {
              "$ref": "rdap_ipnetwork_object.json#"
         }
  ],
  "links": [
    {
      "description": "Get network by ip",
      "href": "/ip/{ip}",
      "hrefSchema": {
        "properties": {
          "ip": {
            "$ref": "rdap_common.json#/definitions/ip"
          }
        }
      },
      "method": "GET",
      "rel": "self",
      "title": "network by ip"
    },
    {
      "description": "Get network by cidr",
      "href": "/ip/{cidrPrefix}/{cidrLength}",
      "hrefSchema": {
          "oneOf": [
          {
                "properties": {
                      "cidrPrefix": {
                        "$ref": "rdap_common.json#/definitions/ipV4"
                      },
                      "cidrLength": {
                        "type": "integer",
                        "minimum": 1,
                        "maximum": 32
                      }
                  }
            },
           {
                "properties": {
                      "cidrPrefix": {
                        "$ref": "rdap_common.json#/definitions/ipV6"
                      },
                      "cidrLength": {
                        "type": "integer",
                        "minimum": 1,
```

```
                        "maximum": 128
                      }
                    }
                }
              ]
      },
      "method": "GET",
      "rel": "self",
      "title": "network by cidr",
      "targetSchema" : {"$ref":"#"}
    }
  ]
}
```

4.9. rdap_entity_object.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_entity_object.json",
  "definitions": {
   "entityClass" : {"const": "entity" },
    "entityRole": {
      "type": "string",
      "description" : "the name is entityRole instead of simply role to distinguish it
from the role defintiion in jcard",
      "enum": [
        "registrant",
        "technical",
        "administrative",
        "abuse",
        "billing",
        "registrar",
        "reseller",
        "sponsor",
        "proxy",
        "notifications",
        "noc"
      ]
    }
  },
              "allOf": [
                {
                  "$ref": "rdap_common.json#"
                },
                {
                  "properties": {
                    "objectClassName": {
                         "$ref": "#/definitions/entityClass"
                    },
                    "handle": {
                      "type": "string",
                      "description" : "this property overrides the same property in
rdap_common.json"
                    },

                    "vcardArray": {
                      "$ref": "jcard.json#/definitions/vcardArray"
                    },
                    "roles": {
                      "type": "array",
                      "items": {
                        "$ref": "#/definitions/entityRole"
                      }
                    },
                    "entities": {
                      "type": "array",
                      "items": {
                        "$ref": "rdap_entity_object.json#"
                      }
```

```
                    },
                    "asEventActor": {
                      "type": "array",
                      "items": {
                        "$ref": "rdap_common.json#/definitions/eventWithoutActor"
                      }
                    },
                    "publicIds": {
                      "$ref": "rdap_common.json#/definitions/publicIds"
                    },
                    "networks": {
                      "type": "array",
                      "items": {
                        "$ref": "rdap_ipnetwork_object.json#"
                      }
                    },
                    "autnums": {
                      "type": "array",
                      "items": {
                        "$ref": "rdap_autnum_object.json#"
                      }
                    }
                  },
                "required": [
                  "objectClassName", "handle"
                ],
                 "not": {
                  "required": [
                    "domainSearchResults",
                    "nameserverSearchResults",
                    "entitySearchResults"
                  ]
                }
            }
          ]
}
```

4.10. rdap_entity.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_ipnetwork.json",
  "allOf": [
           {
                "$ref": "rdap_response_common.json#"
           },
               {
               "$ref": "rdap_ipnetwork_object.json#"
         }
  ],
  "links": [
    {
      "description": "Get network by ip",
      "href": "/ip/{ip}",
      "hrefSchema": {
        "properties": {
          "ip": {
            "$ref": "rdap_common.json#/definitions/ip"
          }
        }
      },
      "method": "GET",
      "rel": "self",
      "title": "network by ip"
    },
    {
      "description": "Get network by cidr",
      "href": "/ip/{cidrPrefix}/{cidrLength}",
      "hrefSchema": {
```

```
        "properties": {
          "cidrPrefix": {
            "$ref": "rdap_common.json#/definitions/ip"
          },
          "cidrLength": {
            "type": "integer",
            "minimum": 1,
            "maximum": 32
          }
        }
      },
      "method": "GET",
      "rel": "self",
      "title": "network by cidr",
      "targetSchema" : {"$ref":"#"}
    }
  ]
}
```

4.11. rdap_nameserver_object.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_nameserver_object.json",
  "definitions" : {
     "nameserverClass" : {"const": "nameserver" }
    },
        "allOf": [
          {
            "$ref": "rdap_common.json#"
          },
          {
            "properties": {
              "objectClassName": {
                "$ref": "#/definitions/nameserverClass"
               },
              "ldhName": {
                "$ref": "rdap_common.json#/definitions/ldhName"
              },
              "unicodeName": {
                "$ref": "rdap_common.json#/definitions/unicodeName"
              },
              "ipAddresses": {
                "type": "object",
                "properties": {
                  "v6": {
                    "type": "array",
                    "items": {
                      "$ref": "rdap_common.json#/definitions/ipv6"
                    }
                  },
                  "v4": {
                    "type": "array",
                    "items": {
                      "$ref": "rdap_common.json#/definitions/ipv4"
                    }
                  }
                }
              },
              "entities": {
                "type": "array",
                "items": {
                  "$ref": "rdap_entity_object.json#"
                }
              }
            },
              "required": [
                "objectClassName","ldhName"
              ],
```

```
                "not": {
                  "required": [
                    "domainSearchResults",
                    "nameserverSearchResults",
                    "entitySearchResults"
                  ]
                }
            }
        ]
}
```

4.12. rdap_nameserver.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_nameserver.json",
  "allOf": [
           {
                   "$ref": "rdap_response_common.json#"
           },
               {
               "$ref": "rdap_nameserver_object.json#"
         }
  ],
  "links": [
    {
      "description": "Get nameserver",
      "href": "/nameserver/{domainName}",
      "hrefSchema": {
        "properties": {
          "domainName": {
            "$ref": "rdap_common.json#/domainName"
          }
        }
      },
      "method": "GET",
      "rel": "self",
      "title": "nameserver",
      "targetSchema" : {"$ref":"#"}
    }
  ]
}
```

4.13. rdap_domain_object.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_domain_object.json",
  "definitions": {
    "domainClass" : {"const": "domain" },
    "variantRelation": {
      "type": "string",
      "enum": [
        "registered",
        "unregistered",
        "registration restricted",
        "open registration",
        "conjoined"
      ]
    },
    "dnssecAlgorithm": {
      "type": "integer",
      "enum": [
        1,
        2,
        3,
        5,
        6,
```

```
      7,
      8,
      10,
      12,
      13,
      14,
      15,
      16
    ]
  },
  "hexString": {
    "type": "string",
    "pattern": "[0-9a-fA-F]+"
  },
  "dsData": {
    "type": "array",
    "items": {
      "type": "object",
      "properties": {
        "keyTag": {
          "$ref": "rdap_common.json#/definitions/positiveInteger"
        },
        "algorithm": {
          "$ref": "#/definitions/dnssecAlgorithm"
        },
        "digest": {
          "$ref": "#/definitions/hexString"
        },
        "digestType": {
          "type": "integer",
          "enum": [
            1,
            2,
            3,
            4
          ]
        },
        "events": {
          "$ref": "rdap_common.json#/definitions/events"
        },
        "links": {
          "$ref": "rdap_common.json#/definitions/links"
        }
      }
    }
  },
  "keyData": {
    "type": "array",
    "items": {
      "type": "object",
      "properties": {
        "flags": {
          "type": "integer",
          "enum": [
            0,
            256,
            257
          ]
        },
        "protocol": {
          "const": 3
        },
        "publicKey": {
          "type": "string"
        },
        "algorithm": {
          "$ref": "#/definitions/dnssecAlgorithm"
        },
        "events": {
```

```
          "$ref": "rdap_common.json#/definitions/events"
        },
        "links": {
          "$ref": "rdap_common.json#/definitions/links"
        }
      }
    }
  },
  "secureDNS": {
    "type": "object",
    "properties": {
      "zoneSigned": {
        "type": "boolean"
      },
      "delegationSigned": {
        "type": "boolean"
      },
      "maxSigLife": {
        "$ref": "rdap_common.json#/definitions/positiveInteger"
      },
      "dsData": {
        "$ref": "#/definitions/dsData"
      },
      "keyData": {
        "$ref": "#/definitions/keyData"
      }
    }
  },
  "variant": {
    "type": "object",
    "properties": {
      "relation": {
        "type": "array",
        "items": {
              "$ref": "#/definitions/variantRelation"
        }
      },
      "idnTable": {
        "type": "string"
      },
      "variantNames": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "ldhName": {
              "$ref": "rdap_common.json#/definitions/ldhName"
            },
            "unicodeName": {
              "$ref": "rdap_common.json#/definitions/unicodeName"
            }
          }
        }
      }
    }
  }
 },
"allOf": [
        {
    "$ref": "rdap_common.json#"
        },
      {
          "properties": {
            "objectClassName": {
              "$ref": "#/definitions/domainClass"
            },
            "ldhName": {
              "$ref": "rdap_common.json#/definitions/ldhName"
            },
```

```
            "unicodeName": {
              "$ref": "rdap_common.json#/definitions/unicodeName"
            },
            "nameservers": {
              "type": "array",
              "items": {
                "$ref": "rdap_nameserver_object.json#"
              }
            },
            "secureDNS": {
              "$ref": "#/definitions/secureDNS"
            },
            "variants": {
              "type": "array",
              "items": {
                "$ref": "#/definitions/variant"
              }
            },
            "entities": {
              "type": "array",
              "items": {
                "$ref": "rdap_entity_object.json#"
              }
            },
            "network": {
              "$ref": "rdap_ipnetwork_object.json#"
            }
          },
           "required": ["objectClassName","ldhName"],
           "not": {
              "required": [
                "domainSearchResults",
                "nameserverSearchResults",
                "entitySearchResults"
              ]
           }
        }
  ]
}
```

4.14. rdap_domain.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_domain.json",
  "allOf": [
          {
                "$ref": "rdap_response_common.json#"
          },
              {
              "$ref": "rdap_domain_object.json#"
        }
  ],
  "links": [
    {
      "description": "Get domain",
      "href": "/domain/{domainName}",
      "hrefSchema": {
        "properties": {
          "domainName": {
            "$ref": "rdap_common.json#/definitions/domainName"
          }
        }
      },
      "method": "GET",
      "rel": "self",
      "title": "domain",
      "targetSchema" : {"$ref":"#"}
    }
```

```
  ]
}
```

4.15. rdap_entities.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_entities.json",
  "allOf": [
    {
      "$ref": "rdap_response_common.json#"
    },
    {
      "properties": {
        "entitySearchResults": {
          "type": "array",
          "items": {
            "$ref": "rdap_entity_object.json#"
          },
          "minItems" : 1
        }
      },
          "required": [
          "entitySearchResults"
        ]
    }
  ],
  "links": [
    {
      "description": "Get entities by handle",
      "href": "/entities{?handle}",
      "hrefSchema": {
        "properties": {
          "name": {
            "type": "string"
          }
        }
      },
      "method": "GET",
      "rel": "self",
      "title": "entities by handle",
      "targetSchema" : {"$ref":"#"}
    },
    {
      "description": "Get entities by fn",
      "href": "/entities{?fn}",
      "hrefSchema": {
        "properties": {
          "fn": {
            "type": "string"
          }
        }
      },
      "method": "GET",
      "rel": "self",
      "title": "entities by fn",
      "targetSchema" : {"$ref":"#"}
    }
  ]
}
```

4.16. rdap_nameservers.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_nameservers.json",
  "allOf": [
    {
```

```
      "$ref": "rdap_response_common.json#"
    },
    {
      "properties": {
        "nameserverSearchResults": {
          "type": "array",
          "items": {
            "$ref": "rdap_nameserver_object.json#"
          },
          "minItems" : 1
        }
      },
       "required": [
         "nameserverSearchResults"
       ]
    }
  ],
  "links": [
    {
      "description": "Get nameservers by name",
      "href": "/nameservers{?name}",
      "hrefSchema": {
        "properties": {
          "name": {
            "type": "string"
          }
        }
      },
      "method": "GET",
      "rel": "self",
      "title": "nameservers by name",
      "targetSchema" : {"$ref":"#"}
    },
    {
      "description": "Get nameservers by ip",
      "href": "/nameservers{?ip}",
      "hrefSchema": {
        "properties": {
          "ip": {
            "type": "string"
          }
        }
      },
      "method": "GET",
      "rel": "self",
      "title": "nameservers by ip",
      "targetSchema" : {"$ref":"#"}
    }
  ]
}
```

4.17. rdap_domains.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_domains.json",
  "allOf": [
    {
      "$ref": "rdap_response_common.json#"
    },
    {
      "properties": {
        "domainSearchResults": {
          "type": "array",
          "items": {
            "$ref": "rdap_domain_object.json#"
                },
           "minItems" : 1
        }
```

```
      },
           "required": [
           "domainSearchResults"
         ]
    }
  ],
  "links": [
    {
      "description": "Get domains by name",
      "href": "/domains{?name}",
      "hrefSchema": {
        "properties": {
          "name": {
            "type": "string"
          }
        }
      },
      "method": "GET",
      "rel": "self",
      "title": "domains by name",
      "targetSchema" : {"$ref":"#"}
    },
    {
      "description": "Get domains by nsLdhName",
      "href": "/domains{?nsLdhName}",
      "hrefSchema": {
        "properties": {
          "nsLdhName": {
            "type": "string"
          }
        }
      },
      "method": "GET",
      "rel": "self",
      "title": "domains by nsLdhName",
      "targetSchema" : {"$ref":"#"}
    },
    {
      "description": "Get domains by nsIp",
      "href": "/domains{?nsIp}",
      "hrefSchema": {
        "properties": {
          "nsIp": {
            "type": "string"
          }
        }
      },
      "method": "GET",
      "rel": "self",
      "title": "domains by nsIp",
      "targetSchema" : {"$ref":"#"}
    }
  ]
}
```

4.18. rdap_response_common.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_response_common.json",
    "type": "object",
    "properties": {
        "rdapConformance": {
          "$ref": "rdap_common.json#/definitions/uniqueStringArray"
        },
        "lang" : {
             "$ref": "rdap_common.json#/definitions/lang"
        },
        "notices" : {
```

```
            "$ref": "rdap_common.json#/definitions/notices"
          }
        }
}
```

4.19. rdap_response.json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "id" : "rdap_response_common.json",
    "type": "object",
    "properties": {
        "rdapConformance": {
          "$ref": "rdap_common.json#/definitions/uniqueStringArray"
        },
        "lang" : {
            "$ref": "rdap_common.json#/definitions/lang"
        },
        "notices" : {
           "$ref": "rdap_common.json#/definitions/notices"
        }
      }
}
```

5. Validation of URI templates

The validation of the URI templates ([RFC6570]) reported in the JSON-Hyper Schemas has been implemented by using the handy-uri-templates library ([URI-TEMPLATES-VALIDATOR]). In the following, the source code of a Junit 4.0 test about the URIs of domain lookup and search queries is shown.

5.1. Validation test

```
package it.nic.rdap.uri_templates.validator.test;

import org.junit.Test;
import static org.junit.Assert.assertTrue;
import com.damnhandy.uri.template.UriTemplate;

public class DomainValidatorTest {

       @Test
       public void testDomainLookupValidation()  {

              String uri =  UriTemplate.fromTemplate("/{domainName}")
                 .set("domainName", "nic.it")
                 .expand();

              assertTrue("testDomainLookupValidation", uri.equals("/nic.it"));

       }

       @Test
       public void testStandardDomainSearchValidation()  {

              String uri =  UriTemplate.fromTemplate("/domains{?name}")
                 .set("name", "nic.it")
                 .expand();

              assertTrue("testStandardDomainSearchValidation - name",
uri.equals("/domains?name=nic.it"));

              uri =  UriTemplate.fromTemplate("/domains{?nsLdhName}")
                 .set("nsLdhName", "ns1.nic.it")
                 .expand();
```

```
            assertTrue("testStandardDomainSearchValidation - nsLdhName",
uri.equals("/domains?nsLdhName=ns1.nic.it"));

            uri =  UriTemplate.fromTemplate("/domains{?nsIp}")
               .set("nsIp", "192.12.193.108")
               .expand();

            assertTrue("testStandardDomainSearchValidation - nsIp",
uri.equals("/domains?nsIp=192.12.193.108"));

      }
}
```

6. Response validation through JSON Schema

The proposed JSON Schema has been used for the validation of a set of RDAP responses. The validation is performed by using the everit-org library ([JSON-SCHEMA-VALIDATOR]). JSON Schema has currently 4 major releases, Draft 3, Draft 4, Draft 6 and Draft 7. This library implements the 3 newer ones.

The best way to denote the JSON Schema version to be used is to include its meta-schema URL in the document root with the "$schema" key. This is a common notation, facilitated by the library to determine which version should be used:

- if there is "$schema": "http://json-schema.org/draft-04/schema" in the schema root, then Draft 4 will be used
- if there is "$schema": "http://json-schema.org/draft-06/schema" in the schema root, then Draft 6 will be used
- if there is "$schema": "http://json-schema.org/draft-07/schema" in the schema root, then Draft 7 will be used
- if none of these is found then Draft 4 will be assumed as default

6.1. Validation test

The validation tests are grouped according to the different RDAP responses:

- **HelpValidatorTest**: validates the help response;
- **ErroreValidatorTest**: validates the error response in two different formats, the basic error structure and an error response including the *rdapConformance* array and *notices* property;
- **AutnumValidatorTest**: validates the response of a autnum lookup query;
- **IpnetworkValidatorTest**: validates the response of a ip network lookup query;
- **DomainValidatorTest**: validates the responses of both domain lookup and search queries;
- **EntityValidatorTest**: validates the responses of both entity lookup and search queries;
- **NameserverValidatorTest**: validates the responses of nameserver lookup and search queries;
- **InvalidResponseValidatorTest**: validates some cases of wrong responses (for example, the case of a search query response having no items in the search results array)

6.2. Error response validation test

In the following, just to give an example of the implemented tests, the source code of a Junit 4.0 test performing the validation of two error responses is reported.

6.2.1. AbstractValidatorTest class

```
package it.nic.rdap.schema.validator.test;

import java.io.BufferedReader;
```

```java
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.List;
import org.bouncycastle.util.Strings;
import org.everit.json.schema.ValidationException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public abstract class AbstractValidatorTest {

	protected final static Logger log =
LoggerFactory.getLogger(AbstractValidatorTest.class);

	// convert InputStream to String
	protected static String getStringFromInputStream(InputStream is) {

		BufferedReader br = null;
		StringBuilder sb = new StringBuilder();

		String line;
		try {
			br = new BufferedReader(new InputStreamReader(is));
			while ((line = br.readLine()) != null) {
				sb.append(line);
			}
		} catch (IOException e) {
			e.printStackTrace();
		} finally {
			if (br != null) {
				try {
					br.close();
				} catch (IOException e) {
					e.printStackTrace();
				}
			}
		}

		return sb.toString();
	}


	private static String _toDeepString(ValidationException e) {

		if (e == null)
			return null;

		String msg = String.format("%s", e.getMessage());
		List<ValidationException> subs = e.getCausingExceptions();
		if (subs == null)
			return msg;
		for (ValidationException sub : subs) {
			msg+=String.format(";%s",_toDeepString(sub));
		}

		return msg;
	}

	protected static String toDeepString(ValidationException e) {

		if (e == null)
			return null;

		String msg = _toDeepString(e);
		String[] items = Strings.split(msg,';');
		String realMsg = "";
		for (String item : items) {

			if (item.isEmpty() ||
```

```
                        item.contains("schema violations found") ||
                        item.contains("extraneous key")          ||
                        item.contains("subschemas matched instead of") ||
                        item.contains("subschema matches out of") ||
                        item.contains("subject must not be valid against schema") ||
                        item.contains("#/objectClassName:") ||
                        item.contains("required key")
                    )
                        continue;

                if (realMsg.isEmpty())
                        realMsg += item;
                else
                        realMsg += ";" + item;

            }

            return realMsg;
        }
}
```

6.2.2. ErrorValidatorTest class

```
package it.nic.rdap.schema.validator.test;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Paths;
import org.everit.json.schema.Schema;
import org.everit.json.schema.ValidationException;
import org.everit.json.schema.loader.SchemaLoader;
import org.json.JSONObject;
import org.json.JSONTokener;
import org.junit.BeforeClass;
import org.junit.Test;

public class ErrorValidatorTest extends AbstractValidatorTest {

        private static Schema schema = null;

        @BeforeClass
        public static void loadSchema() throws FileNotFoundException {

          File file = new File(Paths.get("").toUri().getPath() + getSchemaDir() +
"/rdap_response.json");
          InputStream inputStream = new FileInputStream(file);
          JSONObject rawSchema = new JSONObject(new JSONTokener(inputStream));
          SchemaLoader schemaLoader = SchemaLoader.builder()
                        .schemaJson(rawSchema)
                        .resolutionScope("file:" +
Paths.get("").toUri().getPath().replaceAll(" ", "%20") + getSchemaDir() + "/")
                        .build();
          schema = schemaLoader.load().build();
        }

        @Test
        public void testError1Validation() throws IOException {

                try {
                  String json =
getStringFromInputStream(getClass().getResourceAsStream("/responses/error1.json"));
                  schema.validate(new JSONObject(json));
                } catch (Exception e) {
                        if (e instanceof ValidationException) {
                                System.out.println(toDeepString((ValidationException) e));
```

```
                        }
                        throw e;
                  }
            }


            @Test
            public void testError2Validation() throws IOException {

                  try {
                    String json =
getStringFromInputStream(getClass().getResourceAsStream("/responses/error2.json"));
                    schema.validate(new JSONObject(json));
                  } catch (Exception e) {
                        if (e instanceof ValidationException) {
                              log.info(toDeepString((ValidationException) e));
                        }
                        throw e;
                  }
            }
}
```

6.2.3. Basic error structure (error1.json)

```
{

    "errorCode": 404,
    "title": "Domain bbhbbbbhhb.it not found"

}
```

6.2.4. Extended error structure (error2.json)

```
{
  "rdapConformance": [
    "rdap_level_0"
  ],
  "notices": [
    {
      "title": "Beverage Policy",
      "description": [
        "Beverages with caffeine for keeping horses awake."
      ],
      "links": [
        {
          "value": "http://example.net/ip/192.0.2.0/24",
          "rel": "alternate",
          "type": "text/html",
          "href": "http://www.example.com/redaction_policy.html"
        }
      ]
    }
  ],
  "lang": "en",
  "errorCode": 418,
  "title": "Your beverage choice is not available",
  "description": [
    "I know coffee has more ummpppphhh.",
    "Sorry, dude!"
  ]
}
```

7. RDAP Crawler

RDAP provides a bootstrap method to find which server is authoritative to answer queries for a requested scope, such as domain names, IP addresses, or Autonomous System numbers ([RFC7484]).

The proposed JSON Schema for RDAP has been used to validate the responses of servers listed in the IANA Bootstrap Service Registries ([RDAP-ASN], [RDAP-DSN], [RDAP-IPV4], [RDAP-IPV6]). Such a validation has been performed by a kind of RDAP crawler.

The Bootstrap Service Registries have been made available as JSON objects, which can be retrieved via HTTP from locations specified by IANA. The JSON object for each registry contains a series of members containing metadata about the registry such as a version identifier, a timestamp of the publication date of the registry, and a description.

Additionally, a "services" member contains the registry items themselves, as an array. Each item of the array contains a second level array, with two elements, each of them being a third-level array. Each element of the Services Array is a second-level array with two elements: in order, an Entry Array and a Service URL Array. The Entry Array contains all entries that have the same set of base RDAP URLs. The Service URL Array contains the list of base RDAP URLs usable for the entries found in the Entry Array. Elements within these two arrays are not sorted in any way.

In the following, the current Bootstrap Service Registry for Domain Name Space is shown:


```
{
  "description": "RDAP bootstrap file for Domain Name System registrations",
  "publication": "2017-12-05T17:52:00Z",
  "services": [
    [
      [
        "com",
        "net"
      ],
      [
        "https://rdap-pilot.verisignlabs.com/rdap/v1/"
      ]
    ],
    [
      [
        "ar"
      ],
      [
        "https://rdap.nic.ar"
      ]
    ],
    [
      [
        "cr"
      ],
      [
        "https://rdap.nic.cr/"
      ]
    ],
    [
      [
        "cz"
      ],
      [
        "https://rdap.nic.cz"
      ]
    ],
    [
      [
        "br"
      ],
      [
        "https://rdap.registro.br/"
      ]
    ]
  ],
```

```
  "version": "1.0"
}
```

7.1. Validation test of servers of Bootstrap Service Registry for Domain Name Space

In the following, just to give an example of the implemented tests, the source code of a Junit 4.0 test performing the validation of the responses provided by the servers listed above is shown.

7.1.1. AbstractBootstrapServiceTest class

```
package it.nic.rdap.schema.validator.crawler.test;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Random;
import org.apache.http.Header;
import org.apache.http.impl.client.HttpClientBuilder;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicHeader;
import org.everit.json.schema.Schema;
import org.everit.json.schema.ValidationException;
import org.everit.json.schema.loader.SchemaLoader;
import org.json.JSONObject;
import org.json.JSONTokener;
import be.dnsbelgium.rdap.client.RDAPClient;
import it.nic.rdap.schema.validator.test.AbstractValidatorTest;

public abstract class AbstractBootstrapServiceTest extends AbstractValidatorTest {

	private static Schema schema = null;

	private static void loadSchema() throws FileNotFoundException {

		File file = new File(Paths.get("").toUri().getPath() + getSchemaDir() +
"/rdap_response.json");
		InputStream inputStream = new FileInputStream(file);
		JSONObject rawSchema = new JSONObject(new JSONTokener(inputStream));
		SchemaLoader schemaLoader = SchemaLoader.builder().schemaJson(rawSchema)
				.resolutionScope("file:" +
Paths.get("").toUri().getPath().replaceAll(" ", "%20")
						+ getSchemaDir() + "/")
				.build();
		schema = schemaLoader.load().build();
	}

	protected static RDAPClient getRDAPClient(String url) {

		try {
			HashSet<Header> headers = new HashSet<Header>();
			headers.add(new BasicHeader("Accept", "application/rdap+json"));
			HttpClientBuilder httpClientBuilder =
HttpClients.custom().setDefaultHeaders(headers);

			return new RDAPClient(httpClientBuilder.build(), url);
		} catch (Exception e) {
			System.out.println(e);
		}

		return null;
```

```
	}

	protected static void testBootstrapServer(String query, String json) throws
IOException {

		try {
			if (schema == null)
				loadSchema();

			schema.validate(new JSONObject(json));
			log.info("{};OK", query);
		} catch (Exception e) {
			if (e instanceof ValidationException) {
				ValidationException ex = (ValidationException) e;
				log.info("{};{}", new Object[] { query, toDeepString(ex) });
			}
			throw e;
		}
	}

	protected static int randInt(int min, int max) {

		Random rand = new Random();
		int randomNum = rand.nextInt((max - min) + 1) + min;
		return randomNum;
	}
}
```

7.1.2. DnsServersValidatorTest class

```
package it.nic.rdap.schema.validator.crawler.test;

import java.io.IOException;
import java.util.List;
import org.junit.BeforeClass;
import org.junit.Test;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import be.dnsbelgium.rdap.client.RDAPClient;
import be.dnsbelgium.rdap.core.bootstrap.Bootstrap;

public class DnsServersValidatorTest extends AbstractBootstrapServiceTest {

	private static Bootstrap bootstrap = null;

	@BeforeClass
	public static void loadDnsBootstrap() {

		try {
			RDAPClient rdapClient =
getRDAPClient(Bootstrap.BOOTSTRAP_SERVICE_REGISTRY_BASE_URL);
		  JsonNode json = rdapClient.getBootstrapDnsAsJson();

		  if (json == null)
			return;

		  ObjectMapper mapper = new ObjectMapper();
		  bootstrap = mapper.treeToValue(json, Bootstrap.class);
		} catch(Exception e) {
			System.out.println(e);
		}

	}

	private void testDnsServerValidation(String url, String domainName) throws
IOException {
		try {
		  if (url.endsWith("/"))
```

```
                url = url.substring(0, url.length() - 1);

            RDAPClient rdapClient = getRDAPClient(url);
            JsonNode json = rdapClient.getDomainAsJson(domainName);

            if (json == null)
                return;

            String urlx = (url.endsWith("/")) ? url + "domain/" : url + "/domain/";

            testBootstrapServer(urlx + domainName, json.toString());

        } catch(Exception e) {
                System.out.println(e);
        }

    }

    @Test
    public void testDnsServersValidation() throws IOException {

        if (bootstrap == null)
            return;

        if (bootstrap.getServices() == null)
            return;

        for (List<List<String>> serviceItems : bootstrap.getServices()) {

            List<String> tlds = serviceItems.get(0);
            List<String> urls = serviceItems.get(1);
            String url = urls.get(0);

            for(String tld : tlds) {
                log.info("testDnsServersValidation - url: {} : domain: {}",
url, "nic." + tld);
                testDnsServerValidation(url, "nic." + tld);
            }

        }
    }
}
```

8. References

[JSON-HYPER-SCHEMA] WRIGHT A. AND ANDREWS H., "JSON HYPER-SCHEMA: A VOCABULARY FOR HYPERMEDIA ANNOTATION OF JSON", DRAFT-HANDREWS-JSON-SCHEMA-HYPERSCHEMA-00 (WORK IN PROGRESS), NOVEMBER 2017, HTTPS://TOOLS.IETF.ORG/HTML/DRAFT-HANDREWS-JSON-SCHEMA-HYPERSCHEMA-00

[JSON-SCHEMA] WRIGHT A. AND ANDREWS H., "JSON SCHEMA: A MEDIA TYPE FOR DESCRIBING JSON DOCUMENTS", DRAFT-HANDREWS-JSON-SCHEMA-00 (WORK IN PROGRESS), NOVEMBER 2017, HTTPS://TOOLS.IETF.ORG/HTML/DRAFT-HANDREWS-JSON-SCHEMA-00

[JSON-SCHEMA-VALIDATION] WRIGHT A. , ANDREWS H. AND G. LUFF, "JSON SCHEMA VALIDATION: A VOCABULARY FOR STRUCTURAL VALIDATION OF JSON", DRAFT-HANDREWS-JSON-SCHEMA-VALIDATION-00 (WORK IN PROGRESS), NOVEMBER 2017, HTTPS://TOOLS.IETF.ORG/HTML/DRAFT-HANDREWS-JSON-SCHEMA-VALIDATION-00

[JSON-SCHEMA-POINTER]
ANDREWS H. AND G. LUFF, "RELATIVE JSON POINTERS", DRAFT-HANDREWS-RELATIVE-JSON-POINTER-00 (WORK IN PROGRESS), NOVEMBER 2017, HTTPS://TOOLS.IETF.ORG/HTML/DRAFT-HANDREWS-RELATIVE-JSON-POINTER-00

[JSON-SCHEMA-VALIDATOR]
"JSON SCHEMA VALIDATOR", HTTPS://GITHUB.COM/EVERIT-ORG/JSON-SCHEMA

[RDAP-ASN] "BOOTSTRAP SERVICE REGISTRY FOR AS NUMBER SPACE", HTTP://DATA.IANA.ORG/RDAP/ASN.JSON

[RDAP-DNS] "BOOTSTRAP SERVICE REGISTRY FOR DOMAIN NAME SPACE", HTTP://DATA.IANA.ORG/RDAP/DSN.JSON

[RDAP-IPV4] "BOOTSTRAP SERVICE REGISTRY FOR IPV4 ADDRESS SPACE", HTTP://DATA.IANA.ORG/RDAP/IPV4.JSON

[RDAP-IPV6] "BOOTSTRAP SERVICE REGISTRY FOR IPV6 ADDRESS SPACE", HTTP://DATA.IANA.ORG/RDAP/IPV6.JSON

[RFC6570] GREGORIO J., FIELDING R., HADLEY M., NOTTINGHAM M. AND ORCHARD D., "URI TEMPLATE", RFC 6570, DOI 10.17487/RFC6570, MARCH 2012, <HTTPS://WWW.RFC-EDITOR.ORG/INFO/RFC6570>.

[RFC7482] NEWTON, A. AND S. HOLLENBECK, "REGISTRATION DATA ACCESS PROTOCOL (RDAP) QUERY FORMAT", RFC 7483, 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>.

[RFC7484] M. BLANCHET, "FINDING THE AUTHORITATIVE REGISTRATION DATA (RDAP) SERVICE", RFC 7484, DOI 10.17487/RFC7484, MARCH 2015, <HTTPS://WWW.RFC-EDITOR.ORG/INFO/RFC7484>.

[RFC8259] T. BRAY, ED., "THE JAVASCRIPT OBJECT NOTATION (JSON) DATA INTERCHANGE FORMAT", DECEMBER 2017, <HTTPS://WWW.RFC-EDITOR.ORG/INFO/RFC8259>

[URI-TEMPLATES-VALIDATOR]
"HANDY URI TEMPLATES", HTTPS://GITHUB.COM/DAMNHANDY/HANDY-URI-TEMPLATES