An RDAP With Extensions Media Type

Abstract

This document defines a media type for RDAP that can be used to describe RDAP content with RDAP extensions. Additionally, this document describes the usage of this media type with RDAP.

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 August 2024.

Copyright Notice

Copyright (c) 2024 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. Background

[RF7480] defines the 'application/rdap+json' media type to be used with RDAP. This document defines a new media type to be used in conjunction with the current media type when an RDAP extension needs to be described during HTTP content negotiation.

2. RDAP-X: The RDAP With Extensions Media Type

The media type defined by this document is 'application/rdap-x+json'. This media type has a parameter of "extensions" which is a whitespace-separated list of RDAP extensions as defined in the IANA RDAP Extensions registry.

Here is an example:

application/rdap-x+json;extensions="rdap_level_0 fred"

For readability, this document will refer to this media type, RDAP With Extensions, as RDAP-X.

3. Using The RDAP-X Media Type

[RF7480] specifies the usage of 'application/json', 'application/rdap+json' or both with HTTP accept header. When using the media
type defined by this document, the 'application/rdap+json' media type MUST also be used in the accept header.

An example:

```
accept: application/rdap+json;q=0.9,
    application/rdap-x+json;extensions="rdap_level_0 rdapx fred";q=1
```

When a server is programmed to understand the RDAP-X media type, it SHOULD respond with this media type in the content-type header. By doing so, clients will be able to detect if the server recognizes the media type. Otherwise, the server will use the 'application/rdap+json' media type signalling to the client that the RDAP-X media type is not recognized by the server. This updates the usage of the content-type header with RDAP defined in RFC 7480, but this usage is backward compatible.

If both a client and server support the RDAP-X media type, and the client requests an extension that is unimplemented by the server, the server SHOULD respond with the RDAP-X media type using only extensions implemented by the server. This behavior is backward compatible as RDAP clients must ignore unknown extensions as specified by [RFC9083]. Responding with an HTTP 406 Not Acceptable status code is NOT RECOMMENDED.

When the RDAP-X media type is used in the content-type header, the values in the media type's extension parameter SHOULD match the values in the rdapConformance array in the return JSON. When there is a mismatch between extension parameters and the rdapConformance array, clients SHOULD give preference to the rdapConformance array.

Just as servers should not put extensions into the rdapConformance array for which they do not support, servers SHOULD NOT list extensions in the RDAP-X media type for which they do not support.

Nothing in this specification sidesteps or obviates the HTTP content negotiation defined in [RFC9110] for RDAP. Specifically, if a client gives RDAP-X a lower qvalue than any other media type, that is a signal not to use RDAP-X.

Likewise, nothing in this specification sidesteps or obviates the HTTP caching mechanisms defined in [RFC9110]. Further advice on the vary header may be found in Appendix A.

Some RDAP extensions, such as [I-D.ietf-regext-rdap-openid], have other protocol elements passed from the client to the server, and the presence of these protocol elements may be used by servers to determine a client's capability to handle the RDAP extension. This specification does not require the usage of those extensions.
identifiers in the extensions parameter, though clients SHOULD list the extension identifier in the extensions parameter when using other protocol elements of those extensions. Servers SHOULD NOT require the usage of extension identifiers in the extensions parameter when other extension protocol elements are used.

4. Usage in RDAP Links

Section 4.2 of [RFC9083] defines a link structure used in RDAP.

```
{
  "value" : "https://example.com/context_uri",
  "rel" : "self",
  "href" : "https://example.com/target_uri",
  "hreflang" : [ "en", "ch" ],
  "title" : "title",
  "media" : "screen",
  "type" : "application/json"
}
```

The type attribute signals to a client the expected media type of the resource referenced in the href attribute, and some clients use this information to determine if the URI in the href attribute should be dereferenced.

Servers MAY use the RDAP-X media type in the type attribute if a client has negotiated content with the server using the RDAP-X media type, the resource referenced by the URI matches the RDAP-X media type, and the resource referenced by the URI is served by a server compliant with this specification. Otherwise, use of the application/rdap+json media type is RECOMMENDED when the URI references RDAP resources.

5. RDAP-X Extension

This document defines an RDAP "profile" extension using the identifier "rdapx" (hyphen characters are not allowed in RDAP extension identifiers). This RDAP extension defines no additional RDAP queries or response structures.

The purpose of this RDAP extension is to allow servers to signal support for RDAP-X in rdapConformance arrays of responses to /help (aka "service discovery").

6. Security Considerations

As stated in Section 3, this specification does not override the protocol elements of RDAP security extensions, such as
This specification does contrast with solutions using query parameters in that those solutions require servers to blindly copy query parameters into redirect URLs in situations where such copying could cause harm, such as copying an API key intended for one server into the redirect URL of another server.

7. IANA Considerations

Type name: application

Subtype name: rdap-x+json

Required parameters: This media type has a parameter of "extensions" which is a whitespace-separated list of RDAP extensions as defined in the IANA RDAP Extensions registry.

Optional parameters: N/A

Encoding considerations: See Section 3.1 of [RFC6839].

Security considerations: The media represented by this identifier does not have security considerations beyond that found in Section 12 of [RFC8259].

Interoperability considerations: There are no known interoperability problems regarding this media format.

Published specification: This document.

Applications that use this media type: Implementations of the Registration Data Access Protocol (RDAP) with Extensions.

Additional information: This media type is a product of the IETF REGEXT Working Group. The REGEXT charter, information on the REGEXT mailing list, and other documents produced by the REGEXT Working Group can be found at https://datatracker.ietf.org/wg/regext/.

Person & email address to contact for further information: IESG <iesg@ietf.org>

Intended usage: COMMON

Restrictions on usage: none

Author: Andy Newton

Change controller: IETF
Provisional Registration: No

8. Acknowledgements

Pawel Kowalik and James Mitchell have provided ideas and feedbacks that have contributed to the content of this document.

9. References

9.1. Normative References


9.2. Informative References


Appendix A. Using the Vary Header

Server implementers may want to consider using the vary header depending on the caching behavior desired of shared caches (i.e. middleboxes, not client caches).

Consider the following scenario where user Bob and user Alice send queries to the same RDAP server that is routed through a middlebox network element implementing a shared HTTP cache.

User Bob sends a query for the domain example.com (http://regy.example/domain/example.com) without RDAP-X. The accept header sent for Bob's query would be accept: application/rdap+json or accept: application/json.
User Alice later sends a query for the same domain, however her client uses RDAP-X. The accept header returned by Alice might be accept: application/rdap-x+json, application/rdap+json.

If no vary header is set in the response for these queries, the shared cache will compare only the URL of the query when processing cache items and therefore user Bob and user Alice would receive the same answer. In other words, since both queried http://regy.example/domain/example.com the shared cache would return the answer of the first query to the second query and all other subsequent queries until the item expired out of the cache.

If server implementers do not desire this behaviour and would signal that caches consider each query separately, servers should also return a vary: accept header to inform the cache that the accept header should also be considered when processing cache items. Server implementers should also consult [RFC9110] regarding caching and other uses of the vary header.

Appendix B. Design Considerations

B.1. Not Reusing the Existing Media Type

Section 4.3 of [RFC6838] strongly discourages the creation of new parameters on existing media types to enable new features. As RDAP has always had extensions, it could be argued that adding an "extensions" parameter to the existing application/rdap+json media type is not adding a new feature to RDAP. However, the opposite could be argued that adding the capability for clients to signal desired RDAP extensions is a new feature.
More practically, there is concern that adding a new parameter to the existing media type would not be backward compatible with some server software. That is, servers examining media types as exact string matches may incorrectly conclude that the existing media type with an unknown, new parameter may not be the same as the existing media type without parameters. A similar, though less likely, concern exists for clients.

As servers are required to handle multiple media types according to [RFC7480] and [RFC9110], it therefore seems reasonable to conclude that defining a new media type for use with the existing media type is best to preserve backward compatibility.

B.2. Query Parameters Considered Harmful

Another design approach to communicating RDAP extensions from the client to the server would be the use of URI query parameters:

https://rdap.example/domain/foo.example?extensions=fizzbuzz

B.2.1. Copy and Paste

Consider two RDAP users, Alice and Bob. Alice has an RDAP client that supports the extensions "fizzbuzz", and Bob has an RDAP client that does not support this extension.

Now consider the scenario where Alice copies and pastes the RDAP URL from above into an email and sends it to Bob. When Bob uses that URL with his RDAP client, it will be communicating to the server that the extension "fizzbuzz" is understood by Bob's client when it is not.

In this scenario, Bob's client will be unable to render the RDAP extension regardless of the usage or not of the query parameter. However, if the server is using the query parameter for secondary purposes, such as gathering metrics and statistics, then the capabilities of Bob's client will have been incorrectly signalled to the server.

B.2.2. Redirects

The RDAP ecosystem uses redirects in many situations. [RFC7480] discusses "aggregators", which are RDAP servers used to help clients find authoritative RDAP servers using the RDAP bootstrap registries. Redirects are also heavily used by the RIRs when IP addresses or autonomous system numbers are transferred from one RIR to another.

Within HTTP, URI query parameters are not explicitly preserved during a redirect (probably due to architecture considerations, see
the section below). Specific to RDAP, [RFC7480] instructs RDAP servers to ignore unknown query parameters and instructs clients not to transform a URL of a redirect.

Therefore, query parameters denoting RDAP extensions will not survive redirects. This can be readily observed in currently deployed RDAP servers:

curl -v https://rdap-bootstrap.arin.net/bootstrap/autnum/2830?extension=

To further demonstrate that query parameters do not automatically survive redirects but that media types do, consider the code found here. This code consists of a simple client and a simple server. The client sets both a new media type and query parameters. The servers listens on two ports, redirecting the client from a URL on the first port to a URL on the second port.

Here is the output of the client. It shows that the query parameters are not automatically preserved but that the media type is automatically preserved.

```
2024-01-05T11:34.380989Z INFO client: sending request to http://127.
2024-01-05T11:34.431386Z INFO client: returned content type: "application/rdap-x;extensions="foo bar"
2024-01-05T11:34.431413Z INFO client: status code is 418 I'm a teapot
2024-01-05T11:34.431476Z INFO client: response is {"errorCode":418,"
```

Here is the output of the server. It show that the client, upon redirect, automatically sends the media type but does not automatically preserve the query parameters.

```
2024-01-05T11:15:31.071936Z INFO servers: starting server on port 3000
2024-01-05T11:15:31.071961Z INFO servers: starting server on port 4000
2024-01-05T11:15:34.431476Z INFO servers: [redirecting server] Serving
2024-01-05T11:15:34.431476Z INFO servers: [redirecting server] received
2024-01-05T11:15:34.431476Z INFO servers: [redirecting server] client signaled RDAP extension 'foo'
2024-01-05T11:15:34.431476Z INFO servers: [redirecting server] client signaled RDAP extension 'bar'
2024-01-05T11:15:34.431476Z INFO servers: [authoritative server] Serving
2024-01-05T11:15:34.431476Z INFO servers: [authoritative server] received
2024-01-05T11:15:34.431476Z INFO servers: [authoritative server] client signaled RDAP extension 'foo'
2024-01-05T11:15:34.431476Z INFO servers: [authoritative server] client signaled RDAP extension 'bar'
2024-01-05T11:15:34.431476Z INFO servers: [authoritative server] responding with an unuseful error
```

Preservation of query parameters is not a guaranteed feature of HTTP client and server libraries, whereas preservation of media types is.
B.2.3. Referral Compatibility

It is common in the RDAP ecosystem to link from one RDAP resource to another. These are typically conveyed in the link structure defined in Section 4.2 of [RFC9083] and use the "application/rdap+json" media type. One common usage is to link to a domain registration in a domain registrar from a domain registration in a domain registry.

```json
{
  "value" : "https://regy.example/domain/foo.example",
  "rel" : "related",
  "href" : "https://regr.example/domain/foo.example",
  "type" : "application/rdap+json"
}
```

Usage of the RDAP-X media type does not require clients to conduct further processing of these referrals, whereas a query parameter approach would require clients to process and deconflict any other query parameters if present.

B.2.4. Architectural Violations

As noted in [RFC3986], URI query parameters are meant to be part of the identity of the resource being identified by a URI and pointed to by the location of a URL. RDAP extensions change the portions of JSON returned by the server but are not intended to change the resource being identified. That is, a domain registration is the same domain registration regardless of whether the postal address in that domain registration is communicated via JCard or a new RDAP extension for JSContact.

Changing how the content of a resource is conveyed is called content negotiation and is discussed in detail in [RFC9110] using media types.

Readers should note that protocol design is not a "priestly affair" in which architectural violations are strictly forbidden. Every design decision is a trade-off. However, following the architecture of an ecosystem generally makes re-use of software and systems easier, and often eases the adoption of newer features in the future. When given the choice between two designs, the design that does not violate architecture should be preferred when all other considerations are equal.

B.3. RDAP Extension Versioning

It is beyond the scope of this document to define the versioning of RDAP extensions. However, there is design intent to allow the use of explicitly versioned RDAP extension identifiers where they are also
compatible with the identifiers used in the rdapConformance array of RDAP.

Consider the scenario in which the IETF decides that RDAP extension identifiers suffixed with the character string __V denotes RDAP extensions versioned using a semantic versioning scheme. In this scenario, the RDAP extension identifier fizzbuzz__V is registered with IANA. The __V suffix indicates that when the identifier is used in the rdapConformance array, it must appear appended with a character string denoting the semantic version of the extension.

For example, fizzbuzz__V_2_1 denotes version 2.1 of the fizzbuzz extension. In RDAP JSON, the conformance would appear as:

```
"rdapConformance" : [
   "rdap_level_0",
   "fizzbuzz__V_2_1"
]
```

The usage with the rdap-x media type would be:

```
application/rdap-x+json;extensions="rdap_level_0 fizzbuzz__V_2_1"
```

Readers should note that this scenario is provided to show design intent and is not a full-fledged extension versioning design. Additionally, the new media type defined in this document has utility with existing, opaquely versioned RDAP extensions and does not depend on the definition of a new versioning scheme for RDAP extensions.

Authors' Addresses

Andy Newton
ICANN

Email: andy@hxr.us

Jasdip Singh
ARIN

Email: jasdips@arin.net