Redacted Fields in the Registration Data Access Protocol (RDAP) Response

draft-gould-regext-rdap-redacted-00

Abstract

This document describes an RDAP extension for explicitly identifying
redacted RDAP response fields, using JSONPath as the default
expression language.

Status of This Memo

This Internet-Draft is submitted in full conformance with the
provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering
Task Force (IETF). Note that other groups may also distribute
working documents as Internet-Drafts. The list of current Internet-
Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months
and may be updated, replaced, or obsoleted by other documents at any
time. It is inappropriate to use Internet-Drafts as reference
material or to cite them other than as "work in progress."

This Internet-Draft will expire on 13 January 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal
license-info) in effect on the date of publication of this document.
Please review these documents carefully, as they describe your rights
and restrictions with respect to this document. Code Components
extracted from this document must include Simplified BSD License text
as described in Section 4.e of the Trust Legal Provisions and are
provided without warranty as described in the Simplified BSD License.
1. Introduction

This document describes an RDAP extension for explicitly identifying redacted RDAP response fields, using JSONPath as the default expression language. A redacted RDAP field is one that has data removed from the RDAP response due to the lack of client privilege to receive the field. This extension can be used to identify redacted RDAP fields in any RDAP object class, as defined in [RFC7483], or RDAP fields defined in RDAP extensions. Because an RDAP response may exclude a field due to either the lack of data or based on the lack of RDAP client privileges, this extension is used to explicitly specify which RDAP fields are not included in the RDAP response due to redaction. It thereby provides a capability for disambiguation between redaction and possible other reasons for data or field absence.

JSONPath, as defined in [I-D.ietf-jsonpath-base], is used as the default expression language to reference RDAP fields that have been redacted. The redacted JSON fields will either be removed or have empty values in the RDAP response. JSON is defined by [RFC8259].

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.
3. Redaction Methods

Redaction in RDAP can be handled in multiple ways. The use of placeholder text for the values of the RDAP fields, such as the placeholder text "XXXX", MUST NOT be used for redaction. A placeholder text value will not match the format requirements of each of the RDAP fields and provides an inconsistent and unreliable redaction signal. This section covers the redaction methods that can be used with the redaction signaling defined in Section 4.2.

RDAP responses, as defined in [RFC7483], include a mix of JSON objects and JSON arrays, where JSON arrays are heavily used for entity objects with jCard [RFC7095]. jCard [RFC7095] is a JSON representation of vCard [RFC6350] that inherits its dependency on arrays. An example is the vCard [RFC6350] "ADR" property / jCard [RFC7095] "adr" property that defines a sequence of address components. According to [RFC6350], when an "ADR" property component value is missing, the associated component separator MUST still be specified. jCard [RFC7095] extends the use of arrays with each individual vCard property being represented by an array of three fixed elements, followed by one or more additional elements. The mix of JSON objects and JSON arrays impacts the methods used for redaction in RDAP. The redaction of RDAP fields fall into the two categories of Redaction by Removal Method (Section 3.1) and Redaction by Empty Value Method (Section 3.2), defined in the following subsections.

3.1. Redaction by Removal Method

The Redaction by Removal Method is when the RDAP field is removed from the RDAP response, which is the preferred method. The Redaction by Removal Method can be done for all RDAP response fields other than the JSON arrays used with jCard [RFC7095]. When an RDAP object is redacted by removal, all of the RDAP object’s child fields are also removed. Only the redacted RDAP object needs to be referenced in the list of redacted fields, as defined in Section 4.2. An example of redacting an RDAP object is removing the administrative contact from the RDAP response and including the following "redacted" member:
"redacted": [
    {
        "name": "Administrative Contact",
        "path": "$\cdot entities[?(@.roles[0]=='administrative')]$",
        "method": "removal"
        "reason": "Client request"
    }
]

The Redaction by Removal Method MUST NOT be used to remove a field from a jCard [RFC7095] fixed array position, which will result in a non-conformant jCard [RFC7095] array definition.

3.2. Redaction by Empty Value Method

The Redaction by Empty Value Method is when a redacted field is not removed, but its value is set to an empty value, such as "" for a jCard [RFC7095] Text ("text") property or null for non-Text ("text") properties. The empty jCard [RFC7095] values ("" or null) are referenced in the "redacted" member in place of the jCard [RFC7095] property name, such as referencing the "fn" jCard property value at position 3 instead of referencing the "fn" jCard property name at position 0. The Redaction by Empty Value Method SHOULD be used only when redacting JSON response fields that use jCard [RFC7095] arrays. Optional jCard [RFC7095] properties SHOULD use the Redaction by Removal Method (Section 3.1) to redact the entire property. The required jCard [RFC7095] "fn" property, defined in section 6.2.1 of vCard [RFC6350], MUST use the Redaction by Empty Value Method to redact the property value. Removing the "fn" property would violate vCard [RFC6350] and removing the property value would violate the fixed array positions defined in jCard [RFC7095].

An example of the redacted field "fn" jCard property using the Redaction by Empty Value Method:

```
[  
    "fn",  
    {},  
    "text",  
    ""
]
```

An example of the "redacted" member for the redacted "fn" jCard property value, which is array position 3:
"redacted": [
{
   "name": "Registrant Name",
   "path": "$.entities[?(@.roles[0]==’registrant’)].vcardArray[1][?(@[0]==’fn’)][3]",
   "pathLang": "jsonpPath",
   "method": "emptyValue",
   "reason": "Server policy"
}
]

4. Redacted RDAP Response

4.1. RDAP Conformance

RDAP responses that contain values described in this document MUST indicate conformance with this specification by including an rdapConformance ([RFC7483]) value of "redacted_0". The information needed to register this value in the RDAP Extensions Registry is described in Section 6.1.

Example rdapConformance member with the redacted extension:

"rdapConformance": [
   "rdap_level_0",
   "redacted_0"
]

4.2. "redacted" Member

The "redacted" member MUST be added to the RDAP response when there are redacted fields. The "redacted" member contains an array of redacted objects with the following child members:

"name": A logical name for the redacted field. The logical name used for the redacted field is up to server policy. Conventions used for the chosen logical names MAY be defined in other documents to meet the needs of different RDAP services or industries.

"path": The JSON expression of the redacted field, using the expression language defined by the "pathLang" member. The JSON expression references a removed JSON field or an empty field value based on Section 3.

"pathLang": OPTIONAL JSON path expression language used, with the
default value of "jsonp-path" for JSONPath ([I-D.ietf-jsonpath-base]). Other JSON path expression languages MAY be used based on server policy.

"method":  OPTIONAL redaction method used with "removal" indicating the Redaction By Removed Method (Section 3.1) and "emptyValue" indicating the Redaction by Empty Value Method (Section 3.2), with the default value of "removal".

"reason":  OPTIONAL human readable reason(s) for the redacted field in the language defined by the [RFC7483] "lang" member. The default language is "en" if the [RFC7483] "lang" member is not specified. The "reason" member is provided for informational purposes and MUST NOT be a client processing dependency.

Example unredacted version of RDAP response:

```javascript
{
  "rdapConformance": [ 
    "rdap_level_0"
  ],
  "objectClassName": "domain",
  "handle": "ABC123",
  "ldhName": "example.com",
  "secureDNS": {
    "delegationSigned": false
  },
  "notices": [ 
    { 
      "title": "Terms of Use",
      "description": [ 
        "Service subject to Terms of Use."
      ],
      "links": [ 
        { 
          "rel": "self",
          "href": "https://www.example.com/terms-of-use",
          "type": "text/html",
          "value": "https://www.example.com/terms-of-use"
        }
      ]
    }
  ],
  "nameservers": [ 
    { 
      "objectClassName": "nameserver",
      "ldhName": "ns1.example.com"
    }
  ],
}
```

{  "objectClassName": "nameserver",  "ldhName": "ns2.example.com"  },
"entities": [
    {  "objectClassName": "entity",  "handle": "123",  "roles": [  "registrar"  ],  "publicIds": [  {  "type": "IANA Registrar ID",  "identifier": "1"  }  ],  "vcardArray": [  "vcard",  [  [  "version",  {},  "text",  "4.0"
  ],  [  "fn",  {},  "text",  "Example Registrar Inc."  ],  [  "adr",  {},  "text",  [  "",  "Suite 100",  "123 Example Dr.",  "Dulles",  "VA",  "20166-6503",  "US"  ]  ]
  ]
  ]
]
"email",
{},
"text",
"contact@organization.example"
],
[
"tel",
{
"type": "voice"
},
"uri",
"tel:+1.7035555555;ext=1234"
],
[
"tel",
{
"type": "fax"
},
"uri",
"tel:+1.7035555556"
]
],
"entities": [
{
"objectClassName": "entity",
"roles": [
"abuse"
],
"vcardArray": [
"vcard",
[
["version",
{},
"text",
"4.0"
],
["fn",
{},
"text",
"Abuse Contact"
],
["email",
{},
"text",
"contact@organization.example"
],
["uri",
"tel:+1.7035555555;ext=1234"
],
["uri",
"tel:+1.7035555556"
]
],
"entities": [

"abuse@organization.example",
],
[
"tel",
{
"type": "voice"
},
"uri",
"tel:+1.7035555555;ext=1234"
}
]
]
],
{
"roles": [
"registrant"
],
"vcardArray": [
"vcard",
[
[ "version",
{},
"text",
"4.0"
],
[ "fn",
{},
"text",
"Registrant User"
],
[ "org",
{},
"text",
"Example Inc."
],
[ "adr",
{},
"text",
[ "Suite 1235",
"4321 Rue Somewhere",
]
"Quebec",
"QC",
"G1V 2M2",
"Canada"
],
[
  "email",
  {
    "text",
    "registrant.user@example.com"
  }
],
[
  "tel",
  {
    "type": "voice"
  },
  "uri",
  "tel:+1-555-555-1235;ext=123"
],
[
  "tel",
  {
    "type": "fax"
  },
  "uri",
  "tel:+1-555-555-5321"
]
],
",
{
  "roles": [
    "technical"
  ],
  "vcardArray": [
    "vcard",
    [
      "version",
      {}
    ],
    "text",
    "4.0"
  ],
  [
    "fn",
    {}
  ],
  "text",
  "registrant.user@example.com"}
"Technical User",
]
]
"org",
{}
"text",
"Example Inc."
],
"adr",
{}
"text",
[ "Suite 1234",
"4321 Rue Somewhere",
"Quebec",
"QC",
"G1V 2M2",
"Canada"
],
"email",
{}
"text",
"technical.user@example.com"
],
"tel",
{
"type": "voice"
},
"uri",
"tel:+1-555-555-1234;ext=321"
],
"tel",
{
"type": "fax"
},
"uri",
"tel:+1-555-555-4321"
"roles": [
  "administrative"
],
"vcardArray": [
  "vcard",
  [
    "version",
    {},
    "text",
    "4.0"
  ],
  "fn",
  {},
  "text",
  "Administrative User"
],
  "org",
  {},
  "text",
  "Example Inc."
],
  "adr",
  {},
  "text",
  [
    "Suite 1236",
    "4321 Rue Somewhere",
    "Quebec",
    "QC",
    "G1V 2M2",
    "Canada"
  ],
  "email",
  {},
  "text",
  "administrative.user@example.com"
],
  "tel",
  {
    "type": "voice"
Example redacted version of RDAP response:

```json
{
    "rdapConformance": [
        "rdap_level_0",
        "redacted_0"
    ],
    "objectClassName": "domain",
    "ldhName": "example.com",
    "secureDNS": {
        "uri",
        "tel:+1-555-555-1236;ext=789"
    },
    "events": [
        { "eventAction": "registration", "eventDate": "1997-06-03T00:00:00Z" },
        { "eventAction": "last changed", "eventDate": "2020-05-28T01:35:00Z" },
        { "eventAction": "expiration", "eventDate": "2021-06-03T04:00:00Z" }
    ],
    "status": [
        "server delete prohibited",
        "server update prohibited",
        "server transfer prohibited",
        "client transfer prohibited"
    ]
}
```
"delegationSigned": false,
"notices": [
  {
    "title": "Terms of Use",
    "description": [
      "Service subject to Terms of Use."
    ],
    "links": [
      {
        "rel": "self",
        "href": "https://www.example.com/terms-of-use",
        "type": "text/html",
        "value": "https://www.example.com/terms-of-use"
      }
    ]
  }
],
"nameservers": [
  {
    "objectClassName": "nameserver",
    "ldhName": "ns1.example.com"
  },
  {
    "objectClassName": "nameserver",
    "ldhName": "ns2.example.com"
  }
],
"entities": [
  {
    "objectClassName": "entity",
    "handle": "123",
    "roles": [
      "registrar"
    ],
    "publicIds": [
      {
        "type": "IANA Registrar ID",
        "identifier": "1"
      }
    ],
    "vcardArray": [
      "vcard",
      [
        {
          "version":
        },
        "text",
      ]
    ]
  }
]
"4.0"
],
[ "fn",
 {}
],
"text",
"Example Registrar Inc."
],
[ "adr",
 {}
],
"text",
[ ""
, "Suite 100",
 "123 Example Dr.",
 "Dulles",
 "VA",
 "20166-6503",
 "US"
]
],
[ "email",
 {}
],
"text",
"contact@organization.example"
],
[ "tel",
 {
 "type": "voice"
 },
 "uri",
 "tel:+1.7035555555"
],
[ "tel",
 {
 "type": "fax"
 },
 "uri",
 "tel:+1.7035555556"
]
],
"entities": [ }
}
}
"objectClassName": "entity",
"roles": [
  "abuse"
],
"vcardArray": [
  "vcard",
  [
    [
      "version",
      {},
      "text",
      "4.0"
    ],
    [
      "fn",
      {},
      "text",
      "Abuse Contact"
    ],
    [
      "email",
      {},
      "text",
      "abuse@organization.example"
    ],
    [
      "tel",
      {
        "type": "voice"
      },
      "uri",
      "tel:+1.7035555555"
    ]
  ]
],
"roles": [
  "registrant"
],
"vcardArray": [
  "vcard",
  [
    [
      "version",
      {},
      "text",
      "4.0"
    ],
    [
      "fn",
      {},
      "text",
      "Abuse Contact"
    ],
    [
      "email",
      {},
      "text",
      "abuse@organization.example"
    ],
    [
      "tel",
      {
        "type": "voice"
      },
      "uri",
      "tel:+1.7035555555"
    ]
  ]
]
"text",
"4.0"
],
[ "fn",
{},
"text",
"
],
[ "adr",
{},
"text",
[ "", ",",
"", ",",
"", ",",
"QC",
",",
"Canada"
] ]
] ]
},
{ "roles": [
 "technical"
],
"vcardArray": [
 "vcard",
 [ [ "version",
 {},
 "text",
 "4.0"
 ],
 [ "fn",
 {},
 "text",
 ""
 ],
 [ "org",
 {},
 ""
 ]]
]}
}
]
"text",
  "Example Inc."
],
[  
  "adr",
  {},
  "text",
  [    
    "",
    "Suite 1234",
    "4321 Rue Somewhere",
    "Quebec",
    "QC",
    "G1V 2M2",
    "Canada"
  ]
]
],
"events": [
  {
    "eventAction": "registration",
    "eventDate": "1997-06-03T00:00:00Z"
  },
  {
    "eventAction": "last changed",
    "eventDate": "2020-05-28T01:35:00Z"
  },
  {
    "eventAction": "expiration",
    "eventDate": "2021-06-03T04:00:00Z"
  }
],
"status": [
  "server delete prohibited",
  "server update prohibited",
  "server transfer prohibited",
  "client transfer prohibited"
],
"redacted": [
  {
    "name": "Registry Domain ID",
    "path": ".handle",
    "pathLang": "jsonpath",
    "method": "removal",
    "reason": "Server policy"
},
{
  "name": "Registrant Name",
  "path": "$.entities[?(@.roles[0]==‘registrant’)].vcardArray[1][?(@[0]==‘fn’)][3]",
  "pathLang": "jsonpath",
  "method": "emptyValue",
  "reason": "Server policy"
},
{
  "name": "Registrant Organization",
  "path": "$.entities[?(@.roles[0]==‘registrant’)].vcardArray[1][?(@[0]==‘org’)]",
  "pathLang": "jsonpath",
  "method": "removal",
  "reason": "Server policy"
},
{
  "name": "Registrant Street",
  "path": "$.entities[?(@.roles[0]==‘registrant’)].vcardArray[1][?(@[0]==‘adr’)][3][3]",
  "pathLang": "jsonpath",
  "method": "emptyValue",
  "reason": "Server policy"
},
{
  "name": "Registrant City",
  "path": "$.entities[?(@.roles[0]==‘registrant’)].vcardArray[1][?(@[0]==‘adr’)][3][3]",
  "pathLang": "jsonpath",
  "method": "emptyValue",
  "reason": "Server policy"
},
{
  "name": "Registrant Postal Code",
  "path": "$.entities[?(@.roles[0]==‘registrant’)].vcardArray[1][?(@[0]==‘adr’)][3][5]",
  "pathLang": "jsonpath",
  "method": "emptyValue",
  "reason": "Server policy"
},
{
  "name": "Registrant Email",
  "path": "$.entities[?(@.roles[0]==‘registrant’)].vcardArray[1][?(@[0]==‘email’)]",
  "method": "removal",
  "reason": "Server policy"
}
{
  "name": "Registrant Phone",
  "path": "$.entities[?(@.roles[0]==‘registrant’)].vcardArray[1][?(@[1].type==’voice’)]",
  "method": "removal",
  "reason": "Server policy"
},
{
  "name": "Technical Name",
  "path": "$.entities[?(@.roles[0]==‘technical’)].vcardArray[1][?(@[0]==‘fn’)][3]",
  "method": "emptyValue",
  "reason": "Server policy"
},
{
  "name": "Technical Email",
  "path": "$.entities[?(@.roles[0]==‘technical’)].vcardArray[1][?(@[0]==‘email’)]",
  "method": "removal",
  "reason": "Server policy"
},
{
  "name": "Technical Phone",
  "path": "$.entities[?(@.roles[0]==‘technical’)].vcardArray[1][?(@[1].type==’voice’)]",
  "method": "removal",
  "reason": "Server policy"
},
{
  "name": "Technical Fax",
  "path": "$.entities[?(@.roles[0]==‘technical’)].vcardArray[1][?(@[1].type==’fax’)]",
  "reason": "Client request"
},
{
  "name": "Administrative Contact",
  "path": "$.entities[?(@.roles[0]==‘administrative’)]",
  "method": "removal",
  "reason": "Client request"
}
}
5. JSONPath Considerations

JSONPath [I-D.ietf-jsonpath-base] is the default JSON path expression language. This section covers considerations for servers using [I-D.ietf-jsonpath-base] to identify redacted RDAP fields with the "path" member of redacted objects in the "redacted" member. The list of JSONPath considerations include:

1. Use absolute paths with the ‘$’ JSONPath element. An example is "$.handle" for the "Registry Domain ID".
2. Validate a JSONPath expression using a non-redacted RDAP response, where evaluating the expression results in returning the redacted field.
3. Reference the removed object field when redacting an entire object by the Redaction by Removal Method (Section 3.1), where all of the object’s child fields are explicitly removed. An example is "$.entities[?(@.roles[0]==’administrative’)]]" for the entire "Administrative Contact".
4. Reference the removed field when using the Redaction by Removal Method (Section 3.1). An example is "$.handle" for the "Registry Domain ID".
5. Reference index 0 of the jCard [RFC7095] property array, which is the jCard [RFC7095] "name" property, with a filter expression containing the name of the field, when redacting a jCard [RFC7095] field using the Redaction by Removal Method (Section 3.1). An example is "$.entities[?(@.roles[0]==’registrant’)].vcardArray[1][?(@[0]==’email’)]" for the "Registrant Email".
6. Reference jCard [RFC7095] field value or values redacted by array index 3 and greater, when redacting a jCard [RFC7095] field using the Redaction by Empty Value Method (Section 3.2). The jCard [RFC7095] property array index 3 and greater contain the property values, where the property values set with an empty value are referenced directly in place of the jCard [RFC7095] property name. Servers can then systematically redact jCard [RFC7095] field value or values based on the JSONPath expressions and clients will directly know which jCard [RFC7095] property values have been redacted. An example is "$.entities[?(@.roles[0]==’registrant’)].vcardArray[1][?(@[0]==’fn’)][3]" for the "Registrant Name" or "$.entities[?(@.roles[0]==’registrant’)].vcardArray[1][?(@[0]==’adr’)][3][5]" for the "Registrant Postal Code".

6. IANA Considerations

6.1. RDAP Extensions Registry

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

Extension identifier: redacted_0
Registry operator: Any
Published specification: This document.
Contact: IESG <iesg@ietf.org>
Intended usage: This extension identifies the redacted fields in an RDAP response.

7. Security Considerations

The server including a redacted signal provides an unauthorized client additional information related to the existence of data. Servers MAY exclude the redacted members for RDAP fields that are considered a privacy issue in providing a data existence signal.

8. Acknowledgements

The authors wish to thank the following persons for their feedback and suggestions: Scott Hollenbeck, and Rick Wilhelm.

9. References

9.1. Normative References

[I-D.ietf-jsonpath-base]
Normington, G., Surov, E., Mikulicic, M., and F. Dortmund, "JavaScript Object Notation (JSON) Path", Work in Progress, Internet-Draft, draft-ietf-jsonpath-base-00, 7 March 2021,

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350,
DOI 10.17487/RFC6350, August 2011,

DOI 10.17487/RFC7095, January 2014,
<https://www.rfc-editor.org/info/rfc7095>.

[RFC7483] Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", RFC 7483,
DOI 10.17487/RFC7483, March 2015,


Authors’ Addresses

James Gould
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States of America

Email: jgould@verisign.com
URI: http://www.verisigninc.com

David Smith
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States of America

Email: dsmith@verisign.com
URI: http://www.verisigninc.com

Jody Kolker
GoDaddy Inc.
14455 N. Hayden Rd. #219
Scottsdale, AZ 85260
United States of America

Email: jkolker@godaddy.com
URI: http://www.godaddy.com

Roger Carney
GoDaddy Inc.
14455 N. Hayden Rd. #219
Scottsdale, AZ 85260
United States of America

Email: rcarney@godaddy.com
URI: http://www.godaddy.com
Abstract

Multiple applications related to the registration of names and other identifiers are built around a list of data elements. There is currently no unified public list of these data elements, nor is there an organized and independent change control process. This document codifies the multiple similar but not quite identical lists of data elements into a neutral Data Dictionary to be maintained as an independent IANA Registry. The Data Dictionary defines data elements but does not specify which ones are to be used in any particular application; the Data Dictionary is policy-neutral.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components
Table of Contents

1. Introduction ................................................. 3
   1.1. Requirements Language .............................. 3

2. Data Element Specification ................................. 4
   2.1. Element name: Domain Name .......................... 4
   2.2. Element name: Registry ............................... 4
   2.3. Element name: NS ..................................... 4
   2.4. Element name: Registration Creation Date .......... 5
   2.5. Element name: Registration Expiration Date ........ 5
   2.6. Element name: Registration Updated Date .......... 5
   2.7. Element name: Registration Transfer Date .......... 5
   2.8. Element name: Protection ............................. 5
   2.9. Element name: Nexus .................................. 5
   2.10. Element name: Person ................................ 5
   2.11. Element name: Personal ............................... 5
   2.12. Element name: Status & Locks ....................... 5
   2.13. Element name: Source & Method ..................... 6
   2.14. Element name: Payment History ...................... 6
   2.15. Element name: Transaction History ................. 6
   2.16. Element name: User Account ID ...................... 6
   2.17. Element name: Reserved .............................. 6
   2.18. Element name: Name ................................ 6
   2.19. Element name: Org .................................. 6
   2.20. Element name: Street ................................ 6
   2.21. Element name: City ................................ 6
   2.22. Element name: State/Province ....................... 7
   2.23. Element name: Postal code ......................... 7
   2.24. Element name: Country ............................... 7
   2.25. Element name: Phone ................................ 7
   2.26. Element name: Phone ext ............................ 7
   2.27. Element name: Fax ................................ 7
   2.28. Element name: Fax ext .............................. 7
   2.29. Element name: Email ................................. 8
   2.30. Element name: Email_or_phone ...................... 8
   2.31. Element name: Registry UniqueID ................. 8

3. IANA Considerations ......................................... 8
   3.1. Report Specification ................................. 8
      3.1.1. Designated Expert Evaluation Criteria ........... 8
      3.1.2. Registration Procedure .......................... 9
   3.2. Initial assignments .................................. 11
      3.2.1. Data Element Definition in IANA Registry ........ 11

4. Security Considerations .................................... 12

5. Privacy Considerations ..................................... 12
1. Introduction

The DNS Data Dictionary provides a common set of names and definitions for data elements which may be used in a DNS related protocol. The dictionary is intended to be inclusive and not obligatory. That is, the existence of a data element in this dictionary does not imply the data element must be used or recognized in any particular protocol. The items in this dictionary should represent the union for what is in existing relevant protocols, and should prevent divergence in new protocols. We also expect that each application or protocol may have additional requirements specific to the application or protocol. Such additional requirements should be documented as part of the application or protocol specification.

The data dictionary currently has thirty-one data elements. These data elements include the DNS records, the detailed status of a registration to the details for each of the contacts, and the account details and payment history. The proposed IANA registry lists standard data elements and their syntax for inclusion in the files.

We expect the DNS data dictionary to evolve to meet the needs of various applications. With the exception of correction of errors, we expect the changes to the DNS Data Dictionary to be additions as opposed to deletions or changes.

[Comment: We are looking for additional authors and contributors to add to and improve the data dictionary, keeping in line with the RFC Series Editor statement on authorship. https://www.rfc-editor.org/pipermail/rfc-interest/2015-May/008869.html]

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.
2. Data Element Specification

Each data element is a single unit of information that can be collected and compared during the registration process. The primary purposes of the IANA registry of data elements are to ensure that each data element is assigned a unique name and that the syntax of each data element is specified.

Each data element is assigned to an element type to organize the taxonomy of the data dictionary.

The name of the data element MUST be unique and this characteristic MUST be enforced by the registry. The character encoding recommendation for data elements is specified in Section 3.

The subsections below comprise an initial list of known data elements commonly being used in the templates. The title of the subsection is the data element name for the data element. The combination of data element type and data element name MUST be unique and MUST be processed as case insensitive in the IANA registry.

Note that the legal definition of any of the terms used in the data dictionary, such as 'personally identifiable information' or 'legal person', are to be determined locally. The organization using this dictionary will record their interpretation in the appropriate element.

2.1. Element name: Domain Name

This is the domain name as formatted according to the Internationalized Domain Names for Applications (IDNA) specification.[RFC5890]

See also "Domain name" in [RFC8499].

2.2. Element name: Registry

The name of the registry. This data element is text/string with no naming convention enforced.

See also "Registry" in [RFC8499].

2.3. Element name: NS

The authoritative name server for the domain.[RFC1034]

See also "Authoritative server" in [RFC8499]
2.4. Element name: Registration Creation Date

The date and time of domain object creation. Format TBD.

2.5. Element name: Registration Expiration Date

The date and time identifying the end of the domain object’s registration period. Format TBD.

2.6. Element name: Registration Updated Date

The date and time of the most recent domain-object modification. Format TBD.

2.7. Element name: Registration Transfer Date

The date and time of the most recent successful domain-object transfer. Format TBD.

2.8. Element name: Protection

Definition is TBD.

2.9. Element name: Nexus

Definition is TBD.

2.10. Element name: Person

Definition is TBD.

2.11. Element name: Personal

Definition is TBD.

2.12. Element name: Status & Locks

Examples include the EPP (Section 2.3 of [RFC5731]) and RDAP (Section 10.2.2 of [RFC9083]) codes (ex: clientTransferProhibited) that describe the current state of a registered domain name and the protocol actions that can (or cannot) be performed on the domain name. A registered domain name MAY be associated with multiple status values. Other managed objects, including name server and contact objects, can also have status and lock values.
2.13. Element name: Source & Method

Definition is TBD.

2.14. Element name: Payment History

Definition is TBD.

2.15. Element name: Transaction History

Definition is TBD.

2.16. Element name: User Account ID

Definition is TBD.

2.17. Element name: Reserved

[this field is an artifact of prior use which was determined to not be necessary, but the field was left intact for future use]

2.18. Element name: Name

Individual name is represented using character strings. These strings have a specified minimum length and a specified maximum length. Individual names MAY be provided in either UTF-8 [RFC3629] or a subset of UTF-8 that can be represented in 7-bit ASCII, depending on local needs.

2.19. Element name: Org

Organization name is represented using character strings. These strings have a specified minimum length and a specified maximum length. Organizational names MAY be provided in either UTF-8 [RFC3629] or a subset of UTF-8 that can be represented in 7-bit ASCII, depending on local needs.

2.20. Element name: Street

Postal street address, formatted as per [ISO19160-4].

2.21. Element name: City

Postal city address, formatted as per [ISO19160-4].
2.22. Element name: State/Province

Postal state or province address, formatted as per [ISO19160-4].

2.23. Element name: Postal code

Postal code, formatted as per [ISO19160-4]. Contact postal codes are represented using character strings. These strings have a specified minimum length and a specified maximum length.

2.24. Element name: Country

Country code identifier. Contact country identifiers are represented using two-character identifiers specified in [ISO3166-1].

2.25. Element name: Phone

Telephone number structure is derived from structures defined in [ITU.E164.2005]. Telephone numbers described in this mapping are character strings that MUST begin with a plus sign ("+", ASCII value 0x002B), followed by a country code defined in [ITU.E164.2005], followed by a dot (".", ASCII value 0x002E), followed by a sequence of digits representing the telephone number. An optional "x" attribute is provided to note telephone extension information.

2.26. Element name: Phone ext

This field is intended to represent an "extension" within the phone number to reach the specific person or role desk telephone, appropriate queue or mailbox after successfully dialing the Phone element.

2.27. Element name: Fax

Fax telephone number structure is derived from structures defined in [ITU.E164.2005]. Telephone numbers described in this mapping are character strings that MUST begin with a plus sign ("+", ASCII value 0x002B), followed by a country code defined in [ITU.E164.2005], followed by a dot (".", ASCII value 0x002E), followed by a sequence of digits representing the telephone number.

2.28. Element name: Fax ext

This field is an "extension" within a phone tree or PBX that is necessary to connect to a fax machine after successfully dialing the fax element.
2.29. Element name: Email

   Email address syntax is defined in [RFC5322].

2.30. Element name: Email_or_phone

   There is a requirement that either the phone or email element have
   been confirmed reachable, which this field is intended to represent.

2.31. Element name: Registry UniqueID

   This field represents server-unique identifiers assigned to entities,
   such as clients and contacts. These identifiers are character
   strings that typically have a specified minimum length, a specified
   maximum length, and a specified format.

3. IANA Considerations

   This section describes the format of the IANA Registration Report
   Registry, which has two tables described below, and the procedures
   used to populate and manage the registry entries.

3.1. Report Specification

   This registry uses the "Specification Required" policy described in
   [RFC8126]. An English language version of the extension
   specification is required in the registry, though non-English
   versions of the specification may also be provided.

   The "Specification Required" policy implies review by a "designated
   expert". Section 5.2 of RFC 8126 describes the role of designated
   experts and the function they perform.

3.1.1. Designated Expert Evaluation Criteria

   A high-level description of the role of the designated expert is
   described in Section 5.2 of RFC 8126. Specific guidelines for the
   appointment of designated experts and the evaluation of a new data
   element is provided here.
The IESG SHOULD appoint a small pool of individuals (perhaps 3 - 5) to serve as designated experts, as described in Section 5.2 of RFC 8126. The pool should have a single administrative chair who is appointed by the IESG. The designated experts should use the existing regext mailing list (regext@ietf.org) for public discussion of registration requests. This implies that the mailing list should remain open after the work of the REGEXT working group has concluded.

The results of the evaluation should be shared via email with the registrant and the regext mailing list. Issues discovered during the evaluation can be corrected by the registrant, and those corrections can be submitted to the designated experts until the designated experts explicitly decide to accept or reject the registration request. The designated experts must make an explicit decision and that decision must be shared via email with the registrant and the regext mailing list. If the specification for a data element or report is an IETF Standards Track document, no review is required by the designated expert.

Designated experts should be permissive in their evaluation of requests for data elements and reports that have been implemented and deployed by at least one registry. This implies that it may indeed be possible to register multiple data elements or reports that provide the same functionality. Requests to register data elements or reports that have not been deployed should be evaluated with a goal of reducing duplication. A potential registrant who submits a request to register a new data element or report that includes similar functionality to existing data elements or reports should be made aware of the existing data elements and reports. The registrant should be asked to reconsider their request given the existence of similar data elements or reports. Should they decline to do so, perceived similarity should not be a sufficient reason for rejection as long as all other requirements are met.

3.1.2. Registration Procedure

The registry contains information describing each registered data element or report. Registry entries are created and managed by sending forms to IANA that describe the data element or report for the registry entry.

3.1.2.1. Required Information

The required information must be formatted consistently using the following registration form. Form field names and values may appear on the same line.
3.1.2.1.1. Data Element Definition

Name of data element type

MUST be unique within the registry, enforced to be unique, and MUST be processed as case insensitive

Name of data element

MUST be unique within the registry, enforced to be unique, and MUST be processed as case insensitive

Reference document

MUST define the data element, SHOULD be a URL to a RFC, and SHOULD include the section number (or other detailed internal document reference), MAY be a URL to any document available under equivalent terms

Registrant

Will be IESG for initial entries and all Standards Track specifications; otherwise as specified by the registrant

Status

MUST be one of active, inactive, or unknown

3.1.2.2. Registration Processing

Registrants should send each registration form to IANA with a single record for incorporation into the registry. Send the form via email to iana@iana.org or complete the online form found on the IANA website. The subject line should indicate whether the enclosed form represents an insertion of a new record (indicated by the word "INSERT" in the subject line) or a replacement of an existing record (indicated by the word "MODIFY" in the subject line). At no time can a record be deleted from the registry. On receipt of the registration request, IANA will initiate review by the designated expert(s) if appropriate, who will evaluate the request using the criteria in Section 3.1.1 in consultation with the regext mailing list.
3.1.2.3. Updating Report Definition Registry Entries

When submitting changes to existing registry entries, include text in the "Notes" field of the registration form describing the change. Under normal circumstances, registry entries are only to be updated by the registrant. If the registrant becomes unavailable or otherwise unresponsive, the designated expert can submit a registration form to IANA to update the registrant information. Entries can change state from "Active" to "Inactive" and back again as long as state-change requests conform to the processing requirements identified in this document. In addition to entries that become "Inactive" due to a lack of implementation, entries for which a specification becomes consistently unavailable over time should be marked "Inactive" by the designated expert until the specification again becomes reliably available.

3.2. Initial assignments

3.2.1. Data Element Definition in IANA Registry

--- BEGIN FORM ---

Name of data element:
Domain Name
Reference:
This RFC Section 2.1.
Registrant:
IESG, iesg@ietf.org
Status:
Active
--- END FORM ---
--- BEGIN FORM ---
Name of data element:
............
Reference:
4. Security Considerations

This specification does not consider the issues of distribution or access to the reports that are created and thus does not introduce any new security concerns that are not already present in the local environment in which the report is created.

A security principle to keep in mind as new reports are developed is that it is considered a bad practice to report or disclose security information. In the case of the registration system upon which this reporting mechanism is based, the authInfo code is a specific example of a data element that SHOULD NOT be included in a report.

5. Privacy Considerations

This specification defines a mechanism for policy comparison based on data in a registration system. Some of that data is likely to be considered personally identifiable information (PII) and thus would be subject to privacy protection according to an applicable privacy regulation. It is outside the scope of this specification to address those specific concerns. Implementors are urged to consider these issues with their local legal authority and develop appropriate requirements for their work.

6. Internationalization Considerations

The character encoding for the file contents MUST use UTF-8. Throughout this document A-LABEL is indicated as a SHOULD and that MUST be interpreted as follows. All domain name labels MUST be in A-LABEL format if it is possible to represent it as an A-LABEL, otherwise U-LABEL MAY be used.

7. Draft Change Log

-01: Updated abstract to clarify that this draft does not intend to set policy.
-01: Updated definitions in 2.1, 2.4, 2.5, 2.6, 2.7 to remove normative reference to the EPP spec.

-01: Updated 2. Data Element specification to note local interpretation expected for any legal definitions.

-01: Added TBD to policy-related items, all data-related elements wrt format.

-01: Moved several items from informative to normative references.

8. Acknowledgements

With many thanks to James Galvin and Rod Rasmussen for their advice and feedback on this data dictionary.

9. References

9.1. Informative References


9.2. Normative References


[ITU.E164.2005]


Authors’ Addresses

Heather Flanagan (editor)
Edgemoor Research Institute
Email: hlf@sphericalcowconsulting.com

Steve Crocker
Edgemoor Research Institute
Email: steve@shinkuro.com
Use of Internationalized Email Addresses in the Extensible Provisioning Protocol (EPP)
draft-ietf-regext-epp-eai-08

Abstract

This document describes an EPP extension that permits usage of Internationalized Email Addresses in the EPP protocol and specifies the terms when it can be used by EPP clients and servers. The Extensible Provisioning Protocol (EPP), being developed before appearing the standards for Internationalized Email Addresses (EAI), does not support such email addresses.

TO BE REMOVED on turning to RFC: The document is edited in the dedicated github repo (https://github.com/beldmit/eppeai). Please send your submissions via GitHub.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction ............................................. 3
   1.1. Conventions Used in This Document .................. 3
2. Migrating to Newer Versions of This Extension .............. 3
3. Email Address Specification .............................. 4
4. Functional Extension .................................... 4
5. Internationalized Email Addresses (EAI) Functional Extension ............................................. 5
   5.1. Scope of Functional Extension ........................ 5
   5.2. Signaling Client and Server Support ................. 5
   5.3. Functional Extension Behavior ....................... 5
      5.3.1. EAI Functional Extension Negotiated ............. 5
      5.3.2. EAI Functional Extension Not Negotiated ......... 6
6. IANA Considerations ..................................... 7
   6.1. XML Namespace ........................................ 7
   6.2. EPP Extension Registry .............................. 7
7. Implementation Status ..................................... 8
   7.1. Verisign EPP SDK ................................... 8
8. Security Considerations .................................. 9
9. Acknowledgments ......................................... 9
10. References ................................................ 9
    10.1. Normative References ............................... 9
    10.2. Informative References ............................. 10
Appendix A. Change History .................................. 10
   A.1. Change from 00 to 01 ................................ 11
   A.2. Change from 01 to 02 ................................ 11
   A.3. Change from 02 to 03 ................................ 11
   A.4. Change from 03 to 04 ................................ 11
   A.5. Change from 04 to the regext 01 version ............. 11
   A.6. Change from the regext 01 to regext 02 version ...... 11
   A.7. Change from the regext 02 to regext 03 version ...... 11
   A.8. Change from the regext 03 to regext 04 version ...... 12
   A.9. Change from the regext 04 to regext 05 version ...... 12
   A.10. Change from the regext 05 to regext 06 version ...... 12
   A.11. Change from the regext 06 to regext 07 version ...... 12
   A.12. Change from the regext 07 to regext 08 version ...... 12
Authors’ Addresses .......................................... 12
1. Introduction

[RFC6530] introduced the framework for Internationalized Email Addresses. To make such addresses more widely accepted, the changes to various protocols need to be introduced.

This document describes an Extensible Provisioning Protocol (EPP) extension that permits usage of Internationalized Email Addresses in the EPP protocol and specifies the terms when it can be used by EPP clients and servers. A new form of EPP extension, referred to as a Functional Extension, is defined and used to apply the rules for the handling of email address elements in all of the [RFC5730] extensions negotiated in the EPP session, which include the object and command-responses extensions. The described mechanism can be applied to any object or command-response extension that uses an email address.

The Extensible Provisioning Protocol (EPP) specified in [RFC5730] is a base document for object management operations and an extensible framework that maps protocol operations to objects. The specifics of various objects managed via EPP is described in separate documents. This document is only referring to an email address as a property of a managed object, such as the <contact:email> element in the EPP contact mapping [RFC5733] or the <org:email> element in the EPP organization mapping [RFC8543], and command-response extensions applied to a managed object.

1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Migrating to Newer Versions of This Extension

Servers that implement this extension SHOULD provide a way for clients to progressively update their implementations when a new version of the extension is deployed. A newer version of the extension is expected to use an XML namespace with a higher version number than the prior versions.
3. Email Address Specification

Support of non-ASCII email address syntax is defined in RFC 6530 [RFC6530]. This mapping does not prescribe minimum or maximum lengths for character strings used to represent email addresses. The exact syntax of such addresses is described in Section 3.3 of [RFC6531]. The validation rules introduced in RFC 6531 are considered to be followed.

The definition of email address in the EPP RFCs, including Section 2.6 of [RFC5733] and Section 4.1.2, 4.2.1, and 4.2.5 of [RFC8543], references [RFC5322] for the email address syntax. The XML schema definition in Section 4 of [RFC5733] and Section 5 of [RFC8543] defines the "email" element using the type "eppcom:minTokenType", which is defined in Section 4.2 of [RFC5730] as an XML schema "token" type with minimal length of one. The XML schema "token" type will fully support the use of EAI addresses, so the primary application of the EAI extension is to apply the use of [RFC6531] instead of [RFC5322] for the email address syntax. Other EPP extensions may follow the formal syntax definition using the XML schema type "eppcom:minTokenType" and the [RFC5322] format specification, where this extension applies to all EPP extensions with the same or similar definitions.

The email address format is formally defined in Section 3.4.1 of [RFC5322], which only consists of printable US-ASCII characters for both the local-part and the domain ABNF rules. [RFC6531] extends the Mailbox, Local-part and Domain ABNF rules in [RFC5321] to support "UTF8-non-ascii", defined in Section 3.1 of [RFC6532], for the local-part and U-label, defined in Section 2.3.2.1 of [RFC5890], for the domain. By applying the syntax rules of [RFC5322], the EPP extensions will change from supporting only ASCII characters to supporting Internationalized characters both in the email address local-part and domain-part.

4. Functional Extension

[RFC5730] defines three types of extensions at the protocol, object, and command-response level, which impact the structure of the EPP messages. A Functional Extension applies a functional capability to an existing set of EPP extensions and properties. The scope of the applicable EPP extensions and applicable extension properties are defined in the Functional Extension along with the requirements for the servers and clients that support it. The Functional Extension needs to cover the expected behavior of the supporting client or server when interacting with an unsupporting client or server. Negotiating support for a Functional Extension is handled using the EPP Greeting and EPP Login services.
5. Internationalized Email Addresses (EAI) Functional Extension

5.1. Scope of Functional Extension

The functional extension applies to all object extensions and command-response extensions negotiated in the EPP session that include email address properties. Examples include the <contact:email> element in the EPP contact mapping [RFC5733] or the <org:email> element in the EPP organization mapping [RFC8543]. All registry zones (e.g., top-level domains) authorized for the client in the EPP session apply. There is no concept of a per-client, per-zone, per-extension, or per-field setting that is used to indicate support for EAI, but instead it’s a global setting that applies to the EPP session.

5.2. Signaling Client and Server Support

The client and the server can signal support for the functional extension using a namespace URI in the login and greeting extension services respectively. The namespace URI "urn:ietf:params:xml:ns:epp:eai-1.0" is used to signal support for the functional extension. The client includes the namespace URI in an <svcExtension> <extURI> element of the [RFC5730] <login> Command. The server includes the namespace URI in an <svcExtension> <extURI> element of the [RFC5730] Greeting.

5.3. Functional Extension Behavior

5.3.1. EAI Functional Extension Negotiated

If both client and server have indicated the support of the EAI addresses during the session establishment, it implies possibility to process the EAI address in any message having an email property during the established EPP session. Below are the server and client obligations when the EAI extension has been successfully negotiated in the EPP session.

The server MUST satisfy the following obligations when the EAI extension has been negotiated:

* Accept EAI compatible addresses for all email properties in the EPP session negotiated object extensions and command-response extensions. For example the <contact:email> element in [RFC5733] and the <org:email> element in [RFC8543].

* Accept EAI compatible addresses for all registry zones (e.g., top-level domains) authorized for the client in the EPP session.
* Email address validation based on EAI validation rules defined in Section 3

* Storage of email properties that support internationalized characters.

* Return EAI compatible addresses for all email properties in the EPP responses.

The client MUST satisfy the following obligations when THE EAI extension has been negotiated:

* Provide EAI compatible addresses for all e-mail properties in the EPP session negotiated object extensions and command-response extensions. For example the <contact:email> element in [RFC5733] and the <org:email> element in [RFC8543].

* Provide EAI compatible addresses for all registry zones (e.g., top-level domains) authorized for the client in the EPP session.

* Accept EAI compatible addresses in the EPP responses for all email properties in the EPP session negotiated object extensions and command-response extensions.

5.3.2. EAI Functional Extension Not Negotiated

The lack of EAI support can cause data and functional issues, so an EAI supporting client or server needs to handle cases where the opposite party doesn’t support EAI. Below are the server and client obligations when the EAI extension is not negotiated due to the lack of support by the peer.

The EAI supporting server MUST satisfy the following obligations when the client does not support the EAI extension:

* When the email property is required in the EPP command, the server SHOULD validate the email property sent by the client using the ASCII email validation rules.

* When the email property is optional in the EPP command, if the client supplies the email property the server SHOULD validate the email property using the ASCII email validation rules.

* When the email property is required in the EPP response, the server MUST validate whether the email property is an EAI address and if so return the error code 2308 "Data management policy violation".
* When the email property is optional in the EPP response and is provided, the server MUST validate whether the email property is an EAI address and if so return the error code 2308 "Data management policy violation".

The EAI supporting client MUST satisfy the following obligations when the server does not support the EAI extension:

* When the email property is required in the EPP command and the email property is an EAI address, the client MUST provide an ASCII email address. The provided email address should provide a way to contact the registrant. It can be a secondary ASCII email address or registrar-provided proxy email address.

* When the email property is optional in the EPP command and the email property is an EAI address with no alternative ASCII address, the client SHOULD omit the email property. If the email property is provided, the client MUST provide an ASCII email address. The provided email address should provide a way to contact the registrant. It can be a secondary ASCII email address or registrar-provided proxy email address.

6. IANA Considerations

6.1. XML Namespace

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in RFC 3688 [RFC3688]. The following URI assignment should be made by IANA:

Registration request for the eai namespace:

URI: urn:ietf:params:xml:ns:epp:eai-1.0
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.

Registration request for the eai XML Schema:

Registrant Contact: IESG
XML: See the "Formal Syntax" section of this document.

6.2. EPP Extension Registry

The EPP extension described in this document should be registered by IANA in the "Extensions for the Extensible Provisioning Protocol (EPP)" registry described in RFC 7451 [RFC7451]. The details of the registration are as follows:
7. Implementation Status

Note to RFC Editor: Please remove this section and the reference to RFC 7942 [RFC7942] before publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942 [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942 [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

7.1. Verisign EPP SDK

Organization: Verisign Inc.

Name: Verisign EPP SDK

Description: The Verisign EPP SDK includes both a full client implementation and a full server stub implementation of draft-ietf-regext-epp-eai.

Level of maturity: Development

Coverage: All aspects of the protocol are implemented.
8. Security Considerations

Registries SHOULD validate the domain names syntax in the provided email addresses to reduce the risk of future usability errors. It is RECOMMENDED to validate all code points in the domain names according to IDNA2008 [RFC5892].

9. Acknowledgments

The authors would like to thank Alexander Mayrhofer, Gustavo Lozano, Jody Kolker, John Levine, Klaus Malorny, Marco Schrieck, Mario Loffredo, Patrick Mevzek, Scott Hollenbeck, Taras Heichenko, and Thomas Corte for their careful review and valuable comments.

10. References

10.1. Normative References


10.2. Informative References


Appendix A. Change History
A.1. Change from 00 to 01

1. Changed from update of RFC 5733 to use the "Placeholder Text and a New Email Element" EPP Extension approach.

A.2. Change from 01 to 02

1. Fixed the XML schema and the XML examples based on validating them.

2. Added James Gould as co-author.

3. Updated the language to apply to any EPP object mapping and to use the EPP contact mapping as an example.

4. Updated the structure of document to be consistent with the other Command-Response Extensions.

5. Replaced the use of "eppEAI" in the XML namespace and the XML namespace prefix with "eai".

6. Changed to use a pointed XML namespace with "0.2" instead of "1.0".

A.3. Change from 02 to 03

1. The approach has changed to use the concept of Functional EPP Extension.

2. The examples are removed

A.4. Change from 03 to 04

1. More detailed reference to email syntax is provided

2. The shortened eai namespace reference is removed

A.5. Change from 04 to the regext 01 version

1. Provided the recommended placeholder value

A.6. Change from the regext 01 to regext 02 version

1. Removed the concept of the placeholder value

A.7. Change from the regext 02 to regext 03 version
1. Changed to use a pointed XML namespace with "0.3" instead of "0.2".

2. Some wording improvements

A.8. Change from the regext 03 to regext 04 version

1. Some nitpicking

A.9. Change from the regext 04 to regext 05 version

1. Some nitpicking

2. The "Implementation considerations" section is removed

A.10. Change from the regext 05 to regext 06 version

1. Some nitpicking

A.11. Change from the regext 06 to regext 07 version

1. Namespace version set to 1.0

A.12. Change from the regext 07 to regext 08 version

1. Information about implementations is provided.

2. Acknowledgments section is added.

3. Reference to RFC 7451 is moved to Informative.

4. IPR information is provided

5. Sections are reordered to align with the other regext documents

Authors’ Addresses

Dmitry Belyavskiy
8 marta st.
Moscow
127083
Russian Federation
Phone: +7 916 262 5593
Email: beldmit@gmail.com
Using JSContact in Registration Data Access Protocol (RDAP) JSON Responses

draft-ietf-regext-rdap-jscontact-12

Abstract

This document describes an RDAP extension which represents entity contact information in JSON responses using JSContact.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 November 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.
Table of Contents

1. Introduction ............................................. 3
   1.1. Rationale ........................................... 3
   1.2. Conventions Used in This Document ................. 3
2. JSContact ................................................. 4
3. Using JSCard objects in RDAP Responses .................. 4
   3.1. RDAP Query Parameters ............................... 10
4. Transition Considerations ................................. 10
   4.1. RDAP Features Supporting a Transition Process ... 10
      4.1.1. Notices and Link Relationships ............... 10
      4.1.2. rdapConformance Property .................... 11
      4.1.3. Query Parameters ............................. 11
   4.2. Transition Procedure ................................ 11
      4.2.1. Goals .......................................... 11
      4.2.2. Transition Stages ................................ 11
         4.2.2.1. Stage 1: only jCard provided .......... 12
         4.2.2.2. Stage 2: jCard sunset .................. 12
         4.2.2.3. Stage 3: jCard deprecation ............ 13
         4.2.2.4. Stage 4: jCard deprecated ............. 15
         4.2.2.5. Length ..................................... 15
5. Implementation Status .................................... 15
   5.1. IIT-CNR/Registro.it RDAP Server .................... 15
   5.2. IIT-CNR/Registro.it RDAP Client .................... 16
   5.3. client.rdap.org .................................... 16
   5.4. CentralNic Registry ................................ 16
6. IANA Considerations ..................................... 17
7. Security Considerations .................................. 17
8. Acknowledgements ........................................ 17
9. References ................................................. 17
   9.1. Normative References ................................ 17
   9.2. Informative References ............................. 19
Appendix A. jCard-JSContact Mapping ........................ 19
Appendix B. Change Log ..................................... 21
   B.1. Change from 00 to 01 ................................ 21
   B.2. Change from 01 to 02 ................................ 21
   B.3. Change from 02 to 03 ................................ 22
   B.4. Change from 03 to 04 ................................ 22
   B.5. Initial WG version .................................. 22
   B.6. Change from 00 to 01 ................................ 22
   B.7. Change from 01 to 02 ................................ 22
   B.8. Change from 02 to 03 ................................ 22
   B.9. Change from 03 to 04 ................................ 22
   B.10. Change from 04 to 05 ............................... 23
   B.11. Change from 05 to 06 ............................... 23
   B.12. Change from 06 to 07 ............................... 23
   B.13. Change from 07 to 08 ............................... 23
   B.14. Change from 08 to 09 ............................... 23
1. Introduction

This document specifies an extension to the Registration Data Access Protocol (RDAP) that allows RDAP servers to use JSContact [I-D.ietf-calext-jscontact] to represent the contact information associated with entities in RDAP responses, instead of jCard [RFC7095]. It also describes the process by which an RDAP server can transition from jCard to JSContact. RDAP query and response extensions are defined to facilitate the transition process.

1.1. Rationale

According to the feedback from RDAP Pilot Working Group [RDAP-PILOT-WG], a group of RDAP server implementers representing registries and registrars of generic TLDs, the most commonly raised implementation concern, for both servers and client implementers, related to the use of jCard [RFC7095] to represent the contact information associated with entities. Working Group members reported jCard to be unintuitive, complicated to implement for both clients and servers, and incompatible with best practices for RESTful APIs.

JSContact [I-D.ietf-calext-jscontact] provides a simpler and more efficient representation for contact information with regard to time and effort saved in processing it. In addition, similarly to jCard, it provides a means to represent internationalised and unstructured contact information. Support for internationalised contact information has been recognised being necessary to facilitate the future internationalisation of registration data directory services.

1.2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.
2. JSContact

The JSContact specification defines a data model and JSON representation of contact information that can be used for data storage and exchange in address book or directory applications. It aims to be an alternative to the vCard data format [RFC6350] and to be unambiguous, extendable and simple to process. In contrast with jCard, it is not a direct mapping from the vCard data model and expands semantics where appropriate.

The JSContact specification declares two main object types: "Card", which represents a single contact card, and "CardGroup" which represents a collection of Card objects. For the purpose of this document, only Card objects are considered. To avoid confusion, in the following of this document, the term "JSCard" is used to refer to "JSContact Card".

JSCard differs from jCard in that it:

* follows an object-oriented rather than array-oriented approach;
* is simple to process;
* requires no extra work in serialization/deserialization from/to a data model;
* includes no "jagged" arrays;
* prefers maps rather than arrays to implement collections.

[I-D.ietf-calext-jscontact-vcard] provides informational guidance on the conversion of jCard into JSCard, and vice versa. Appendix A shows JSContact counterparts for the most commonly used jCard properties in an RDAP response.

3. Using JSCard objects in RDAP Responses

Entity objects in RDAP responses MAY include a "jscard_0" property whose value is a JSCard object instead of the "vCardArray" property defined in [RFC9083].

Servers returning the "jscard_0" property in their response MUST include "jscard_0" in the "rdapConformance" array.

The JSCard "uid" property SHOULD contain the same value as the RDAP "handle" property.

Since most of the JSCard collections are represented as maps, map keys must be defined. To aid interoperability, RDAP providers are RECOMMENDED to use as map keys the following string values and labels defined in [RFC5733]:

Loffredo & Brown Expires 3 November 2022 [Page 4]
* "org" in the "organizations" map when there is a single <contact:org> element. If both internationalised and localized forms exist, the key MUST be used for the internationalised form;
* "addr" in the "addresses" map when there is a single <contact:addr> element. If both internationalised and localized forms exist, the key MUST be used for the internationalised form;
* "email" in the "emails" map for the <contact:email> element;
* "voice" in the "phones" map for the <contact:voice> element;
* "fax" in the "phones" map for the <contact:fax> element.

If present, the localized versions of name, organization and postal address MUST be inserted into the "localizations" map. The following is an elided example of an RDAP entity lookup response including a JSCard object that presents the "localizations" map (See PDF for non-ASCII character string).

```json
"jscard_0": {
    "@type": "Card",
    "uid": "7e0636f5-e48f-4a32-ab96-b57e9c07c7aa",
    "fullName": "Vasya Pupkin",
    "organizations": {
        "org": {
            "@type": "Organization",
            "name": "My Company"
        }
    },
    "addresses": {
        "addr": {
            "@type": "Address",
            "street": [ {
                "@type": "StreetComponent",
                "value": "1 Street"
            }, {
                "@type": "StreetComponent",
                "type": "postOfficeBox",
                "value": "01001"
            } ],
            "locality": "Kyiv",
            "countryCode": "UA"
        }
    },
    "localizations": {
        "ua": {
            "/jscard_0/addresses/addr": {
                "@type": "Address",
                "street": [ {
                    
```
"type" : "StreetComponent",
"value" : "1, "
}, {
  "type" : "postOfficeBox",
  "value" : "01001"
} ],
"locality" : "",
"countryCode" : "UA"
},
"jscard_0/fullName" : "",
"jscard_0/organizations/org" : {
@type" : "Organization",
"name" : ""
}
}
}
...

Figure 1: Example of handling localizations in JSContact

Implementers MAY use different mapping schemes to define keys for additional entries of the aforementioned maps or others. For example, a mapping scheme may consist in using a trivial sequential number (e.g. "url-1", "url-2", etc.)

The following is an example of an RDAP entity including a JSCard object that has been converted from the example in section 5.1 of [RFC9083].

{
  "rdapConformance": [
    "rdap_level_0",
    "jscard_0"
  ],
  "objectClassName" : "entity",
  "handle": "XXXX",
  "jscard_0": {
    "@type": "Card",
    "uid": "XXXX",
    "fullName": "Joe User",
    "name": {
      "@type": "Name",
      "components": [
      
        "@type": "NameComponent",
      ]
    }
  }
}
"type": "surname",
"value": "User"
},
{
"@type": "NameComponent",
"type": "personal",
"value": "Joe"
},
{
"@type": "NameComponent",
"type": "suffix",
"value": "ing. jr"
},
{
"@type": "NameComponent",
"type": "suffix",
"value": "M.Sc."
}
]
},
"kind": "individual",
"preferredContactLanguages": {
"fr": {
"@type": "ContactLanguage",
"pref": 1
},
"en": {
"@type": "ContactLanguage",
"pref": 2
}
},
"organizations": {
"org": {
"@type": "Organization",
"name": "Example"
}
},
"titles": {
"title": {
"@type": "Title",
"title": "Research Scientist"
},
"role": {
"@type": "Title",
"title": "Project Lead"
}
},
"addresses": {
"addr": {
  "@type": "Address",
  "contexts": {
    "work": true
  },
  "street": [
    {
      "@type": "StreetComponent",
      "type": "name",
      "value": "4321 Rue Somewhere"
    },
    {
      "@type": "StreetComponent",
      "type": "extension",
      "value": "Suite 1234"
    }
  ],
  "locality": "Quebec",
  "region": "QC",
  "postcode": "G1V 2M2",
  "country": "Canada",
  "countryCode": "CA",
  "coordinates": "geo:46.772673,-71.282945",
  "timeZone": "Etc/GMT+5"
},
"home": {
  "@type": "Address",
  "contexts": {
    "private": true
  },
  "fullAddress": "123 Maple Ave\nSuite 90001\nVancouver\nBC\n1239\n"
},
"phones": {
  "voice": {
    "@type": "Phone",
    "contexts": {
      "work": true
    },
    "features": {
      "voice": true,
      "cell": true,
      "video": true,
      "text": true
    },
    "pref": 1,
    "phone": "tel:+1-555-555-1234;ext=102"
  }
}


```
{
  "emails": {
    "email": {
      "@type": "EmailAddress",
      "contexts": {
        "work": true
      },
      "email": "joe.user@example.com"
    }
  },
  "online": {
    "key": {
      "@type": "Resource",
      "contexts": {
        "work": true
      },
      "type": "publicKey",
      "resource": "http://www.example.com/joe.user/joe.asc"
    },
    "url": {
      "@type": "Resource",
      "contexts": {
        "private": true
      },
      "type": "uri",
      "resource": "http://example.org"
    }
  },
  "roles": ["registrar"],
  "publicIds": [
    {
      "type": "IANA Registrar ID",
      "identifier": "1"
    }
  ],
  "remarks": [
    {
      "description": [
        "She sells sea shells down by the sea shore.",
        "Originally written by Terry Sullivan."
      ]
    }
  ],
  "links": [
    {
      "value": "http://example.com/entity/XXXX",
      "rel": "self",
```
"href":"http://example.com/entity/XXXX",
"type" : "application/rdap+json"
}

"events": [  
{  
"eventAction":"registration",
"eventDate":"1990-12-31T23:59:59Z"
  }
],

"asEventActor": [  
{  
"eventAction":"last changed",
"eventDate":"1991-12-31T23:59:59Z"
  }
]
}

Figure 2: Example of using JSContact in RDAP response

3.1. RDAP Query Parameters

Two new query parameters are defined for the purpose of this document.

The query parameters are OPTIONAL extensions of path segments defined in [RFC9082]. They are as follows:

* "jscard": a boolean value that allows a client to request the "jscard_0" property in the RDAP response;
* "jcard": a boolean value that allows a client to request the "vcardArray" property in the RDAP response.

These parameters are furtherly explained in Section 4.

4. Transition Considerations

4.1. RDAP Features Supporting a Transition Process

4.1.1. Notices and Link Relationships

RDAP allows servers to communicate service information to clients through notices. According to Section 4.3 of [RFC9083], an RDAP response may contain one or more notice objects. Each notice may include a set of link objects, which can be used to provide clients with references and documentation. These link objects may have a "rel" property which defines the relationship type, as described in [RFC8288], Section 4. The transition process outlined in this
document uses two link relation types, namely "related" and "alternate", described in [RFC8288].

4.1.2. rdapConformance Property

The information about the specifications used in the construction of the response is also described by the strings which appear in the "rdapConformance" property of the RDAP response.

4.1.3. Query Parameters

Clients can ask servers to use the query parameters defined in Section 3.1 in accordance with [RFC9082].

4.2. Transition Procedure

The principles of the procedure for jCard to JSCard transition are based on the best practices in [API-DEPRECATION].

The procedure consists of four contiguous stages. During the procedure, the presence of "jscard_0" tag in the rdapConformance array indicates that JSCard is returned instead of jCard. The date and time format used to notify clients about the stages of this procedure is defined in [RFC3339].

4.2.1. Goals

The procedure described in this document aims to achieve the following goals:

* only one contact representation would be included in the response;
* the response would always be compliant to [RFC9083] because:
  - being the "jscard_0" property a response extension, its presence would be signaled by the "jscard_0" conformance tag;
  - being "vcardArray" property optional in a response, its absence would be allowed;
* clients would be informed about the transition timeline;
* the backward compatibility would be guaranteed throughout the transition;
* servers and clients could execute their transitions independently.

4.2.2. Transition Stages
4.2.2.1. Stage 1: only jCard provided

This stage corresponds to providing jCard as the default contact card [RFC9083]. The RDAP server is not able to provide an alternate contact card. The rdapConformance array MUST NOT contain the "jcard_0" tag.

4.2.2.2. Stage 2: jCard sunset

During this stage, the server uses jCard by default, but the RDAP server will return JSCard if the client sets the query parameter "jscard" to 1/true/yes. The rdapConformance array MUST contain the "jcard_0" tag if JSCard is returned.

From this stage on, the RDAP server MUST include the "jcard_0" tag in the rdapConformance array of the help response to signal clients that JSCard can be returned instead of jCard.

The RDAP server SHOULD include a notice titled "jCard sunset end". Such a notice includes a description reporting the jCard sunset end date and time and two OPTIONAL links:

* "related": a link to a URI-identified resource documenting the transition procedure;
* "alternate": if JSCard is not requested, a link to an alternate result view identified by the current query string plus the parameter "jscard" set to 1/true/yes (Figure 3); otherwise, only the "related" link can be provided (Figure 4).

"notices": [
{
  "title": "jCard sunset end",
  "description": "[2022-07-01T00:00:00Z]",
  "links": [
    {
      "value": "http://example.net/entity/XXXX",
      "rel": "related",
      "type": "text/html",
      "href": "http://www.example.com/jcard_deprecation.html"
    },
    {
      "value": "http://example.net/entity/XXXX",
      "rel": "alternate",
      "type": "application/rdap+json",
      "href": "http://example.net/entity/XXXX?jscard=1"
    }
  ]
} ]
4.2.2.3. Stage 3: jCard deprecation

This stage corresponds to the provisioning of JSCard by default, but the RDAP will return jCard if the client sets the query parameter "jcard" to 1/true/yes. The rdapConformance array contains the "jscard_0" tag unless jCard is returned. The "jcard" query parameter MUST be ignored.

The RDAP server SHOULD return a notice titled "jCard deprecation end". Such a notice includes a description reporting the jCard deprecation end date and time and two OPTIONAL links:

* "related": a link to a URI-identified resource documenting the transition procedure;
* "alternate": if jCard is not requested, a link to an alternate result view identified by the current query string plus the parameter "jcard" set to 1/true/yes (Figure 5); otherwise, a link to the result view identified by the current query string without the parameter "jcard" (Figure 6).
"notices": [
  {
    "title": "jCard deprecation end",
    "description": ["2022-12-31T23:59:59Z"],
    "links": [
      {
        "value": "http://example.net/entity/XXXX",
        "rel": "related",
        "type": "text/html",
        "href": "http://www.example.com/jcard_deprecation.html"
      },
      {
        "value": "http://example.net/entity/XXXX",
        "rel": "alternate",
        "type": "application/rdap+json",
        "href": "http://example.net/entity/XXXX?jcard=1"
      }
    ]
  }
]

Figure 5: jCard deprecation - jCard not requested

"notices": [
  {
    "title": "jCard deprecation end",
    "description": ["2022-12-31T23:59:59Z"],
    "links": [
      {
        "value": "http://example.net/entity/XXXX?jcard=1",
        "rel": "related",
        "type": "text/html",
        "href": "http://www.example.com/jcard_deprecation.html"
      },
      {
        "value": "http://example.net/entity/XXXX?jcard=1",
        "rel": "alternate",
        "type": "application/rdap+json",
        "href": "http://example.net/entity/XXXX"
      }
    ]
  }
]

Figure 6: jCard deprecation - jCard requested
4.2.2.4. Stage 4: jCard deprecated

This stage corresponds to providing JSCard as default contact card. The RDAP server is not able to provide an alternate contact card. The rdapConformance array always contains "jscard_0" tag. The RDAP server doesn't include any notice about the jCard deprecation process. Both "jscard" and "jcard" query parameters MUST be ignored.

4.2.2.5. Length

The length of both jCard sunset and jCard deprecation periods are not fixed by this specification. Best practices in REST API deprecation suggest that, depending on the deprecated API’s reach, user base and service offering, a convenient time could be anywhere between 3 - 8 months. Anyway, RDAP providers are RECOMMENDED to monitor the server log to figure out whether declared times need to be changed to meet client requirements.

5. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942 [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

5.1. IIT-CNR/Registro.it RDAP Server

* Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it
* Location: https://rdap.pubtest.nic.it/
* Description: This implementation includes support for RDAP queries using data from the public test environment of .it ccTLD.
* Level of Maturity: This is an "alpha" test implementation.
* Coverage: This implementation includes all of the features described in this specification.
* Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

5.2. IIT-CNR/Registro.it RDAP Client

* Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it
* Location: https://web-rdap.pubtest.nic.it/
* Description: This is a Javascript web-based RDAP client. RDAP responses are retrieved from RDAP servers by the browser, parsed into an HTML representation, and displayed in a format improving the user experience. RDAP responses containing JSCard objects are handled identically to those containing jCard objects. Raw versions of RDAP responses including either jCard or JSCard objects are provided.
* Level of Maturity: This is an "alpha" test implementation.
* Coverage: This implementation includes all of the features described in this specification.
* Contact Information: Francesco Donini, francesco.donini@iit.cnr.it

5.3. client.rdap.org

* Location: https://client.rdap.org/
* Description: This is a web-based "single page" RDAP client. RDAP responses are retrieved from RDAP servers by the browser, and parsed into an HTML representation. RDAP responses containing JSCard objects are handled identically to those containing jCard objects.
* Level of Maturity: This is an "alpha" test implementation.
* Coverage: This implementation implements client support for parsing JSCard objects in RDAP responses.
* Contact Information: Gavin Brown, feedback@rdap.org

5.4. CentralNic Registry

* Responsible Organization: CentralNic Group PLC
* Location: https://rdap.centralnic.com/{tld}
* Description: This server is the product RDAP service for all top-level domains on the CentralNic registry platform.
* Level of Maturity: Production quality.
* Coverage: This implementation includes all of the features described in this specification.
* Contact Information: support@centralnic.com
6. IANA Considerations

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

* Extension identifier: jscard_0
* Registry operator: Any
* Published specification: This document.
* Contact: IETF <iesg@ietf.org>
* Intended usage: This extension represents a contact card provided in an RDAP response according to the JSContact specification [I-D.ietf-calext-jscontact].

7. Security Considerations

Unlike jCard, the formatted name as well as any other personally identifiable information is not required in JSCard. The only mandatory property, namely "uid", is not a sensitive information as it happens, instead, for the "fn" property in jCard. Therefore, redacted properties can be merely excluded without using placeholder values. This means that, with reference to what is described in [I-D.ietf-regext-rdap-redacted], only the "Removal" method can be used for redacting JSContact properties whereas the "Empty Value" is also used for redacting jCard.

8. Acknowledgements

The authors would like to acknowledge the following individuals for their contributions to this document: Jasdip Singh and Francesco Donini.

9. References

9.1. Normative References

[I-D.ietf-calext-jscontact]

[I-D.ietf-calext-jscontact-vcard]
Internet-Draft           Using JSContact in RDAP                May 2022


Appendix A.  jCard-JSContact Mapping

Provided that the keys defined in Section 3 are used for the 
JSContact maps, the mapping between the most commonly used jCard 
properties in an RDAP response and their JSContact counterparts is 
shown in the following.  The mapping is done through the use of a 
JSONPath expression [I-D.ietf-jsonpath-base].

jCard property:  fn
Reference:  Section 6.2.1 of [RFC6350]
Path:  $.vcardArray[1][?(@[0]=='fn')][3]
JSContact property:  fullName
Reference:  Section 2.2.2 of [I-D.ietf-calext-jscontact]
Path:  $.jscard_0.fullName

jCard property:  org
Reference:  Section 6.6.4 of [RFC6350]
Path:  $.vcardArray[1][?(@[0]=='org')][3]
JSContact property:  Organization.name
Reference:  Section 2.2.4 of [I-D.ietf-calext-jscontact]
Path: $..jscard_0.organizations.org.name

jCard property: tel with type="voice"
Reference: Section 6.4.1 of [RFC6350]
Path: $..vcardArray[1][?(@[1].type=='voice')][3]
JSContact property: Phone.phone
Reference: Section 2.3.2 of [I-D.ietf-calext-jscontact]
Path: $..jscard_0.phones.voice.phone

jCard property: tel with type="fax"
Reference: Section 6.4.1 of [RFC6350]
Path: $..vcardArray[1][?(@[1].type=='fax')][3]
JSContact property: Phone.phone
Reference: Section 2.3.2 of [I-D.ietf-calext-jscontact]
Path: $..jscard_0.phones.fax.phone

jCard property: email
Reference: Section 6.4.2 of [RFC6350]
Path: $..vcardArray[1][?(@[0]=='email')][3]
JSContact property: Email.email
Reference: Section 2.3.1 of [I-D.ietf-calext-jscontact]
Path: $..jscard_0.emails.email.email

jCard property: "post office box" component of adr
Reference: Section 6.3.1 of [RFC6350]
Path: $..vcardArray[1][?(@[0]=='adr')][3][1]
JSContact property: "postOfficeBox" StreetComponent of Address.street
Reference: Section 2.4.1 of [I-D.ietf-calext-jscontact]
Path: $..jscard_0.addresses.addr.street[?(@.type=='postOfficeBox')].value

jCard property: "street address" component of adr
Reference: Section 6.3.1 of [RFC6350]
Path: $..vcardArray[1][?(@[0]=='adr')][3][2]
JSContact property: "name" StreetComponent of Address.street
Reference: Section 2.4.1 of [I-D.ietf-calext-jscontact]
Path: $..jscard_0.addresses.addr.street[?(@.type=='name')].value

jCard property: "locality" component of adr
Reference: Section 6.3.1 of [RFC6350]
Path: $..vcardArray[1][?(@[0]=='adr')][3][3]
JSContact property: Address.locality
Reference: Section 2.4.1 of [I-D.ietf-calext-jscontact]
Path: $..jscard_0.addresses.addr.locality

jCard property: "region" component of adr
Reference: Section 6.3.1 of [RFC6350]
Path: $..vcardArray[1][?(@[0]=='adr')][3][4]
JSContact property: Address.region
Reference: Section 2.4.1 of [I-D.ietf-calext-jscontact]
Path: $..jscard_0.addresses.addr.region

jCard property: "postal code" component of adr
Reference: Section 6.3.1 of [RFC6350]
Path: $..vcardArray[1][?(@[0]=='adr')][3][5]
JSContact property: Address.postcode
Reference: Section 2.4.1 of [I-D.ietf-calext-jscontact]
Path: $..jscard_0.addresses.addr.postcode

jCard property: "country name" component of adr
Reference: Section 6.3.1 of [RFC6350]
Path: $..vcardArray[1][?(@[0]=='adr')][3][6]
JSContact property: Address.country
Reference: Section 2.4.1 of [I-D.ietf-calext-jscontact]
Path: $..jscard_0.addresses.addr.country

jCard property: "cc" parameter of adr
Reference: Section 3.1 of [RFC8605]
Path: $..vcardArray[1][?(@[0]=='adr')][1].cc
JSContact property: Address.countryCode
Reference: Section 2.4.1 of [I-D.ietf-calext-jscontact]
Path: $..jscard_0.addresses.addr.countryCode

Appendix B. Change Log

B.1. Change from 00 to 01

1. Changed category from "Best Current Practice" to "Standards Track"
2. Replaced the example of Figure 2
3. Changed the title of the "Migration from JCard to JSCard" section to "Transition Considerations"
4. Added Section 3.1
5. Updated Section 6
6. Updated Section 7
7. Rearranged the description of stage 1 in Section 4.2.2
8. Changed the names of the transition stages 1 and 2
9. Corrected Figure 3, Figure 5, Figure 6
10. Changed the rdapConformance tag "jscard_level_0" to "jscard"
11. Removed the "Best Practices for deprecating a REST API features" section, but added a useful reference.

B.2. Change from 01 to 02
1. Removed the sentence "which cannot be represented using jCard" in Section 1.1.

B.3. Change from 02 to 03

1. Updated section "Conventions Used in This Document".
2. Updated the contact in "IANA Considerations" section.
4. Added reference to RFC8174.
5. Other minor edits.

B.4. Change from 03 to 04

1. Updated the reference draft-dalal-deprecation-header to draft-ietf-httpapi-deprecation-header.

B.5. Initial WG version


B.6. Change from 00 to 01

1. Updated Section 3 and Figure 2.

B.7. Change from 01 to 02

1. Updated Section 2 and Figure 2.

B.8. Change from 02 to 03

1. Replaced references to obsolete RFC7482 and RFC7483 with RFC9082 and RFC9083.
2. Updated Section 3 and Figure 2.

B.9. Change from 03 to 04

1. Changed the references to Internet Drafts.
2. Added an example showing how localizations are treated in JSContact.
3. Changed the position of section "Goals" in Section 4.2.
4. Added three more implementations to Section 5.
5. Changed the rdapConformance tag "jcard" to "jscard_0"
6. Added clarifications addressing the feedback provided by Jasdip Singh about version -03.
7. Added Section 8.
8. Other minor edits.
B.10. Change from 04 to 05

1. Updated Figure 2 to make it compliant with draft-ietf-jmap-jscontact-09.

B.11. Change from 05 to 06

1. Reviewed the notices presented in Section 4.2.2.2 and Section 4.2.2.3.

B.12. Change from 06 to 07

1. Corrected the JSON Pointer expressions in Figure 1.
2. Other minor edits.

B.13. Change from 07 to 08

1. Corrected a nit in Figure 1.
2. Removed the reference to draft-ietf-httpapi-deprecation-header.
3. Replaced the "deprecation" link relation type with "related".
4. Moved the references to JSContact drafts to the "Normative References" section.

B.14. Change from 08 to 09

1. Updated the references to JSContact drafts due to the transfer from JMAP to CalExt.

B.15. Change from 09 to 10

1. Updated Figure 2 to make it compliant with draft-ietf-calext-jscontact-02.

B.16. Change from 10 to 11

1. Added Appendix "jCard-JSContact Mapping".

B.17. Change from 11 to 12

1. Renamed the "jcard" property to "jcard_0".
2. Corrected JSONPath expressions in Appendix A.

Authors’ Addresses
Mario Loffredo  
IIT-CNR/Registro.it  
Via Moruzzi,1  
56124 Pisa  
Italy  
Email: mario.loffredo@iit.cnr.it  
URI:   http://www.iit.cnr.it

Gavin Brown  
CentralNic Group plc  
Saddlers House, 44 Gutter Lane  
London  
EC2V 6BR  
United Kingdom  
Phone: +44 20 33 88 0600  
Email: gavin.brown@centralnic.com  
URI:   https://www.centralnic.com
Federated Authentication for the Registration Data Access Protocol (RDAP) using OpenID Connect
draft-ietf-regext-rdap-openid-13

Abstract

The Registration Data Access Protocol (RDAP) provides "RESTful" web services to retrieve registration metadata from domain name and regional internet registries. RDAP allows a server to make access control decisions based on client identity, and as such it includes support for client identification features provided by the Hypertext Transfer Protocol (HTTP). Identification methods that require clients to obtain and manage credentials from every RDAP server operator present management challenges for both clients and servers, whereas a federated authentication system would make it easier to operate and use RDAP without the need to maintain server-specific client credentials. This document describes a federated authentication system for RDAP based on OpenID Connect.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 19 November 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction ......................................................... 3
   1.1. Problem Statement ............................................. 3
   1.2. Proposal ...................................................... 4
2. Conventions Used in This Document ................................. 4
3. Federated Authentication for RDAP .................................. 4
   3.1. RDAP and OpenID Connect ..................................... 5
      3.1.1. Terminology ............................................. 5
      3.1.2. Overview ................................................. 5
      3.1.3. RDAP Authentication and Authorization Steps ............ 6
         3.1.3.1. Provider Discovery ................................ 7
         3.1.3.2. Authentication Request ............................... 7
         3.1.3.3. End-User Authorization ............................... 8
         3.1.3.4. Authorization Response and Validation ............... 8
         3.1.3.5. Token Processing ................................... 8
         3.1.3.6. Delivery of User Information ......................... 8
      3.1.4. Specialized Claims for RDAP ............................. 9
         3.1.4.1. Stated Purposes ..................................... 9
         3.1.4.2. Do Not Track ....................................... 10
4. Protocol Parameters ................................................. 11
   4.1. Data Structures ................................................ 11
      4.1.1. Session ................................................ 11
      4.1.2. Device Info ............................................. 12
      4.1.3. OpenID Connect Configuration ............................ 13
   4.2. Client Login .................................................. 14
      4.2.1. End-User Identifier .................................... 14
      4.2.2. OP Issuer Identifier ................................... 15
      4.2.3. Login Response ......................................... 15
      4.2.4. Clients with Limited User Interfaces ................... 17
         4.2.4.1. UI-constrained Client Login ....................... 17
         4.2.4.2. UI-constrained Client Login Polling ............... 19
   4.3. RDAP Query Parameters ......................................... 19
      4.3.1. RDAP Query Purpose ...................................... 20
      4.3.2. RDAP Do Not Track ...................................... 20
   4.4. Session Status ................................................. 20
   4.5. Session Refresh ............................................... 22
   4.6. Client Logout ................................................ 23
   4.7. Parameter Processing ......................................... 25
1. Introduction

The Registration Data Access Protocol (RDAP) provides "RESTful" web services to retrieve registration metadata from domain name and regional internet registries. RDAP allows a server to make access control decisions based on client identity, and as such it includes support for client identification features provided by the Hypertext Transfer Protocol (HTTP) [RFC7230].

RDAP is specified in multiple documents, including "HTTP Usage in the Registration Data Access Protocol (RDAP)" [RFC7480], "Security Services for the Registration Data Access Protocol (RDAP)" [RFC7481], "Registration Data Access Protocol Query Format" [RFC9082], and "JSON Responses for the Registration Data Access Protocol (RDAP)" [RFC9083]. RFC 7481 describes client identification and authentication services that can be used with RDAP, but it does not specify how any of these services can (or should) be used with RDAP.

1.1. Problem Statement

The traditional "user name and password" authentication method does not scale well in the RDAP ecosystem. Assuming that all domain name and address registries will eventually provide RDAP service, it is impractical and inefficient for users to secure login credentials from the hundreds of different server operators. Authentication methods based on user names and passwords do not provide information that describes the user in sufficient detail (while protecting the personal privacy of the user) for server operators to make fine-
grained access control decisions based on the user’s identity. The authentication system used for RDAP needs to address all of these needs.

1.2. Proposal

A basic level of RDAP service can be provided to users who possess an identifier issued by a recognized provider who is able to authenticate and validate the user. The identifiers issued by social media services, for example, can be used. Users who require higher levels of service (and who are willing to share more information about them self to gain access to that service) can secure identifiers from specialized providers who are or will be able to provide more detailed information about the user. Server operators can then make access control decisions based on the identification information provided by the user.

A federated authentication system in which an RDAP server outsources identification and authentication services to a trusted OpenID Provider would make it easier to operate and use RDAP by re-using existing identifiers to provide a basic level of access. It can also provide the ability to collect additional user identification information, and that information can be shared with the consent of the user. This type of system allows an RDAP server to make access control decisions based on the nature of a query and the identity, authentication, and authorization information that is received from the OpenID Provider. This document describes a federated authentication system for RDAP based on OpenID Connect [OIDC] that meets all of these needs.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Federated Authentication for RDAP

RDAP itself does not include native security services. Instead, RDAP relies on features that are available in other protocol layers to provide needed security services including access control, authentication, authorization, availability, data confidentiality, data integrity, and identification. A description of each of these security services can be found in "Internet Security Glossary, Version 2" [RFC4949]. This document focuses on a federated authentication system for RDAP that provides services for
authentication, authorization, and identification, allowing a server operator to make access control decisions. Section 3 of RFC 7481 [RFC7481] describes general considerations for RDAP access control, authentication, and authorization.

The traditional client-server authentication model requires clients to maintain distinct credentials for every RDAP server. This situation can become unwieldy as the number of RDAP servers increases. Federated authentication mechanisms allow clients to use one credential to access multiple RDAP servers and reduce client credential management complexity.

3.1. RDAP and OpenID Connect

OpenID Connect 1.0 [OIDCC] is a decentralized, single sign-on (SSO) federated authentication system that allows users to access multiple web resources with one identifier instead of having to create multiple server-specific identifiers. Users acquire identifiers from OpenID Providers, or OPs. Relying Parties, or RPs, are applications (such as RDAP) that outsource their user authentication function to an OP. OpenID Connect is built on top of the authorization framework provided by the OAuth 2.0 [RFC6749] protocol.

The OAuth authorization framework describes a method for users to access protected web resources without having to hand out their credentials. Instead, clients are issued Access Tokens by authorization servers with the permission of the resource owners. Using OpenID Connect and OAuth, multiple RDAP servers can form a federation and clients can access any server in the federation by providing one credential registered with any OP in that federation. The OAuth authorization framework is designed for use with HTTP and thus can be used with RDAP.

3.1.1. Terminology

This document uses the terms "client" and "server" defined by RDAP [RFC7480]. An RDAP client performs the role of an OpenID Connect Core [OIDCC] Entity or End-User. An RDAP server performs the role of an OpenID Connect Core Relying Party (RP). Additional terms from Section 1.2 of the OpenID Connect Core specification are incorporated by reference.

3.1.2. Overview

At a high level, RDAP authentication of a browser-like client using OpenID Connect requires completion of the following steps:
1. An RDAP client sends an RDAP "help" query to an RDAP server to determine the type and capabilities of the OpenID Authorization Servers that are used by the RDAP server. This information is returned in the rdapConformance section of the response. A value of "roidc1" indicates support for the extension described in this specification. If one or more remote Authorization Servers are supported, the RDAP client SHOULD evaluate the additional information described in Section 4.1.3 in order to discover the capabilities of the RDAP server and optionally obtain the set of supported OPs.

2. An RDAP client (acting as an OpenID End-User) sends an RDAP "login" request to an RDAP server as described in Section 4.2.

3. The RDAP server (acting as an OpenID Relying Party (RP)) prepares an Authentication Request containing the desired request parameters.

4. The RDAP server sends the RDAP client and Authentication Request to an Authorization Server operated by an OpenID Provider (OP) using an HTTP redirect.

5. The Authorization Server authenticates the End-User.


8. The RDAP server requests a response using the Authorization Code at the Token Endpoint.

9. The RDAP server receives a response that contains an ID Token and Access Token in the response body.

10. The RDAP server validates the ID Token and retrieves the claims associated with the End-User’s identity.

The RDAP server can then make identification, authorization, and access control decisions based on End-User identity information and local policies. Note that OpenID Connect describes different process flows for other types of clients, such as script-based or command line clients.

3.1.3. RDAP Authentication and Authorization Steps

End-Users MAY present an identifier (an OpenID) issued by an OP to use OpenID Connect with RDAP. If the RDAP server supports a default Authorization Server or End-User identifier discovery is not supported, the End-User identifier MAY be omitted. An OP SHOULD include support for the claims described in Section 3.1.4 to provide additional information needed for RDAP End-User authorization. OpenID Connect requires RPs to register with OPs to use OpenID Connect services for an End-User. The registration process is often completed using out-of-band methods, but it is also possible to use the automated method described by the "OpenID Connect Dynamic Client Registration" protocol [OIDCR]. The parties involved can use any
method that is mutually acceptable.

3.1.3.1. Provider Discovery

An RDAP server/RP needs to be able to map an End-User’s identifier to an OP. This can be accomplished using the OPTIONAL "OpenID Connect Discovery" protocol [OIDCD], but that protocol is not widely implemented. Out-of-band methods are also possible and can be more dependable. For example, an RP can support a limited number of OPs and maintain internal associations of those identifiers with the OPs that issued them.

Alternatively, if mapping of an End-User’s identifier is not possible, or not supported by the RDAP server, the RDAP server SHOULD support explicit specification of a remote OP by the RDAP client in the form of a query parameter as described in Section 4.2.2. An RDAP server SHOULD provide information about its capabilities and supported OPs in the "help" query response in the "openidcConfiguration" data structure described in Section 4.1.3.

An RP can also ask an End-User to identify the OP that issued their identifier as part of an RDAP query workflow. In this case, the RP will need to maintain state for the association between the user’s identifier and the OP in order to process later queries that rely on passing the access token and user identifier as authorization parameters. An RDAP server MUST support at least one of these methods of OP discovery.

3.1.3.2. Authentication Request

Once the OP is known, an RP MUST form an Authentication Request and send it to the OP as described in Section 3 of the OpenID Connect Core protocol [OIDCC]. The authentication path followed (authorization, implicit, or hybrid) will depend on the Authentication Request response_type set by the RP. The remainder of the processing steps described here assume that the Authorization Code Flow is being used by setting "response_type=code" in the Authentication Request.

The benefits of using the Authorization Code Flow for authenticating a human user are described in Section 3.1 of the OpenID Connect Core protocol. The Implicit Flow is more commonly used by clients implemented in a web browser using a scripting language; it is described in Section 3.2 of the OpenID Connect Core protocol. The Hybrid Flow (described in Section 3.3 of the OpenID Connect Core protocol) combines elements of the Authorization and Implicit Flows by returning some tokens from the Authorization Endpoint and others from the Token Endpoint.
An Authentication Request can contain several parameters. REQUIRED
parameters are specified in Section 3.1.2.1 of the OpenID Connect
Core protocol [OIDCC]. Apart from these parameters, it is
RECOMMENDED that the RP include the optional "login_hint" parameter
in the request, with the value being that of the "id" query parameter
of the End-User’s RDAP "login" request, if provided. Passing the
"login_hint" parameter allows a client to pre-fill login form
information, so logging in can be more convenient for users. Other
parameters MAY be included.

The OP receives the Authentication Request and attempts to validate
it as described in Section 3.1.2.2 of the OpenID Connect Core
protocol [OIDCC]. If the request is valid, the OP attempts to
authenticate the End-User as described in Section 3.1.2.3 of the
OpenID Connect Core protocol [OIDCC]. The OP returns an error
response if the request is not valid or if any error is encountered.

3.1.3.3. End-User Authorization

After the End-User is authenticated, the OP MUST obtain authorization
information from the End-User before releasing information to the
RDAP Server/RP. This process is described in Section 3.1.2.4 of the
OpenID Connect Core protocol [OIDCC].

3.1.3.4. Authorization Response and Validation

After the End-User is authenticated, the OP will send a response to
the RP that describes the result of the authorization process in the
form of an Authorization Grant. The RP MUST validate the response.
This process is described in Sections 3.1.2.5 - 3.1.2.7 of the OpenID
Connect Core protocol [OIDCC].

3.1.3.5. Token Processing

The RP sends a Token Request using the Authorization Grant to a Token
Endpoint to obtain a Token Response containing an Access Token, ID
Token, and an OPTIONAL Refresh Token. The RP MUST validate the Token
Response. This process is described in Section 3.1.3 of the OpenID
Connect Core protocol [OIDCC].

3.1.3.6. Delivery of User Information

The set of claims can be retrieved by sending a request to a UserInfo
Endpoint using the Access Token. The claims MAY be returned in the
ID Token. The process of retrieving claims from a UserInfo Endpoint
is described in Section 5.3 of the OpenID Connect Core protocol
[OIDCC].
OpenID Connect specifies a set of standard claims in Section 5.1. Additional claims for RDAP are described in Section 3.1.4.

3.1.4. Specialized Claims for RDAP

OpenID Connect claims are pieces of information used to make assertions about an entity. Section 5 of the OpenID Connect Core protocol [OIDCC] describes a set of standard claims that can be used to identify a person. Section 5.1.2 notes that additional claims MAY be used, and it describes a method to create them. The set of claims that are specific to RDAP are associated with an OAuth scope request parameter value (see Section 3.3 of RFC 6749 ([RFC6749])) of "rdap".

3.1.4.1. Stated Purposes

There are communities of RDAP users and operators who wish to make and validate claims about a user’s "need to know" when it comes to requesting access to a resource. For example, a law enforcement agent or a trademark attorney may wish to be able to assert that they have a legal right to access a protected resource, and a server operator will need to be able to receive and validate that claim. These needs can be met by defining and using an additional "rdap_allowed_purposes" claim.

The "rdap_allowed_purposes" claim identifies the purposes for which access to a protected resource can be requested by an End-User. Use of the "rdap_allowed_purposes" claim is OPTIONAL; processing of this claim is subject to server acceptance of the purposes, the trust level assigned to this claim by the server, and successful authentication of the End-User. Unrecognized purpose values MUST be ignored and the associated query MUST be processed as if the unrecognized purpose value was not present at all.

The "rdap_allowed_purposes" claim is represented as an array of case-sensitive StringOrURI values as specified in Section 2 of the JSON Web Token (JWT) specification ([RFC7519]). An example:

"rdap_allowed_purposes": ["domainNameControl","dnsTransparency"]

Individual purpose values are registered with IANA. Each entry in the registry contains the following fields:

Value: the purpose string value being registered. Value strings can contain upper case characters from "A" to "Z", lower case ASCII characters from "a" to "z", and the underscore ("_") character. Value strings contain at least one character and no more than 64 characters.
Description: a one- or two-sentence description of the meaning of the purpose value, how it might be used, and/or how it should be interpreted by clients and servers.

This registry is operated under the "Specification Required" policy defined in RFC 5226 ([RFC5226]). The set of initial values used to populate the registry as described in Section 8.3 are taken from the final report (https://www.icann.org/en/system/files/files/final-report-06jun14-en.pdf) produced by the Expert Working Group on gTLD Directory Services chartered by the Internet Corporation for Assigned Names and Numbers (ICANN).

3.1.4.2. Do Not Track

There are also communities of RDAP users and operators who wish to make and validate claims about a user’s wish to not have their queries logged, tracked, or recorded. For example, a law enforcement agent may wish to be able to assert that their queries are part of a criminal investigation and should not be tracked due to a risk of query exposure compromising the investigation, and a server operator will need to be able to receive and validate that claim. These needs can be met by defining and using an additional "do not track" claim.

The "do not track" ("rdap_dnt_allowed") claim can be used to identify an End-User that is authorized to perform queries without the End-User’s association with those queries being logged, tracked, or recorded by the server. Client use of the "rdap_dnt_allowed" claim is OPTIONAL. Server operators MUST NOT log, track, or record any association of the query and the End-User’s identity if the End-User is successfully identified and authorized, the "rdap_dnt_allowed" claim is present, the value of the claim is "true", and accepting the claim complies with local regulations regarding logging and tracking.

The "rdap_dnt_allowed" value is represented as a JSON boolean literal. An example:

```
rdap_dnt_allowed: true
```

No special query tracking processing is required if this claim is not present or if the value of the claim is "false". Use of this claim MUST be limited to End-Users who are granted "do not track" privileges in accordance with service policies and regulations. Specification of these policies and regulations is beyond the scope of this document.
4. Protocol Parameters

This specification adds the following protocol parameters to RDAP:

1. Data structures to return information that describes an established session, the information needed to establish a session for a UI-constrained device, and the RDAP server’s OpenID Connect extension configuration.
2. A query parameter to request authentication for a specific End-User identity.
3. A query parameter to identify the purpose of the query.
4. A query parameter to request that the server not log or otherwise record information about the identity associated with a query.
5. Path segments to start, stop, refresh, and determine the status of an authenticated session for a specific End-User identity.

4.1. Data Structures

This specification describes three new data structures that are used to return information to a client: a "roidc1_session" data structure that contains information that describes an established session, a "roidc1_deviceInfo" data structure that contains information that describes an active attempt to establish a session on a UI-constrained device, and a "roidc1_openidcConfiguration" data structure that describes the OpenID Connect configuration and related extension features supported by the RDAP server.

4.1.1. Session

The "roidc1_session" data structure is an object that contains two sub-objects:

1. A "userClaims" object that contains the set of claims associated with the End-User's identity as used/requested by the RDAP server to make access control decisions. The set of possible values is determined by OP policy.
2. A "sessionInfo" object that contains two members:
   a. "tokenExpiration": an integer value that represents the number of seconds from the current time for which the Access Token remains valid, and
   b. "tokenRefresh": A boolean value that indicates if the OP supports refresh tokens. As described in RFC 6749 [RFC6749], support for refresh tokens is OPTIONAL.

An example of a "roidc1_session" data structure:
"roidc1_session": {
  "userClaims": {
    "sub": "103892603076825016132",
    "name": "User Person",
    "given_name": "User",
    "family_name": "Person",
    "picture": "https://lh3.example.com/a-/AOh14=s96-c",
    "email": "user@example.com",
    "email_verified": true,
    "locale": "en",
    "rdap_allowed_purposes": [
      "domainNameControl",
      "personalDataProtection"
    ],
    "rdap_dnt_allowed": false
  },
  "sessionInfo": {
    "tokenExpiration": 3599,
    "tokenRefresh": true
  }
}

Figure 1

4.1.2. Device Info

The flow described in Section 3.1.3 requires an End-User to interact with a server using a user interface that can process HTTP. This will not work well in situations where the client is automated or an End-User is using a command line user interface such as curl (http://curl.haxx.se/) or wget (https://www.gnu.org/software/wget/). This limitation can be addressed using a web browser on a second device. The information that needs to be entered using the web browser is contained in the "roidc1_deviceInfo" data structure.

The "roidc1_deviceInfo" data structure is an object that contains three members:

1. "verification_url": the URL that the End-User needs to visit using the web browser,
2. "user_code": the string value that the End-User needs to enter on the form presented in the web browser, and
3. "expires_in": an integer value that represents the number of seconds after which the opportunity to visit the URL and enter the user_code will expire.

An example of a "roidc1_deviceInfo" data structure:
"roidc1_deviceInfo": {
  "verification_url": "https://www.example.com/device",
  "user_code": "NJJQ-GJFC",
  "expires_in": "1800"
}

Figure 2

4.1.3. OpenID Connect Configuration

The "roidc1_openidcConfiguration" data structure is an object with the following members:

1. "dntSupported": (MANDATORY) a boolean value that describes RDAP server support for the "roidc1_dnt" query parameter (see Section 4.3.2).
2. "endUserIdentifierDiscoverySupported": (OPTIONAL) a boolean value that describes RDAP server support for discovery of End-User identifiers. The default value is "true".
3. "issuerIdentifierSupported": (OPTIONAL) a boolean value that describes RDAP server support for explicit client specification of an Issuer Identifier. The default value is "true".
4. "openidcProviders": (OPTIONAL) a list of objects with the following members that describes the set of OPs that are supported by the RDAP server. This data is RECOMMENDED if the value of issuerIdentifierSupported is "true":
   a. "iss": (MANDATORY) a string value equal to Issuer Identifier of the OP as per OpenID Connect Core specification [OIDCC]
   b. "name": (MANDATORY) a string value representing the human-friendly name of the OP.
   c. "default": (OPTIONAL) a boolean value that describes RDAP server support for an OPTIONAL default OP that will be used when a client omits the "roidc1_id" and "roidc1_iss" query parameters from a "roidc1_session/login" request. Only one member of this set can be identified as the default OP by setting a value of "true". The default value is "false".

An example of a "roidc1_openidcConfiguration" data structure:
"roidc1_openidcConfiguration": {
  "dntSupported": false,
  "endUserIdentifierDiscoverySupported": true,
  "issuerIdentifierSupported": true,
  "openidcProviders": [
    {
      "iss": "https://idp.example.com",
      "name": "Example IDP"
    },
    {
      "iss": "https://accounts.example.net",
      "name": "Login with EXAMPLE"
    },
    {
      "iss": "https://auth.nic.example/auth/realms/rdap",
      "name": "Default OP for the Example RDAP server",
      "default": "true"
    }
  ]
}

Figure 3

4.2. Client Login

Client authentication is requested by sending a "roidc1_session/login" request to an RDAP server. If the RDAP server supports only remote Authorization Servers, the "roidc1_session/login" request MUST include at least one of an End-User Identifier or an OP Issuer Identifier.

4.2.1. End-User Identifier

The End-User identifier is delivered using one of two methods: by adding a query component to an RDAP request URI using the syntax described in Section 3.4 of RFC 3986 [RFC3986], or by including an HTTP authorization header for the Basic authentication scheme as described in RFC 7617 [RFC7617]. Clients can use either of these methods to deliver the End-User identifier to a server that supports remote Authorization Servers and End-User identifier discovery. Servers that support remote Authorization Servers and End-User identifier discovery MUST accept both methods. If the RDAP server supports a default Authorization Server or End-User identifier discovery is not supported, the End-User identifier MAY be omitted.
The query used to request client authentication is represented as an
OPTIONAL "key=value" pair using a key value of "roidc1_id" and a
value component that contains the client identifier issued by an OP.
An example for client identifier "user.idp.example":

https://example.com/rdap/roidc1_session/
login?roidc1_id=user.idp.example

The authorization header for the Basic authentication scheme contains
a Base64-encoded representation of the client identifier issued by an
OP. No password is provided. An example for client identifier
"user.idp.example":

https://example.com/rdap/roidc1_session/login
Authorization: Basic dXNlci5pZHAuZXhhbXBsZQ==

An example for use with a default Authorization Server:

https://example.com/rdap/roidc1_session/login

4.2.2. OP Issuer Identifier

The OP’s Issuer Identifier is delivered by adding a query component
to an RDAP request URI using the syntax described in Section 3.4 of
RFC 3986 [RFC3986]. If the RDAP server supports a default
Authorization Server, the Issuer Identifier MAY be omitted.

The query used to request client authentication is represented as an
OPTIONAL "key=value" pair using a key value of "roidc1_iss" and a
value component that contains the Issuer Identifier associated with
an OP. An RDAP server MAY accept Issuer Identifiers not specified in
the "roidc1_openidcConfiguration" data structure and MAY also decide
to accept specific Issuer Identifiers only from specific clients.

An example for Issuer Identifier "https://idp.example.com":

https://example.com/rdap/roidc1_session/
login?roidc1_iss=https%3A%2F%2Fidp.example.com

4.2.3. Login Response

The response to the login request MUST use the response structures
specified in RFC 9083 [RFC9083]. In addition, the response MUST
include an indication of the requested operation’s success or failure
in the "notices" data structure (including the client identifier),
and, if successful, a "roidc1_session" data structure.
An example of a successful "roidc1_session/login" response:

```json
{
    "rdapConformance": [
        "roidc1"
    ],
    "lang": "en-US",
    "notices": {
        "title": "Login Result",
        "description": [
            "Login succeeded",
            "user.idp.example"
        ]
    },
    "roidc1_session": {
        "userClaims": {
            "sub": "103892603076825016132",
            "name": "User Person",
            "given_name": "User",
            "family_name": "Person",
            "picture": "https://lh3.example.com/a/-/AOh14=s96-c",
            "email": "user@example.com",
            "email_verified": true,
            "locale": "en",
            "rdap_allowed_purposes": [
                "domainNameControl",
                "personalDataProtection"
            ],
            "rdap_dnt_allowed": false
        },
        "sessionInfo": {
            "tokenExpiration": 3599,
            "tokenRefresh": true
        }
    }
}
```

Figure 4

An example of a failed "roidc1_session/login" response:
4.2.4. Clients with Limited User Interfaces

The "OAuth 2.0 Device Authorization Grant" [RFC8628] provides an OPTIONAL method to request user authorization from devices that have an Internet connection, but lack a suitable browser for a more traditional OAuth flow. This method requires an End-User to use a second device (such as a smart telephone) that has access to a web browser for entry of a code sequence that is presented on the UI-constrained device.

4.2.4.1. UI-constrained Client Login

Client authentication is requested by sending a "roidc1_session/device" request to an RDAP server. If the RDAP server supports only remote Authorization Servers, the "roidc1_session/device" request MUST include an End-User identifier that’s delivered using one of two methods: by adding a query component to an RDAP request URI using the syntax described in Section 3.4 of RFC 3986 [RFC3986], or by including an HTTP authorization header for the Basic authentication scheme as described in RFC 7617 [RFC7617]. If the RDAP server supports a default Authorization Server, the End-User identifier MAY be omitted. Clients can use either of these methods. Servers MUST support both methods.

The query used to request client authentication is represented as an OPTIONAL "key=value" pair using a key value of "id" and a value component that contains the client identifier issued by an OP.

An example using wget for client identifier "user.idp.example":

    wget -qO- --keep-session-cookies --save-cookies\  
    https://example.com/rdap/session/device?roidc1_id=user.idp.example
Figure 6

The authorization header for the Basic authentication scheme contains a Base64-encoded representation of the client identifier issued by an OP. No password is provided.

An example using curl and an authorization header:

```
curl -H "Authorization: Bearer dXNlci5pZHAuZXhhbXBsZQ=="\
     -c cookies.txt https://example.com/rdap/session/device
```

Figure 7

The response to this request MUST use the response structures specified in RFC 9083 [RFC9083]. In addition, the response MUST include an indication of the requested operation’s success or failure in the "notices" data structure (including the client identifier), and, if successful, a "roidc1_deviceInfo" data structure.

An example of a "roidc1_session/device" response:

```
{
   "rdapConformance": [
      "roidc1"
   ],
   "lang": "en-US",
   "notices": {
      "title": "Device Login Result",
      "description": [
         "Login succeeded",
         "user.idp.example"
      ]
   },
   "roidc1_deviceInfo": {
      "verification_url": "https://www.example.com/device",
      "user_code": "NJJQ-GJFC",
      "expires_in": 1800
   }
}
```

Figure 8
4.2.4.2. UI-constrained Client Login Polling

After successful processing of the "roidc1_session/device" request, the client MUST send a "roidc1_session/devicepoll" request to the RDAP server to continue the login process. This request performs the polling function described in RFC 8628 [RFC8628], allowing the RDAP server to wait for the End-User to enter the information returned from the "roidc1_session/device" request using the interface on their second device. After the End-User has completed that process, or if the process fails or times out, the OP will respond to the polling requests with an indication of success or failure.

An example using wget:

```
wget -qO- --load-cookies cookies.txt\https://example.com/rdap/roidc1_session/devicepoll
```

Figure 9

An example using curl:

```
curl -b cookies.txt https://example.com/rdap/roidc1_session/devicepoll
```

Figure 10

The response to this request MUST use the response structures described in Section 4.2. RDAP query processing can continue normally on the UI-constrained device once the "login" process has been completed.

4.3. RDAP Query Parameters

This specification describes two OPTIONAL query parameters for use with RDAP queries that request access to information associated with protected resources:

1. "roidcl_qp": A query parameter to identify the purpose of the query.
2. "roidcl_dnt": A query parameter to request that the server not log or otherwise record information about the identity associated with a query.

One or both of these parameters MAY be added to an RDAP request URI using the syntax described in Section 3.4 of RFC 3986 [RFC3986].
4.3.1. RDAP Query Purpose

This query is represented as a "key=value" pair using a key value of "roidc1_qp" and a value component that contains a single query purpose string from the set of allowed purposes associated with the End-User’s identity (see Section 3.1.4.1). If present, the server SHOULD compare the value of the parameter to the "rdap_allowed_purposes" claim values associated with the End-User’s identity and ensure that the requested purpose is present in the set of allowed purposes. The RDAP server MAY choose to ignore both requested purpose and the "rdap_allowed_purposes" claim values if they are inconsistent with local server policy. The server MUST return an HTTP 403 (Forbidden) response if the requested purpose is not an allowed purpose. If this parameter is not present, the server MUST process the query and make an access control decision based on any other information known to the server about the End-User and the information they are requesting. For example, a server MAY treat the request as one performed by an unidentified or unauthenticated user and return either an error or an appropriate subset of the available data. An example domain query using the "roidc1_qp" query parameter:

https://example.com/rdap/domain/example.com?roidc1_qp=legalActions

4.3.2. RDAP Do Not Track

This query is represented as a "key=value" pair using a key value of "roidc1_dnt" and a value component that contains a single boolean value. A value of "true" indicates that the End-User is requesting that their query not be tracked or logged in accordance with server policy. A value of "false" indicates that the End-User is accepting that their query can be tracked or logged in accordance with server policy. The server MUST return an HTTP 501 (Not Implemented) response if the server is unable to perform the action requested by this query parameter. An example domain query using the "roidc1_dnt" query parameter:

https://example.com/rdap/domain/example.com?roidc1_dnt=true

4.4. Session Status

Clients MAY send a query to an RDAP server to determine the status of an existing login session using a "roidc1_session/status" path segment. An example "roidc1_session/status" request:

https://example.com/rdap/roidc1_session/status
The response to this query MUST use the response structures specified in RFC 9083 [RFC9083]. In addition, the response MUST include an indication of the requested operation’s success or failure in the "notices" data structure (including the client identifier), and, if successful, a "roidc1_session" data structure.

An example of a "roidc1_session/status" response:

```json
{
    "rdapConformance": [
        "roidc1"
    ],
    "lang": "en-US",
    "notices": {
        "title": "Session Status Result",
        "description": [
            "Session status succeeded",
            "user.idp.example"
        ]
    },
    "roidc1_session": {
        "userClaims": {
            "sub": "103892603076825016132",
            "name": "User Person",
            "given_name": "User",
            "family_name": "Person",
            "picture": "https://lh3.example.com/a-/AOh14=s96-c",
            "email": "user@example.com",
            "email_verified": true,
            "locale": "en",
            "rdap_allowed_purposes": [
                "domainNameControl",
                "personalDataProtection"
            ],
            "rdap_dnt_allowed": false
        },
        "sessionInfo": {
            "tokenExpiration": 3490,
            "tokenRefresh": true
        }
    }
}
```

Figure 11
4.5. Session Refresh

Clients MAY send a request to an RDAP server to refresh, or extend, an existing login session using a "roidc1_session/refresh" path segment. The RDAP server MAY attempt to refresh the access token associated with the current session as part of extending the session for a period of time determined by the RDAP server. As described in RFC 6749 [RFC6749], OP support for refresh tokens is OPTIONAL. An RDAP server MUST determine if the OP supports token refresh and process the refresh request by either requesting refresh of the access token or by returning a response that indicates that token refresh is not supported by the OP in the "notices" data structure. An example "roidc1_session/refresh" request:

https://example.com/rdap/roidc1_session/refresh

The response to this request MUST use the response structures specified in RFC 9083 [RFC9083]. In addition, the response MUST include an indication of the requested operation’s success or failure in the "notices" data structure (including the client identifier), and, if successful, a "roidc1_session" data structure.

An example of a "roidc1_session/refresh" response:
Clients MAY send a request to an RDAP server to terminate an existing login session. Termination of a session is requested using a "roidc1_session/logout" path segment. Access and refresh tokens can be revoked during the "roidc1_session/logout" process as described in RFC 7009 [RFC7009] if supported by the OP (token revocation endpoint support is OPTIONAL per RFC 8414 [RFC8414]). If supported, this feature SHOULD be used to ensure that the tokens are not mistakenly associated with a future RDAP session. Alternatively, an RDAP server
MAY attempt to logout from the OP using the "OpenID Connect RP-Initiated Logout" protocol ([OIDCL]) if that protocol is supported by the OP.

An example "roidc1_session/logout" request:

https://example.com/rdap/roidc1_session/logout

The response to this request MUST use the response structures specified in RFC 9083 [RFC9083]. In addition, the response MUST include an indication of the requested operation’s success or failure in the "notices" data structure (including the client identifier). The "notices" data structure MUST also include an indication of the success or failure of any attempt to logout from the OP or to revoke the tokens issued by the OP.

An example of a "roidc1_session/logout" response:

```json
{
   "rdapConformance": [
      "roidc1"
   ],
   "lang": "en-US",
   "notices": {
      "title": "Logout Result",
      "description": [
         "Logout succeeded",
         "user.idp.example",
         "Provider logout failed: Not supported by provider."
      ],
      "Token revocation successful."
   }
}
```

Figure 13

In the absence of a "logout" request, an RDAP session MUST be terminated by the RDAP server after a server-defined period of time. The server should also take appropriate steps to ensure that the tokens associated with the terminated session cannot be reused. This SHOULD include revoking the tokens or logging out from the OP if either operation is supported by the OP.
4.7.  Parameter Processing

Unrecognized query parameters MUST be ignored. An RDAP server that processes an authenticated query MUST determine if the End-User identification information is associated with an OP that is recognized and supported by the server. Servers MUST reject queries that include identification information that is not associated with a supported OP by returning an HTTP 501 (Not Implemented) response. An RDAP server that receives a query containing identification information associated with a recognized OP MUST perform the steps required to authenticate the user with the OP, process the query, and return an RDAP response that is appropriate for the End-User's level of authorization and access.

5.  Token Exchange

ID tokens include an audience parameter that contains the OAuth 2.0 client_id of the RP as an audience value. In some operational scenarios (such as a client that is providing a proxy service), an RP can receive tokens with an audience value that does not include the RP's client_id. These tokens might not be trusted by the RP, and the RP might refuse to accept the tokens. This situation can be remedied by having the RP exchange these tokens with the OP for a set of trusted tokens that reset the audience parameter. This token exchange protocol is described in RFC 8693 [RFC8693]. This issue is not visible to the RDAP client and should be managed by the OpenID implementation used by the RDAP server.

6.  RDAP Query Processing

Once an RDAP session is active, an RDAP server MUST determine if the End-User is authorized to perform any queries that are received during the duration of the session. This MAY include rejecting queries outright, and it MAY include omitting or otherwise redacting information that the End-User is not authorized to receive. Specific processing requirements are beyond the scope of this document. A client can end a session explicitly by sending a "roidc1_session/logout" request to the RDAP server. A session can also be ended implicitly by the server after a server-defined period of time. The status of a session can be determined at any time by sending a "roidc1_session/status" query to the RDAP server.

An RDAP server MUST maintain session state information for the duration of an active session. This is commonly done using HTTP cookies as described in RFC 6265 [RFC6265]. Doing so allows End-User to submit queries without having to explicitly identify and authenticate themselves for each and every query.
7. RDAP Conformance

RDAP responses that contain values described in this document MUST indicate conformance with this specification by including an rdapConformance ([RFC9083]) value of "roidc1" (RDAP OpenID Connect version 1). The information needed to register this value in the RDAP Extensions Registry is described in Section 8.1.

Example rdapConformance structure with extension specified:

```
"rdapConformance" :
[  
  "rdap_level_0",
  "roidc1"
]
```

Figure 14

8. IANA Considerations

8.1. RDAP Extensions Registry

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

- Extension identifier: roidc1
- Registry operator: Any
- Published specification: This document.
- Contact: IESG <iesg@ietf.org>
- Intended usage: This extension describes version 1 of a federated authentication method for RDAP using OAuth 2.0 and OpenID Connect.

8.2. JSON Web Token Claims Registry

IANA is requested to register the following values in the JSON Web Token Claims Registry:

- Claim Name: "rdap_allowed_purposes"
- Claim Description: This claim describes the set of RDAP query purposes that are available to an identity that is presented for access to a protected RDAP resource.
- Change Controller: IESG
- Specification Document(s): Section 3.1.4.1 of this document.

- Claim Name: "rdap_dnt_allowed"
Claim Description: This claim contains a JSON boolean literal that
describes a "do not track" request for server-side tracking,
logging, or recording of an identity that is presented for access
to a protected RDAP resource.
Change Controller: IESG
Specification Document(s): Section 3.1.4.2 of this document.

8.3. RDAP Query Purpose Registry

IANA is requested to create a new protocol registry to manage RDAP
query purpose values. This registry should be named "Registration
Data Access Protocol (RDAP) Query Purpose Values" and should appear
under the "Registration Data Access Protocol (RDAP)" section of
IANA’s protocol registries. The information to be registered and the
procedures to be followed in populating the registry are described in
Section 3.1.4.1.

Section at http://www.iana.org/protocols: Registration Data Access
Protocol (RDAP)

Name of registry: Registration Data Access Protocol (RDAP) Query
Purpose Values

Registration Procedure: Specification Required

Reference: This document

Required information: See Section 3.1.4.1.

Review process: "Specification Required" as described in RFC 5226
[RFC5226].

Size, format, and syntax of registry entries: See Section 3.1.4.1.

Initial assignments and reservations:

-----BEGIN FORM-----

Value: domainNameControl

Description: Tasks within the scope of this purpose include
creating and managing and monitoring a registrant’s own domain
name, including creating the domain name, updating information
about the domain name, transferring the domain name, renewing the
domain name, deleting the domain name, maintaining a domain name
portfolio, and detecting fraudulent use of the Registrant’s own
contact information.
Value: personalDataProtection

Description: Tasks within the scope of this purpose include identifying the accredited privacy/proxy provider associated with a domain name and reporting abuse, requesting reveal, or otherwise contacting the provider.

Value: technicalIssueResolution

Description: Tasks within the scope of this purpose include (but are not limited to) working to resolve technical issues, including email delivery issues, DNS resolution failures, and website functional issues.

Value: domainNameCertification

Description: Tasks within the scope of this purpose include a Certification Authority (CA) issuing an X.509 certificate to a subject identified by a domain name.

Value: individualInternetUse

Description: Tasks within the scope of this purpose include identifying the organization using a domain name to instill consumer trust, or contacting that organization to raise a customer complaint to them or file a complaint about them.

Value: businessDomainNamePurchaseOrSale
Description: Tasks within the scope of this purpose include making purchase queries about a domain name, acquiring a domain name from a registrant, and enabling due diligence research.

-----END FORM-----

-----BEGIN FORM-----

Value: academicPublicInterestDNSRResearch

Description: Tasks within the scope of this purpose include academic public interest research studies about domain names published in the registration data service, including public information about the registrant and designated contacts, the domain name’s history and status, and domain names registered by a given registrant (reverse query).

-----END FORM-----

-----BEGIN FORM-----

Value: legalActions

Description: Tasks within the scope of this purpose include investigating possible fraudulent use of a registrant’s name or address by other domain names, investigating possible trademark infringement, contacting a registrant/licensee’s legal representative prior to taking legal action and then taking a legal action if the concern is not satisfactorily addressed.

-----END FORM-----

-----BEGIN FORM-----

Value: regulatoryAndContractEnforcement

Description: Tasks within the scope of this purpose include tax authority investigation of businesses with online presence, Uniform Dispute Resolution Policy (UDRP) investigation, contractual compliance investigation, and registration data escrow audits.

-----END FORM-----

-----BEGIN FORM-----

Value: criminalInvestigationAndDNSAbuseMitigation
Description: Tasks within the scope of this purpose include reporting abuse to someone who can investigate and address that abuse, or contacting entities associated with a domain name during an offline criminal investigation.

------END FORM------

------BEGIN FORM------

Value: dnsTransparency

Description: Tasks within the scope of this purpose involve querying the registration data made public by registrants to satisfy a wide variety of use cases around informing the general public.

------END FORM------

9. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942 [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Version -09 of this specification introduced changes that are incompatible with earlier implementations. Implementations that are consistent with this specification will be added as they are identified.
9.1. Editor Implementation

Location: https://procuratus.net/rdap/
Description: This implementation is a functionally-limited RDAP server that supports only the path segments described in this specification. It uses the "jumbojett/OpenID-Connect-PHP" library found on GitHub, which appears to no longer be under active development. The library was modified to add support for the device authorization grant. Session variable management is still a little buggy. Supported OPs include Google (Gmail) and Yahoo.
Level of Maturity: This is a "proof of concept" research implementation.
Coverage: This implementation includes all of the features described in this specification.
Version compatibility: Version -11+ of this specification.
Contact Information: Scott Hollenbeck, shollenbeck@verisign.com

9.2. Verisign Labs

Responsible Organization: Verisign Labs
Location: https://rdap.verisignlabs.com/
Description: This implementation includes support for domain registry RDAP queries using live data from the .cc and .tv country code top-level domains and the .career generic top-level domain. Three access levels are provided based on the authenticated identity of the client:
1. Unauthenticated: Limited information is returned in response to queries from unauthenticated clients.
2. Basic: Clients who authenticate using a publicly available identity provider like Google Gmail or Microsoft Hotmail will receive all of the information available to an unauthenticated client plus additional registration metadata, but no personally identifiable information associated with entities.
3. Advanced: Clients who authenticate using a more restrictive identity provider will receive all of the information available to a Basic client plus whatever information the server operator deems appropriate for a fully authorized client. Currently supported identity providers include those developed by Verisign Labs (https://testprovider.rdap.verisignlabs.com/) and CZ.NIC (https://www.mojeid.cz/).
Level of Maturity: This is a "proof of concept" research implementation.
Coverage: This implementation includes all of the features described in this specification.
Version compatibility: Version -07 of this specification.
Contact Information: Scott Hollenbeck, shollenbeck@verisign.com
9.3. Viagenie

Responsible Organization: Viagenie
Location: https://auth.viagenie.ca
Description: This implementation is an OpenID identity provider enabling users and registries to connect to the federation. It also includes a barebone RDAP client and RDAP server in order to test the authentication framework. Various level of purposes are available for testing.
Level of Maturity: This is a "proof of concept" research implementation.
Coverage: This implementation includes most features described in this specification as an identity provider.
Version compatibility: Version -07 of this specification.
Contact Information: Marc Blanchet, marc.blanchet@viagenie.ca

10. Security Considerations

Security considerations for RDAP can be found in RFC 7481 [RFC7481]. Security considerations for OpenID Connect Core [OIDCC] and OAuth 2.0 [RFC6749] can be found in their reference specifications. OpenID Connect defines optional mechanisms for robust signing and encryption that can be used to provide data integrity and data confidentiality services as needed.

10.1. Authentication and Access Control

Having completed the client identification, authorization, and validation process, an RDAP server can make access control decisions based on a comparison of client-provided information and local policy. For example, a client who provides an email address (and nothing more) might be entitled to receive a subset of the information that would be available to a client who provides an email address, a full name, and a stated purpose. Development of these access control policies is beyond the scope of this document.

11. Acknowledgments

The author would like to acknowledge the following individuals for their contributions to the development of this document: Marc Blanchet, Tom Harrison, Russ Housley, Jasdip Singh, Rhys Smith, Jaromir Talir, Rick Wilhelm, and Alessandro Vesely. In addition, the Verisign Registry Services Lab development team of Joseph Harvey, Andrew Kaizer, Sai Mogali, Anurag Saxena, Swapneel Sheth, Nitin Singh, and Zhao Zhao provided critical "proof of concept" implementation experience that helped demonstrate the validity of the concepts described in this document.
Pawel Kowalik and Mario Loffredo provided significant text contributions that led to welcome improvements in several sections of this document. Their contributions are greatly appreciated.

12. References

12.1. Normative References


12.2. Informative References


Appendix A. Change Log

00: Initial working group version ported from draft-hollenbeck-regext-rdap-openid-10.
01: Modified ID Token delivery approach to note proper use of an HTTP bearer authorization header.
02: Modified token delivery approach (Access Token is the bearer token) to note proper use of an HTTP bearer authorization header, fixing the change made in -01.
03: Updated OAuth 2.0 Device Authorization Grant description and reference due to publication of RFC 8628.
04: Updated OAuth 2.0 token exchange description and reference due to publication of RFC 8693. Corrected the RDAP conformance identifier to be registered with IANA.
05: Keepalive refresh.
06: Keepalive refresh.
07: Added "login_hint" description to Section 3.1.3.2. Added some text to Section 3.1.4.2 to note that "do not track" requires compliance with local regulations.
08: Rework of token management processing in Sections 4 and 5.
09: Updated RDAP specification references. Added text to describe
both default and remote Authorization Server processing. Removed
text that described passing of ID Tokens as query parameters.
10: Updated Section 3.1.3.1. Replaced token processing queries with
"login", "session", and "logout" queries.
11: Replaced queries with "session/*" queries. Added description of
"rdap" OAuth scope. Added implementation status information.
12: Updated data structure descriptions. Updated Section 8. Minor
formatting changes due to a move to xml2rfc-v3 markup.
13: Added support for OP discovery via OP’s Issuer Identifier.
Modified the RDAP conformance text to use "roidc1", and added that
value to extension path segments, data structures, and query
parameters. Changed the "purpose" and "dnt" claims to
"rdap_allowed_purposes" (making it an array) and
"rdap_dnt_allowed". Added the "roidc1_qp" and "roidc1_dnt" query
parameters. Changed the descriptions of "local" OPs to "default"
OPs.

Author’s Address

Scott Hollenbeck
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
United States of America
Email: shollenbeck@verisign.com
URI: http://www.verisignlabs.com/
Registration Protocols Extensions
M. Loffredo
Internet-Draft
M. Martinelli
Intended status: Standards Track
IIT-CNR/Registro.it
Expires: 3 November 2022
2 May 2022

Registration Data Access Protocol (RDAP) Reverse search capabilities
draft-ietf-regext-rdap-reverse-search-11

Abstract

The Registration Data Access Protocol (RDAP) does not include query
capabilities for finding the list of domains related to a set of
entities matching a given search pattern. In the RDAP context, an
entity can be associated with any defined object class. Moreover,
other relationships between object classes exist and might be used
for providing a reverse search capability. Therefore, a reverse
search can be applied to other use cases than the classic domain-
entity scenario. This document describes an RDAP extension that
allow servers to provide a reverse search feature based on the
relationship defined in RDAP between an object class for search and
any related object class. The reverse search based on the domain-
entity relationship is treated as a particular case.

Status of This Memo

This Internet-Draft is submitted in full conformance with the
provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering
Task Force (IETF). Note that other groups may also distribute
working documents as Internet-Drafts. The list of current Internet-
Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months
and may be updated, replaced, or obsoleted by other documents at any
time. It is inappropriate to use Internet-Drafts as reference
material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 November 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the
document authors. All rights reserved.
1. Introduction

Reverse Whois is a service provided by many web applications that allows users to find domain names owned by an individual or a company starting from the owner's details, such as name and email. Even if it has been considered useful for some legal purposes (e.g. uncovering trademark infringements, detecting cybercrimes), its availability as a standardized Whois capability has been objected to for two main reasons, which now don’t seem to conflict with an RDAP implementation.

The first objection concerns the potential risks of privacy violation. However, the domain name community is considering a new generation of Registration Directory Services [ICANN-RDS1] [ICANN-RDS2] [ICANN-RA], which provide access to sensitive data under...
some permissible purposes and in accordance with appropriate policies for requestor accreditation, authentication and authorization. RDAP’s reliance on HTTP means that it can make use of common HTTP-based approaches to authentication and authorization, making it more useful than Whois [RFC3912] in the context of such directory services. Since RDAP consequently permits a reverse search implementation complying with privacy protection principles, this objection is not well-founded.

The other objection to the implementation of a reverse search capability has been connected with its impact on server processing. However, the core RDAP specifications already define search queries, with similar processing requirements, so the distinction on which this objection is based is not clear.

Reverse searches, such as finding the list of domain names associated with contacts or nameservers, may be useful to registrars as well. Usually, registries adopt out-of-band solutions to provide results to registrars asking for reverse searches on their domains. Possible reasons for such requests are:

* the loss of synchronization between the registrar database and the registry database;
* the need for such data to perform bulk EPP [RFC5730] updates (e.g. changing the contacts of a set of domains, etc.).

Currently, RDAP does not provide any means for a client to search for the collection of domains associated with an entity [RFC9082]. A query (lookup or search) on domains can return the array of entities related to a domain with different roles (registrant, registrar, administrative, technical, reseller, etc.), but the reverse operation is not allowed. Only reverse searches to find the collection of domains related to a nameserver (ldhName or ip) can be requested. Since an entity can be in relationship with any RDAP object [RFC9083], the availability of a reverse search as largely intended can be common to all the object classes allowed for search. Through a further step of generalization, the meaning of reverse search in the RDAP context can be extended to include any query for retrieving all the objects in relationship with another matching a given search pattern.

The protocol described in this specification aims to extend the RDAP query capabilities to enable reverse search based on the relationships defined in RDAP between an object class for search and a related object class. The reverse search based on the domain-entity relationship is treated as a particular case of such a generic query model.
1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. RDAP Path Segment Specification

A generic reverse search path is described by the syntax:

{searchable-resource-type}/reverse_search_0/{related-resource-type}?<search-condition>

The path segments are defined as in the following:

* searchable-resource-type: it MUST be one of the resource types for search defined in Section 3.2 of [RFC9082] (i.e. "domains", "nameservers" and "entities") or a resource type extension;
* related-resource-type: it MUST be one of the resource types for lookup defined in Section 3.1 of [RFC9082] (i.e. "domain", "nameserver", "entity", "ip" and "autnum") or a resource type extension;
* search-condition: a sequence of "property=search pattern" predicates separated by the ampersand character (‘&’, US-ASCII value 0x0026). Each "property" represents a JSON object property of the RDAP object class corresponding to "related-resource-type". Objects are only included in the search results if they satisfy all included predicates. This includes predicates that are for the same property: it is necessary in such a case for the related object to match against each of those predicates. Based on their policy, servers MAY restrict the usage of predicates to make a valid search condition, by returning a 400 (Bad Request) response when a problematic request is received.

While related-resource-type is defined as having one of a number of different values, the only searches defined in this document are for a related-resource-type of "entity". Searches for the other resource types specified in [RFC9082] and resource type extensions may be defined by future documents.

Partial string matching in search patterns is allowed as defined in section 4.1 of [RFC9082].
3. RDAP Response Specification

Reverse search responses use the formats defined in section 8 of [RFC9083], which correspond to the searchable resource types defined in Section 2.

4. Reverse Searches Based on Entity Details

Since in RDAP, an entity can be associated with any other object class, the most common kind of reverse search is one based on an entity’s details. Such reverse searches arise from the query model by setting the related resource type to "entity".

By selecting a specific searchable resource type, the resulting reverse search aims at retrieving all the objects (e.g. all the domains) that are related to any entity object matching the search conditions.

This section defines the following reverse search properties servers SHOULD support regardless of the searchable resource type being selected:

- **Reverse search property: role**
  - RDAP property: $..entities[*].roles
  - Reference: Section 10.2.4 of [RFC9083]

- **Reverse search property: handle**
  - RDAP property: $..entities[*].handle
  - Reference: Section 5.1 of [RFC9083]

- **Reverse search property: fn**
  - RDAP property: $..entities[*].vcardArray[1][?(@[0]=='fn')][3]
  - Reference: Section 6.2.1 of [RFC6350]

- **Reverse search property: email**
  - RDAP property: $..entities[*].vcardArray[1][?(@[0]=='email')][3]
  - Reference: Section 6.4.2 of [RFC6350]

The mapping between the reverse search property and the corresponding RDAP response property is done through the use of a JSONPath expression [I-D.ietf-jsonpath-base].

The presence of a predicate on the reverse search property "role" means that the RDAP response property "roles" must contain at least the specified role.
The last two properties are related to jCard elements [RFC7095], but the field references are to vCard [RFC6350], since jCard is the JSON format for vCard.

Examples of reverse search paths based on the domain-entity relationship are presented in Figure 1.

```
/domains/reverse_search_0/entity?handle=CID-40*&role=technical
```

```
/domains/reverse_search_0/entity?fn=Bobby*&role=registrant
```

```
/domains/reverse_search_0/entity?handle=RegistrarX&role=registrar
```

Figure 1

Documents that deprecate or restructure RDAP responses such that one or more of the properties listed above becomes invalid MUST either note that the relevant reverse search is no longer available (in the case of deprecation) or describe how to continue supporting the relevant search by way of some new RDAP property (in the case of restructuring).

A server that includes additional fields in its objects in accordance with the extensibility provisions of section 6 of [RFC7480] MAY support the use of those fields in search conditions, in the same way as for the search conditions defined in this section. Support for such fields in the reverse search context MUST be documented in the extension specification.

5. RDAP Conformance

Servers complying with this specification MUST include the value "reverse_search_0" in the rdapConformance property of the help response [RFC9083]. The information needed to register this value in the "RDAP Extensions" registry is described in Section 8.

6. Implementation Considerations

To limit the impact of processing the search predicates, servers are RECOMMENDED to make use of indexes and similar functionality in their underlying data store. In addition, risks with respect to performance degradation or result set generation can be mitigated by adopting practices used for standard searches, e.g. restricting the search functionality, limiting the rate of search requests according to the user’s authorization, truncating and paging the results, and returning partial responses.
7. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

7.1. IIT-CNR/Registro.it RDAP Server

* Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it
* Location: https://rdap.pubtest.nic.it/
* Description: This implementation includes support for RDAP queries using data from the public test environment of .it ccTLD. Reverse search is allowed to authenticated users. Registrar users are allowed to perform reverse searches on their own domains and contacts. This is achieved by adding an implicit predicate to the search condition.
* Level of Maturity: This is an "alpha" test implementation.
* Coverage: This implementation includes all of the features described in this specification.
* Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

7.2. IIT-CNR/Registro.it RDAP Client

* Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it
* Location: https://web-rdap.pubtest.nic.it/
* Description: This is a Javascript web-based RDAP client. RDAP responses are retrieved from RDAP servers by the browser, parsed into an HTML representation, and displayed in a format improving the user experience. Reverse search is allowed to authenticated users.
* Level of Maturity: This is an "alpha" test implementation.
* Coverage: This implementation includes all of the features described in this specification.
* Contact Information: Francesco Donini, francesco.donini@iit.cnr.it

8. IANA Considerations

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

* Extension identifier: reverse_search_0
* Registry operator: Any
* Published specification: This document.
* Contact: IETF <iesg@ietf.org>
* Intended usage: This extension describes reverse search query patterns for RDAP.

9. Privacy Considerations

The search functionality defined in this document may affect the privacy of entities in the registry (and elsewhere) in various ways: see [RFC6973] for a general treatment of privacy in protocol specifications. Registry operators should be aware of the tradeoffs that result from implementation of this functionality.

Many jurisdictions have laws or regulations that restrict the use of "Personal Data", per the definition in [RFC6973]. Given that, registry operators should ascertain whether the regulatory environment in which they operate permits implementation of the functionality defined in this document.

In general, given the sensitivity of this functionality, it SHOULD be accessible to authorized users only, and for specific use cases only.

Since reverse search requests and responses could contain Personally Identifiable Information (PII), reverse search functionality SHOULD be available over HTTPS only.

Providing reverse search in RDAP carries the following threats as described in [RFC6973]:

* Correlation
* Disclosure
* Misuse of information

Therefore, RDAP providers are REQUIRED to mitigate the risk of those threats by implementing appropriate measures supported by security services (see Section 10).

10. Security Considerations

Security services required to provide controlled access to the operations specified in this document are described in [RFC7481]. A non-exhaustive list of access control paradigms an RDAP provider can implement is presented in Appendix A.

The specification of the relationship within the reverse search path allows the RDAP servers to implement different authorization policies on a per-relationship basis.

11. Acknowledgements

The authors would like to acknowledge the following individuals for their contributions to this document: Francesco Donini, Scott Hollenbeck, Francisco Arias, Gustavo Lozano, Eduardo Alvarez, Ulrich Wisser and James Gould.

Tom Harrison and Jasdip Singh provided relevant feedback and constant support to the implementation of this proposal. Their contributions have been greatly appreciated.

12. References

12.1. Normative References


12.2. Informative References

[I-D.ietf-jsonpath-base]
Gössner, S., Normington, G., and C. Bormann, "JSONPath: Query expressions for JSON", Work in Progress, Internet-
Appendix A. Paradigms to Enforce Access Control on Reverse Search in RDAP

Access control can be implemented according to different paradigms introducing increasingly stringent rules. The paradigms reported here in the following leverage the capabilities either supported natively or provided as extensions by the OpenID Connect [OIDCC]:

* Role-Based Access Control: access rights are granted depending on roles. Generally, this is done by grouping users into fixed categories and assigning static grants to each category. A more dynamic approach can be implemented by using the OpenID Connect "scope" claim;
* Purpose-Based Access Control: access rules are based on the notion of purpose, being the intended use of some data by a user. It can be implemented by tagging a request with the usage purpose and
making the RDAP server check the compliance between the given purpose and the control rules applied to the data to be returned. The purpose can be stated within an out-of-band process by setting the OpenID Connect RDAP-specific "purpose" claim as defined in [I-D.ietf-regext-rdap-openid];

* Attribute-Based Access Control: rules to manage access rights are evaluated and applied according to specific attributes describing the context within which data are requested. It can be implemented by setting within an out-of-band process additional OpenID Connect claims describing the request context and making the RDAP server check the compliance between the given context and the control rules applied to the data to be returned;

* Time-Based Access Control: data access is allowed for a limited time only. It can be implemented by assigning the users with temporary credentials linked to access grants whose scope is limited.

Appendix B. Change Log

00: Initial working group version ported from draft-loffredo-regext-rdap-reverse-search-04
01: Updated "Privacy Considerations" section.
02: Revised the text.
03: Refactored the query model.
04: Keepalive refresh.
05: Reorganized "Abstract". Corrected "Conventions Used in This Document" section. Added "RDAP Conformance" section. Changed "IANA Considerations" section. Added references to RFC7095 and RFC8174. Other minor edits.
06: Updated "Privacy Considerations", "Security Considerations" and "Acknowledgements" sections. Added some normative and informative references. Added Appendix A.
07: Updated normative references.
08: Changed "Implementation Status" section. Updated informative references.
09: Extended the query model to represent a reverse search based on any relationship between the RDAP object classes. Changed the path segment "role" into a query parameter.
10: Updated "Reverse Searches Based on Entity Details" section to consider the use of JSContact format instead of jCard. Added references to JSContact documents.
11: Updated the document based on Tom Harrison and James Gould feedback:
   * Updated section "RDAP Path Segment Specification":
     - Clarified how servers must evaluate a reverse search including predicates that are for the same property.
- Specified the error response servers must return when receiving a wrong reverse search request according to their policy.
- Clarified that searches for the related-resource-type values other than "entity" may be defined in future documents.
* Reviewed text in section "Reverse Searches Based on Entity Details" about reverse searches based on custom response extensions.
* Removed references to JSContact documents in section "Reverse Searches Based on Entity Details". Moved the mapping between jCard properties used in the RDAP response and JSContact counterparts to draft-ietf-regext-rdap-jscontact.
* Added section "RDAP Response Specification".
* Changed the text to present reverse search as a single extension with multiple features.
* Changed the definition of searchable-resource-type and related-resource-type to consider also the resource type extensions.
* Replaced "reverse" with "reverse_search_0" in the generic reverse search path. Updated Figure 1 accordingly.
* Removed the phrase "but with a special focus on its privacy implications" from both the "Abstract" and the "Introduction". Moved the mapping between jCard properties used in the RDAP response and JSContact counterparts to draft-ietf-regext-rdap-jscontact.
* Reviewed the text of "Privacy Considerations" section.
* Text cleaning.

Authors’ Addresses

Mario Loffredo
IIT-CNR/Registro.it
Via Moruzzi,1
56124 Pisa
Italy
Email: mario.loffredo@iit.cnr.it
URI: http://www.iit.cnr.it

Maurizio Martinelli
IIT-CNR/Registro.it
Via Moruzzi,1
56124 Pisa
Italy
Email: maurizio.martinelli@iit.cnr.it
URI: http://www.iit.cnr.it
Finding the Authoritative Registration Data (RDAP) Service
draft-ietf-regext-rfc7484bis-06

Abstract

This document specifies a method to find which Registration Data Access Protocol (RDAP) server is authoritative to answer queries for a requested scope, such as domain names, IP addresses, or Autonomous System numbers. This document obsoletes RFC7484.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 1 August 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.
1. Introduction

Querying and retrieving registration data from registries are defined in Registration Data Access Protocol (RDAP) [RFC7480] [RFC7481] [RFC9082] [RFC9083]. These documents do not specify where to send the queries. This document specifies a method to find which server is authoritative to answer queries for the requested scope.
Top-Level Domains (TLDs), Autonomous System (AS) numbers, and network blocks are delegated by IANA to Internet registries such as TLD registries and Regional Internet Registries (RIRs) that then issue further delegations and maintain information about them. Thus, the bootstrap information needed by RDAP clients is best generated from data and processes already maintained by IANA; the relevant registries already exist at [ipv4reg], [ipv6reg], [asreg], and [domainreg]. This document obsoletes [RFC7484].

Per this document, IANA has created new registries based on a JSON format specified in this document, herein named RDAP Bootstrap Service Registries. These new registries are based on the existing entries of the above-mentioned registries. An RDAP client fetches the RDAP Bootstrap Service Registries, extracts the data, and then performs a match with the query data to find the authoritative registration data server and appropriate query base URL.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Structure of the RDAP Bootstrap Service Registries

The RDAP Bootstrap Service Registries, as specified in Section 13 below, have been made available as JSON [RFC8259] 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 ordered in any way.

An example structure of the JSON output of a RDAP Bootstrap Service Registry is illustrated:
The formal syntax is described in Section 10.

The "version" corresponds to the format version of the registry. This specification defines version "1.0".

The syntax of the "publication" value conforms to the Internet date/time format [RFC3339]. The value is the latest update date of the registry by IANA.

The optional "description" string can contain a comment regarding the content of the bootstrap object.

Per [RFC7258], in each array of base RDAP URLs, the secure versions of the transport protocol SHOULD be preferred and tried first. For example, if the base RDAP URLs array contains both HTTPS and HTTP URLs, the bootstrap client SHOULD try the HTTPS version first.

Base RDAP URLs MUST have a trailing "/" character because they are concatenated to the various segments defined in [RFC9082].

JSON names MUST follow the format recommendations of section 6 of [RFC7480]. Any unrecognized JSON object properties or values MUST be ignored by implementations.

Internationalized Domain Name labels used as entries or base RDAP URLs in the registries defined in this document MUST be only represented using their A-label form as defined in [RFC5890].
All Domain Name labels used as entries or base RDAP URLs in the registries defined in this document MUST be only represented in lowercase.

4. Bootstrap Service Registry for Domain Name Space

The JSON output of this registry contains domain label entries attached to the root, grouped by base RDAP URLs, as shown in this example.

```
{
    "version": "1.0",
    "publication": "2024-01-07T10:11:12Z",
    "description": "Some text",
    "services": [
        [
            ["net", "com"],
            ["https://registry.example.com/myrdap/"
        ],
        [
            ["org", "mytld"],
            ["https://example.org/"
        ],
        [
            ["xn--zckzah"],
            ["https://example.net/rdap/xn--zckzah/",
             "http://example.net/rdap/xn--zckzah/
        ]
    ]
}
```

The domain name’s authoritative registration data service is found by doing the label-wise longest match of the target domain name with the domain values in the Entry Arrays in the IANA Bootstrap Service Registry for Domain Name Space. The match is done per label, from right to left. If the longest match results in multiple entries, then those entries are considered equivalent. The values contained in the Service URL Array of the matching second-level array are the valid base RDAP URLs as described in [RFC9082].
For example, a domain RDAP query for a.b.example.com matches the com entry in one of the arrays of the registry. The base RDAP URL for this query is then taken from the second element of the array, which is an array of base RDAP URLs valid for this entry. The client chooses one of the base URLs from this array; in this example, it chooses the only one available, "https://registry.example.com/myrdap/". The segment specified in [RFC9082] is then appended to the base URL to complete the query. The complete query is then "https://registry.example.com/myrdap/domain/a.b.example.com".

If a domain RDAP query for a.b.example.com matches both com and example.com entries in the registry, then the longest match applies and the example.com entry is used by the client.

If the registry contains entries such as com and goodexample.com, then a domain RDAP query for example.com only matches the com entry because matching is done on a per-label basis.

The entry for the root of the domain name space is specified as "".

5. Bootstrap Service Registries for Internet Numbers

This section discusses IPv4 and IPv6 address space and Autonomous System numbers.

For IP address space, the authoritative registration data service is found by doing a longest match of the target address with the values of the arrays in the corresponding RDAP Bootstrap Service Registry for Address Space. The longest match is done the same way as in packet forwarding: the addresses are converted in binary form and then the binary strings are compared to find the longest match up to the specified prefix length. The values contained in the second element of the array are the base RDAP URLs as described in [RFC9082]. The longest match method enables covering prefixes of a larger address space pointing to one base RDAP URL while more specific prefixes within the covering prefix are being served by another base RDAP URL.

5.1. Bootstrap Service Registry for IPv4 Address Space

The JSON output of this registry contains IPv4 prefix entries, specified in Classless Inter-domain Routing (CIDR) format [RFC4632] and grouped by RDAP URLs, as shown in this example.


```
{
    "version": "1.0",
    "publication": "2024-01-07T10:11:12Z",
    "description": "RDAP Bootstrap file for example registries.",
    "services": [
        [
            ["198.51.100.0/24", "192.0.0.0/8"],
            ["https://rir1.example.com/myrdap/"]
        ],
        [
            ["203.0.113.0/24", "192.0.2.0/24"],
            ["https://example.org/"]
        ],
        [
            ["203.0.113.0/28"],
            ["https://example.net/rdaprir2/",
             "http://example.net/rdaprir2/"]
        ]
    ]
}
```

For example, a query for "192.0.2.1/25" matches the "192.0.0.0/8" entry and the "192.0.2.0/24" entry in the example registry above. The latter is chosen by the client because it is the longest match. The base RDAP URL for this query is then taken from the second element of the array, which is an array of base RDAP URLs valid for this entry. The client chooses one of the base URLs from this array; in this example, it chooses the only one available, "https://example.org/". The {resource} specified in [RFC9082] is then appended to the base URL to complete the query. The complete query is then "https://example.org/ip/192.0.2.1/25".

5.2. Bootstrap Service Registry for IPv6 Address Space

The JSON output of this registry contains IPv6 prefix entries, using [RFC5952] text representation of the address prefixes format, grouped by base RDAP URLs, as shown in this example.
For example, a query for "2001:db8:1000::/48" matches the "2001:db8::/34" entry and the "2001:db8:1000::/36" entry in the example registry above. The latter is chosen by the client because it is the longest match. The base RDAP URL for this query is then taken from the second element of the array, which is an array of base RDAP URLs valid for this entry. The client chooses one of the base URLs from this array; in this example, it chooses "https://example.net/rdapr2/" because it’s the secure version of the protocol. The segment specified in [RFC9082] is then appended to the base URL to complete the query. The complete query is, therefore, "https://example.net/rdapr2/ip/2001:db8:1000::/48". If the target RDAP server does not answer, the client can then use another URL prefix from the array.
5.3. Bootstrap Service Registry for AS Number Space

The JSON output of this registry contains Autonomous Systems number ranges entries, grouped by base RDAP URLs, as shown in this example. The Entry Array is an array containing the list of AS number ranges served by the base RDAP URLs found in the second element. Each element of the array contains two AS numbers represented in decimal format, separated by a hyphen, that represents the range of AS numbers between the two AS numbers (inclusive), where values are in increasing order (e.g. 100-200, not 200-100). A single AS number is represented as a range of two identical AS numbers. AS numbers are represented as ‘asplain’ as defined in [RFC5396]. Ranges MUST NOT overlap.

```
{
  "version": "1.0",
  "publication": "2024-01-07T10:11:12Z",
  "description": "RDAP Bootstrap file for example registries.",
  "services": [
    [
      "64496-64496",
      "https://rir3.example.com/myrdap/
    ],
    [
      "64497-64510", "65536-65551"],
      "https://example.org/
    ],
    [
      "64512-65534",
      "http://example.net/rdaprir2/",
      "https://example.net/rdaprir2/
    ]
  ]
}
```

For example, a query for AS 65411 matches the 64512-65534 entry in the example registry above. The base RDAP URL for this query is then taken from the second element of the array, which is an array of base RDAP URLs valid for this entry. The client chooses one of the base URLs from this array; in this example, it chooses "https://example.net/rdaprir2/". The segment specified in [RFC9082] is then appended to the base URL to complete the query. The complete
query is, therefore, "https://example.net/rdaprir2/autnum/65411". If the server does not answer, the client can then use another URL prefix from the array.

6. Entity

Entities (such as contacts, registrants, or registrars) can be queried by handle as described in [RFC9082]. Since there is no global namespace for entities, this document does not describe how to find the authoritative RDAP server for entities. However, it is possible that, if the entity identifier was received from a previous query, the same RDAP server could be queried for that entity, or the entity identifier itself is a fully qualified URL that can be queried. The mechanism described in [RFC8521] MAY also be used.

7. Non-existent Entries or RDAP URL Values

The registries may not contain the requested value. In these cases, there is no known RDAP server for that requested value, and the client SHOULD provide an appropriate error message to the user.

8. Deployment and Implementation Considerations

This method relies on the fact that RDAP clients are fetching the IANA registries to then find the servers locally. Clients SHOULD NOT fetch the registry on every RDAP request. Clients SHOULD cache the registry, but use underlying protocol signaling, such as the HTTP Expires header field [RFC7234], to identify when it is time to refresh the cached registry.

Some authorities of registration data may work together on sharing their information for a common service, including mutual redirection [REDIRECT-RDAP].

When a new object is allocated, such as a new AS range, a new TLD, or a new IP address range, there is no guarantee that this new object will have an entry in the corresponding bootstrap RDAP registry, since the setup of the RDAP server for this new entry may become live and registered later. Therefore, the clients should expect that even if an object, such as TLD, IP address range, or AS range is allocated, the existence of the entry in the corresponding bootstrap registry is not guaranteed.
9. Limitations

This method does not provide a direct way to find authoritative RDAP servers for any other objects than the ones described in this document. In particular, the following objects are not bootstrapped with the method described in this document:

* entities

* queries using search patterns that do not contain a terminating string that matches some entries in the registries

* nameservers

* help

10. Formal Definition

This section is the formal definition of the registries. The structure of JSON objects and arrays using a set of primitive elements is defined in [RFC8259]. Those elements are used to describe the JSON structure of the registries.

10.1. Imported JSON Terms

* OBJECT: a JSON object, defined in Section 4 of [RFC8259]

* MEMBER: a member of a JSON object, defined in Section 4 of [RFC8259]

* MEMBER-NAME: the name of a MEMBER, defined as a "string" in Section 4 of [RFC8259]

* MEMBER-VALUE: the value of a MEMBER, defined as a "value" in Section 4 of [RFC8259]

* ARRAY: an array, defined in Section 5 of [RFC8259]

* ARRAY-VALUE: an element of an ARRAY, defined in Section 5 of [RFC8259]

* STRING: a "string", as defined in Section 7 of [RFC8259]

10.2. Registry Syntax

Using the above terms for the JSON structures, the syntax of a registry is defined as follows:
* **rdap-bootstrap-registry**: an OBJECT containing a MEMBER version and a MEMBER publication, an optional MEMBER description, and a MEMBER services-list

* **version**: a MEMBER with MEMBER-NAME "version" and MEMBER-VALUE a STRING

* **publication**: a MEMBER with MEMBER-NAME "publication" and MEMBER-VALUE a STRING

* **description**: a MEMBER with MEMBER-NAME "description" and MEMBER-VALUE a STRING

* **services-list**: a MEMBER with MEMBER-NAME "services" and MEMBER-VALUE a services-array

* **services-array**: an ARRAY, where each ARRAY-VALUE is a service

* **service**: an ARRAY of 2 elements, where the first ARRAY-VALUE is an entry-list and the second ARRAY-VALUE is a service-uri-list

* **entry-list**: an ARRAY, where each ARRAY-VALUE is an entry

* **entry**: a STRING

* **service-uri-list**: an ARRAY, where each ARRAY-VALUE is a service-uri

* **service-uri**: a STRING

11. Security Considerations

By providing a bootstrap method to find RDAP servers, this document helps to ensure that the end users will get the RDAP data from an authoritative source, instead of from rogue sources. The method has the same security properties as the RDAP protocols themselves. The transport used to access the registries uses TLS [RFC8446].

Additional considerations on using RDAP are described in [RFC7481].

12. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942].
The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

12.1. RDAP Browser Mobile Application

Responsible Organization: Viagenie

Author: Marc Blanchet

Location: https://viagenie.ca/rdapbrowser/

Description: RDAP Browser is an RDAP client for domain names, IP addresses and AS numbers fetching the IANA registries described in this document to find the right authoritative RDAP server. End user can query any domain name, IP address or AS number and the registration data will be shown on the screen.

Level of Maturity: Production (i.e. in the Android and iOS App stores since August 2019)

Contact Information: rdapbrowser@viagenie.ca

Information last updated: March 2021

12.2. ICANN Lookup Web Application

Responsible Organization: ICANN

Location: https://lookup.icann.org
Description: ICANN’s Domain Name Registration Data Lookup is an RDAP client for domain names fetching the IANA registries described in this document to find the right authoritative RDAP server. End user can query any domain name and the registration data will be shown on the screen.

Level of Maturity: Production

Information last updated: March 2021

12.3. ARIN Implementation

Responsible Organization: ARIN

Base URL: https://rdap-bootstrap.arin.net/bootstrap (Sample query: https://rdap-bootstrap.arin.net/bootstrap/autnum/1)

Description: ARIN RDAP Bootstrap server aids clients by reading the bootstrapping information published by IANA and using it to send HTTP redirects to RDAP queries. RDAP clients https://search.arin.net/ and NicInfo (https://github.com/arineng/nicinfo) use this bootstrap service. The underlying server software is open-sourced at https://github.com/arineng/rdap_bootstrap_server.

Level of Maturity: Production

Contact Information: info@arin.net

Information Last Updated: Nov 2020

13. IANA Considerations

IANA has created the RDAP Bootstrap Services Registries, listed below, and made them available as JSON objects. The contents of these registries are described in Section 3, Section 4, and Section 5, with the formal syntax specified in Section 10. The registries MUST be accessible only through HTTPS (TLS [RFC8446]) transport.

The process for adding or updating entries in these registries differs from the normal IANA registry processes: these registries are generated from the data, processes, and policies maintained by IANA in their allocation registries ([ipv4reg], [ipv6reg], [asreg], and [domainreg]), with the addition of new RDAP server information.

IANA updates RDAP Bootstrap Services Registries entries from the allocation registries as those registries are updated.
This document does not change any policies related to the allocation registries; IANA has provided a mechanism for collecting the RDAP server information.

IANA has created a new top-level category on the Protocol Registries page, <https://www.iana.org/protocols>. The group is called "Registration Data Access Protocol (RDAP)". Each of the RDAP Bootstrap Services Registries has been made available for general public on-demand download in the JSON format, and that registry’s URI is listed directly on the Protocol Registries page.

Other normal registries will be added to this group by other documents, but the reason the URIs for these registries are clearly listed on the main page is to make those URIs obvious to implementers -- these are registries that will be accessed by software, as well as by humans using them for reference information.

Because these registries will be accessed by software, the download demand for the RDAP Bootstrap Services Registries may be unusually high compared to normal IANA registries. The technical infrastructure by which registries are published has been put in place by IANA to support the load. Since the publication of [RFC7484], no issue have been reported regarding the load or the service.

As discussed in Section 8, software that accesses these registries will depend on the HTTP Expires header field to limit their query rate. It is, therefore, important for that header field to be properly set to provide timely information as the registries change, while maintaining a reasonable load on the IANA servers.

The HTTP Content-Type returned to clients accessing these JSON-formatted registries MUST be "application/json", as defined in [RFC8259].

Because of how information in the RDAP Bootstrap Services Registries is grouped and formatted, the registry entries may not be sortable. It is, therefore, not required or expected that the entries be ordered in any way.

NOTE TO IANA: Please update the registries to reference this new RFC instead of RFC 7484 once this document is approved by the IESG and published by the RFC Editor. RFC-Editor, please remove this paragraph before publication.
13.1. Bootstrap Service Registry for IPv4 Address Space

Entries in this registry contain at least the following:

* a CIDR [RFC4632] specification of the network block being registered.
* one or more URLs that provide the RDAP service regarding this registration.

13.2. Bootstrap Service Registry for IPv6 Address Space

Entries in this registry contain at least the following:

* an IPv6 prefix [RFC5952] specification of the network block being registered.
* one or more URLs that provide the RDAP service regarding this registration.

13.3. Bootstrap Service Registry for AS Number Space

Entries in this registry contain at least the following:

* a range of Autonomous System numbers being registered.
* one or more URLs that provide the RDAP service regarding this registration.

13.4. Bootstrap Service Registry for Domain Name Space

Entries in this registry contain at least the following:

* a domain name attached to the root being registered.
* one or more URLs that provide the RDAP service regarding this registration.

14. References

14.1. Normative References


14.2. Informative References

IANA, "Root Zone Database",
<https://www.iana.org/domains/root/db>.

IANA, "IPv4 Address Space Registry",
<https://www.iana.org/assignments/ipv4-address-space>.

IANA, "IPv6 Global Unicast Address Assignments",
<https://www.iana.org/assignments/ipv6-unicast-address-assignments>.


Borenstein, N. and M. Kucherawy, "A Media Type for Reputation Interchange", RFC 7071, DOI 10.17487/RFC7071,

RFC 7234, DOI 10.17487/RFC7234, June 2014,

Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)!", STD 95,
RFC 7481, DOI 10.17487/RFC7481, March 2015,

Blanchet, M., "Finding the Authoritative Registration Data (RDAP) Service", RFC 7404, DOI 10.17487/RFC7404, March 2015,

Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205,
RFC 7942, DOI 10.17487/RFC7942, July 2016,


Hollenbeck, S. and A. Newton, "Registration Data Access Protocol (RDAP) Object Tagging", BCP 221, RFC 8521,
DOI 10.17487/RFC8521, November 2018,
Acknowledgements

The WEIRDS working group had multiple discussions on this topic, including a session during IETF 84, where various methods such as in-DNS and others were debated. The idea of using IANA registries was discovered by the author during discussions with his colleagues as well as by a comment from Andy Newton. All the people involved in these discussions are herein acknowledged. Linlin Zhou, Jean-Philippe Dionne, John Levine, Kim Davies, Ernie Dainow, Scott Hollenbeck, Arturo Servin, Andy Newton, Murray Kucherawy, Tom Harrison, Naoki Kambe, Alexander Mayrhofer, Edward Lewis, Pete Resnick, Alessandro Vesely, Bert Greevenbosch, Barry Leiba, Jari Arkko, Kathleen Moriarty, Stephen Farrell, Richard Barnes, and Jean-Francois Tremblay have provided input and suggestions to this document. Guillaume Leclanche was a coauthor of this document for some revisions; his support is therein acknowledged and greatly appreciated. The section on formal definition was inspired by Section 6.2 of [RFC7071]. This new version got comments and suggestions from: Gavin Brown, Patrick Mevzek, John Levine, Jas dip Singh, George Michaelson, Scott Hollenbeck, Russ Housley, Joel Halpern, Lars Eggert, Benjamin Kaduk, Scott Kelly, Eric Vyncke, John Scudder, Erik Kline, Robert Wilton. Errata of RFC7484 were submitted by Pieter Vandepitte and were applied to this version.

Changes since RFC7484

There are no substantive changes except for updates to the implementation status and minor clarifications. This update is primarily to meet the requirements for moving to Internet Standard.

Author’s Address

Marc Blanchet
Viagenie
246 Aberdeen
Quebec QC G1R 2E1
Canada
Email: Marc.Blanchet@viagenie.ca
URI:   https://viagenie.ca
Abstract

Domain name registries (the producer) and registrars (the consumer) report to each other by sharing bulk information through files. This document creates two IANA registries to establish a standard reporting mechanism between domain name registries and registrars. The first IANA registry lists standard data elements and their syntax for inclusion in the files. The second IANA registry lists standard reports based on the standard data elements. Each report is a file formatted as a CSV file. The advantage of this reporting mechanism is that a report, each file, can be imported by recipients without any prior knowledge of their contents, although reporting is enhanced with a minimum of knowledge about the files. The mechanism for the distribution of and access of the files is a matter of local policy.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
Table of Contents

1.  Introduction ..................................................... 3

1.1.  Requirements Language ........................................ 4

2.  Data Element Specification ....................................... 4

2.1.  General Information Data Elements .............................. 5

2.1.1.  TLD .......................................................... 5

2.1.2.  Server_TRID ................................................ 5

2.1.3.  Domain ...................................................... 5

2.1.4.  Transaction_Type ............................................ 5

2.1.5.  Object_Type ................................................ 5

2.1.6.  DateTime .................................................... 5

2.1.7.  Term ........................................................ 5

2.1.8.  Fee .......................................................... 6

2.1.9.  Currency ..................................................... 6

2.1.10.  Status ...................................................... 6

2.1.11.  Registrar ................................................... 6

2.1.12.  Period ...................................................... 6

2.1.13.  Description ............................................... 6

2.2.  Domain Price Data Elements ..................................... 6

2.2.1.  Domain_Create ............................................... 6

2.2.2.  Domain_Renew ................................................ 6

2.2.3.  Domain_Transfer ............................................. 7

2.2.4.  Domain_Restore .............................................. 7

2.2.5.  Trade ........................................................ 7

2.3.  Timestamp Data Elements ........................................ 7

2.3.1.  Start_Date .................................................. 7

2.3.2.  Deleted_Date ................................................ 7

2.3.3.  RGP_Date .................................................... 7

2.3.4.  Purge_Date .................................................. 7

2.3.5.  Updated_Date ................................................ 7

2.3.6.  Create_Date ................................................ 8

2.3.7.  Expiry_Date .................................................. 8

2.4.  Registration Information Data Elements ........................ 8

2.4.1.  Registrar_ID ............................................... 8

2.4.2.  Server_Registrant_ID ....................................... 8

2.4.3.  DNSSEC ...................................................... 8

2.4.4.  Server_Contact_ID ......................................... 8

2.4.5.  Contact_Type ............................................... 8
1. Introduction

Currently, domain name registry operators (the producer) create and set their own domain name registration reports for use by their registrars (the consumer). Among the distinctions that vary by producer is the syntax of the data provided, e.g., date formats, and the format of the collection of the data provided, e.g., the report may be a CSV file that tends to allow for straightforward importation or a PDF file that can be problematic to import. In addition, although there are a number of best practice reports that have evolved, these are not currently documented as such, which results in a fair amount of customization on the part of the consumers to import data.
This document standardizes the name and syntax of the data elements to be used across all existing domain name registration reports and creates an IANA registry of them to facilitate their evolution, including adding additional data elements as needed. In addition, a known set of existing standard reports using the aforementioned data elements is specified in another IANA registry to facilitate the evolution of the reports and adding additional report definitions as needed.

Each report definition MUST use only the data elements defined in the data element aforementioned data element registry, including all future reports. Note that a produced report MAY include data elements that are not registered, as specified below. Future reports and future data elements may be specified in their own individual documents, updating the IANA registries as needed.

The mechanism for the distribution of and access to the files is a matter of local policy.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Data Element Specification

Data elements are grouped into categories for convenience. There is no other significance to the groupings.

Each data element conceptually represents the column heading in a printed report. It is a single unit of information that can be passed from the producer to the consumer. The primary purposes of the IANA registry of data elements are to ensure that each data element is assigned a unique name and that the syntax of each data element is specified.

The name of the data element MUST be unique and this characteristic MUST be enforced by the registry. The name is used in the report definition (in the next section) to alert the consumer as to what to expect in the file and how to import the data element. Character encoding recommendation for data elements is specified in Section 7.
The subsections below comprise an initial list of known data elements commonly being used between producers and consumers as of the date of publication of this document. The title of the subsection is the data element name for the data element. Data element names in the IANA registry MUST be unique and MUST be processed as case insensitive.

2.1. General Information Data Elements

2.1.1. TLD

The string of the top level domain involved that MUST be in A-label format as defined by RFC 5890 [RFC5890].

2.1.2. Server_TRID

The transaction identifier issued by an EPP Server. The format MUST conform to "type:trIDStringType" as specified in RFC 5730 [RFC5730].

2.1.3. Domain

This is the domain name in an EPP RFC 5731 [RFC5731] domain object and it MUST be in A-Label format.

2.1.4. Transaction_Type

The type of transform action made to the domain object (e.g., create, delete, update, transfer, renew) as specified in RFC 5730 [RFC5730] Section 2.9.3.

2.1.5. Object_Type

The object type involved in the report. In the EPP environment, an object could be domain RFC 5731 [RFC5731], contact RFC 5733 [RFC5733], or host RFC 5732 [RFC5732].

2.1.6. DateTime

The timestamp of the transaction recorded in the system. Dates and Times MUST be expressed as defined in RFC 5731 [RFC5731] Section 2.4.

2.1.7. Term

The number of units added to the domain registration period in <domain:period> RFC 5731 [RFC5731] in create, renew or transfer transforms. If there is no <domain:period>, the default value set out-of-band by the registry should be used.
2.1.8. Fee

The amount of money charged or returned (shown as a negative value) to the registrar. The numeric format MUST conform to the currency specified below in Section 2.1.9. The format must conform to "balanceType" as defined in RFC 8748 [RFC8748].

2.1.9. Currency

The currency used in the money charged as documented above in Section 2.1.8. The currency code should follow the ISO 4217 [ISO4217] standard.

2.1.10. Status

The status of the domain object. It MUST be one of the values specified in RFC 5731 [RFC5731] Section 2.3.

2.1.11. Registrar

The name of the registrar. This data element is text/string with no naming convention enforced.

2.1.12. Period

The type of time (year, month) in 'Term' described above in Section 2.1.7. The value of 'year' and 'month' are referenced to pUnitType value 'y' and 'm' respectively. pUnitType is specified in RFC 5731 [RFC5731].

2.1.13. Description

Additional information regarding the current entry in the report. It is provided by the producer and its actual value is a matter of local policy. This data element is text/string with no naming convention enforced.

2.2. Domain Price Data Elements

2.2.1. Domain_Create

The fee charged to create the domain. The format must conform to "balanceType" as defined in RFC 8748 [RFC8748].

2.2.2. Domain_Renew

The fee charged to renew the domain. The format must conform to "balanceType" as defined in RFC 8748 [RFC8748].
2.2.3. Domain_Transfer

The fee charged to transfer the domain. The format must conform to "balanceType" as defined in RFC 8748 [RFC8748].

2.2.4. Domain_Restore

The fee charged to restore the domain. The format must conform to "balanceType" as defined in RFC 8748 [RFC8748].

2.2.5. Trade

The fee charged to trade the domain. The format must conform to "balanceType" as defined in RFC 8748 [RFC8748].

2.3. Timestamp Data Elements

2.3.1. Start_Date

The timestamp of when the domain object becomes available. The date and time format follows the "type=dateTime" specification as defined in RFC 5731 [RFC5731].

2.3.2. Deleted_Date

The timestamp of when the domain was deleted. The date and time format follows the "type=dateTime" specification as defined in RFC 5731 [RFC5731].

2.3.3. RGP_Date

The timestamp of when the domain will complete its redemption grace period. The date and time format follows the "type=dateTime" specification as defined in RFC 5731 [RFC5731].

2.3.4. Purge_Date

The timestamp of when the domain will be purged and become available again. The date and time format follows the "type=dateTime" specification as defined in RFC 5731 [RFC5731].

2.3.5. Updated_Date

The timestamp of the last time the domain object was updated. The date and time format follows the "type=dateTime" specification as defined in RFC 5731 [RFC5731].
2.3.6.  Create_Date

The timestamp of when the domain object was allocated. The date and
time format follows the "type=dateTime" specification as defined in
RFC 5731 [RFC5731].

2.3.7.  Expiry_Date

The timestamp of when the domain object will expire. The date and
time format follows the "type=dateTime" specification as defined in
RFC 5731 [RFC5731].

2.4.  Registration Information Data Elements

2.4.1.  Registrar_ID

The identifier assigned to the registrar. If the registrar is
accredited under ICANN, it MUST be the registrar’s IANA ID
[IANA_Registrar_IDs]. Otherwise it is a value known between the
producer and the consumer, set via an out-of-band mechanism and
unique within all reports of the producer.

2.4.2.  Server_Registrant_ID

The identifier, issued by EPP server, assigned to the contact object
that is associated as registrant of the domain name that MUST conform
to "clIDType" specified in RFC 5730 [RFC5730].

2.4.3.  DNSSEC

The value MUST be either 'YES' or 'NO' to indicate whether the domain
is DNSSEC signed.

2.4.4.  Server_Contact_ID

The identifier of the contact object assigned by the registry system
and MUST conform to "clIDType" specified in RFC 5730 [RFC5730].

2.4.5.  Contact_Type

The value MUST be one of value as defined by "contactAttrType" in RFC
5731 [RFC5731].

2.4.6.  Contact_Name

The name of the contact object. Usually it is the name of an
individual or an organization as described in RFC 5733 [RFC5733]
Section 2.3.
2.4.7. In_use

The value MUST be either "YES" or "NO" to indicate whether the contact object is associated with a domain object.

2.4.8. Nameserver_Host

The full domain name of the host object as defined in RFC 5732 [RFC5732] Section 2.1. The name MUST be in A-label format as defined by RFC5890 [RFC5890].

2.4.9. Nameserver_IP

The IP address of the host object. The syntax of the IPv4 address MUST conform to RFC 791 [RFC0791]. The syntax of the IPv6 address MUST conform to RFC 4291 [RFC4291].

2.4.10. Client_Contact_ID

The identifier of the contact object assigned by the registrar and MUST conform to "clIDType" specified in RFC 5730 [RFC5730].

3. Report Definition Specification

Each report specification conceptually represents a file of comma separated values [RFC4180] (commonly called a CSV file) where the values are selected from the data elements specified above. The first row of the file is a comma separated list of data element names as specified in the data element registry. The remaining rows of the file are the unordered sets of data elements, one set per row, where each row is one transaction in the report.

Each data element in a set conceptually represents the column heading in a printed report.

A consumer MUST be able to receive data elements that are not recognized and MAY skip them accordingly, both in the header row and in the transaction rows.

A report is specified in the report registry with two pieces of information. First is the name of the report. This can be whatever is appropriate as defined by the producer of the report. The name of the report MUST be unique and this characteristic MUST be enforced by registry.
Second is the ordered list of data element names of what is included in the report. The data element names MUST be listed in the data element registry specified above. The data element names and the data MUST appear in the report in the order listed in the report registry.

The subsections below comprise an initial list of standard reports commonly being used between producers and consumers as of the date of publication of this document. The title of the subsection is the report name. The report name in the IANA registry MUST be unique and MUST be processed as case insensitive.

3.1. Domain Transaction

Name of report: domain_transaction

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLD</td>
<td>RFC XXXX Section 2.1.1</td>
</tr>
<tr>
<td>Server_TRID</td>
<td>Section 2.1.2</td>
</tr>
<tr>
<td>Domain</td>
<td>Section 2.1.3</td>
</tr>
<tr>
<td>DateTime</td>
<td>Section 2.1.6</td>
</tr>
<tr>
<td>Registrar_ID</td>
<td>Section 2.4.1</td>
</tr>
<tr>
<td>Registrar</td>
<td>Section 2.1.11</td>
</tr>
<tr>
<td>Transaction_Type</td>
<td>Section 2.1.4</td>
</tr>
<tr>
<td>Period</td>
<td>Section 2.1.12</td>
</tr>
<tr>
<td>Term</td>
<td>Section 2.1.7</td>
</tr>
<tr>
<td>Fee</td>
<td>Section 2.1.8</td>
</tr>
<tr>
<td>Currency</td>
<td>Section 2.1.9</td>
</tr>
<tr>
<td>Description</td>
<td>Section 2.1.13</td>
</tr>
</tbody>
</table>

Table 1: Transaction Report Definition Table
3.2. Premium Name

Name of report: premium_name

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLD</td>
<td>RFC XXXX Section 2.1.1</td>
</tr>
<tr>
<td>Domain</td>
<td>Section 2.1.3</td>
</tr>
<tr>
<td>Status</td>
<td>Section 2.1.10</td>
</tr>
<tr>
<td>Description</td>
<td>Section 2.1.13</td>
</tr>
<tr>
<td>Currency</td>
<td>Section 2.1.9</td>
</tr>
<tr>
<td>Domain_Create</td>
<td>Section 2.2.1</td>
</tr>
<tr>
<td>Domain_Renew</td>
<td>Section 2.2.2</td>
</tr>
<tr>
<td>Domain_Transfer</td>
<td>Section 2.2.3</td>
</tr>
<tr>
<td>Domain_Restore</td>
<td>Section 2.2.4</td>
</tr>
<tr>
<td>Start_Date</td>
<td>Section 2.3.1</td>
</tr>
</tbody>
</table>

Table 2: Premium Name Report Definition Table

3.3. Domain RGP

Name of report: domain_rgp
Table 3: Domain RGP Report Definition

3.4. Reserved Domain

Name of report: reserved_domain

Table 4: Reserved Domain Report Definition Table

3.5. Domain Inventory

Name of report: domain_inventory
Table 5: Domain Inventory Report Definition Table

3.6. Contact Inventory

Name of report: contact_inventory
Table 6: Contact Inventory Report
Definition Table

3.7. Host Inventory

Name of report: host_inventory

Table 7: Host Inventory Report
Definition Table

4. IANA Considerations

This section describes the format of the IANA Registration Report Registry, which has two tables described below, and the procedures used to populate and manage the registry entries.
4.1. Report Specification

This registry uses the "Specification Required" policy described in RFC 8126 [RFC8126]. An English language version of the extension specification is required in the registry, though non-English versions of the specification may also be provided.

The "Specification Required" policy implies review by a "designated expert". Section 5.2 of RFC 8126 [RFC8126] describes the role of designated experts and the function they perform.

4.1.1. Designated Expert Evaluation Criteria

A high-level description of the role of the designated expert is described in Section 5.2 of RFC 8126 [RFC8126]. Specific guidelines for the appointment of designated experts and the evaluation of a Registration Report is provided here.

The IESG SHOULD appoint a small pool of individuals (perhaps 3 - 5) to serve as designated experts, as described in Section 5.2 of RFC 8126 [RFC8126]. The pool should have a single administrative chair who is appointed by the IESG. The designated experts should use the existing regext mailing list (regext@ietf.org) for public discussion of registration requests. This implies that the mailing list should remain open after the work of the REGEXT working group has concluded.

The results of the evaluation should be shared via email with the registrant and the regext mailing list. Issues discovered during the evaluation can be corrected by the registrant, and those corrections can be submitted to the designated experts until the designated experts explicitly decide to accept or reject the registration request. The designated experts must make an explicit decision and that decision must be shared via email with the registrant and the regext mailing list. If the specification for a data element or report is an IETF Standards Track document, no review is required by the designated expert.

Designated experts should be permissive in their evaluation of requests for data elements and reports that have been implemented and deployed by at least one registry. This implies that it may indeed be possible to register multiple data elements or reports that provide the same functionality. Requests to register data elements or reports that have not been deployed should be evaluated with a goal of reducing duplication. A potential registrant who submits a request to register a new data element or report that includes similar functionality to existing data elements or reports should be made aware of the existing data elements and reports. The registrant should be asked to reconsider their request given the existence of
similar data elements or reports. Should they decline to do so, perceived similarity should not be a sufficient reason for rejection as long as all other requirements are met.

4.1.2. Registration Procedure

The registry contains information describing each registered data element or report. Registry entries are created and managed by sending forms to IANA that describe the data element or report for the registry entry.

4.1.2.1. Required Information

The required information must be formatted consistently using the following registration form. Form field names and values may appear on the same line.

4.1.2.1.1. Data Element Definition

Name of data element
   MUST be unique within the registry, enforced to be unique, and MUST be processed as case insensitive

Reference document
   MUST define the data element, SHOULD be a URL to a RFC, and SHOULD include the section number (or other detailed internal document reference), MAY be a URL to any document available under equivalent terms

Registrant
   Will be IESG for initial entries and all Standards Track specifications; otherwise as specified by the registrant

Status
   MUST be one of active, inactive, or unknown

4.1.2.1.2. Report Definition

Name of Report
   MUST be unique within the registry, enforced to be unique, and MUST be processed as case insensitive

Document Status
   MUST be one of active, inactive, or unknown
Reference document
MUST define the report, SHOULD be a URL to a RFC and SHOULD include the section number (or other detailed internal document reference), MAY be a URL to any document available under equivalent terms

Registrant
Will be IESG for initial entries and all Standards Track specifications; otherwise as specified by the registrant

TLD
MUST be "ANY" if the report is intended to be generally applicable or MAY be any top level domain formatted as defined by RFC 5890 [RFC5890] (or comma separated list of domains) and each MUST be an A-LABEL if the report is intended to have that scope

Status:active

4.1.2.2. Registration Processing

Registrants should send each registration form to IANA with a single record for incorporation into the registry. Send the form via email to iana@iana.org or complete the online form found on the IANA website. The subject line should indicate whether the enclosed form represents an insertion of a new record (indicated by the word "INSERT" in the subject line) or a replacement of an existing record (indicated by the word "MODIFY" in the subject line). At no time can a record be deleted from the registry. On receipt of the registration request, IANA will initiate review by the designated expert(s) if appropriate, who will evaluate the request using the criteria in Section 4.1.1 in consultation with the regext mailing list.

4.1.2.3. Updating Report Definition Registry Entries

When submitting changes to existing registry entries, include text in the "Notes" field of the registration form describing the change. Under normal circumstances, registry entries are only to be updated by the registrant. If the registrant becomes unavailable or otherwise unresponsive, the designated expert can submit a registration form to IANA to update the registrant information. Entries can change state from "Active" to "Inactive" and back again as long as state-change requests conform to the processing requirements identified in this document. In addition to entries that become "Inactive" due to a lack of implementation, entries for which a specification becomes consistently unavailable over time should be marked "Inactive" by the designated expert until the
specification again becomes reliably available.

4.2. Initial assignments

4.2.1. Data Element Definition in IANA Registry

---- BEGIN FORM ----

Name of data element:
TLD

Reference:
This RFC Section 2.1.1

Registrant:
IESG, iesg@ietf.org

Status:
Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:
Server_TRID

Reference:
This RFC Section 2.1.2

Registrant:
IESG, iesg@ietf.org

Status:
Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:
Domain

Reference:
This RFC Section 2.1.3

Registrant:
IESG, iesg@ietf.org
Status:
  Active

Name of data element:
  Transaction_Type

Reference:
  This RFC Section 2.1.4

Registrant:
  IESG, iesg@ietf.org

Status:
  Active

Name of data element:
  Object_Type

Reference:
  This RFC Section 2.1.5

Registrant:
  IESG, iesg@ietf.org

Status:
  Active

Name of data element:
  DateTime

Reference:
  This RFC Section 2.1.6

Registrant:
  IESG, iesg@ietf.org
Status:
  Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:
  Currency

Reference:
  This RFC Section 2.1.9

Registrant:
  IESG, iesg@ietf.org

Status:
  Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:
  Status

Reference:
  This RFC Section 2.1.10

Registrant:
  IESG, iesg@ietf.org

---- END FORM ----

---- BEGIN FORM ----

Name of data element:
  Term

Reference:
  This RFC Section 2.1.7

Registrant:
  IESG, iesg@ietf.org

Status:
  Active

---- END FORM ----

---- BEGIN FORM ----
Status:
   Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:
   Registrar

Reference:
   This RFC Section 2.1.11

Registrant:
   IESG, iesg@ietf.org

Status:
   Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:
   Period

Reference:
   This RFC Section 2.1.12

Registrant:
   IESG, iesg@ietf.org

Status:
   Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:
   Description

Reference:
   This RFC Section 2.1.13

Registrant:
   IESG, iesg@ietf.org
Status:
  Active

Name of data element:
  Domain_Create

Reference:
  This RFC Section 2.2.1

Registrant:
  IESG, iesg@ietf.org

Status:
  Active

Name of data element:
  Domain_Renew

Reference:
  This RFC Section 2.2.2

Registrant:
  IESG, iesg@ietf.org

Status:
  Active

Name of data element:
  Domain_Transfer

Reference:
  This RFC Section 2.2.3

Registrant:
  IESG, iesg@ietf.org
Status:
  Active

Name of data element:
  Domain_Restore

Reference:
  This RFC Section 2.2.4

Registrant:
  IESG, iesg@ietf.org

Status:
  Active

Name of data element:
  Start_Date

Reference:
  This RFC Section 2.3.1

Registrant:
  IESG, iesg@ietf.org

Status:
  Active

Name of data element:
  Deleted_Date

Reference:
  This RFC Section 2.3.2

Registrant:
  IESG, iesg@ietf.org
Status:
   Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:
   RGP_Date

Reference:
   This RFC Section 2.3.3

Registrant:
   IESG, iesg@ietf.org

Status:
   Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:
   Purge_Date

Reference:
   This RFC Section 2.3.4

Registrant:
   IESG, iesg@ietf.org

Status:
   Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:
   Updated_Date

Reference:
   This RFC Section 2.3.5

Registrant:
   IESG, iesg@ietf.org
Status:
    Active

Name of data element:
Create_Date

Reference:
    This RFC Section 2.3.6

Registrant:
    IESG, iesg@ietf.org

Status:
    Active

Name of data element:
Expiry_Date

Reference:
    This RFC Section 2.3.7

Registrant:
    IESG, iesg@ietf.org

Status:
    Active

Name of data element:
Registrar_ID

Reference:
    This RFC Section 2.4.1

Registrant:
    IESG, iesg@ietf.org
Status:
  Active

Name of data element:
  Server_Registrant_ID

Reference:
  This RFC Section 2.4.2

Registrant:
  IESG, iesg@ietf.org

Status:
  Active

Name of data element:
  DNSSEC

Reference:
  This RFC Section 2.4.3

Registrant:
  IESG, iesg@ietf.org

Status:
  Active

Name of data element:
  Server_Contact_ID

Reference:
  This RFC Section 2.4.4

Registrant:
  IESG, iesg@ietf.org
Name of data element: Contact_Type
Reference:
   This RFC Section 2.4.5
Registrant:
   IESG, iesg@ietf.org

Status:
   Active

Name of data element: Contact_Name
Reference:
   This RFC Section 2.4.6
Registrant:
   IESG, iesg@ietf.org

Status:
   Active

Name of data element: INUSE
Reference:
   This RFC Section 2.4.7
Registrant:
   IESG, iesg@ietf.org
Status:
  Active

Name of data element:
  Nameserver_Host

Reference:
  This RFC Section 2.4.8

Registrant:
  IESG, iesg@ietf.org

Status:
  Active

Name of data element:
  Nameserver_IP

Reference:
  This RFC Section 2.4.9

Registrant:
  IESG, iesg@ietf.org

Status:
  Active

Name of data element:
  Client_Contact_ID

Reference:
  This RFC Section 2.4.10

Registrant:
  IESG, iesg@ietf.org
4.2.2. Report Definition in IANA Registry

----- BEGIN FORM -----
Name of report:
  domain_transaction
Reference:
  This RFC Table 1
Registrant:
  IESG, iesg@ietf.org
TLD:
  any
Status:
  Active
----- END FORM -----

----- BEGIN FORM -----
Name of report:
  premium_name
Reference:
  This RFC Section 3.2
Registrant:
  IESG, iesg@ietf.org
TLD:
  any
Status:
  Active
----- END FORM -----
----- BEGIN FORM -----

Yee & Galvin            Expires 8 September 2022               [Page 29]
Name of report:
domain_rgp

Reference:
This RFC Section 3.3

Registrant:
IESG, iesg@ietf.org

TLD:
any

Status:
Active

---- END FORM ----

---- BEGIN FORM ----

Name of report:
reserved_domain

Reference:
This RFC Section 3.4

Registrant:
IESG, iesg@ietf.org

TLD:
any

Status:
Active

---- END FORM ----

---- BEGIN FORM ----

Name of report:
domain_inventory

Reference:
This RFC Section 3.5

Registrant:
IESG, iesg@ietf.org
TLD:
    any
Status:
    Active

---- END FORM ----

---- BEGIN FORM ----

Name of report:
    contact_inventory
Reference:
    This RFC Section 3.6
Registrant:
    IESG, iesg@ietf.org

TLD:
    any
Status:
    Active

---- END FORM ----

---- BEGIN FORM ----

Name of report:
    host_inventory
Reference:
    This RFC Section 3.7
Registrant:
    IESG, iesg@ietf.org

TLD:
    any
Status:
    Active

---- END FORM ----
5. Security Considerations

This specification does not consider the issues of distribution or access to the reports that are created and thus does not introduce any new security security concerns that are not already present in the local environment in which the report is created.

A security principle to keep in mind as new reports are developed is that it is considered a bad practice to report or disclose security information. In the case of the registration system upon which this reporting mechanism is based, the authInfo code is a specific example of a data element that SHOULD NOT be included in a report.

6. Privacy Considerations

This specification defines a mechanism for creating reports based on data in a registration system. Some of that data is likely to be considered personally identifiable information (PII) and thus would be subject to privacy protection according to an applicable privacy regulation. It is outside the scope of this specification to address those specific concerns. Implementors are urged to consider these issues with their local legal authority and develop appropriate requirements for their work.

As expressly noted in the Introduction, distribution of and access to the reports created by this specification is expressly outside the scope of this specification. However, to the extent a report contains PII, implementors are urged to consider these issues with their local legal authority and develop appropriate requirements for their work.

7. Internationalization Considerations

The character encoding for the file contents MUST use UTF-8.

Throughout this document A-LABEL is indicated as a SHOULD and that MUST be interpreted as follows. All domain name labels MUST be in A-LABEL format if it is possible to represent it as an A-LABEL, otherwise U-LABEL MAY be used.

8. References

8.1. Normative References


8.2. Informative References

Appendix A.  Acknowledgements

The authors would like to thank Roger Carney, Jody Kolker, Tobias Sattler, and bestpractice.domains for their reviews and suggestions.

Appendix B.  File Naming Convention

TBD on file naming convention suggestion

Authors’ Addresses

Joseph Yee (editor)
Donuts
Toronto
Canada
Email: jyee@afilias.info

James Galvin
Donuts
Horsham, PA
United States
Email: jgalvin@donuts.email
Abstract

This document describes how the Extensible Provisioning Protocol (EPP) is mapped over the Hypertext Transfer Protocol (HTTP). This mapping requires the use of the Transport Layer Security (TLS) protocol to protect information exchanged between an EPP client and an EPP server.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.
1. Introduction

Although the Extensible Provisioning Protocol (EPP) core specification [RFC5730] does not state the transport protocol, only the mapping over TCP [RFC5734] has been standardized thus far. Nevertheless, some EPP implementations leverage HTTP due to its ease of use and simplicity. This document describes the reasons behind using HTTP as the transport protocol for EPP and how EPP is mapped over HTTP preserving the semantics of commands.

HTTP is defined in some IETF documents according to the versions currently in use: HTTP/1.1 [RFC7230], HTTP/2 [RFC7540], HTTP/3 [I-D.ietf-quic-http]. As the differences among such versions do not affect the EPP mapping, hereinafter the version number is omitted except for presenting the special features in the underlying layers of the HTTP stack.

Security services beyond those defined in EPP are provided by the Transport Layer Security (TLS) protocol [RFC8446].
1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Reasons behind Using HTTP

Unlike TCP, HTTP is loosely coupled with the network and provides client-server cross-platform technology communication. Indeed, since an HTTP connection is a higher-level abstraction of a network connection, there is no need to take over all of the lower-level details of TCP. For example, while in TCP the data transmission between a client and a server starts only after having established a connection through a 3-way handshake (i.e. SYN, SYN-ACK, ACK), HTTP uses a one-way communication so that a client can directly issue a request to a server and then receive a response.

All the burden needed to manage the HTTP connections is usually performed by an application server, which a service can be deployed on. Service implementors are only required to process the requests and return the responses. Definitively, HTTP ease of use and simplicity reduces the development time.

While TCP is connection-oriented, HTTP is stateless but not session less. This means that, by making an EPP session untied from the network connection, the EPP communication over HTTP is more flexible and efficient than over TCP.

The main reason supporting the usage of TCP has always been its speed. TCP has been significantly faster than HTTP as HTTP was initially built on top of TCP so that every HTTP request should be issued on a new TCP connection. However, subsequent HTTP versions have been defined over time to increase the protocol speed and reduce the gap with TCP:

* Compared to the original HTTP specification, HTTP/1.1 introduced the "keep-alive" connection by default to enable a request-response sequence on a single TCP connection without repeating the connection handshake at each request;
* As opposed to HTTP/1.1, which keeps all requests and responses in plain text format, HTTP/2 defined the binary framing layer to encapsulate all messages in binary format;
* HTTP/3 is based on QUIC transport protocol [I-D.ietf-quic-transport]. QUIC uses UDP [RFC768] instead of TCP to exchange packets between the client and the server. It
incorporates TLS whereas HTTP/1.1 and HTTP/2 define TLS as an add-on. So doing, HTTP/3 can provide a very quick handshake to establish a secure connection.

In the perspective of moving to the cloud to achieve scalability and cost reduction, it should be further noted that application protocols that aren’t based on HTTP can be hardly migrated by using cloud-native features, both on client side and server side. In addition, from the security point of view, registries would be limited in terms of the third-party security services available to protect their EPP servers.

Finally, some considerations should be done about load balancing which is generally used by EPP operators to distribute the requests across a pool of servers and, consequently, provide an efficient domains registration and maintenance service. While HTTP load balancers are very common and are quite often software, TCP load balancers are usually implemented in dedicated hardware. In addition, HTTP load balancers don’t merely forward the traffic but can make high-level routing decisions based on the message content. With regard to the performance, although HTTP load balancers do more work, their throughput is evaluated considerably fast.

Additional notes on how EPP sessions can be managed in HTTP load balancing are included in Appendix A.

3. Message Exchange

EPP describes client-server interaction as a command-response exchange where the client sends one command to the server and the server returns one response to the client. A client MUST use the POST method (Section 3.3 of [RFC7231]) to issue an EPP command through the request body. A server receiving a request MUST return an EPP message in the response body using the "Content-Length" entity-header field to indicate the length in decimal number of OCTETs of the entity-body. No EPP message information MUST be issued through any other part of the request or the response. If the HTTP connection is closed after a server receives and successfully processes a command but before the response can be returned to the client, the server MAY attempt to undo the effects of the command to ensure a consistent state between the client and the server.

Commands MUST be processed independently and in the same order as received from the server. An EPP client MAY issue multiple EPP commands to an EPP server on an HTTP connection by relying on the HTTP keep-alive capability. A server SHOULD limit a client to a maximum number of HTTP connections based on server capabilities and operational load.
A client might be able to realize a slight performance gain by pipelining the requests, but this feature does not change the basic single command, single response operating mode of the EPP protocol. A server SHOULD limit the amount of time required for a client to issue a well-formed EPP command and, consequently close an open HTTP connection.

4. Session Management

The EPP session is implemented by using the mechanism described in [RFC6265]. An EPP session is started by the client issuing an EPP <login> command. A server receiving an EPP <login> command MUST use the "Set-Cookie" response header to send the client a token that the client will return in future requests within the scope of the EPP session. For example (Figure 1), the server can send the client a "session identifier" (a.k.a "session ID") named SID. The client then returns the session ID in the "Cookie" header of the subsequent requests.

```
== Server -> Client ==
Set-Cookie: SID=52ceb07c2a824f09a1c6f9c45574097d

== Client -> Server ==
Cookie: SID=52ceb07c2a824f09a1c6f9c45574097d
```

Figure 1

The name of the cookie attribute identifying the session ID is not relevant and depends on the implementations. Examples of the names that some programming languages use to represent the session ID include JSESSIONID (Java EE), PHPSESSID (PHP), and ASPSESSIONID (Microsoft ASP).

An EPP session is ended by the client issuing an EPP <logout> command. A server receiving an EPP <logout> command MUST end the EPP session invalidating it after having issued the <logout> response.

A client MAY open multiple EPP sessions and distribute commands from a single EPP session over multiple HTTP connections. A server SHOULD limit a client to a maximum number of EPP sessions based on server capabilities and operational load.

EPP sessions that are inactive for more than a server-defined period MAY be ended by a server invalidating the session.
Clients MAY issue the <hello> command outside an EPP session. In such a case, servers MUST return the <greeting> response without starting a session. To accomplish this, a server MAY return no cookie at all or provide the client with an expired cookie so that it cannot be used for further communication with the server. Clients MAY also issue the <hello> command within an EPP session to keep it alive.

The mechanism implemented by a server to maintain the relationship between a session and the EPP information negotiated with the client through the <login> command (e.g. the language, the namespace URIs representing both the objects and the extensions to be managed during the session) is out of the scope of this document.

The state machine described in Section 2 of [RFC5730] is updated as shown in Figure 2.
Figure 2
5. Return Codes

Servers MUST NOT use HTTP return codes to signal clients about the failure of the EPP commands. The HTTP code 200 MUST be used for both successful and unsuccessful EPP requests. Servers MUST use HTTP codes to signal clients about the failure of the HTTP requests.

Servers MUST return a 2002 response (i.e. Command use error) if the client issues an EPP command other than the <hello> and the <login> commands through HTTP requests including either an empty or an invalid session ID. Servers receiving a <login> command through an HTTP request including a session ID MAY return a 2002 response (i.e. Command use error) or simply ignore the incoming session ID.

6. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

6.1. IIT-CNR/Registro.it EPP Server

* Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it
* Location: https://epp.nic.it/ EPP endpoint available only "per IP address" basis.
* Description: The .it EPP server is deployed on WildFly Application Server. TLS versions supported are 1.2 and 1.3. Load balancing is implemented with NGINX. EPP sessions are maintained on a Redis cluster.
* Level of Maturity: This is a live implementation.
* Coverage: This implementation includes all of the features described in this specification except for the media type that is currently set to "text/xml".
* Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

6.2. .pl domain Registry (NASK) EPP Server

* Responsible Organization: .pl domain Registry (NASK)/dns.pl
* Location: https://dns.pl EPP endpoint available only "per IP address" basis.
* Description: It is an implementation of the EPP protocol that is used by .pl Registry.
* Level of Maturity: This is a live implementation.
* Coverage: This implementation includes all of the features described in this specification.
* Contact Information: Marcin Machnio, info@dns.pl

7. Transport Considerations

Section 2.1 of the EPP core specification [RFC5730] describes considerations to be addressed by protocol transport mappings. This document addresses each of the considerations using a combination of features described in this document and features provided by HTTP as follows:

* Sections 3.3.3 and 3.9.3 of [RFC8095] includes features to provide reliability, flow control, ordered delivery, and congestion control of, respectively, UDP and HTTP over TCP as Pseudotransport.
* Section 3 and Section 4 of this document describe how the stateful nature of EPP is preserved through controlled message exchanges and managed sessions.
* Section 3 of this document notes that command pipelining is possible with HTTP, though batch-oriented processing (combining multiple EPP commands in a single HTTP request) is not permitted.

8. IANA Considerations

This document has no actions for IANA.
9. Internationalization Considerations

Servers MUST use the "charset" attribute in the HTTP "Content-Type" response header field to specify the UTF-8 character encoding (e.g. Content-Type: application/epp+xml; charset=UTF-8).

10. Security Considerations

Since clients credentials are included in the EPP <login> command, the HTTP over TLS [RFC8740] MUST be used to protect them from disclosure while in transit. As well, the transfer over TLS prevents from sniffing the session ID and, consequently, impersonating a client to perform actions on registrars’ objects.

Anyway, servers are RECOMMENDED to implement additional measures to verify the client. These measures include IP whitelisting and locking the session ID to the client’s IP address.

As a further measure to enforce the security, servers MAY require clients to present a digital certificate. Clients who possess and present a valid X.509 digital certificate, issued by a recognized Certification Authority (CA), could be identified and authenticated by a server who trusts the corresponding CA. This certificate-based mechanism is supported by HTTPS and can be used with EPP over HTTP. The TLS protocol describes the specification of a client certificate in Section 7.4.6 of [RFC8446].

With regard to sessions, session IDs SHOULD be randomly generated to mitigate the risk of obtaining a valid one through a brute-force search. A session ID SHOULD be at least 128 bits or 16 bytes long. An example of a reliable session ID is the Universally Unique Identifier (UUID). Servers MAY limit the lifetime of active sessions to avoid them being exchanged for a long time.

The following measures MAY also be taken to control cookies usage:

- restricting their scope through the "Domain" and "Path" attributes;
- limiting their lifetime through the "Max-Age" and "Expire" attributes.

Other attributes that are normally used to secure the cookies and prevent them to be accessed from unintended parties or scripts, such as "HttpOnly" and "Secure", are meaningless in this context.
Finally, servers are RECOMMENDED to perform additional checks to limit the rate of open EPP sessions and HTTP connections to mitigate the risk of congestion of requests. Here again, IP whitelisting could also be implemented to prevent DDoS attacks.

If the EPP server is configured as a load balancer routing the requests to a pool of backend servers, some of the aforementioned checks SHOULD be implemented on the load balancer side.

11. Acknowledgements

The authors would like to acknowledge the following individuals for their contributions to this document: Cristian Lucchesi, Stefano Ruberti, Luca Vasarelli, Roberto Ravazzolo from IIT-CNR/Registro.it and Adrian Prokop, Sławomir Mateuszczynk from NASK/.pl Registry.

12. References

12.1. Normative References


12.2. Informative References


Appendix A. Notes on Load Balancing

An EPP server should be able to serve a large number of concurrent requests from clients and return the responses in a fast and reliable manner. In addition, since EPP is extensible, EPP servers might be updated and the replacement of an EPP server with a new version should take place in accordance with the service level agreement negotiated between the registry and the registrars. To cost-effectively scale high volumes of requests and redeploy a server without affecting its functioning, best practice in providing a software service generally requires using load balancing. This section presents two possible approaches to the implementation of a HTTP load balancing solution for an EPP server.

An EPP server made up of a server pool must always operate with respect to the constraint that, once an EPP session is established, all the requests related to that session should be processed by the servers in the pool as long as the session is alive.

One possible approach is using sticky sessions. In this case, the load balancer assigns an identifier to each client issuing a request. Then, according to such identifier, the load balancer can route all of the requests of a given client to the backend server that started the session for its entire duration. This approach requires each backend server to maintain the EPP information connected to the sessions opened by that server. This means that when a backend server is stopped and then restarted after its update, all the sessions currently active and managed by that server are lost.
A more efficient solution consists in releasing the sessions from the server pool. According to this approach, every session is stored somewhere outside the server pool. The load balancer distributes the request based on the load of each backend server and according to a specific algorithm. When a server receives a request, it first retrieves the session information by the session ID and, if any, processes the request. Sessions are normally stored in a cluster of NO-SQL databases so that performance and efficiency requirements are fulfilled. In this approach, only the ongoing requests are lost when a backend server is stopped and restarted. Moreover, maintaining the sessions on a persistent data storage results in supporting a virtually unlimited number of concurrent sessions.

Authors’ Addresses

Mario Loffredo  
IIT-CNR/Registro.it  
Via Moruzzi,1  
56124 Pisa  
Italy  
Email: mario.loffredo@iit.cnr.it  
URI: https://www.iit.cnr.it

Lorenzo Luconi Trombacchi  
IIT-CNR/Registro.it  
Via Moruzzi,1  
56124 Pisa  
Italy  
Email: lorenzo.luconi@iit.cnr.it  
URI: https://www.iit.cnr.it

Maurizio Martinelli  
IIT-CNR/Registro.it  
Via Moruzzi,1  
56124 Pisa  
Italy  
Email: maurizio.martinelli@iit.cnr.it  
URI: https://www.iit.cnr.it

Jan Romanowski  
NASK/.pl Registry  
Kolska 12  
01-045 Warszawa  
Poland  
Email: jan.romanowski@nask.pl
URI: https://www.dns.pl/

Marcin Machnio
NASK/.pl Registry
Kolska 12
01-045 Warszawa
Poland
Email: info@dns.pl
URI: https://www.dns.pl/