

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 13 January 2022

J.G. Gould
D.S. Smith
VeriSign, Inc.
J.K. Kolker
R.C. Carney
GoDaddy Inc.
12 July 2021

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 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 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.

Table of Contents

1. Introduction	2
2. Conventions Used in This Document	2
3. Redaction Methods	3
3.1. Redaction by Removal Method	3
3.2. Redaction by Empty Value Method	4
4. Redacted RDAP Response	5
4.1. RDAP Conformance	5
4.2. "redacted" Member	5
5. JSONPath Considerations	21
6. IANA Considerations	21
6.1. RDAP Extensions Registry	21
7. Security Considerations	22
8. Acknowledgements	22
9. References	22
9.1. Normative References	22
Authors' Addresses	23

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.

The JSON examples include extra line breaks and whitespace. For instance, the JSONPath expressions are broken out into multiple lines when required for illustration.

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 sub-sections.

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": "$.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": "jsonpath",
    "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 "jsonpath" 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:

```
{
  "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",
```



```
        "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",
```

```
        "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"
      }
    ]
  ]
]
```

```

        },
        "uri",
        "tel:+1-555-555-1236;ext=789"
      ],
      [
        "tel",
        {
          "type": "fax"
        },
        "uri",
        "tel:+1-555-555-6321"
      ]
    ]
  }
],
"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"
]
}

```

Example redacted version of RDAP response:

```

{
  "rdapConformance": [
    "rdap_level_0",
    "redacted_0"
  ],
  "objectClassName": "domain",
  "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"
      ],
      [
        "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",
        ""
      ],
      [
        "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,
<<https://tools.ietf.org/html/draft-ietf-jsonpath-base-00>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6350] Perreault, S., "vCard Format Specification", RFC 6350,
DOI 10.17487/RFC6350, August 2011,
<<https://www.rfc-editor.org/info/rfc6350>>.
- [RFC7095] Kewisch, P., "jCard: The JSON Format for vCard", RFC 7095,
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,
<<https://www.rfc-editor.org/info/rfc7483>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.

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>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 5 October 2022

D. Belyavskiy

J. Gould
VeriSign, Inc.
3 April 2022

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:

URI: urn:ietf:params:xml:schema:epp:eai-1.0
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:

Name of Extension: Use of Internationalized Email Addresses
in EPP protocol
Document status: Standards Track
Reference: TBA
Registrant Name and Email Address: IESG, <iesg@ietf.org>
Top-Level Domains(TLDs): Any
IPR Disclosure: None
Status: Active
Notes: None

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.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

URL: https://www.verisign.com/en_US/channel-resources/domain-registry-products/epp-sdks

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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.27487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.27487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <<https://www.rfc-editor.org/info/rfc5321>>.
- [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <<https://www.rfc-editor.org/info/rfc5322>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.27487/RFC5730, August 2009, <<https://www.rfc-editor.org/info/rfc5730>>.

- [RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.27487/RFC5733, August 2009, <<https://www.rfc-editor.org/info/rfc5733>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<https://www.rfc-editor.org/info/rfc5890>>.
- [RFC6530] Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, DOI 10.27487/RFC6530, February 2012, <<https://www.rfc-editor.org/info/rfc6530>>.
- [RFC6531] Yao, J. and W. Mao, "SMTP Extension for Internationalized Email", RFC 6531, DOI 10.17487/RFC6531, February 2012, <<https://www.rfc-editor.org/info/rfc6531>>.
- [RFC6532] Yang, A., Steele, S., and N. Freed, "Internationalized Email Headers", RFC 6532, DOI 10.17487/RFC6532, February 2012, <<https://www.rfc-editor.org/info/rfc6532>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.27487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

10.2. Informative References

- [RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.27487/RFC5892, August 2010, <<https://www.rfc-editor.org/info/rfc5892>>.
- [RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.27487/RFC7451, February 2015, <<https://www.rfc-editor.org/info/rfc7451>>.
- [RFC8543] Zhou, L., Kong, N., Yao, J., Gould, J., and G. Zhou, "Extensible Provisioning Protocol (EPP) Organization Mapping", RFC 8543, DOI 10.27487/RFC8543, March 2019, <<https://www.rfc-editor.org/info/rfc8543>>.

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: belldmit@gmail.com

James Gould
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States of America
Email: jgould@verisign.com
URI: <http://www.verisigninc.com>

Internet Engineering Task Force (IETF)
Internet-Draft
Intended status: Standards Track
Expires: December 17, 2021

T. Sattler
R. Carney
J. Kolker
GoDaddy Inc.
October 11, 2021

Registry Maintenance Notification for the
Extensible Provisioning Protocol (EPP)
draft-ietf-regext-epp-registry-maintenance-19

Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension called "Registry Maintenance Notification", used by EPP servers to notify EPP clients and allow EPP clients to query EPP servers regarding maintenance events.

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 December 17, 2021.

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 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 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.

Table of Contents

1.	Introduction	3
1.1.	Terminology and Definitions	3
2.	Migrating to Newer Versions of This Extension	4
3.	Object Attributes	4
3.1.	Internationalized Domain Names	4
3.2.	Dates and Times	4
3.3.	Maintenance Elements	4
4.	EPP Command Mapping	7
4.1.	EPP Query Commands	7
4.1.1.	EPP <info> Command	7
4.1.1.1.	Info Maintenance Item	7
4.1.1.2.	Info Maintenance List	9
4.1.2.	EPP <poll> Command	10
4.2.	EPP Transform Commands	12
5.	Formal Syntax	12
5.1.	Registry Maintenance Notification EPP Mapping Schema	12
6.	IANA Considerations	17
6.1.	XML Namespace	17
6.2.	EPP Extension Registry	17
7.	Security Considerations	18
8.	Implementation Status	18
8.1.	GoDaddy Registry	18
8.2.	TANGO Registry Services	19
9.	References	19
9.1.	Normative References	19
9.2.	Informative References	20
	Appendix A. Change History	20
A.1.	Change from draft-sattler-epp-poll-maintenance-response to draft-sattler-epp-registry-maintenance	20
A.2.	Change from draft-sattler-epp-registry-maintenance to draft-ietf-regext-epp-registry-maintenance	20
A.3.	Change from 00 to 01	21
A.4.	Change from 01 to 02	21
A.5.	Change from 02 to 03	21
A.6.	Change from 03 to 04	21
A.7.	Change from 04 to 05	21
A.8.	Change from 05 to 06	21
A.9.	Change from 06 to 07	21
A.10.	Change from 07 to 08	21
A.11.	Change from 08 to 09	21
A.12.	Change from 09 to 10	21
A.13.	Change from 10 to 11	22
A.14.	Change from 11 to 12	22
A.15.	Change from 12 to 13	22
A.16.	Change from 13 to 14	22
A.17.	Change from 14 to 15	22
A.18.	Change from 15 to 16	22
A.19.	Change from 16 to 17	22
A.20.	Change from 17 to 18	22
A.21.	Change from 18 to 19	22
	Acknowledgments	22
	Authors' Addresses	23

1. Introduction

The Extensible Provisioning Protocol (EPP), as defined in [RFC5730], is a protocol whose original motivation is to provide a standard Internet domain name registration protocol for use between registries and registrars.

Registries routinely update systems to ensure a higher quality of service, implement new services, or upgrade protocols to the latest standards. These updates are pushed to various registry environments during time frames communicated to registrars as "maintenance events". Maintenance events may require making services unavailable for some limited time while the upgrade happens. Registries usually inform registrars about maintenance events in various formats, none of which are standardized between registries.

The DNS namespace expansion has led to many additional registries that registrars must interact with, adding more maintenance events and formats. It is now desirable to provide an efficient approach to notify registrars.

This document describes an extension mapping for version 1.0 of the EPP to provide a mechanism by which EPP servers may notify EPP clients of and allow EPP clients to query EPP servers on upcoming maintenance events.

1.1. Terminology and Definitions

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.

XML [W3C.REC-xml11-20060816] is case-sensitive. Unless stated otherwise, XML specification and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.

"maint" is used as an abbreviation for "urn:ietf:params:xml:ns:epp:maintenance-1.0". The XML namespace prefix "maint" is used, but implementations MUST NOT depend on it. Instead, they are to employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

"ote" is an abbreviation for "Operational Test and Evaluation".

In examples, "C:" represents lines sent by a protocol client, and "S:" represents lines returned by a protocol server. Indentation and white space in examples are provided only to illustrate element relationships and are not a required feature of this protocol.

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.

Servers SHOULD (for a temporary migration period up to server policy) provide support for older versions of the extension in parallel to the newest version and allow clients to execute their preferred version of the <info> command based on the maintenance <objURI> elements of the server <greeting>. The version of the maintenance <info> response MUST match the version of the maintenance <info> command executed by the server.

Servers MUST return a Registry Maintenance Notification poll message matching the newest negotiated version of the maintenance extension, based on an intersection of the maintenance <objURI> elements in the server <greeting> and the client <login> command. If the intersection of the maintenance <objURI> elements of the server <greeting> and the client <login> command results in an empty set, the server MUST return the newest version of the Registry Maintenance Notification poll message supported by the server based on "Usage with Poll-Message EPP Responses" in Section 6 of [RFC9038].

3. Object Attributes

3.1. Internationalized Domain Names

Names of affected hosts MUST be provided in A-label form, according to [RFC5891].

3.2. Dates and Times

All date and time attribute values MUST be expressed in Universal Coordinated Time (UTC) using the Gregorian calendar. The date-time format defined as "date-time" in [RFC3339], with time-offset="Z", MUST be used.

3.3. Maintenance Elements

The <maint:item> element describes a single registry maintenance event during a specific period. This element is used in a maintenance item EPP <info> command and response, and <poll> response.

If an element is not marked as optional, it is mandatory.

<maint:id>

The server unique identifier for the maintenance event with the OPTIONAL "name" attribute that includes a human-readable name of the event. The server unique identifier SHALL NOT be changed if the event is updated or deleted. When the "name" attribute is set, the OPTIONAL "lang" attribute MAY be present to identify the

language if the negotiated value is something other than the default value of "en" (English).

<maint:type>

Zero or more OPTIONAL types of the maintenance event, with the possible set of values defined by server policy, such as "Routine Maintenance", "Software Update", "Software Upgrade", or "Extended Outage". The OPTIONAL "lang" attribute MAY be present to identify the language if the negotiated value is something other than the default value of "en" (English).

<maint:pollType>

The OPTIONAL <maint:pollType> element for a Registry Maintenance Notification poll message; values MUST either be "create", "update", "delete", "courtesy", or "end". For the "create" and "update" types, the server includes the state of the maintenance event after the creation or update. For the "delete" type, the server includes the state of the event before the delete. The "courtesy" provides a reminder of an event, and the "end" provides a notification of the end of the event without updating the maintenance object and includes the latest state of the event. This element MUST be present only for poll messages.

<maint:systems>

One or more <maint:system> elements that are affected by the maintenance event.

<maint:system>

The <maint:system> element contains the following child elements:

<maint:name>

The name of the affected system, such as "EPP", "WHOIS", "DNS", "Portal", "RDAP", etc.

<maint:host>

The OPTIONAL affected maintained system's hostname, which SHALL be in A-label form, according to [RFC5891].

<maint:impact>

The impact level; the values MUST either be "full", "partial", or "none". If access is expected to be intermittently unavailable, it is "partial". If access is expected to be completely unavailable, it is "full". If access is not affected, it is "none".

<maint:environment>

The type of the affected system; the attribute "type" is REQUIRED and MUST either be "production", "ote", "staging", "dev" or "custom". For extensibility, the <maint:environment> element includes the OPTIONAL "name" attribute that can define the name of the custom environment when the <maint:environment> element "type" attribute has the "custom" value. For example, for the custom "marketing" environment, the <maint:environment> element should

<maint:start>

The date and time of the start of the maintenance event.

<maint:end>

The date and time of the end of the maintenance event. The <maint:end> element MUST be greater than the <maint:start> element.

<maint:reason>

The reason behind the maintenance event; the values MUST either be "planned" or "emergency".

<maint:detail>

The OPTIONAL URI to the detailed maintenance event description, formatted according to [RFC3986].

<maint:description>

Zero or more OPTIONAL free-form descriptions of the maintenance event, usable without creating and traversing an external resource as defined by the <maint:detail> element. The OPTIONAL "lang" attribute MAY be present to identify the language if the negotiated value is something other than the default value of "en" (English). The OPTIONAL "type" attribute MAY be present to identify the format of the description. It MUST either be "plain" for plain text or "html" for HTML text that is defined in [W3C.REC-html5-20141028] and XML-escaped, with a default value of "plain".

<maint:tlds>

The OPTIONAL <maint:tlds> element contains one or more <maint:tld> child elements. If the <maint:tlds> is not present, the entire system is affected.

<maint:tld>

The affected top-level domain or registry zone, which SHALL be in A-label form, according to [RFC5891].

<maint:intervention>

The OPTIONAL <maint:intervention> element contains the following child elements:

<maint:connection>

The value SHALL be boolean and indicates if a client needs to perform a connection-related action, such as a reconnect. The attribute should only be used as a flag to indicate connections will be affected. Servers SHOULD include a description of how the connections are affected in the <maint:description> element or use the <maint:detail> element above.

<maint:implementation>

The value SHALL be boolean and indicates if a client needs to perform an implementation-related action, such as a code

change. The attribute should only be used as a flag to indicate implementation will be affected. Servers SHOULD include a description of how the implementation is affected in the <maint:description> element or use the <maint:detail> element above.

<maint:crDate>

The date and time of the maintenance object creation.

<maint:upDate>

The OPTIONAL date and time of the most recent maintenance object modification. This element MUST NOT be present if the maintenance object has never been modified.

4. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730]. The command mappings described here are specifically used to notify registrars of registry maintenance events and object mapping.

4.1. EPP Query Commands

EPP [RFC5730] provides three commands to retrieve object information: <check> to determine if an object is known to the server, <info> to retrieve detailed information associated with an object, and <transfer> to retrieve object transfer status information.

This extension does not add any elements to EPP <check> and <transfer> commands or responses.

4.1.1. EPP <info> Command

EPP provides the <info> command that is used to retrieve registry maintenance information. In addition to the standard EPP command elements, the <info> command MUST contain a <maint:info> element that identifies the maintenance namespace.

The <maint:info> element MUST contain a child element. It is either the <maint:id> child element, described in Section 4.1.1.1, to query for a specific maintenance item or the <maint:list> child element, described in Section 4.1.1.2, to query all maintenance items.

4.1.1.1. Info Maintenance Item

The information regarding a specific maintenance item can be retrieved by using the <info> command with the <maint:info> element and the <maint:id> child element, defined in Section 3.3. If the maintenance identifier does not exist, the server MUST return an EPP error result code of 2303 ("Object does not exist") [RFC5730].

Example to retrieve a specific maintenance item in an <info> command.

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <maint:info
C:        xmlns:maint="urn:ietf:params:xml:ns:epp:maintenance-1.0">
C:          <maint:id>2e6df9b0-4092-4491-bcc8-9fb2166dcee6</maint:id>
C:        </maint:info>
C:      </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When an `<info>` command has been processed successfully, the EPP `<resData>` element MUST contain a child `<maint:infData>` element that identifies the maintenance namespace. The `<maint:infData>` element contains the `<maint:item>` element defined in Section 3.3.

Example of returning a specific maintenance item in an `<info>` response.

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <maint:infData
S:        xmlns:maint="urn:ietf:params:xml:ns:epp:maintenance-1.0">
S:        <maint:item>
S:          <maint:id>2e6df9b0-4092-4491-bcc8-9fb2166dcee6
S:          </maint:id>
S:          <maint:type lang="en">Routine Maintenance</maint:type>
S:          <maint:systems>
S:            <maint:system>
S:              <maint:name>EPP</maint:name>
S:              <maint:host>epp.registry.example
S:              </maint:host>
S:              <maint:impact>full</maint:impact>
S:            </maint:system>
S:          </maint:systems>
S:          <maint:environment type="production"/>
S:          <maint:start>2021-12-30T06:00:00Z</maint:start>
S:          <maint:end>2021-12-30T07:00:00Z</maint:end>
S:          <maint:reason>planned</maint:reason>
S:          <maint:detail>
S:            https://www.registry.example/notice?123
S:          </maint:detail>
S:          <maint:description lang="en">free-text
S:          </maint:description>
S:          <maint:description lang="de">Freitext
S:          </maint:description>
```

```

S:      <maint:tlds>
S:      <maint:tld>example</maint:tld>
S:      <maint:tld>test</maint:tld>
S:      </maint:tlds>
S:      <maint:intervention>
S:      <maint:connection>>false</maint:connection>
S:      <maint:implementation>>false</maint:implementation>
S:      </maint:intervention>
S:      <maint:crDate>2021-11-08T22:10:00Z</maint:crDate>
S:      </maint:item>
S:      </maint:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>

```

4.1.1.2. Info Maintenance List

The information for a list of maintenance items can be retrieved by using the `<info>` command with the `<maint:info>` element and the empty `<maint:list>` child element. Server policy determines if completed maintenance events will be included in the list of maintenance items.

Example to retrieve the list of maintenance items in an `<info>` command.

```

C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <maint:info
C:        xmlns:maint="urn:ietf:params:xml:ns:epp:maintenance-1.0">
C:        <maint:list/>
C:      </maint:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>

```

When an `<info>` command has been processed successfully, the EPP `<resData>` element MUST contain a child `<maint:infData>` element that identifies the maintenance namespace. The `<maint:infData>` element contains the `<maint:list>` element with zero or more `<maint:listItem>` child elements. The `<maint:listItem>` element contains the following child elements:

`<maint:id>`
The `<maint:id>` element defined in Section 3.3.

`<maint:start>`
The `<maint:start>` element defined in Section 3.3.

<maint:end>

The <maint:end> element defined in Section 3.3.

<maint:crDate>

The <maint:crDate> element defined in Section 3.3.

<maint:upDate>

The OPTIONAL <maint:upDate> element defined in Section 3.3.

Example of returning the list of maintenance items in an <info> response.

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <maint:infData
S:        xmlns:maint="urn:ietf:params:xml:ns:epp:maintenance-1.0">
S:        <maint:list>
S:          <maint:listItem>
S:            <maint:id>2e6df9b0-4092-4491-bcc8-9fb2166dcee6
S:          </maint:id>
S:            <maint:start>2021-12-30T06:00:00Z</maint:start>
S:            <maint:end>2021-12-30T07:00:00Z</maint:end>
S:            <maint:crDate>2021-11-08T22:10:00Z</maint:crDate>
S:          </maint:listItem>
S:          <maint:listItem>
S:            <maint:id>91e9dabf-c4e9-4c19-a56c-78e3e89c2e2f
S:          </maint:id>
S:            <maint:start>2021-12-15T04:30:00Z</maint:start>
S:            <maint:end>2021-12-15T05:30:00Z</maint:end>
S:            <maint:crDate>2021-11-08T22:11:00Z</maint:crDate>
S:            <maint:upDate>2021-11-17T15:00:00Z</maint:upDate>
S:          </maint:listItem>
S:        </maint:list>
S:      </maint:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

4.1.2. EPP <poll> Command

The EPP <poll> command and response are defined in Section 2.9.2.3 of [RFC5730]. The Registry Maintenance Notification is included in the EPP <poll> response of [RFC5730].

There are five types of poll messages for the Registry Maintenance Notification, defined by the <maint:pollType> element in Section 3.3. A poll message might be generated when a maintenance event is created, updated, or deleted. A courtesy poll message can be sent as a reminder of an upcoming maintenance event. An end poll message can be sent when the maintenance event is completed. In the case of a Registry Maintenance specific message, a <maint:infData> element, that identifies the maintenance namespace will be included within the <resData> element of the standard <poll> response. The <maint:infData> element contains the <maint:item> element defined in Section 3.3.

Example <poll> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <poll op="req"/>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <poll> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:    </result>
S:    <msgQ count="1" id="12345">
S:      <qDate>2021-11-08T22:10:00Z</qDate>
S:      <msg lang="en">Registry Maintenance Notification</msg>
S:    </msgQ>
S:    <resData>
S:      <maint:infData
S:        xmlns:maint="urn:ietf:params:xml:ns:epp:maintenance-1.0">
S:        <maint:item>
S:          <maint:id>2e6df9b0-4092-4491-bcc8-9fb2166dcee6</maint:id>
S:          <maint:pollType>create</maint:pollType>
S:          <maint:systems>
S:            <maint:system>
S:              <maint:name>EPP</maint:name>
S:              <maint:host>epp.registry.example
S:            </maint:host>
S:            <maint:impact>full</maint:impact>
S:          </maint:system>
S:        </maint:systems>
S:        <maint:environment type="production"/>
S:        <maint:start>2021-12-30T06:00:00Z</maint:start>
S:        <maint:end>2021-12-30T07:00:00Z</maint:end>
S:        <maint:reason>planned</maint:reason>
```



```

S:      <maint:detail>
S:        https://www.registry.example/notice?123
S:      </maint:detail>
S:      <maint:tlds>
S:        <maint:tld>example</maint:tld>
S:        <maint:tld>test</maint:tld>
S:      </maint:tlds>
S:      <maint:intervention>
S:        <maint:connection>false</maint:connection>
S:        <maint:implementation>false</maint:implementation>
S:      </maint:intervention>
S:      <maint:crDate>2021-11-08T22:10:00Z</maint:crDate>
S:    </maint:item>
S:  </maint:infData>
S: </resData>
S: <trID>
S:   <clTRID>ABC-12345</clTRID>
S:   <svTRID>54321-XYZ</svTRID>
S: </trID>
S: </response>
S: </epp>

```

4.2. EPP Transform Commands

EPP provides five commands to transform objects: <create> to create an instance of an object, <delete> to delete an instance of an object, <renew> to extend the validity period of an object, <transfer> to manage object sponsorship changes, and <update> to change information associated with an object.

This extension does not add any elements to the EPP <create>, <delete>, <renew>, <transfer>, and <update>.

5. Formal Syntax

The EPP Registry Maintenance Notification schema is presented here.

The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances. The <CODE BEGINS> and <CODE ENDS> tags are not part of the schema; they are used to note the beginning and end of the schema for URI registration purposes.

5.1. Registry Maintenance Notification EPP Mapping Schema

```

<CODE BEGINS>
<?xml version="1.0" encoding="UTF-8"?>
  <schema targetNamespace="urn:ietf:params:xml:ns:epp:
    maintenance-1.0"
    xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
    xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
    xmlns:maint="urn:ietf:params:xml:ns:epp:maintenance-1.0"
    xmlns="https://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">

```

```
<!--
Import common element types
-->
<import namespace="urn:ietf:params:xml:ns:eppcom-1.0"/>
<import namespace="urn:ietf:params:xml:ns:epp-1.0"/>
<annotation>
  <documentation>
    Extensible Provisioning Protocol v1.0
    Registry Maintenance Notification Mapping Schema.
  </documentation>
</annotation>
<!--
Child elements found in EPP commands.
-->
<element name="info" type="maint:infoType"/>
<!--
Child elements of the <info> command.
-->
<complexType name="infoType">
  <sequence>
    <choice>
      <element name="list"/>
      <element name="id" type="maint:idType"/>
    </choice>
  </sequence>
</complexType>
<!--
Human-readable text may describe the maintenance
-->
<complexType name="idType">
  <simpleContent>
    <extension base="token">
      <attribute name="name" type="token"/>
      <attribute name="lang" type="language" default="en"/>
    </extension>
  </simpleContent>
</complexType>
<!--
Info Response element
-->
<element name="infData" type="maint:infDataType"/>
<!--
<info> response elements.
-->
<complexType name="infDataType">
  <choice>
    <element name="list" type="maint:listDataType"/>
    <element name="item" type="maint:maintDataType"/>
  </choice>
</complexType>
<!--
Attributes associated with the list info response
-->
```

```

<complexType name="listDataType">
  <sequence>
    <element name="listItem" type="maint:maintItemType"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
<!--
  Attributes associated with the list item info response
-->
<complexType name="maintItemType">
  <sequence>
    <element name="id" type="maint:idType"/>
    <element name="start" type="dateTime"/>
    <element name="end" type="dateTime"/>
    <element name="crDate" type="dateTime"/>
    <element name="upDate" type="dateTime" minOccurs="0"/>
  </sequence>
</complexType>
<!--
  Attributes associated with the maintenance info response
-->
<complexType name="maintDataType">
  <sequence>
    <element name="id" type="maint:idType"/>
    <element name="type" type="maint:typeType" minOccurs="0"
      maxOccurs="unbounded"/>
    <element name="pollType" type="maint:pollType" minOccurs="0"/>
    <element name="systems" type="maint:systemsType"/>
    <element name="environment" type="maint:envType"/>
    <element name="start" type="dateTime"/>
    <element name="end" type="dateTime"/>
    <element name="reason" type="maint:reasonEnum"/>
    <element name="detail" type="anyURI" minOccurs="0"/>
    <element name="description" type="maint:descriptionType"
      minOccurs="0" maxOccurs="unbounded"/>
    <element name="tlds" type="maint:tldsType" minOccurs="0"/>
    <element name="intervention" type="maint:interventionType"
      minOccurs="0"/>
    <element name="crDate" type="dateTime"/>
    <element name="upDate" type="dateTime" minOccurs="0"/>
  </sequence>
</complexType>
<!--
  systems element
-->
<complexType name="systemsType">
  <sequence>
    <element name="system" type="maint:systemType"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>
<!--
  Enumerated list of poll types
-->

```

```
<simpleType name="pollType">
  <restriction base="token">
    <enumeration value="create"/>
    <enumeration value="update"/>
    <enumeration value="delete"/>
    <enumeration value="courtesy"/>
    <enumeration value="end"/>
  </restriction>
</simpleType>
<!--
Enumerated list of impacts
-->
<simpleType name="impactEnum">
  <restriction base="token">
    <enumeration value="none"/>
    <enumeration value="partial"/>
    <enumeration value="full"/>
  </restriction>
</simpleType>
<!--
description element
-->
<complexType name="descriptionType">
  <simpleContent>
    <extension base="string">
      <attribute name="lang" type="language" default="en"/>
      <attribute name="type" type="maint:descEnum" default="plain"
        />
    </extension>
  </simpleContent>
</complexType>
<!--
Enumerated list of description mime types
-->
<simpleType name="descEnum">
  <restriction base="token">
    <enumeration value="plain"/>
    <enumeration value="html"/>
  </restriction>
</simpleType>
<!--
type element
-->
<complexType name="typeType">
  <simpleContent>
    <extension base="string">
      <attribute name="lang" type="language" default="en"/>
    </extension>
  </simpleContent>
</complexType>
<!--
system element
-->
```

```
<complexType name="systemType">
  <sequence>
    <element name="name" type="token"/>
    <element name="host" type="eppcom:labelType" minOccurs="0"/>
    <element name="impact" type="maint:impactEnum"/>
  </sequence>
</complexType>
<!--
  Enumerated list of environments
-->
<simpleType name="envEnum">
  <restriction base="token">
    <enumeration value="production"/>
    <enumeration value="ote"/>
    <enumeration value="staging"/>
    <enumeration value="dev"/>
    <enumeration value="custom"/>
  </restriction>
</simpleType>
<!--
  environment element
-->
<complexType name="envType">
  <simpleContent>
    <extension base="token">
      <attribute name="type" type="maint:envEnum" use="required"/>
      <attribute name="name" type="token" use="optional"/>
    </extension>
  </simpleContent>
</complexType>
<!--
  Enumerated list of reasons
-->
<simpleType name="reasonEnum">
  <restriction base="token">
    <enumeration value="planned"/>
    <enumeration value="emergency"/>
  </restriction>
</simpleType>
<!--
  tlds element
-->
<complexType name="tldsType">
  <sequence>
    <element name="tld" type="eppcom:labelType"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>
<!--
  intervention element
-->
<complexType name="interventionType">
  <sequence>
```

```
<element name="connection" type="boolean"/>
<element name="implementation" type="boolean"/>
</sequence>
</complexType>
<!--
End of schema.
-->
</schema>
<CODE ENDS>
```

6. IANA Considerations

6.1. XML Namespace

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism defined in [RFC3688].

Registration request for the maintenance namespace:

URI: urn:ietf:params:xml:ns:epp:maintenance-1.0

Registrant Contact: IESG

XML: None. Namespace URIs do not represent an XML specification.

Registration request for the maintenance schema:

URI: urn:ietf:params:xml:schema:epp:maintenance-1.0

Registrant Contact: IESG

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

6.2. EPP Extension Registry

The following registration of the EPP Extension Registry, described in [RFC7451], is requested:

Name of Extension: Registry Maintenance Notification for the Extensible Provisioning Protocol (EPP)

Document status: Standards Track

Reference: (insert the reference to RFC version of this document)

Registrant Name and Email Address: IESG <iesg@ietf.org>

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

7. Security Considerations

The security considerations of [RFC5730] apply in this document. Additionally, a server MUST only provide maintenance information to clients that are authorized. Suppose a client queries a maintenance identifier that it is not authorized to access per Section 4.1.1.1 "Info Maintenance Item". In that case, the server SHOULD return an EPP error result code of 2201 ("Authorization error") or 2303 ("Object does not exist") [RFC5730]. The list of top-level domains or registry zones returned in the "Info Maintenance Item" response SHOULD be filtered based on the top-level domains or registry zones for which the client is authorized. Authorization of poll messages is done at the time of poll message insertion and not at the time of poll message consumption.

8. Implementation Status

Note to RFC Editor: Please remove this section and the reference to [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 [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".

8.1. GoDaddy Registry

Organization: GoDaddy Registry

Name: GoDaddy Registry

Description: GoDaddy Registry provides maintenance notifications to their registrars.

Level of maturity: Production

Coverage: All aspects of the protocol according to the draft version 2 are implemented with further updates to come.

Licensing: Proprietary

Contact: quoc@registry.godaddy

URL: <https://registry.godaddy>

8.2. TANGO Registry Services

Name: TANGO Registry Services

Description: TANGO Registry Services provides maintenance notifications to their registrars.

Level of maturity: Beta

Coverage: All aspects of the protocol according to the draft version 12 are implemented with further updates to come.

Licensing: Proprietary

Contact: Michael.Bauland@knipp.de

URL: <https://tango-rs.com>

9. References

9.1. Normative References

[W3C.REC-html5-20141028]

Hickson, I., Berjon, R., Faulkner, S., Leithead, T., Doyle Navara, E., O'Connor, E., and S. Pfeiffer, "HTML5", W3C Recommendation REC-html5-20141028, October 2014, <<https://www.w3.org/TR/2014/REC-html5-20141028/>>.

Latest version available at <<https://www.w3.org/TR/html/>>.

[W3C.REC-xml11-20060816]

Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., Yergeau, F., and J. Cowan, "Extensible Markup Language (XML) 1.1 (Second Edition)", World Wide Web Consortium Recommendation REC-xml11-20060816, 16 August 2006, <<https://www.w3.org/TR/2006/REC-xml11-20060816>>.

Latest version available at
<<https://www.w3.org/TR/xml11/>>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<https://www.rfc-editor.org/info/rfc5730>>.
- [RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <<https://www.rfc-editor.org/info/rfc5891>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC9038] Gould, J. and M. Casanova, "Extensible Provisioning Protocol (EPP) Unhandled Namespaces", RFC 9038, DOI 10.17487/RFC9038, May 2021, <<https://www.rfc-editor.org/info/rfc9038>>.

9.2. Informative References

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <<https://www.rfc-editor.org/info/rfc7451>>.

Appendix A. Change History

A.1. Change from draft-sattler-epp-poll-maintenance-response to draft-sattler-epp-registry-maintenance

Updated to be EPP based instead of JSON document.

A.2. Change from draft-sattler-epp-registry-maintenance to draft-ietf-regext-epp-registry-maintenance

Adopted by the REGEXT working group.

A.3. Change from 00 to 01

Clarified maint:description and maint:environment. Changed maint:description from complexType to simpleType. Fixed typo. Added acknowledgment.

A.4. Change from 01 to 02

Update language from Domain Name Registry to Registry. Clarified XML namespace urn:ietf:params:xml:ns:maintenance-1.0. Changed host to contain hostName and hostAddr. Changed maint:tlds from MUST to SHOULD. Fixed maint:status in Schema. Changed UUID to a server unique id.

A.5. Change from 02 to 03

Changed maint:connection from MUST to SHOULD.

A.6. Change from 03 to 04

A lot of clarifications and editorial changes.

A.7. Change from 04 to 05

Changed XML namespace from urn:ietf:params:xml:ns:maintenance-1.0 to urn:ietf:params:xml:ns:epp:maintenance-0.1. Removed <maint:status>. Clarified <maint:info> for retrieving maintenance items and the list.

A.8. Change from 05 to 06

Changed dates in examples to more recent dates. Renamed Query Maintenance Item and List to Info Maintenance Item and List. Removed blackout in favor of full. Added GoDaddy Registry implementation.

A.9. Change from 06 to 07

Removed IP addresses for <maint:host>. Editorial changes.

A.10. Change from 07 to 08

Editorial changes. Changed XML namespace and schema from 0.1 to 0.2. Added pollType to reflect create, update, or delete maintenance poll messages.

A.11. Change from 08 to 09

Editorial changes. Added new section "Migrating to Newer Versions of This Extension".

A.12. Change from 09 to 10

Editorial changes. Renamed "msg" to "name". Added "courtesy" and "end" to pollType.

A.13. Change from 10 to 11

Editorial changes. Added mime type to description.

A.14. Change from 11 to 12

Editorial changes. Changed XML namespace from 0.2 to 0.3.

A.15. Change from 12 to 13

Editorial changes. Added TANGO Registry Services to Section 8. Added Michael Bauland to acknowledgments. Added "none" to <maint:impact>.

A.16. Change from 13 to 14

Accepted in WGLC. Changed XML namespace from 0.3 to 1.0.

A.17. Change from 14 to 15

Editorial changes, added feedback from the document shepherd.

A.18. Change from 15 to 16

Editorial changes, added feedback from area director.

A.19. Change from 16 to 17

Editorial changes, added last call feedback. Changed schema URI to urn:ietf:params:xml:schema:epp:maintenance-1.0. Changed dates in examples to more recent dates.

A.20. Change from 17 to 18

Editorial changes.

A.21. Change from 18 to 19

Editorial changes.

Acknowledgments

The authors wish to thank the following persons for their feedback and suggestions: James Gould, Michael Bauland, Patrick Mevzek, Quoc-Anh Pham, Raymond Zylstra, Christopher Martens, Anthony Eden, Neal McPherson, Craig Marchant, and Andreas Huber.

Authors' Addresses

Tobias Sattler

Email: mail@tobiassattler.com

URI: <https://tobiassattler.com>

Roger Carney

GoDaddy Inc.

14455 N. Hayden Rd. #219

Scottsdale, AZ 85260

US

Email: rcarney@godaddy.com

URI: <https://www.godaddy.com>

Jody Kolker

GoDaddy Inc.

14455 N. Hayden Rd. #219

Scottsdale, AZ 85260

US

Email: jkolker@godaddy.com

URI: <https://www.godaddy.com>

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: 3 November 2022

M. Loffredo
IIT-CNR/Registro.it
G. Brown
CentralNic Group plc
2 May 2022

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

B.15. Change from 09 to 10	23
B.16. Change from 10 to 11	23
B.17. Change from 11 to 12	23
Authors' Addresses	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]:

- * "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).

```
...
"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",
        "type" : "name",
        "value" : "1 Street"
      }, {
        "@type" : "StreetComponent",
        "type" : "postOfficeBox",
        "value" : "01001"
      } ],
      "locality" : "Kyiv",
      "countryCode" : "UA"
    }
  },
  "localizations" : {
    "ua" : {
      "/jscard_0/addresses/addr" : {
        "@type" : "Address",
        "street" : [ {
```

```

        "@type" : "StreetComponent",
        "type" : "name",
        "value" : "1, "
    }, {
        "@type" : "StreetComponent",
        "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 "jscard_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 "jscard_0" tag if JSCard is returned.

From this stage on, the RDAP server MUST include the "jscard_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"  
    }  
  ]  
}
```


Figure 3: jCard sunset - JSCard not requested

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

Figure 4: jCard sunset - JSCard requested

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 "jscard" to 1/true/yes. The rdapConformance array contains the "jscard_0" tag unless jCard is returned. The "jscard" 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 "jscard" set to 1/true/yes (Figure 5); otherwise, a link to the result view identified by the current query string without the parameter "jscard" (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]
Stepanek, R. and M. Loffredo, "JSContact: A JSON representation of contact data", Work in Progress, Internet-Draft, draft-ietf-calext-jscontact-00, 17 January 2020, <<https://www.ietf.org/archive/id/draft-ietf-calext-jscontact-00.txt>>.

[I-D.ietf-calext-jscontact-vcard]
Loffredo, M. and R. Stepanek, "JSContact: Converting from and to vCard", Work in Progress, Internet-Draft, draft-ietf-calext-jscontact-vcard-00, 7 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-calext-jscontact-vcard-00.txt>>.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.17487/RFC5733, August 2009, <<https://www.rfc-editor.org/info/rfc5733>>.
- [RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <<https://www.rfc-editor.org/info/rfc6350>>.
- [RFC7095] Kewisch, P., "jCard: The JSON Format for vCard", RFC 7095, DOI 10.17487/RFC7095, January 2014, <<https://www.rfc-editor.org/info/rfc7095>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <<https://www.rfc-editor.org/info/rfc8288>>.
- [RFC8605] Hollenbeck, S. and R. Carney, "vCard Format Extensions: ICANN Extensions for the Registration Data Access Protocol (RDAP)", RFC 8605, DOI 10.17487/RFC8605, May 2019, <<https://www.rfc-editor.org/info/rfc8605>>.
- [RFC9082] Hollenbeck, S. and A. Newton, "Registration Data Access Protocol (RDAP) Query Format", STD 95, RFC 9082, DOI 10.17487/RFC9082, June 2021, <<https://www.rfc-editor.org/info/rfc9082>>.
- [RFC9083] Hollenbeck, S. and A. Newton, "JSON Responses for the Registration Data Access Protocol (RDAP)", STD 95, RFC 9083, DOI 10.17487/RFC9083, June 2021, <<https://www.rfc-editor.org/info/rfc9083>>.

9.2. Informative References

[API-DEPRECATION]

Sandoval, K., "How to Smartly Sunset and Deprecate APIs", August 2019, <<https://web.archive.org/web/20200417084255/https://nordicapis.com/how-to-smartly-sunset-and-deprecate-apis/>>.

[I-D.ietf-jsonpath-base]

Gössner, S., Normington, G., and C. Bormann, "JSONPath: Query expressions for JSON", Work in Progress, Internet-Draft, draft-ietf-jsonpath-base-03, 16 January 2022, <<https://www.ietf.org/archive/id/draft-ietf-jsonpath-base-03.txt>>.

[I-D.ietf-regex-rdap-redacted]

Gould, J., Smith, D., Kolker, J., and R. Carney, "Redacted Fields in the Registration Data Access Protocol (RDAP) Response", Work in Progress, Internet-Draft, draft-ietf-regex-rdap-redacted-02, 18 November 2021, <<https://www.ietf.org/archive/id/draft-ietf-regex-rdap-redacted-02.txt>>.

[RDAP-PILOT-WG]

ICANN RDAP Pilot WG, "RDAP Pilot Report", April 2019, <<https://www.icann.org/en/system/files/files/rdap-pilot-report-25apr19-en.pdf>>.

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.adr.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.adr.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.adr.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.
 3. Changed the reference draft-loffredo-calext-jscontact-vcard to draft-ietf-calext-jscontact-vcard.
 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
1. Ported from draft-loffredo-regext-rdap-jcard-deprecation-04 renamed to draft-ietf-regext-rdap-jscontact-00.
- 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 "jscard" to "jscard_0"
 6. Added clarifications addressing the feedback provided by Jasdeep 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-calex-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 "jscard" property to "jscard_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>

REGEXT Working Group
Internet-Draft
Intended status: Standards Track
Expires: 24 September 2022

S. Hollenbeck
Verisign Labs
23 March 2022

Federated Authentication for the Registration Data Access Protocol
(RDAP) using OpenID Connect
draft-ietf-regext-rdap-openid-12

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 24 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 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 Purpose	9
3.1.4.2. Do Not Track	10
4. Protocol Parameters	10
4.1. Data Structures	11
4.1.1. Session	11
4.1.2. Device Info	12
4.2. Client Login	13
4.2.1. Clients with Limited User Interfaces	15
4.2.1.1. UI-constrained Client Login	15
4.2.1.2. UI-constrained Client Login Polling	17
4.3. Session Status	17
4.4. Session Refresh	18
4.5. Client Logout	20
4.6. Parameter Processing	21
5. Token Exchange	21
6. RDAP Query Processing	21
7. RDAP Conformance	22
8. IANA Considerations	22
8.1. RDAP Extensions Registry	22
8.2. JSON Web Token Claims Registry	23
8.3. RDAP Query Purpose Registry	23

9. Implementation Status	26
9.1. Editor Implementation	27
9.2. Verisign Labs	27
9.3. Viagenie	28
10. Security Considerations	28
10.1. Authentication and Access Control	29
11. Acknowledgments	29
12. References	29
12.1. Normative References	29
12.2. Informative References	31
Appendix A. Change Log	32
Author's Address	32

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 of OpenID Authorization Server that's used by the RDAP server. This information is returned in the `rdapConformance` section of the response. A value of `"rdap_openidc_local_level_0"` indicates that the server uses a local Authorization Server. A value of `"rdap_openidc_remote_level_0"` indicates that the server uses a remote Authorization Server.
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.
6. The Authorization Server obtains End-User consent/authorization.
7. The Authorization Server sends the RDAP Client back to the RDAP server with an Authorization Code using an HTTP redirect.
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 MUST possess an identifier (an OpenID) issued by an OP to use OpenID Connect with RDAP. 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. 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 RP MAY use any provider discovery approach that is suitable for its operating environment.

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. 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 Purpose

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 "purpose" claim.

The "purpose" claim identifies the purpose for which access to a protected resource is being requested. Use of the "purpose" claim is OPTIONAL; processing of this claim is subject to server acceptance of the purpose 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 "purpose" value is a case-sensitive string containing a StringOrURI value as specified in Section 2 of the JSON Web Token (JWT) specification ([RFC7519]). An example:

```
{"purpose" : "domainNameControl"}
```

Purpose values are themselves 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" ("dnt") 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 "dnt" 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 "dnt" claim is present, the value of the claim is "true", and accepting the claim complies with local regulations regarding logging and tracking.

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

```
{"dnt" : 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 and the information needed to establish a session for a UI-constrained device.

2. A query parameter to request authentication for a specific End-User identity.
3. 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 two new data structures that are used to return information to a client: a "session" data structure that contains information that describes an established session, and a "deviceInfo" data structure that contains information that describes an active attempt to establish a session on a UI-constrained device.

4.1.1. Session

The "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 "session" data structure:

```
"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",
    "purpose": "domainNameControl",
    "dnt": 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 "deviceInfo" data structure.

The "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 "deviceInfo" data structure:


```
"deviceInfo": {  
  "verification_url": "https://www.example.com/device",  
  "user_code": "NJJQ-GJFC",  
  "expires_in": "1800"  
}
```

Figure 2

4.2. Client Login

Client authentication is requested by sending a "session/login" request to an RDAP server. If the RDAP server supports only remote Authorization Servers, the "session/login" 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]. Clients can use either of these methods to deliver the End-User identifier to a server that supports remote Authorization Servers. Servers that support remote Authorization Servers MUST accept both methods. If the RDAP server supports a local Authorization Server, 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 "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/session/login?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/session/login
```

```
Authorization: Basic dXNlci5pZHAuZXhhbXBsZQ==
```

An example for use with a local Authorization Server:

```
https://example.com/rdap/session/login
```

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 "session" data structure.

An example of a successful "session/login" response:

```
{
  "rdapConformance": [
    "rdap_openidc_remote_level_0"
  ],
  "lang": "en-US",
  "notices": {
    "title": "Login Result",
    "description": [
      "Login succeeded",
      "user.idp.example"
    ],
  },
  "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",
      "purpose": "domainNameControl",
      "dnt": false
    },
    "sessionInfo": {
      "tokenExpiration": 3599,
      "tokenRefresh": true
    }
  }
}
```

Figure 3

An example of a failed "session/login" response:

```
{
  "rdapConformance": [
    "rdap_openidc_remote_level_0"
  ],
  "lang": "en-US",
  "notices": {
    "title": "Login Result",
    "description": [
      "Login failed",
      "user.idp.example"
    ]
  }
}
```

Figure 4

4.2.1. 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.

4.2.1.1. UI-constrained Client Login

Client authentication is requested by sending a "session/device" request to an RDAP server. If the RDAP server supports only remote Authorization Servers, the "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 local 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?id=user.idp.example
```

Figure 5

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 6

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 "deviceInfo" data structure.

An example of a "session/device" response:

```
{  
  "rdapConformance": [  
    "rdap_openidc_remote_level_0"  
  ],  
  "lang": "en-US",  
  "notices": {  
    "title": "Device Login Result",  
    "description": [  
      "Login succeeded",  
      "user.idp.example"  
    ]  
  },  
  "deviceInfo": {  
    "verification_url": "https://www.example.com/device",  
    "user_code": "NJJQ-GJFC",  
    "expires_in": 1800  
  }  
}
```

Figure 7

4.2.1.2. UI-constrained Client Login Polling

After successful processing of the "session/device" request, the client MUST send a "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 "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/session/devicepoll
```

Figure 8

An example using curl:

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

Figure 9

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. Session Status

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

```
https://example.com/rdap/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 "session" data structure.

An example of a "session/status" response:

```
{
  "rdapConformance": [
    "rdap_openidc_remote_level_0"
  ],
  "lang": "en-US",
  "notices": {
    "title": "Session Status Result",
    "description": [
      "Session status succeeded",
      "user.idp.example"
    ]
  },
  "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",
      "purpose": "domainNameControl",
      "dnt": false
    },
    "sessionInfo": {
      "tokenExpiration": 3490,
      "tokenRefresh": true
    }
  }
}
```

Figure 10

4.4. Session Refresh

Clients MAY send a request to an RDAP server to refresh, or extend, an existing login session using a "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 "session/refresh" request:

`https://example.com/rdap/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 "session" data structure.

An example of a "session/refresh" response:

```
{
  "rdapConformance": [
    "rdap_openidc_remote_level_0"
  ],
  "lang": "en-US",
  "notices": {
    "title": "Session Refresh Result",
    "description": [
      "Session refresh succeeded",
      "user.idp.example",
      "Token refresh succeeded."
    ]
  },
  "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",
      "purpose": "domainNameControl",
      "dnt": false
    },
    "sessionInfo": {
      "tokenExpiration": 3599,
      "tokenRefresh": true
    }
  }
}
```

Figure 11

4.5. Client Logout

Clients MAY send a request to an RDAP server to terminate an existing login session. Termination of a session is requested using a "session/logout" path segment. Access and refresh tokens can be revoked during the "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 "session/logout" request:

`https://example.com/rdap/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 "session/logout" response:

```
{
  "rdapConformance": [
    "rdap_openidc_remote_level_0"
  ],
  "lang": "en-US",
  "notices": {
    "title": "Logout Result",
    "description": [
      "Logout succeeded",
      "user.idp.example",
      "Provider logout failed: Not supported by provider.",
      "Token revocation successful."
    ]
  }
}
```

Figure 12

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.

4.6. 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 "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 "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 `"rdap_openidc_remote_level_0"` (to indicate support for one or more remote Authorization Servers), `"rdap_openidc_local_level_0"` (to indicate support for a local Authorization Server), or both values if the server supports both remote and local OpenID Authorization Servers. The information needed to register these values in the RDAP Extensions Registry is described in Section 8.1.

Example `rdapConformance` structure with extension specified:

```
"rdapConformance" :  
  [  
    "rdap_level_0",  
    "rdap_openidc_remote_level_0"  
  ]
```

Figure 13

8. IANA Considerations

8.1. RDAP Extensions Registry

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

Extension identifier: `rdap_openidc_remote_level_0`
Registry operator: Any
Published specification: This document.
Contact: IESG <iesg@ietf.org>
Intended usage: This extension describes a federated authentication method for RDAP using OAuth 2.0, OpenID Connect, and a remote Authorization Server.

Extension identifier: `rdap_openidc_local_level_0`
Registry operator: Any
Published specification: This document.
Contact: IESG <iesg@ietf.org>

Intended usage: This extension describes a federated authentication method for RDAP using OAuth 2.0, OpenID Connect, and a local Authorization Server.

8.2. JSON Web Token Claims Registry

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

Claim Name: "purpose"

Claim Description: This claim describes the stated purpose for submitting a request to access a protected RDAP resource.

Change Controller: IESG

Specification Document(s): Section 3.1.4.1 of this document.

Claim Name: "dnt"

Claim Description: This claim contains a JSON boolean literal that describes an End-User's "do not track" preference for identity tracking, logging, or recording when accessing 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.

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

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

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.

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

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

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 web site functional issues.

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

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

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.

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

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

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.

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

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

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: academicPublicInterestDNSRRResearch

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.

Mario Loffredo provided significant feedback based on implementation experience that led to welcome improvements in several sections of this document. His contributions are greatly appreciated.

12. References

12.1. Normative References

- [OIDC] OpenID Foundation, "OpenID Connect",
<<http://openid.net/connect/>>.
- [OIDCC] OpenID Foundation, "OpenID Connect Core incorporating
errata set 1", November 2014,
<http://openid.net/specs/openid-connect-core-1_0.html>.
- [OIDCD] OpenID Foundation, "OpenID Connect Discovery 1.0
incorporating errata set 1", November 2014,
<http://openid.net/specs/openid-connect-discovery-1_0.html>.
- [OIDCL] OpenID Foundation, "OpenID Connect RP-Initiated Logout 1.0
- draft 01", August 2020, <https://openid.net/specs/openid-connect-rpinitiated-1_0.html>.

- [OIDCR] OpenID Foundation, "OpenID Connect Dynamic Client Registration 1.0 incorporating errata set 1", November 2014, <http://openid.net/specs/openid-connect-registration-1_0.html>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.
- [RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <<https://www.rfc-editor.org/info/rfc6265>>.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <<https://www.rfc-editor.org/info/rfc6749>>.
- [RFC7009] Lodderstedt, T., Ed., Dronia, S., and M. Scurtescu, "OAuth 2.0 Token Revocation", RFC 7009, DOI 10.17487/RFC7009, August 2013, <<https://www.rfc-editor.org/info/rfc7009>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", STD 95, RFC 7480, DOI 10.17487/RFC7480, March 2015, <<https://www.rfc-editor.org/info/rfc7480>>.
- [RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", STD 95, RFC 7481, DOI 10.17487/RFC7481, March 2015, <<https://www.rfc-editor.org/info/rfc7481>>.

- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <<https://www.rfc-editor.org/info/rfc7519>>.
- [RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <<https://www.rfc-editor.org/info/rfc7617>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8628] Denniss, W., Bradley, J., Jones, M., and H. Tschofenig, "OAuth 2.0 Device Authorization Grant", RFC 8628, DOI 10.17487/RFC8628, August 2019, <<https://www.rfc-editor.org/info/rfc8628>>.
- [RFC8693] Jones, M., Nadalin, A., Campbell, B., Ed., Bradley, J., and C. Mortimore, "OAuth 2.0 Token Exchange", RFC 8693, DOI 10.17487/RFC8693, January 2020, <<https://www.rfc-editor.org/info/rfc8693>>.
- [RFC9082] Hollenbeck, S. and A. Newton, "Registration Data Access Protocol (RDAP) Query Format", STD 95, RFC 9082, DOI 10.17487/RFC9082, June 2021, <<https://www.rfc-editor.org/info/rfc9082>>.
- [RFC9083] Hollenbeck, S. and A. Newton, "JSON Responses for the Registration Data Access Protocol (RDAP)", STD 95, RFC 9083, DOI 10.17487/RFC9083, June 2021, <<https://www.rfc-editor.org/info/rfc9083>>.

12.2. Informative References

- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <<https://www.rfc-editor.org/info/rfc4949>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <<https://www.rfc-editor.org/info/rfc8414>>.

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 local 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.

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
Internet-Draft
Intended status: Standards Track
Expires: 3 November 2022

M. Loffredo
M. Martinelli
IIT-CNR/Registro.it
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.

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	2
1.1. Conventions Used in This Document	4
2. RDAP Path Segment Specification	4
3. RDAP Response Specification	5
4. Reverse Searches Based on Entity Details	5
5. RDAP Conformance	6
6. Implementation Considerations	6
7. Implementation Status	7
7.1. IIT-CNR/Registro.it RDAP Server	7
7.2. IIT-CNR/Registro.it RDAP Client	7
8. IANA Considerations	8
9. Privacy Considerations	8
10. Security Considerations	9
11. Acknowledgements	9
12. References	9
12.1. Normative References	9
12.2. Informative References	10
Appendix A. Paradigms to Enforce Access Control on Reverse Search in RDAP	11
Appendix B. Change Log	12
Authors' Addresses	13

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

- [OIDCC] OpenID Foundation, "OpenID Connect Core incorporating errata set 1", November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, DOI 10.17487/RFC3912, September 2004, <<https://www.rfc-editor.org/info/rfc3912>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<https://www.rfc-editor.org/info/rfc5730>>.

- [RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <<https://www.rfc-editor.org/info/rfc6350>>.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/info/rfc6973>>.
- [RFC7095] Kewisch, P., "jCard: The JSON Format for vCard", RFC 7095, DOI 10.17487/RFC7095, January 2014, <<https://www.rfc-editor.org/info/rfc7095>>.
- [RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", STD 95, RFC 7480, DOI 10.17487/RFC7480, March 2015, <<https://www.rfc-editor.org/info/rfc7480>>.
- [RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", STD 95, RFC 7481, DOI 10.17487/RFC7481, March 2015, <<https://www.rfc-editor.org/info/rfc7481>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC9082] Hollenbeck, S. and A. Newton, "Registration Data Access Protocol (RDAP) Query Format", STD 95, RFC 9082, DOI 10.17487/RFC9082, June 2021, <<https://www.rfc-editor.org/info/rfc9082>>.
- [RFC9083] Hollenbeck, S. and A. Newton, "JSON Responses for the Registration Data Access Protocol (RDAP)", STD 95, RFC 9083, DOI 10.17487/RFC9083, June 2021, <<https://www.rfc-editor.org/info/rfc9083>>.

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-

Draft, draft-ietf-jsonpath-base-03, 16 January 2022,
<<https://www.ietf.org/archive/id/draft-ietf-jsonpath-base-03.txt>>.

[I-D.ietf-regext-rdap-openid]

Hollenbeck, S., "Federated Authentication for the Registration Data Access Protocol (RDAP) using OpenID Connect", Work in Progress, Internet-Draft, draft-ietf-regext-rdap-openid-08, 8 November 2021,
<<https://www.ietf.org/archive/id/draft-ietf-regext-rdap-openid-08.txt>>.

[ICANN-RA] Internet Corporation For Assigned Names and Numbers, "Registry Agreement", July 2017,
<<https://newgtlds.icann.org/sites/default/files/agreements/agreement-approved-31jul17-en.pdf>>.

[ICANN-RDS1]

Internet Corporation For Assigned Names and Numbers, "Final Report from the Expert Working Group on gTLD Directory Services: A Next-Generation Registration Directory Service (RDS)", June 2014,
<<https://www.icann.org/en/system/files/files/final-report-06jun14-en.pdf>>.

[ICANN-RDS2]

Internet Corporation For Assigned Names and Numbers, "Final Issue Report on a Next-Generation gTLD RDS to Replace WHOIS", October 2015,
<<http://whois.icann.org/sites/default/files/files/final-issue-report-next-generation-rds-07oct15-en.pdf>>.

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>

Network Working Group
Internet-Draft
Obsoletes: 7484 (if approved)
Intended status: Standards Track
Expires: 1 August 2022

M. Blanchet
Viagenie
28 January 2022

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.

Table of Contents

1. Introduction	2
2. Conventions Used in This Document	3
3. Structure of the RDAP Bootstrap Service Registries	3
4. Bootstrap Service Registry for Domain Name Space	5
5. Bootstrap Service Registries for Internet Numbers	6
5.1. Bootstrap Service Registry for IPv4 Address Space	6
5.2. Bootstrap Service Registry for IPv6 Address Space	7
5.3. Bootstrap Service Registry for AS Number Space	9
6. Entity	10
7. Non-existent Entries or RDAP URL Values	10
8. Deployment and Implementation Considerations	10
9. Limitations	11
10. Formal Definition	11
10.1. Imported JSON Terms	11
10.2. Registry Syntax	11
11. Security Considerations	12
12. Implementation Status	12
12.1. RDAP Browser Mobile Application	13
12.2. ICANN Lookup Web Application	13
12.3. ARIN Implementation	14
13. IANA Considerations	14
13.1. Bootstrap Service Registry for IPv4 Address Space	16
13.2. Bootstrap Service Registry for IPv6 Address Space	16
13.3. Bootstrap Service Registry for AS Number Space	16
13.4. Bootstrap Service Registry for Domain Name Space	16
14. References	16
14.1. Normative References	16
14.2. Informative References	17
Acknowledgements	19
Changes since RFC7484	19
Author's Address	19

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:

```
{
  "version": "1.0",
  "publication": "YYYY-MM-DDTHH:MM:SSZ",
  "description": "Some text",
  "services": [
    [
      ["entry1", "entry2", "entry3"],
      [
        "https://registry.example.com/myrdap/",
        "http://registry.example.com/myrdap/"
      ]
    ],
    [
      ["entry4"],
      [
        "https://example.org/"
      ]
    ]
  ]
}
```

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.


```
{
  "version": "1.0",
  "publication": "2024-01-07T10:11:12Z",
  "description": "RDAP Bootstrap file for example registries.",
  "services": [
    [
      ["2001:db8::/34"],
      [
        "https://rir2.example.com/myrdap/"
      ]
    ],
    [
      ["2001:db8:4000::/36", "2001:db8:ffff::/48"],
      [
        "https://example.org/"
      ]
    ],
    [
      ["2001:db8:1000::/36"],
      [
        "https://example.net/rdaprir2/",
        "http://example.net/rdaprir2/"
      ]
    ]
  ]
}
```

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/rdaprir2/" 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/rdaprir2/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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <<https://www.rfc-editor.org/info/rfc4632>>.
- [RFC5396] Huston, G. and G. Michaelson, "Textual Representation of Autonomous System (AS) Numbers", RFC 5396, DOI 10.17487/RFC5396, December 2008, <<https://www.rfc-editor.org/info/rfc5396>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<https://www.rfc-editor.org/info/rfc5890>>.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <<https://www.rfc-editor.org/info/rfc5952>>.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.
- [RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", STD 95, RFC 7480, DOI 10.17487/RFC7480, March 2015, <<https://www.rfc-editor.org/info/rfc7480>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.

14.2. Informative References

- [asreg] IANA, "Autonomous System (AS) Numbers", <<https://www.iana.org/assignments/as-numbers>>.

- [domainreg] IANA, "Root Zone Database",
<<https://www.iana.org/domains/root/db>>.
- [ipv4reg] IANA, "IPv4 Address Space Registry",
<<https://www.iana.org/assignments/ipv4-address-space>>.
- [ipv6reg] IANA, "IPv6 Global Unicast Address Assignments",
<<https://www.iana.org/assignments/ipv6-unicast-address-assignments>>.
- [REDIRECT-RDAP] Martinez, C., Zhou, L., and G. Rada, "Redirection Service for Registration Data Access Protocol", Work in Progress, draft-ietf-weirds-redirects-04, July 2014.
- [RFC7071] Borenstein, N. and M. Kucherawy, "A Media Type for Reputation Interchange", RFC 7071, DOI 10.17487/RFC7071, November 2013, <<https://www.rfc-editor.org/info/rfc7071>>.
- [RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <<https://www.rfc-editor.org/info/rfc7234>>.
- [RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", STD 95, RFC 7481, DOI 10.17487/RFC7481, March 2015, <<https://www.rfc-editor.org/info/rfc7481>>.
- [RFC7484] Blanchet, M., "Finding the Authoritative Registration Data (RDAP) Service", RFC 7484, DOI 10.17487/RFC7484, March 2015, <<https://www.rfc-editor.org/info/rfc7484>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8521] Hollenbeck, S. and A. Newton, "Registration Data Access Protocol (RDAP) Object Tagging", BCP 221, RFC 8521, DOI 10.17487/RFC8521, November 2018, <<https://www.rfc-editor.org/info/rfc8521>>.

- [RFC9082] Hollenbeck, S. and A. Newton, "Registration Data Access Protocol (RDAP) Query Format", STD 95, RFC 9082, DOI 10.17487/RFC9082, June 2021, <<https://www.rfc-editor.org/info/rfc9082>>.
- [RFC9083] Hollenbeck, S. and A. Newton, "JSON Responses for the Registration Data Access Protocol (RDAP)", STD 95, RFC 9083, DOI 10.17487/RFC9083, June 2021, <<https://www.rfc-editor.org/info/rfc9083>>.

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 Moriaty, 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, Jasdip 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>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 30 December 2021

J. Gould
R. Wilhelm
VeriSign, Inc.
28 June 2021

Extensible Provisioning Protocol (EPP) Secure Authorization Information
for Transfer
draft-ietf-regext-secure-authinfo-transfer-07

Abstract

The Extensible Provisioning Protocol (EPP), in RFC 5730, defines the use of authorization information to authorize a transfer of an EPP object, such as a domain name, between clients that are referred to as registrars. Object-specific, password-based authorization information (see RFC 5731 and RFC 5733) is commonly used, but raises issues related to the security, complexity, storage, and lifetime of authentication information. This document defines an operational practice, using the EPP RFCs, that leverages the use of strong random authorization information values that are short-lived, not stored by the client, and stored by the server using a cryptographic hash that provides for secure authorization information that can safely be used for object transfers.

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 30 December 2021.

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 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 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.

Table of Contents

1. Introduction	3
1.1. Conventions Used in This Document	4
2. Registrant, Registrar, Registry	5
3. Signaling Client and Server Support	6
4. Secure Authorization Information	7
4.1. Secure Random Authorization Information	7
4.2. Authorization Information Time-To-Live (TTL)	8
4.3. Authorization Information Storage and Transport	8
4.4. Authorization Information Matching	9
5. Create, Transfer, and Secure Authorization Information	10
5.1. Create Command	10
5.2. Update Command	12
5.3. Info Command and Response	15
5.4. Transfer Request Command	17
6. Transition Considerations	18
6.1. Transition Phase 1 - Features	20
6.2. Transition Phase 2 - Storage	21
6.3. Transition Phase 3 - Enforcement	21
7. IANA Considerations	21
7.1. XML Namespace	21
7.2. EPP Extension Registry	22
8. Implementation Status	22
8.1. Verisign EPP SDK	23
8.2. RegistryEngine EPP Service	23
9. Security Considerations	24
10. Acknowledgements	24
11. References	24
11.1. Normative References	24
11.2. Informative References	25
Appendix A. Change History	26
A.1. Change from 00 to 01	26
A.2. Change from 01 to 02	26
A.3. Change from 02 to 03	26
A.4. Change from 03 to REGEXT 00	28
A.5. Change from REGEXT 00 to REGEXT 01	28
A.6. Change from REGEXT 01 to REGEXT 02	28
A.7. Change from REGEXT 02 to REGEXT 03	28

A.8. Change from REGEXT 03 to REGEXT 04	28
A.9. Change from REGEXT 04 to REGEXT 05	28
A.10. Change from REGEXT 05 to REGEXT 06	29
A.11. Change from REGEXT 06 to REGEXT 07	29
Authors' Addresses	31

1. Introduction

The Extensible Provisioning Protocol (EPP), in [RFC5730], defines the use of authorization information to authorize a transfer of an EPP object, such as a domain name, between clients that are referred to as registrars. The authorization information is object-specific and has been defined in the EPP Domain Name Mapping, in [RFC5731], and the EPP Contact Mapping, in [RFC5733], as password-based authorization information. Other authorization mechanisms can be used, but in practice the password-based authorization information has been used at the time of object create, managed with the object update, and used to authorize an object transfer request. What has not been considered is the security of the authorization information that includes the complexity of the authorization information, the time-to-live (TTL) of the authorization information, and where and how the authorization information is stored.

The current/original lifecycle for authorization information involves long-term storage of encrypted (not hashed) passwords, which presents a significant latent risk of password compromise and is not consistent with current best practices. The mechanisms in this document provide a way to avoid long-term password storage entirely, and to only require the storage of hashed (not retrievable) passwords instead of encrypted passwords.

This document defines an operational practice, using the EPP RFCs, that leverages the use of strong, random authorization information values that are short-lived, that are not stored by the client, and that are stored by the server using a cryptographic hash to provide secure authorization information used for transfers. This operational practice can be used to support transfers of any EPP object, where the domain name object defined in [RFC5731] is used in this document for illustration purposes. Elements of the practice may be used to support the secure use of the authorization information for purposes other than transfer, but any other purposes and the applicable elements are out-of-scope for this document.

The overall goal is to have strong, random authorization information values, that are short-lived, and that are either not stored or stored as a cryptographic hash values by the non-responsible parties. In a registrant, registrar, and registry model, the registrant registers the object through the registrar to the registry. The

registrant is the responsible party and the registrar and the registry are the non-responsible parties. EPP is a protocol between the registrar and the registry, where the registrar is referred to as the client and the registry is referred to as the server. The following are the elements of the operational practice and how the existing features of the EPP RFCs can be leveraged to satisfy them:

- "Strong Random Authorization Information": The EPP RFCs define the password-based authorization information value using an XML schema "normalizedString" type, so they don't restrict what can be used in any substantial way. This operational practice defines the recommended mechanism for creating a strong random authorization value, that would be generated by the client.
- "Short-Lived Authorization Information": The EPP RFCs don't explicitly support short-lived authorization information or a time-to-live (TTL) for authorization information, but there are EPP RFC features that can be leveraged to support short-lived authorization information. All of these features are compatible with the EPP RFCs, though not mandatory to implement. In section 2.6 of [RFC5731] it states that authorization information is assigned when a domain object is created, which results in long-lived authorization information. This specification changes the nature of the authorization information to be short-lived. If authorization information is set only when there is a transfer in process, the server needs to support an empty authorization information value on create, support setting and unsetting authorization information, and support automatically unsetting the authorization information upon a successful transfer. All of these features can be supported by the EPP RFCs.
- "Storing Authorization Information Securely": The EPP RFCs don't specify where and how the authorization information is stored in the client or the server, so there are no restrictions to define an operational practice for storing the authorization information securely. The operational practice will require the client to not store the authorization information and will require the server to store the authorization information using a cryptographic hash, with at least a 256-bit hash function, such as SHA-256 [FIPS-180-4], and with a per-authorization information random salt, with at least 128 bits. Returning the authorization information set in an EPP info response will not be supported.

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.

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.

In examples, "C:" represents lines sent by a protocol client and "S:" represents lines returned by a protocol server. Indentation and white space in examples are provided only to illustrate element relationships and are not a required feature of this protocol.

The examples reference XML namespace prefixes that are used for the associated XML namespaces. Implementations MUST NOT depend on the example XML namespaces and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents. The example namespace prefixes used and their associated XML namespaces include:

"domain": urn:ietf:params:xml:ns:domain-1.0
"contact": urn:ietf:params:xml:ns:contact-1.0

2. Registrant, Registrar, Registry

The EPP RFCs refer to client and server, but when it comes to transfers, there are three types of actors that are involved. This document will refer to the actors as registrant, registrar, and registry. [RFC8499] defines these terms formally for the Domain Name System (DNS). The terms are further described below to cover their roles as actors of using the authorization information in the transfer process of any object in the registry, such as a domain name or a contact:

"registrant": [RFC8499] defines the registrant as "an individual or organization on whose behalf a name in a zone is registered by the registry". The registrant can be the owner of any object in the registry, such as a domain name or a contact. The registrant interfaces with the registrar for provisioning the objects. A transfer is coordinated by the registrant to transfer the sponsorship of the object from one registrar to another. The authorization information is meant to authenticate the registrant as the owner of the object to the non-sponsoring registrar and to authorize the transfer.

"registrar": [RFC8499] defines the registrar as "a service provider that acts as a go-between for registrants and registries". The registrar interfaces with the registrant for the provisioning of objects, such as domain names and contacts, and with the registries to satisfy the registrant's provisioning requests. A registrar may directly interface with the registrant or may indirectly interface with the registrant, typically through one

or more resellers. Implementing a transfer using secure authorization information extends through the registrar's reseller channel up to the direct interface with the registrant. The registrar's interface with the registries uses EPP. The registrar's interface with its reseller channel or the registrant is registrar-specific. In the EPP RFCs, the registrar is referred to as the "client", since EPP is the protocol used between the registrar and the registry. The sponsoring registrar is the authorized registrar to manage objects on behalf of the registrant. A non-sponsoring registrar is not authorized to manage objects on behalf of the registrant. A transfer of an object's sponsorship is from one registrar, referred to as the losing registrar, to another registrar, referred to as the gaining registrar.

"registry": [RFC8499] defines the registry as "the administrative operation of a zone that allows registration of names within the zone". The registry typically interfaces with the registrars over EPP and generally does not interact directly with the registrant. In the EPP RFCs, the registry is referred to as the "server", since EPP is the protocol used between the registrar and the registry. The registry has a record of the sponsoring registrar for each object and provides the mechanism (over EPP) to coordinate a transfer of an object's sponsorship between registrars.

3. Signaling Client and Server Support

This document does not define new protocol but an operational practice using the existing EPP protocol, where the client and the server can signal support for the operational practice using a namespace URI in the login and greeting extension services. The namespace URI "urn:ietf:params:xml:ns:epp:secure-authinfo-transfer-1.0" is used to signal support for the operational practice. 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.

A client that receives the namespace URI in the server's Greeting extension services can expect the following supported behavior by the server:

1. Support an empty authorization information value with a create command.
2. Support unsetting authorization information with an update command.
3. Support validating authorization information with an info command.

4. Support not returning an indication whether the authorization information is set or unset to the non-sponsoring registrar.
5. Support returning an empty authorization information value to the sponsoring registrar when the authorization information is set in an info response.
6. Support allowing for the passing of a matching non-empty authorization information value to authorize a transfer.
7. Support automatically unsetting the authorization information upon a successful completion of transfer.

A server that receives the namespace URI in the client's <login> Command extension services, can expect the following supported behavior by the client:

1. Support generation of authorization information using a secure random value.
2. Support only setting the authorization information when there is a transfer in process.

4. Secure Authorization Information

The authorization information in the EPP RFCs ([RFC5731] and [RFC5733]) that support transfer use password-based authorization information ([RFC5731] with the <domain:pw> element and [RFC5733] with the <contact:pw> element). Other EPP objects that support password-based authorization information for transfer can use the Secure Authorization Information defined in this document. For the authorization information to be secure, it must be generated using a strong random value and have a short time-to-live (TTL). The security of the authorization information is defined in the following sections.

4.1. Secure Random Authorization Information

For authorization information to be secure, it MUST be generated using a secure random value. The authorization information is treated as a password, and the required length L of a password, rounded up to the largest whole number, is based on the size N of the set of characters and the desired entropy H , in the equation $L = \text{ROUNDUP}(H / \log_2 N)$. Given a target entropy, the required length can be calculated after deciding on the set of characters that will be randomized. In accordance with current best practices and noting that the authorization information is a machine-generated value, the implementation SHOULD use at least 128 bits of entropy as the value of H . The lengths below are calculated using that value.

Calculation of the required length with 128 bits of entropy and with the set of all printable ASCII characters except space (0x20), which consists of the 94 characters 0x21-0x7E.

$\text{ROUNDUP}(128 / \log_2 94) \approx \text{ROUNDUP}(128 / 6.55) \approx \text{ROUNDUP}(19.54) = 20$

Calculation of the required length with 128 bits of entropy and with the set of case insensitive alphanumeric characters, which consists of 36 characters (a-z A-Z 0-9).

$\text{ROUNDUP}(128 / \log_2 36) \approx \text{ROUNDUP}(128 / 5.17) \approx \text{ROUNDUP}(24.76) = 25$

The strength of the random authorization information is dependent on the random number generator. Suitably strong random number generators are available in a wide variety of implementation environments, including the interfaces listed in Sections 7.1.2 and 7.1.3 of [RFC4086]. In environments that do not provide interfaces to strong random number generators, the practices defined in [RFC4086] and section 4.7.1 of the NIST Federal Information Processing Standards (FIPS) Publication 140-2 [FIPS-140-2] can be followed to produce random values that will be resistant to attack.

4.2. Authorization Information Time-To-Live (TTL)

The authorization information SHOULD only be set when there is a transfer in process. This implies that the authorization information has a Time-To-Live (TTL) by which the authorization information is cleared when the TTL expires. The EPP RFCs have no definition of TTL, but since the server supports the setting and unsetting of the authorization information by the sponsoring registrar, the sponsoring registrar can apply a TTL based on client policy. The TTL client policy may be based on proprietary registrar-specific criteria, which provides for a transfer-specific TTL tuned for the particular circumstances of the transaction. The sponsoring registrar will be aware of the TTL and the sponsoring registrar MUST inform the registrant of the TTL when the authorization information is provided to the registrant.

4.3. Authorization Information Storage and Transport

To protect the disclosure of the authorization information, the following requirements apply:

1. The authorization information MUST be stored by the registry using a strong one-way cryptographic hash, with at least a 256-bit hash function, such as SHA-256 [FIPS-180-4], and with a per-authorization information random salt, with at least 128 bits.

2. Empty authorization information MUST be stored as an undefined value that is referred to as a NULL value. The representation of a NULL (undefined) value is dependent on the type of database used.
3. The authorization information MUST NOT be stored by the losing registrar.
4. The authorization information MUST only be stored by the gaining registrar as a "transient" value in support of the transfer process.
5. The plain text version of the authorization information MUST NOT be written to any logs by a registrar or the registry, nor otherwise recorded where it will persist beyond the transfer process.
6. All communication that includes the authorization information MUST be over an encrypted channel, such as defined in [RFC5734] for EPP.
7. The registrar's interface for communicating the authorization information with the registrant MUST be over an authenticated and encrypted channel.

4.4. Authorization Information Matching

To support the authorization information TTL, as defined in Section 4.2, the authorization information must have either a set or unset state. Authorization information that is unset is stored with a NULL (undefined) value. Based on the requirement to store the authorization information using a strong one-way cryptographic hash, as defined in Section 4.3, authorization information that is set is stored with a non-NULL hashed value. The empty authorization information is used as input in both the create command (Section 5.1) and the update command (Section 5.2) to define the unset state. The matching of the authorization information in the info command (Section 5.3) and the transfer request command (Section 5.4) is based on the following rules:

1. Any input authorization information value MUST NOT match an unset authorization information value. This includes empty authorization information, such as <domain:null/> or <domain:pw/> in [RFC5731], and non-empty authorization information, such as <domain:pw>2fooBAR</domain:pw> in [RFC5731].
2. An empty input authorization information value MUST NOT match any set authorization information value.
3. A non-empty input authorization information value MUST be hashed and matched against the set authorization information value, which is stored using the same hash algorithm.

5. Create, Transfer, and Secure Authorization Information

To secure the transfer process using secure authorization information, as defined in Section 4, the client and server need to implement steps where the authorization information is set only when a transfer is actively in process and ensure that the authorization information is stored securely and transported only over secure channels. The steps in management of the authorization information for transfers include:

1. Registrant requests to register the object with the registrar. Registrar sends the create command, with an empty authorization information value, to the registry, as defined in Section 5.1.
2. Registrant requests from the losing registrar the authorization information to provide to the gaining registrar.
3. Losing registrar generates a secure random authorization information value, sends it to the registry as defined in Section 5.2, and provides it to the registrant.
4. Registrant provides the authorization information value to the gaining registrar.
5. Gaining registrar optionally verifies the authorization information with the info command to the registry, as defined in Section 5.3.
6. Gaining registrar sends the transfer request with the authorization information to the registry, as defined in Section 5.4.
7. If the transfer successfully completes, the registry automatically unsets the authorization information; otherwise the losing registrar unsets the authorization information when the TTL expires, as defined in Section 5.2.

The following sections outline the practices of the EPP commands and responses between the registrar and the registry that supports secure authorization information for transfer.

5.1. Create Command

For a create command, the registry **MUST** allow for the passing of an empty authorization information value and **MAY** disallow for the passing of a non-empty authorization information value. By having an empty authorization information value on create, the object is initially not in the transfer process. Any EPP object extension that supports setting the authorization information with a "eppcom:pwAuthInfoType" element can have an empty authorization information value passed. Examples of such extensions are [RFC5731] and [RFC5733].

Example of passing an empty authorization information value in an [RFC5731] domain name create command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:authInfo>
C:          <domain:pw/>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example of passing an empty authorization information value in an [RFC5733] contact create command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <contact:create
C:        xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
C:        <contact:id>sh8013</contact:id>
C:        <contact:postalInfo type="int">
C:          <contact:name>John Doe</contact:name>
C:          <contact:addr>
C:            <contact:city>Dulles</contact:city>
C:            <contact:cc>US</contact:cc>
C:          </contact:addr>
C:        </contact:postalInfo>
C:        <contact:email>jdoe@example.com</contact:email>
C:        <contact:authInfo>
C:          <contact:pw/>
C:        </contact:authInfo>
C:      </contact:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```


5.2. Update Command

For an update command, the registry MUST allow for the setting and unsetting of the authorization information. The registrar sets the authorization information by first generating a strong, random authorization information value, based on Section 4.1, and setting it in the registry in the update command. The importance of generating strong authorization information values cannot be overstated: secure transfers are very important to the Internet to mitigate damage in the form of theft, fraud, and other abuse. It is critical that registrars only use strong, randomly generated authorization information values.

Because of this, registries may validate the randomness of the authorization information based on the length and character set required by the registry. For example, validating an authorization value contains a combination of upper-case, lower-case, and non-alphanumeric characters, in an attempt to assess the strength of the value, and return an EPP error result of 2202 if the check fails.

Such checks are, by their nature, heuristic and imperfect, and may identify well-chosen authorization information values as being not sufficiently strong. Registrars, therefore, must be prepared for an error response of 2202, "Invalid authorization information", and respond by generating a new value and trying again, possibly more than once.

Often, the registrar has the "clientTransferProhibited" status set, so to start the transfer process, the "clientTransferProhibited" status needs to be removed, and the strong, random authorization information value needs to be set. The registrar MUST define a time-to-live (TTL), as defined in Section 4.2, where if the TTL expires the registrar will unset the authorization information.

Example of removing the "clientTransferProhibited" status and setting the authorization information in an [RFC5731] domain name update command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>example.com</domain:name>
C:          <domain:rem>
C:            <domain:status s="clientTransferProhibited"/>
C:          </domain:rem>
C:          <domain:chg>
C:            <domain:authInfo>
C:              <domain:pw>LuQ7Bu@w9?%+_HK3cayg$55$LSft3MPP
C:            </domain:pw>
C:          </domain:authInfo>
C:          </domain:chg>
C:        </domain:update>
C:      </update>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C:</epp>
```

When the registrar-defined TTL expires, the sponsoring registrar MUST cancel the transfer process by unsetting the authorization information value and MAY add back statuses like the "clientTransferProhibited" status. Any EPP object extension that supports setting the authorization information with a "eppcom:pwAuthInfoType" element, can have an empty authorization information value passed. Examples of such extensions are [RFC5731] and [RFC5733]. Setting an empty authorization information value unsets the authorization information. [RFC5731] supports an explicit mechanism of unsetting the authorization information, by passing the <domain:null> authorization information value. The registry MUST support unsetting the authorization information by accepting an empty authorization information value and accepting an explicit unset element if it is supported by the object extension.

Example of adding the "clientTransferProhibited" status and unsetting the authorization information explicitly in an [RFC5731] domain name update command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>example.com</domain:name>
C:          <domain:add>
C:            <domain:status s="clientTransferProhibited"/>
C:          </domain:add>
C:          <domain:chg>
C:            <domain:authInfo>
C:              <domain:null/>
C:            </domain:authInfo>
C:          </domain:chg>
C:        </domain:update>
C:      </update>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C:</epp>
```

Example of unsetting the authorization information with an empty authorization information value in an [RFC5731] domain name update command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>example.com</domain:name>
C:          <domain:add>
C:            <domain:status s="clientTransferProhibited"/>
C:          </domain:add>
C:          <domain:chg>
C:            <domain:authInfo>
C:              <domain:pw/>
C:            </domain:authInfo>
C:          </domain:chg>
C:        </domain:update>
C:      </update>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C:</epp>
```

Example of unsetting the authorization information with an empty authorization information value in an [RFC5733] contact update command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <contact:update
C:        xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
C:        <contact:id>sh8013</contact:id>
C:        <contact:chg>
C:          <contact:authInfo>
C:            <contact:pw/>
C:          </contact:authInfo>
C:        </contact:chg>
C:      </contact:update>
C:    </update>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C:</epp>
```

5.3. Info Command and Response

For an info command, the registry MUST allow for the passing of a non-empty authorization information value for verification. The gaining registrar can pre-verify the authorization information provided by the registrant prior to submitting the transfer request with the use of the info command. The registry compares the hash of the passed authorization information with the hashed authorization information value stored for the object. When the authorization information is not set or the passed authorization information does not match the previously set value, the registry MUST return an EPP error result code of 2202 [RFC5730].

Example of passing a non-empty authorization information value in an [RFC5731] domain name info command to verify the authorization information value:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <domain:info
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>example.com</domain:name>
C:          <domain:authInfo>
C:            <domain:pw>LuQ7Bu@w9?%+_HK3cayg$55$LSft3MPP
C:          </domain:pw>
C:        </domain:authInfo>
C:      </domain:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

The info response in object extensions, such as [RFC5731] and [RFC5733], MUST NOT include the optional authorization information element with a non-empty authorization value. The authorization information is stored as a hash in the registry, so returning the plain text authorization information is not possible, unless a valid plain text authorization information is passed in the info command. The registry MUST NOT return any indication of whether the authorization information is set or unset to the non-sponsoring registrar by not returning the authorization information element in the response. The registry MAY return an indication to the sponsoring registrar that the authorization information is set by using an empty authorization information value. The registry MAY return an indication to the sponsoring registrar that the authorization information is unset by not returning the authorization information element.

Example of returning an empty authorization information value in an [RFC5731] domain name info response to indicate to the sponsoring registrar that the authorization information is set:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:authInfo>
S:          <domain:pw/>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5.4. Transfer Request Command

For a Transfer Request Command, the registry MUST allow for the passing of a non-empty authorization information value to authorize a transfer. The registry compares the hash of the passed authorization information with the hashed authorization information value stored for the object. When the authorization information is not set or the passed authorization information does not match the previously set value, the registry MUST return an EPP error result code of 2202 [RFC5730]. Whether the transfer occurs immediately or is pending is up to server policy. When the transfer occurs immediately, the registry MUST return the EPP success result code of 1000 and when the transfer is pending, the registry MUST return the EPP success result code of 1001. The losing registrar MUST be informed of a successful transfer request using an EPP poll message.

Example of passing a non-empty authorization information value in an [RFC5731] domain name transfer request command to authorize the transfer:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <transfer op="request">
C:      <domain:transfer
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>example1.com</domain:name>
C:          <domain:authInfo>
C:            <domain:pw>LuQ7Bu@w9?%+_HK3cayg$55$LSft3MPP
C:          </domain:pw>
C:        </domain:authInfo>
C:      </domain:transfer>
C:    </transfer>
C:  <clTRID>ABC-12345</clTRID>
C: </command>
C:</epp>
```

Upon successful completion of the transfer, the registry **MUST** automatically unset the authorization information. If the transfer request is not submitted within the time-to-live (TTL) (Section 4.2) or the transfer is cancelled or rejected, the registrar **MUST** unset the authorization information as defined in Section 5.2.

6. Transition Considerations

The goal of the transition considerations to the practice defined in this document, referred to as the Secure Authorization Information Model, is to minimize the impact to the registrars by supporting incremental steps of adoption. The transition steps are dependent on the starting point of the registry. Registries may have different starting points, since some of the elements of the Secure Authorization Information Model may have already been implemented. The considerations assume a starting point, referred to as the Classic Authorization Information Model, that have the following steps in the management of the authorization information for transfers:

1. Registrant requests to register the object with the registrar. Registrar sends the create command, with a non-empty authorization information value, to the registry. The registry stores the authorization information as an encrypted value and requires a non-empty authorization information value for the life of the object. The registrar may store the long-lived authorization information.
2. At the time of transfer, Registrant requests from the losing registrar the authorization information to provide to the gaining registrar.

3. Losing registrar retrieves the locally stored authorization information or queries the registry for authorization information using the info command, and provides it to the registrant. If the registry is queried, the authorization information is decrypted and the plain text authorization information is returned in the info response to the registrar.
4. Registrant provides the authorization information value to the gaining registrar.
5. Gaining registrar optionally verifies the authorization information with the info command to the registry, by passing the authorization information in the info command to the registry.
6. Gaining registrar sends the transfer request with the authorization information to the registry. The registry will decrypt the stored authorization information to compare to the passed authorization information.
7. If the transfer successfully completes, the authorization information is not touched by the registry and may be updated by the gaining registrar using the update command. If the transfer is cancelled or rejected, the losing registrar may reset the authorization information using the update command.

The gaps between the Classic Authorization Information Model and the Secure Authorization Information Model include:

1. Registry requirement for a non-empty authorization information value on create and for the life of the object versus the authorization information not being set on create and only being set when a transfer is in process.
2. Registry not allowing the authorization information to be unset versus supporting the authorization to be unset in the update command.
3. Registry storing the authorization information as an encrypted value versus as a hashed value.
4. Registry support for returning the authorization information versus not returning the authorization information in the info response.
5. Registry not touching the authorization information versus the registry automatically unsetting the authorization information upon a successful transfer.
6. Registry may validate a shorter authorization information value using password complexity rules versus validating the randomness of a longer authorization information value that meets the required bits of entropy.

The transition can be handled in the three phases defined in the sub-sections Section 6.1, Section 6.2, Section 6.3.

6.1. Transition Phase 1 - Features

The goal of the "Transition Phase 1 - Features" is to implement the needed features in EPP so that the registrar can optionally implement the Secure Authorization Information Model. The features to implement are broken out by the command and responses below:

Create Command: Change the create command to make the authorization information optional, by allowing both a non-empty value and an empty value. This enables a registrar to optionally create objects without an authorization information value, as defined in Section 5.1.

Update Command: Change the update command to allow unsetting the authorization information, as defined in Section 5.2. This enables the registrar to optionally unset the authorization information when the TTL expires or when the transfer is cancelled or rejected.

Transfer Approve Command and Transfer Auto-Approve: Change the transfer approve command and the transfer auto-approve to automatically unset the authorization information. This sets the default state of the object to not have the authorization information set. The registrar implementing the Secure Authorization Information Model will not set the authorization information for an inbound transfer and the registrar implementing the Classic Authorization Information Model will set the new authorization information upon the successful transfer.

Info Response: Change the info command to not return the authorization information in the info response, as defined in Section 5.3. This sets up the implementation of "Transition Phase 2 - Storage", since the dependency in returning the authorization information in the info response will be removed. This feature is the only one that is not an optional change to the registrar that has the potential of breaking the client, so it's recommended that the registry provide notice of the change.

Info Command and Transfer Request: Change the info command and the transfer request to ensure that a registrar cannot get an indication that the authorization information is set or not set by returning the EPP error result code of 2202 when comparing a passed authorization to a non-matching set authorization information value or an unset value.

6.2. Transition Phase 2 - Storage

The goal of the "Transition Phase 2 - Storage" is to transition the registry to use hashed authorization information instead of encrypted authorization information. There is no direct impact to the registrars, since the only visible indication that the authorization information has been hashed is by not returning the set authorization information in the info response, which is addressed in Transition Phase 1 - Features (Section 6.1). There are three steps to transition the authorization information storage, which includes:

Hash New Authorization Information Values: Change the create command and the update command to hash instead of encrypting the authorization information.

Supporting Comparing Against Encrypted and Hashed Authorization Information: Change the info command and the transfer request command to be able to compare a passed authorization information value with either a hashed or encrypted authorization information value. This requires that the stored values are self-identifying as being in hashed or encrypted form.

Hash Existing Encrypted Authorization Information Values: Convert the encrypted authorization information values stored in the registry database to hashed values. The update is not a visible change to the registrar. The conversion can be done over a period of time depending on registry policy.

6.3. Transition Phase 3 - Enforcement

The goal of the "Transition Phase 3 - Enforcement" is to complete the implementation of the "Secure Authorization Information Model", by enforcing the following:

Disallow Authorization Information on Create Command: Change the create command to not allow for the passing of a non-empty authorization information value. This behavior has the potential of breaking the client, so it's recommended that the registry provide notice of the change.

Validate the Strong Random Authorization Information: Change the validation of the authorization information in the update command to ensure at least 128 bits of entropy.

7. IANA Considerations

7.1. XML Namespace

This document uses URNs to describe XML namespaces conforming to a registry mechanism described in [RFC3688]. The following URI assignment is requested of IANA:

Registration request for the secure authorization information for transfer namespace:

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

7.2. EPP Extension Registry

The EPP operational practice described in this document should be registered by the IANA in the EPP Extension Registry described in [RFC7451]. The details of the registration are as follows:

Name of Extension: "Extensible Provisioning Protocol (EPP) Secure Authorization Information for Transfer"

Document status: Standards Track

Reference: (insert reference to RFC version of this document)

Registrant Name and Email Address: IESG, <iesg@ietf.org>

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

8. 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".

8.1. 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-secure-authinfo-transfer.

Level of maturity: Development

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

URL: https://www.verisign.com/en_US/channel-resources/domain-registry-products/epp-sdks

8.2. RegistryEngine EPP Service

Organization: CentralNic

Name: RegistryEngine EPP Service

Description: Generic high-volume EPP service for gTLDs, ccTLDs and SLDs

Level of maturity: Deployed in CentralNic's production environment as well as two other gTLD registry systems, and two ccTLD registry systems.

Coverage: Authorization Information is "write only" in that the registrars can set the Authorization Information, but not get the Authorization Information in the Info Response.

Licensing: Proprietary In-House software

Contact: epp@centralnic.com

URL: <https://www.centralnic.com>

9. Security Considerations

Section 4.1 defines the use a secure random value for the generation of the authorization information. The client SHOULD choose a length and set of characters that results in at least 128 bits of entropy.

Section 4.2 defines the use of an authorization information Time-To-Live (TTL). The registrar SHOULD only set the authorization information during the transfer process by the server support for setting and unsetting the authorization information. The TTL value is up to registrar policy and the sponsoring registrar MUST inform the registrant of the TTL when providing the authorization information to the registrant.

Section 4.3 defines the storage and transport of authorization information. The losing registrar MUST NOT store the authorization information and the gaining registrar MUST only store the authorization information as a "transient" value during the transfer process, where the authorization information MUST NOT be stored after the end of the transfer process. The registry MUST store the authorization information using a one-way cryptographic hash of at least 256 bits and with a per-authorization information random salt, with at least 128 bits. All communication that includes the authorization information MUST be over an encrypted channel. The plain text authorization information MUST NOT be written to any logs by the registrar or the registry.

Section 4.4 defines the matching of the authorization information values. The registry stores an unset authorization information as a NULL (undefined) value to ensure that an empty input authorization information never matches it. The method used to define a NULL (undefined) value is database specific.

10. Acknowledgements

The authors wish to thank the following persons for their feedback and suggestions: Michael Bauland, Martin Casanova, Scott Hollenbeck, Benjamin Kaduk, Jody Kolker, Barry Leiba, Patrick Mevzek, Matthew Pozun, Srikanth Veeramachaneni, and Ulrich Wissner.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<https://www.rfc-editor.org/info/rfc5730>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <<https://www.rfc-editor.org/info/rfc5731>>.
- [RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.17487/RFC5733, August 2009, <<https://www.rfc-editor.org/info/rfc5733>>.
- [RFC5734] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Transport over TCP", STD 69, RFC 5734, DOI 10.17487/RFC5734, August 2009, <<https://www.rfc-editor.org/info/rfc5734>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.

11.2. Informative References

[FIPS-140-2]

National Institute of Standards and Technology, U.S. Department of Commerce, "NIST Federal Information Processing Standards (FIPS) Publication 140-2", May 2001, <<https://csrc.nist.gov/publications/detail/fips/140/2/final>>.

[FIPS-180-4]

National Institute of Standards and Technology, U.S. Department of Commerce, "Secure Hash Standard, NIST Federal Information Processing Standards (FIPS) Publication 180-4", August 2015, <<https://csrc.nist.gov/publications/detail/fips/180/4/final>>.

[RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <<https://www.rfc-editor.org/info/rfc7451>>.

Appendix A. Change History

A.1. Change from 00 to 01

1. Filled in the "Implementation Status" section with the inclusion of the "Verisign EPP SDK" and "RegistryEngine EPP Service" implementations.
2. Made small wording corrections based on private feedback.
3. Added content to the "Acknowledgements" section.

A.2. Change from 01 to 02

1. Revised the language used for the storage of the authorization information based on the feedback from Patrick Mevzek and Jody Kolker.

A.3. Change from 02 to 03

1. Updates based on the feedback from the interim REGEXT meeting held at ICANN-66:
 1. Section 3.3, include a reference to the hash algorithm to use. Broke the requirements into a list and included a the reference the text ', with at least a 256-bit hash function, such as SHA-256'.
 2. Add a Transition Considerations section to cover the transition from the classic authorization information security model in the EPP RFCs to the model defined in the document.

3. Add a statement to the Introduction that elements of the practice can be used for purposes other than transfer, but with a caveat.
2. Updates based on the review by Michael Bauland, that include:
 1. In section 2, change 'there are three actors' to 'there are three types of actors' to cover the case with transfers that has two registrar actors (losing and gaining).
 2. In section 3.1, change the equations equals to be approximately equal by using '=' instead of '=', where applicable.
 3. In section 3.3, change 'MUST be over an encrypted channel, such as RFC5734' to 'MUST be over an encrypted channel, such as defined in RFC5734'.
 4. In section 4.1, remove the optional RFC 5733 elements from the contact create, which includes the <contact:voice>, <contact:fax>, <contact:disclose>, <contact:org>, <contact:street>, <contact:sp>, and <contact:cc> elements.
 5. In section 4.2, changed 'Example of unsetting the authorization information explicitly in an [RFC5731] domain name update command.' to 'Example of adding the "clientTransferProhibited" status and unsetting the authorization information explicitly in an [RFC5731] domain name update command.'
 6. In section 4.3, cover a corner case of the ability to return the authorization information when it's passed in the info command.
 7. In section 4.4, change 'If the transfer does not complete within the time-to-live (TTL)' to 'If the transfer is not initiated within the time-to-live (TTL)', since the TTL is the time between setting the authorization information and when it's successfully used in a transfer request. Added the case of unsetting the authorization information when the transfer is cancelled or rejected.
3. Updates based on the authorization information messages by Martin Casanova on the REGEXT mailing list, that include:
 1. Added section 3.4 'Authorization Information Matching' to clarify how the authorization information is matched, when there is set and unset authorization information in the database and empty and non-empty authorization information passed in the info and transfer commands.
 2. Added support for signaling that the authorization information is set or unset to the sponsoring registrar with the inclusion of an empty authorization information element in the response to indicate that the authorization information is set and the exclusion of the authorization information element in the response to indicate that the authorization information is unset.

4. Made the capitalization of command and response references consistent by uppercasing section and item titles and lowercasing references elsewhere.

A.4. Change from 03 to REGEXT 00

1. Changed to regext working group draft by changing draft-gould-regext-secure-authinfo-transfer to draft-ietf-regext-secure-authinfo-transfer.

A.5. Change from REGEXT 00 to REGEXT 01

1. Added the "Signaling Client and Server Support" section to describe the mechanism to signal support for the BCP by the client and the server.
2. Added the "IANA Considerations" section with the registration of the secure authorization for transfer XML namespace and the registration of the EPP Best Current Practice (BCP) in the EPP Extension Registry.

A.6. Change from REGEXT 01 to REGEXT 02

1. Added inclusion of random salt for the hashed authorization information, based on feedback from Ulrich Wisser.
2. Added clarification that the representation of a NULL (undefined) value is dependent on the type of database, based on feedback from Patrick Mevzek.
3. Filled in the Security Considerations section.

A.7. Change from REGEXT 02 to REGEXT 03

1. Updated the XML namespace to urn:ietf:params:xml:ns:epp:secure-authinfo-transfer-1.0, which removed bcp from the namespace and bumped the version from 0.1 and 1.0. Inclusion of bcp in the XML namespace was discussed at the REGEXT interim meeting.
2. Replaced Auhtorization with Authorization based on a review by Jody Kolker.

A.8. Change from REGEXT 03 to REGEXT 04

1. Converted from xml2rfc v2 to v3.
2. Updated Acknowledgements to match the approach taken by the RFC Editor with draft-ietf-regext-login-security.
3. Changed from Best Current Practice (BCP) to Standards Track based on mailing list discussion.

A.9. Change from REGEXT 04 to REGEXT 05

1. Fixed IDNITS issues, including moving RFC7451 to Informative References section.

A.10. Change from REGEXT 05 to REGEXT 06

Updates based on the Barry Leiba (AD) feedback:

1. Simplified the abstract based on the proposal provided.
2. In the Introduction, split the first paragraph by starting a new paragraph at "This document".
3. In section 1.1, updated to use the new BCP 14 boilerplate and add a normative reference to RFC 8174.
4. In section 4, Updated the phrasing to "For the authorization information to be secure it must be generated using a strong random value and have a short time-to-live (TTL).".
5. In section 4.1, removed the first two unnecessary calculations and condensed the introduction of the section.
6. In section 4.1, added the use of the normative SHOULD for use of at least 128 bits of entropy.
7. Added an informative reference to FIPS 180-4 for the SHA-256 references.
8. Normalized the way that the "empty and non-empty authorization information values" are referenced, which a few exceptions.
9. In section 4, revised the first sentence to explicitly reference the use of the <domain:pw> and <contact:pw> elements for password-based authorization information.
10. In section 4.4, revised the language associated with the storage of the authorization information to be cleaner.
11. In section 4.4, added "set" in the sentence "An empty input authorization information value MUST NOT match any set authorization information value."
12. In section 5.1 and 5.2, clarified the references to RFC5731 and RFC5733 as examples of object extensions that use the "eppcom:pwAuthInfoType" element.
13. In section 5.2, updated language for the validation of the randomness of the authorization information, based on an offline review by Barry Leiba, Benjamin Kaduk, and Roman Danyliw.
14. In section 9, changed "49 bits of entropy" to "128 bits of entropy".

In section 3, replaced the reference to BCP with operational practice, since the draft is not defined as a BCP.

A.11. Change from REGEXT 06 to REGEXT 07

1. Updates based on the Lars Eggert feedback:

1. Updated Section 1, Paragraph 4 to read "The operational practice will require the client to not store the authorization information and".
 2. Updated each of the example references to end with a colon instead of a period.
 3. Updated Section 1, Paragraph 3 to read "provide secure authorization information used for transfers."
 4. Updated Section 3, Paragraph 3 to read "extension services can expect".
 5. Updated Section 4, Paragraph 2 to read "authorization information to be secure, it must".
 6. Updated Section 4.2, Paragraph 2 to read "authorization information by the sponsoring registrar, the".
 7. Updated Section 4.2, Paragraph 2 to read "proprietary registrar-specific criteria, which".
 8. Updated Section 4.3, Paragraph 3 to read "256-bit hash function, such as SHA-256".
 9. Updated Section 4.3, Paragraph 3 to read "a NULL (undefined) value".
 10. Updated Section 5, Paragraph 2 to read "To secure the transfer process using secure authorization".
 11. Updated Section 5.2, Paragraph 6 to read "Often, the registrar has the "clientTransferProhibited" status set".
 12. Updated Section 5.2, Paragraph 9 to read "MUST cancel cancel the transfer process by unsetting the authorization information value and MAY add back statuses".
 13. Updated Section 5.2, Paragraph 9 to read ""eppcom:pwAuthInfoType" element can have".
2. Updated the first sentence of the abstract and introduction based on the Rob Wilton feedback to help non-EPP readers on the what and the who for transfers.
 3. Removed the duplicate first paragraph of section 5.2 based on feedback from Francesca Palombini.
 4. Updates based on the Benjamin Kaduk feedback:
 1. Added the second paragraph in the Introduction to provide high-level motivation for the work.
 2. Updated Section 1, changed "in any way" to "in any substantial way".
 3. Updated Section 1 by adding the sentence "All of these features are compatible with the EPP RFCs, though not mandatory to implement." for the "Short-Lived Authorization Information".

4. Updated the description of "Short-Lived Authorization Information" in Section 1 to reference section 2.6 of RFC5731 and change in nature of the authorization information.
5. Updated Section 4.1, Paragraph 1 and 2 were merged with modified language proposed by Benjamin Kaduk, which included removing the reference to RFC4086 for length and entropy.
6. Updated rule #1 of Section 4.1 to add a second clarifying sentence for what is meant by input authorization information.
7. Updated Section 4.1 by replacing the last paragraph "The strength of the random..." with a revised version.
8. Updated "retrieves the stored authorization information locally" with "retrieves the locally stored authorization information".
9. Updated Section 6.1 to include the recommendation that the registry provide notice of the Info Response change.
10. Updated Section 6.2 to include the sentence "This requires that the stored values are self-identifying as being in hashed or encrypted form" for the "Supporting Comparing Against Encrypted and Hashed Authorization Information" step.
11. Updated Section 6.3 to include the recommendation that the registry provide notice of the Create Command change.
12. Updated "written to any logs by the registrar or the registry" to "written to any logs by a registrar or the registry" to cover both the losing and the gaining registrar.
13. Updated references to "with a random salt" to "with a per-authorization information random salt, with at least 128 bits" to address sharing of salts and the size of the salts.
14. Updated the first paragraph of Section 9 to remove the reference to defining a server policy for the length and set of characters that are included in the randomization to target the target entropy level.
15. Updated Section 9 by removing the sentence "A random number generator (RNG) is preferable over the use of a pseudorandom number generator (PRNG) when creating the authorization information value."
16. Changed FIPS-140-2 from a normative reference to an informative reference.

Authors' Addresses

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

Email: jgould@verisign.com
URI: <http://www.verisign.com>

Richard Wilhelm
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States of America

Email: rwilhelm@verisign.com
URI: <http://www.verisign.com>

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: 8 September 2022

J. Yee, Ed.
J. Galvin
Donuts
7 March 2022

Simple Registration Reporting
draft-ietf-regext-simple-registration-reporting-06

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.

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. 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

2.4.6.	Contact_Name	8
2.4.7.	In_use	9
2.4.8.	Nameserver_Host	9
2.4.9.	Nameserver_IP	9
2.4.10.	Client_Contact_ID	9
3.	Report Definition Specification	9
3.1.	Domain Transaction	10
3.2.	Premium Name	11
3.3.	Domain RGP	11
3.4.	Reserved Domain	12
3.5.	Domain Inventory	12
3.6.	Contact Inventory	13
3.7.	Host Inventory	14
4.	IANA Considerations	14
4.1.	Report Specification	15
4.1.1.	Designated Expert Evaluation Criteria	15
4.1.2.	Registration Procedure	16
4.1.2.1.	Required Information	16
4.1.2.2.	Registration Processing	17
4.1.2.3.	Updating Report Definition Registry Entries	17
4.2.	Initial assignments	18
4.2.1.	Data Element Definition in IANA Registry	18
4.2.2.	Report Definition in IANA Registry	29
5.	Security Considerations	32
6.	Privacy Considerations	32
7.	Internationalization Considerations	32
8.	References	32
8.1.	Normative References	32
8.2.	Informative References	33
Appendix A.	Acknowledgements	34
Appendix B.	File Naming Convention	34
Authors'	Addresses	34

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

Data Element	Reference
TLD	RFC XXXX Section 2.1.1
Server_TRID	Section 2.1.2
Domain	Section 2.1.3
DateTime	Section 2.1.6
Registrar_ID	Section 2.4.1
Registrar	Section 2.1.11
Transaction_Type	Section 2.1.4
Period	Section 2.1.12
Term	Section 2.1.7
Fee	Section 2.1.8
Currency	Section 2.1.9
Description	Section 2.1.13

Table 1: Transaction Report Definition Table

3.2. Premium Name

Name of report: premium_name

Data Element	Reference
TLD	RFC XXXX Section 2.1.1
Domain	Section 2.1.3
Status	Section 2.1.10
Description	Section 2.1.13
Currency	Section 2.1.9
Domain_Create	Section 2.2.1
Domain_Renew	Section 2.2.2
Domain_Transfer	Section 2.2.3
Domain_Restore	Section 2.2.4
Start_Date	Section 2.3.1

Table 2: Premium Name Report Definition
Table

3.3. Domain RGP

Name of report: domain_rgp

Data Element	Reference
TLD	RFC XXXX Section 2.1.1
Domain	Section 2.1.3
Deleted_Date	Section 2.3.2
RGP_Date	Section 2.3.3
Purge_Date	Section 2.3.4

Table 3: Domain RGP Report Definition
Table

3.4. Reserved Domain

Name of report: reserved_domain

Data Element	Reference
TLD	RFC XXXX Section 2.1.1
Domain	Section 2.1.3
Status	Section 2.1.10

Table 4: Reserved Domain Report
Definition Table

3.5. Domain Inventory

Name of report: domain_inventory

Data Element	Reference
TLD	RFC XXXX Section 2.1.1
Domain	Section 2.1.3
Updated_Date	Section 2.3.5
Registrar_ID	Section 2.4.1
Create_Date	Section 2.3.6
Expiry_Date	Section 2.3.7
Server_Registrant_ID	Section 2.4.2
DNSSEC	Section 2.4.3
Status	Section 2.1.10

Table 5: Domain Inventory Report Definition Table

3.6. Contact Inventory

Name of report: contact_inventory

Data Element	Reference
Server_Contact_ID	Section 2.4.4
Client_Contact_ID	Section 2.4.10
TLD	Section 2.1.1
Domain	Section 2.1.3
Contact_Type	Section 2.4.5
Contact_Name	Section 2.4.6
Updated_Date	Section 2.3.5
INUSE	Section 2.4.7
Registrar_ID	Section 2.4.1

Table 6: Contact Inventory Report
Definition Table

3.7. Host Inventory

Name of report: host_inventory

Data Element	Reference
TLD	RFCXXXX Section 2.1.1
Nameserver_Host	Section 2.4.8
Nameserver_IP	Section 2.4.9

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 regex mailing list (regex@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 regex 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 regex 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 web site. 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

---- END FORM ----

---- BEGIN FORM ----

Name of data element:

Transaction_Type

Reference:

This RFC Section 2.1.4

Registrant:

IESG, iesg@ietf.org

Status:

Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:

Object_Type

Reference:

This RFC Section 2.1.5

Registrant:

IESG, iesg@ietf.org

Status:

Active

---- END FORM ----

---- BEGIN FORM ----

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:

Term

Reference:

This RFC Section 2.1.7

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

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

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

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

Name of data element:

Domain_Create

Reference:

This RFC Section 2.2.1

Registrant:

IESG, iesg@ietf.org

Status:

Active

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

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

Name of data element:

Domain_Renew

Reference:

This RFC Section 2.2.2

Registrant:

IESG, iesg@ietf.org

Status:

Active

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

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

Name of data element:

Domain_Transfer

Reference:

This RFC Section 2.2.3

Registrant:

IESG, iesg@ietf.org

Status:

Active

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

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

Name of data element:

Domain_Restore

Reference:

This RFC Section 2.2.4

Registrant:

IESG, iesg@ietf.org

Status:

Active

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

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

Name of data element:

Start_Date

Reference:

This RFC Section 2.3.1

Registrant:

IESG, iesg@ietf.org

Status:

Active

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

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

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

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

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

Name of data element:

Create_Date

Reference:

This RFC Section 2.3.6

Registrant:

IESG, iesg@ietf.org

Status:

Active

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

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

Name of data element:

Expiry_Date

Reference:

This RFC Section 2.3.7

Registrant:

IESG, iesg@ietf.org

Status:

Active

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

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

Name of data element:

Registrar_ID

Reference:

This RFC Section 2.4.1

Registrant:

IESG, iesg@ietf.org

Status:

Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:

Server_Registrant_ID

Reference:

This RFC Section 2.4.2

Registrant:

IESG, iesg@ietf.org

Status:

Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:

DNSSEC

Reference:

This RFC Section 2.4.3

Registrant:

IESG, iesg@ietf.org

Status:

Active

---- END FORM ----

---- BEGIN FORM ----

Name of data element:

Server_Contact_ID

Reference:

This RFC Section 2.4.4

Registrant:

IESG, iesg@ietf.org

Status:

Active

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

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

Name of data element:

Contact_Type

Reference:

This RFC Section 2.4.5

Registrant:

IESG, iesg@ietf.org

Status:

Active

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

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

Name of data element:

Contact_Name

Reference:

This RFC Section 2.4.6

Registrant:

IESG, iesg@ietf.org

Status:

Active

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

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

Name of data element:

INUSE

Reference:

This RFC Section 2.4.7

Registrant:

IESG, iesg@ietf.org

Status:

Active

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

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

Name of data element:

Nameserver_Host

Reference:

This RFC Section 2.4.8

Registrant:

IESG, iesg@ietf.org

Status:

Active

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

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

Name of data element:

Nameserver_IP

Reference:

This RFC Section 2.4.9

Registrant:

IESG, iesg@ietf.org

Status:

Active

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

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

Name of data element:

Client_Contact_ID

Reference:

This RFC Section 2.4.10

Registrant:

IESG, iesg@ietf.org

Status:
Active

---- END FORM ----

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 ----

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

- [ISO4217] International Organization for Standardization, "ISO 4217 Currency Codes", 2018, <https://www.currency-iso.org/dam/downloads/lists/list_one.xml>.

- [RFC0791] Postel, J., "Internet Protocol", September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4180] Shafranovich, Y., "IP Version 6 Addressing Architecture", February 2006, <<https://www.rfc-editor.org/info/rfc4180>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", August 2009, <<https://www.rfc-editor.org/info/rfc5730>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", August 2009, <<https://www.rfc-editor.org/info/rfc5731>>.
- [RFC5732] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Host Mapping", August 2009, <<https://www.rfc-editor.org/info/rfc5732>>.
- [RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", August 2009, <<https://www.rfc-editor.org/info/rfc5733>>.
- [RFC5890] Klensin, J., "Extensible Provisioning Protocol (EPP) Contact Mapping", August 2009, <<https://www.rfc-editor.org/info/rfc5890>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8748] Carney, R. and J. Frakes, "Registry Fee Extension for the Extensible Provisioning Protocol", March 2021, <<https://www.rfc-editor.org/info/rfc8748>>.

8.2. Informative References

- [IANA_Registrar_IDs]
Internet Assigned Numbers Authority, "IANA Assignments - Registrar IDs", 2020, <<https://www.iana.org/assignments/registrar-ids/registrar-ids.xhtml>>.

- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <<https://www.rfc-editor.org/info/rfc2629>>.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.

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@afiliias.info

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

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: 8 January 2022

U. Wisser
The Swedish Internet Foundation
7 July 2021

Registry Lock Extension for the Extensible Provisioning Protocol (EPP)
draft-wisser-registrylock-04

Abstract

This extensions defines an additional protective layer for changes to domain [RFC5731], host [RFC5732] and contact [RFC5733] objects managed through EPP.

EPP allows changes to objects only by the sponsoring client. EPP objects are usually managed by the sponsoring client on behalf of the sponsoring clients customers. All of these interactions are ususally fully automated.

In case of a system breach, there is no protection in EPP to changes to any object by the intruder.

This extension defines a protective layer that aims to break automated changes and work flows by requiring manual intervention.

The actual form of manual intervention is out-of-scope for this document. By whom and how changes can be made is up to the registry and registrars to decide.

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 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 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 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.

Table of Contents

1. Introduction	3
1.1. Conventions Used in This Document	3
2. Object Protection	4
2.1. Out-of-band Authorization	4
2.2. In-band Authorization	4
2.3. Command Execution Restrictions	4
2.4. Temporary Unlock	5
3. Object Attributes	6
3.1. Locking Status	6
4. EPP Command Mapping	6
4.1. EPP Query Commands	6
4.1.1. EPP <check> Command	6
4.1.2. EPP <info> Command	6
4.1.3. EPP <transfer> Command	9
4.2. EPP Transform Commands	9
4.2.1. EPP <create> Command	9
4.2.2. EPP <delete> Command	10
4.2.3. EPP <renew> Command	10
4.2.4. EPP <transfer> Command	11
4.2.5. EPP <update> Command	11
5. Formal Syntax	12
5.1. Registry Lock Extension Schema	12
6. IANA Considerations	13
6.1. XML Namespace	13
6.2. EPP Extension Registry	14
7. Implementation Status	14
8. Security Considerations	14
9. Acknowledgements	15
10. Normative References	15
Appendix A. Change History	16
A.1. Change from 00 to 01	16
A.2. Change from 01 to 02	16

A.3. Change from 02 to 03	16
A.4. Change from 03 to 04	16
Author's Address	16

1. Introduction

This extensions defines an additional protective layer for changes to domain [RFC5731], host [RFC5732] and contact [RFC5733] objects managed through EPP.

EPP allows changes to objects only by the sponsoring client. EPP objects are usually managed by the sponsoring client on behalf of the sponsoring clients customers. All of these interactions are ususally fully automated.

In case of a system breach, there is no protection in EPP to changes to any object by the intruder.

This extension defines a protective layer that aims to break automated changes and work flows by requiring manual intervention.

The actual form of manual intervention is out-of-scope for this document. By whom and how changes can be made is up to the registry and registrars to decide.

1.1. Conventions Used in This Document

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 [RFC2119].

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.

In examples, "C:" represents lines sent by a protocol client and "S:" represents lines returned by a protocol server. Indentation and white space in examples are provided only to illustrate element relationships and are not a REQUIRED feature of this protocol.

"regLock" is used as an abbreviation for "urn:ietf:params:xml:ns:epp:registryLock-1.0". The XML namespace prefix "reglock" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

2. Object Protection

This extension provides additional protection to objects managed by a sponsoring client on behalf of a registrant. This is achieved by requiring additional authorization for transform commands.

Solutions can be broadly categorized as in-band or out-of-band authorizations. Where in-band authorizations would provide authorization through EPP. Whereas out-of-band solutions provide authorization by some other means.

- * either by temporarily unlocking the object for changes
- * or by authorizing pending changes after they have been submitted to the server

2.1. Out-of-band Authorization

Out-of-band Authorization is not covered in this document. By definition out-of-band authorization will not use EPP and therefore is not subject of consideration here.

Registries must provide means for the registrar or registrant to temporarily unlock the domain, to remove registry lock or to authorize changes submitted to the server through some means than EPP.

2.2. In-band Authorization

Currently defined authorization schemes are not deemed secure enough for in-band change authorization. Therefore this document does not allow in-band authorization. This is left as a future development once secure enough authorization schemes have been defined.

The current defined authorization scheme is based on static passwords. This would mean that once a password is known any change can be made. Security here is once again dependent on the security of all automatic systems involved.

2.3. Command Execution Restrictions

Once an object has Registry Lock enabled all transform commands except <renew> MUST only be executed if a proper authorization has been made.

Otherwise the command MUST be rejected with EPP result code 2201 "Authorization error" or 1001 "Command completed successfully; action pending" [RFC5730] section 3 in depending on the chosen out-of-band authorization.

if the server has returned a 1001 "Command completed successfully; action pending" answer, it MUST follow [RFC5731], [RFC5732], [RFC5733] in handling succeeded or failed commands.

The following EPP flags must be set.

- * serverDeleteProhibited
- * serverTransferProhibited
- * serverUpdateProhibited

If the object is unlocked the flags SHOULD be cleared and the server should answer to an <info> request with the according information.

OPEN QUESTION: If a domain is under registry lock, can a subordinate host be updated?

- * I got one "no" answer - hosts might not be owned by domain owner
- * In .se/.nu all subordinate hosts are automatically owned by the domain owner and locked if the domain is locked.

We need more input!

If the object is temporarily unlocked only <update> commands are allowed. <delete> and <transfer> are explicitly not allowed. For the time of the temporary unlock the serverUpdateProhibited status should be cleared.

2.4. Temporary Unlock

While an object is locked some situations could require a change. To fully unlock the object would remove all protection and could not provide any guarantee that the object is protected again after the desired changes have been made.

Temporarily unlocking the object allows for a more fine grained security model for all objects.

Any temporary unlocking of the object has to be time limited. After that time has passed no further changes are possible.

Additionally the number of allowed EPP commands can be specified to further limit the changes possible.

Registries and registrars can further limit the possibilities changes, e.g. not allowing owner changes even for temporarily unlocked Domain objects.

IS THE LAST PARAGRAPH A GOOD IDEA? INPUT NEEDED!!!

When an object is temporarily unlocked the `serverUpdateProhibited` SHOULD be cleared while changes are possible.

When either the time for the temporary unlock has passed or the maximum amount of EPP changes has been made the object MUST return to a fully locked status. The `serverUpdateProhibited` flag MUST be set again and the `infData` response MUST no longer contain a `<unlockedUntil>` element.

3. Object Attributes

3.1. Locking Status

Locking Status information indicates if the additional protection of Registry Lock is enabled for an object.

Boolean values MUST be represented in the XML Schema format described in Part 2 of the W3C XML Schema recommendation [W3C.REC-xmlschema-2-20041028].

4. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730].

4.1. EPP Query Commands

4.1.1. EPP `<check>` Command

This extension does not add any elements to the EPP `<check>` command or `<check>` response described in the EPP mappings [RFC5731], [RFC5732] or [RFC5733].

4.1.2. EPP `<info>` Command

This extension does not add any elements to the EPP `<info>` command described in the EPP domain mapping [RFC5731], host mapping [RFC5732] or contact mapping [RFC5733] However, additional elements are defined for the `<info>` response.

When an `<info>` command has been processed successfully, the EPP `<resData>` element MUST contain child elements as described in the EPP object mappings.

In addition, the EPP `<extension>` element SHOULD contain a child `<regLock:infData>` element that identifies the extension namespace the epp client has indicated support for the extension in the `<login>` command.

The <regLock:infData> element contains the following child elements:

- * Exactly one <locked> element that indicates if Registry Lock is enabled for the object.
- * An OPTIONAL <unlockedUntil> element if the object currently can be changed by the sponsoring client. The field indicates the time stamp when the lock will become active again.
- * An OPTIONAL <eppCmdCount> attribute that indicates the number of EPP <update> commands that will be executed.

Example <domain:info> Response, domain not locked

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
S:  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
...
S:    </domain:infData>
S:  </resData>
S:  <extension>
S:    <regLock:infData
S:      xmlns:regLock="urn:ietf:params:xml:ns:epp:registryLock-1.0">
S:        <regLock:locked>0</regLock:locked>
S:      </regLock:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <domain:info> Response, domain locked

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
S:  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
...
S:    </domain:infData>
S:  </resData>
S:  <extension>
S:    <regLock:infData
S:      xmlns:regLock="urn:ietf:params:xml:ns:epp:registryLock-1.0">
S:        <regLock:locked>1</regLock:locked>
S:      </regLock:infData>
S:    </extension>
S:  <trID>
S:    <clTRID>ABC-12345</clTRID>
S:    <svTRID>54322-XYZ</svTRID>
S:  </trID>
S: </response>
S:</epp>
```

Example <domain:info> Response, domain temporary unlocked

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
S:  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
...
S:        </domain:infData>
S:      </resData>
S:    <extension>
S:      <regLock:infData
S:        xmlns:regLock="urn:ietf:params:xml:ns:epp:registryLock-1.0">
S:          <regLock:locked>1</regLock:locked>
S:<regLock:unlockedUntil eppCmdCount="1">20000101T000000+0000
S:</regLock:unlockedUntil>
S:      </regLock:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

4.1.3. EPP <transfer> Command

This extension does not add any elements to the EPP <transfer> command or <transfer> response described in the EPP mapping [RFC5731], [RFC5732] or [RFC5733].

4.2. EPP Transform Commands

4.2.1. EPP <create> Command

This extension is intended to be used within the scope of the object creation. It does not define a <create> command of its own.

This extension adds elements to the EPP <create> command as described in the EPP [RFC5730].

When submitting a <create> command to the server, the client MAY include in the <extension> element a <registryLock:lock> element to create the domain in a locked state. The extension includes the following element:

- * A <regLock:lock> element indicating that the domain MUST be created in a locked state.

When a <create> command has been processed successfully, the EPP response is as described in the EPP objects mappings [RFC5731], [RFC5732], [RFC5733].

Example <host:create> command

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <host:create
C:        xmlns:host="urn:ietf:params:xml:ns:host-1.0">
C:          <host:name>ns1.example.com</host:name>
C:          <host:addr ip="v4">192.0.2.2</host:addr>
C:          <host:addr ip="v4">192.0.2.29</host:addr>
C:          <host:addr ip="v6">1080:0:0:0:8:800:200C:417A</host:addr>
C:        </host:create>
C:      </create>
C:    <extension>
C:      <regLock:lock
C:        xmlns:regLock="urn:ietf:params:xml:ns:epp:registryLock-1.0" />
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

4.2.2. EPP <delete> Command

This extension does not add any elements to the EPP <delete> command or <delete> response described in the EPP mappings [RFC5731], [RFC5732] or [RFC5733].

If the object is locked, the EPP <delete> command MUST be rejected with EPP response code 2201 "Authorization error" [RFC5730] section 3. See Section 2.3

4.2.3. EPP <renew> Command

This extension does not add any elements to the EPP <renew> command or <renew> response described in the EPP mappings [RFC5731], [RFC5732] or [RFC5733].

Execution of the EPP <renew> command is not restricted by this extension.

4.2.4. EPP <transfer> Command

This extension does not add any elements to the EPP <transfer> command or <transfer> response described in the EPP mappings [RFC5731], [RFC5732] or [RFC5733].

If the object is locked, the EPP <transfer> command MUST be rejected with EPP response code 2201 "Authorization error" [RFC5730] section 3. See Section 2.3

4.2.5. EPP <update> Command

This extension adds elements to the EPP <update> command as described in [RFC5730].

If the object is not locked, the <update> command can be used to lock the object, similarly to the <create> command.

If the object is in locked state, but temporarily unlocked, the server MUST execute the command as if the object were unlocked.

If the object is locked the server can handle <update> commands in two ways

- * answering the command with EPP response code 1001 "Command completed successfully; action pending" [RFC5730] section 3
- * rejecting with EPP response code 2201 "Authorization error" [RFC5730] section 3

Registries can narrow down allowed changes when a domain is locked. Registries could prohibit changes of registrant for domains even if the domain is temporarily unlocked or password authorization is given.

When a <update> command has been processed successfully, the EPP response is as described in the EPP objects mappings [RFC5731], [RFC5732], [RFC5733].

Example <domain:update> command, locking domain

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>example.com</domain:name>
C:        </domain:update>
C:      </update>
C:    <extension>
C:      <regLock:lock
C:        xmlns:regLock="urn:ietf:params:xml:ns:epp:registryLock-1.0" />
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

5. Formal Syntax

One schema is presented here that is the EPP Registry Lock Extension schema.

The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances. The BEGIN and END tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.

5.1. Registry Lock Extension Schema

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="urn:ietf:params:xml:ns:epp:registryLock-1.00"
  xmlns:regLock="urn:ietf:params:xml:ns:epp:registryLock-1.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

  <xs:annotation>
    <xs:documentation>
      Registry Lock Extension to the
      Extensible Provisioning Protocol v1.0
    </xs:documentation>
  </xs:annotation>

  <!-- child elements found in EPP commands -->

  <xs:element name="lock" />

  <!-- child elements found in EPP responses -->

  <xs:element name="infData" type="regLock:infDataType"/>

  <!-- child element of the response -->

  <xs:complexType name="infDataType">
    <xs:sequence>
      <xs:element name="locked" type="xs:boolean"/>
      <xs:element name="unlockedUntil" type="xs:dateTime" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="eppCmdCount" type="xs:positiveInteger" minOccurs="0" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
END
```

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 [RFC3688]. The following URI assignment is requested of IANA:

Registration request for the registryLock namespace:

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

Registration request for the registryLock XML schema:

URI: urn:ietf:params:xml:schema:epp:registryLock-1.0
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 the IANA in the EPP Extension Registry described in [RFC7451]. The details of the registration are as follows:

Name of Extension: "Registry Lock Extension for the Extensible Provisioning Protocol (EPP)"

Document status: Standards Track

Reference: (insert reference to RFC version of this document)

Registrant Name and Email Address: IESG, <iesg@ietf.org>

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

7. Implementation Status

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

Implemented by .SE since 2019.

8. Security Considerations

The security properties of EPP from [RFC5730] are preserved.

This extensions introduces an additional security layer for changes of objects managed through EPP. The overall security of these measures depends on the security of the out-of-band authorization. Registries and registrars are therefore advised to select secure forms of authorization.

Current EPP authorizations schemes are not secure enough to allow in-band authorization. Registries and registrars therefore MUST not implment in-band command authorization.

9. Acknowledgements

The authors wish to thank the following persons for their feedback and suggestions:

10. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<https://www.rfc-editor.org/info/rfc5730>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <<https://www.rfc-editor.org/info/rfc5731>>.
- [RFC5732] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Host Mapping", STD 69, RFC 5732, DOI 10.17487/RFC5732, August 2009, <<https://www.rfc-editor.org/info/rfc5732>>.
- [RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.17487/RFC5733, August 2009, <<https://www.rfc-editor.org/info/rfc5733>>.
- [RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <<https://www.rfc-editor.org/info/rfc7451>>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.

[W3C.REC-xmlschema-2-20041028] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, 28 October 2004, <<https://www.w3.org/TR/2004/REC-xmlschema-2-20041028>>.

Appendix A. Change History

A.1. Change from 00 to 01

1. Corrected information for the <create/> command.
2. Minor fixes in wording.
3. Introduces resData element.

A.2. Change from 01 to 02

1. Multiple spelling errors fixed.
2. Moved response from resData to extension part of the EPP response.
3. Clarification of password and out-of-band usage.
4. Updated XML schema and examples
5. Changed security considerations for password authorization.
6. Added unlockUntil to create command
7. Forbid temporarily unlock for password authorization.

A.3. Change from 02 to 03

1. Fix list styles for better readability
2. Fix reference to W3C XML Schema

A.4. Change from 03 to 04

1. Remove references to in-band authorization
2. Remove special response elements
3. Add command counter to temporary unlock
4. Fix formatting and XML schema

Author's Address

Ulrich Wisser
The Swedish Internet Foundation
Box 92073
SE-12007 Stockholm
Sweden

Email: ulrich@wisser.se
URI: <https://www.internetstiftelsen.se>