Abstract

The Registration Data Access Protocol (RDAP) includes a method that can be used to identify the authoritative server for processing domain name, IP address, and autonomous system number queries. The method does not describe how to identify the authoritative server for processing other RDAP query types, such as entity queries. This limitation exists because the identifiers associated with these query types are typically unstructured. This document describes an operational practice that can be used to add structure to RDAP identifiers that makes it possible to identify the authoritative server for additional RDAP queries.

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 September 27, 2018.

Copyright Notice

Copyright (c) 2018 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.
1. Introduction

The Registration Data Access Protocol (RDAP) includes a method ([RFC7484]) that can be used to identify the authoritative server for processing domain name, IP address, and autonomous system number (ASN) queries. This method works because each of these data elements is structured in a way that facilitates automated parsing of the element and association of the data element with a particular RDAP service provider. For example, domain names include labels (such as "com", "net", and "org") that are associated with specific service providers.

As noted in Section 9 of RFC 7484 [RFC7484], the method does not describe how to identify the authoritative server for processing entity queries, name server queries, help queries, or queries using certain search patterns. This limitation exists because the identifiers bound to these queries are typically not structured in a way that makes it easy to associate an identifier with a specific service provider. This document describes an operational practice that can be used to add structure to RDAP identifiers that makes it...
Tagging object identifiers with a service provider tag makes it possible to identify the authoritative server for processing an RDAP query using the method described in RFC 7484 [RFC7484]. A service provider tag is constructed by prepending the Unicode TILDE character "~" (U+007E, described as an "unreserved" character in RFC 3986 [RFC3986]) to an IANA-registered value that represents the service provider. For example, a tag for a service provider identified by the string value "ARIN" is represented as "~ARIN".

Service provider tags are concatenated to the end of RDAP query object identifiers to unambiguously identify the authoritative server for processing an RDAP query. Building on the example from Section 3.1.5 of RFC 7482 [RFC7482], an RDAP entity handle can be constructed that allows an RDAP client to bootstrap an entity query. The following identifier is used to find information for the entity associated with handle "XXXX" at service provider "ARIN":

XXXX~ARIN

Clients that wish to bootstrap an entity query can parse this identifier into distinct handle and service provider identifier elements. Handles can themselves contain TILDE characters; the service provider identifier is found following the last TILDE character in the tagged identifier. The service provider identifier is used to retrieve a base RDAP URL from an IANA registry. The base URL and entity handle are then used to form a complete RDAP query path segment. For example, if the base RDAP URL "https://example.com/rdap/" is associated with service provider "YYYY" in an IANA registry, an RDAP client will parse a tagged entity identifier "XXXX~YYYY" into distinct handle ("XXXX") and service provider ("YYYY") identifiers. The service provider identifier "YYYY" is used to query an IANA registry to retrieve the base RDAP URL "https://example.com/rdap/". The base RDAP URL is concatenated to the entity handle to create a complete RDAP query path segment of "https://example.com/rdap/entity/XXXX~YYYY".

Implementation of this practice requires tagging of unstructured potential query identifiers in RDAP responses. Consider these elided examples from Section 5.3 of RFC 7483 [RFC7483] in which the handle identifiers have been tagged with a service provider tag:

```json
{
    "objectClassName" : "domain",

```
"handle" : "XXXX-RIR",
"ldhName" : "0.2.192.in-addr.arpa",
"nameservers" :
[ ...
],
"secureDNS":
{ ...
},
"remarks" :
[ ...
],
"links" :
[ ...
],
"events" :
[ ...
],
"entities" :
[ {
   "objectClassName" : "entity",
   "handle" : "XXXX-RIR",
   "vcardArray":
   [ ...
 ],
   "roles" : [ "registrant" ],
   "remarks" :
   [ ...
 ],
   "links" :
   [ ...
 ],
   "events" :
   [ ...
 ]
}]

"network" :
"objectClassName": "ip network",
"handle": "XXXX~RIR",
"startAddress": "192.0.2.0",
"endAddress": "192.0.2.255",
"ipVersion": "v4",
"name": "NET-RTR-1",
"type": "DIRECT ALLOCATION",
"country": "AU",
"parentHandle": "YYYY~RIR",
"status": [ "active" ]
}
}

Figure 1

{
"objectClassName": "domain",
"handle": "XXXX~DNR",
"ldhName": "xn--fo-5ja.example",
"unicodeName": "foo.example",
"variants": [
...
],
"status": [ "locked", "transfer prohibited" ],
"publicIds": [
...
],
"nameservers": [
{
  "objectClassName": "nameserver",
  "handle": "XXXX~DNR",
  "ldhName": "ns1.example.com",
  "status": [ "active" ],
  "ipAddresses": [
    ...
  ],
  "remarks": [
    ...
  ],
  "links": [
    ...
  ],
}
}
"events" :
[
  ...
],
"objectClassName" : "nameserver",
"handle" : "XXXX-DNR",
"ldhName" : "ns2.example.com",
"status" : [ "active" ],
"ipAddresses" :
[
  ...
],
"remarks" :
[
  ...
],
"links" :
[
  ...
],
"events" :
[
  ...
]
},
"secureDNS":
{
  ...
},
"remarks" :
[
  ...
],
"links" :
[
  ...
],
"port43" : "whois.example.net",
"events" :
[
  ...
],
"entities" :
[
  ["..."],
  ["..."]]}
"objectClassName": "entity",
"handle": "XXXX~ABC",
"vcardArray": [
...],
"status": [ "validated", "locked" ],
"roles": [ "registrant" ],
"remarks": [
...],
"links": [
...],
"events": [
...]
}
}

Figure 2

As described in Section 5 of RFC 7483 [RFC7483], RDAP responses can contain "self" links. Service provider tags and self references SHOULD be consistent. If they are inconsistent, the service provider tag is processed with higher priority when using these values to identify a service provider.

There is a risk of unpredictable processing behavior if the TILDE character is used for naturally occurring, non-separator purposes in an entity handle. This could lead to a client mistakenly assuming that a TILDE character represents a separator and the text that follows TILDE is a service provider identifier. A client that queries the IANA registry for what they assume is a valid service provider will likely receive an unexpected invalid result. As a consequence, the TILDE character MUST NOT be used in an entity handle for any purpose other than to separate an object identifier from a service provider tag.

The TILDE character was chosen as a separator for two reasons: 1) to avoid collisions with characters that are commonly found in entity handles, and 2) to avoid collisions with URI-reserved characters. The list of unreserved characters specified in Section 2.3 of RFC 3986 [RFC3986] provided multiple options for consideration as follows:
unreserved = ALPHA / DIGIT / "-" / "." / "_" / "~"

ALPHA and DIGIT characters were excluded because they are commonly used in entity handles. The "-" (HYPHEN MINUS, U+002D) and "_" (LOW LINE, U+005F) characters were also excluded as a result of being observed in current operational use. The TILDE character was chosen over the "." (FULL STOP, U+002E) character due to the authors’ belief that it is less likely to be in use in entity handles as of the time of this writing.

3. Bootstrap Service Registry for RDAP Service Providers

The bootstrap service registry for the RDAP service provider space is represented using the structure specified in Section 3 of RFC 7484 [RFC7484]. The JSON output of this registry contains alphanumeric identifiers that identify RDAP service providers, grouped by base RDAP URLs, as shown in this example.

```json
{
    "version": "1.0",
    "publication": "YYYY-MM-DDTHH:MM:SSZ",
    "description": "RDAP service provider bootstrap values",
    "services": [
        [
            "YYYY",
            [
                "https://example.com/rdap/"
            ]
        ],
        [
            "ZZ54",
            [
                "http://rdap.example.org/"
            ]
        ],
        [
            "1754",
            [
                "https://example.net/rdap/",
                "http://example.net/rdap/"
            ]
        ]
    ]
}
```

Figure 3
Alphanumeric service provider identifiers conform to the syntax specified in the IANA registry of Extensible Provisioning Protocol (EPP) Repository Identifiers [1].

3.1. Registration Procedure

The service provider registry is populated using the "First Come First Served" policy defined in RFC 5226 [RFC5226]. Provider identifier values can be derived and assigned by IANA on request. Registration requests include the requested service provider identifier (or an indication that IANA should assign an identifier) and one or more base RDAP URLs to be associated with the service provider identifier.

4. IANA Considerations

IANA is requested to create the RDAP Bootstrap Services Registry listed below and make it available as JSON objects. The contents of this registry is described in Section 3, with the formal syntax specified in Section 10 of RFC 7484 [RFC7484].

4.1. Bootstrap Service Registry for RDAP Service Providers

Entries in this registry contain at least the following:

- An alphanumeric value that identifies the RDAP service provider being registered.
- One or more URLs that provide the RDAP service regarding this registration.

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. 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. Client authentication is required to receive entity information in query responses.
Level of Maturity: This is a "proof of concept" research implementation.
Coverage: This implementation includes all of the features described in this specification.
Contact Information: Scott Hollenbeck, shollenbeck@verisign.com

5.2. OpenRDAP

Responsible Organization: OpenRDAP
Location: https://www.openrdap.org
Description: RDAP client implementing bootstrapping for entity handles with a service provider tag. A test Bootstrap Services Registry file is currently used in lieu of an official one.
Level of Maturity: Alpha
Coverage: Implements draft 04+, supports the TILDE separator character only.
Contact Information: Tom Harwood, tfh@skip.org

6. Security Considerations

This practice helps to ensure that end users will get RDAP data from an authoritative source using a bootstrap method to find authoritative RDAP servers, reducing the risk of sending queries to non-authoritative sources. The method has the same security properties as the RDAP protocols themselves. The transport used to access the IANA registries can be more secure by using TLS [RFC5246], which IANA supports. Additional considerations associated with RDAP are described in RFC 7481 [RFC7481].
7. Acknowledgements

The author would like to acknowledge the following individuals for their contributions to the development of this document: Tom Harrison, and Marcos Sanz. In addition, the authors would like to recognize the Regional Internet Registry (RIR) operators (AFRINIC, APNIC, ARIN, LACNIC, and RIPE) that have been implementing and using the practice of tagging handle identifiers for several years. Their experience provided significant inspiration for the development of this document.

8. References

8.1. Normative References


8.2. Informative References


8.3. URIs


Appendix A. Change Log

00: Initial version.
01: Changed separator character from HYPHEN MINUS to COMMERCIAL AT.
   Added a recommendation to maintain consistency between service
   provider tags and "self" links (suggestion received from Tom
   Harrison). Fixed a spelling error, and corrected the network
   example in Section 2 (editorial erratum reported for RFC 7483 by
   Marcos Sanz). Added acknowledgements.
02: Changed separator character from COMMERCIAL AT to TILDE.
   Clarity updates and fixed an example handle. Added text to
   describe the risk of separator characters appearing naturally in
   entity handles and being misinterpreted as separator characters.
03: Added Implementation Status section (Section 5).
04: Keepalive refresh.
05: Added OpenRDAP implementation information to Section 5.
00: Initial working group version.
01: Added text to describe why the TILDE character was chosen as the
   separator character.

Authors’ Addresses

Scott Hollenbeck
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
USA

Email: shollenbeck@verisign.com
URI: http://www.verisignlabs.com/
Andrew Lee Newton
American Registry for Internet Numbers
PO Box 232290
Centreville, VA 20120
US

Email: andy@arin.net
URI: http://www.arin.net