Workgroup:	Network Working Group
Internet-Draft:	draft-wullink-rpp-core-01
Published:	20 July 2025
Intended Status:	Standards Track
Expires:	21 January 2026
---	---	---
Authors:	M. Wullink	P. Kowalik
	SIDN Labs	*DENIC*

RESTful Provisioning Protocol (RPP)

Abstract

This document describes the endpoints for the RESTful Provisioning Protocol, used for the provisioning and management of objects in a shared database.

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 21 January 2026.

Copyright Notice

Table of Contents

1. Introduction

This document describes an Application Programming Interface (API) API based on the HTTP protocol [RFC2616] and the principles of [REST]. Conforming to the REST constraints is generally referred to as being "RESTful". Hence the API is dubbed: "'RESTful Provisioning Protocol" or "RPP" for short.

RPP is data format agnostic, this document describes a framework describing protocol messages in any data format. the client uses server-driven content negotiation. Allowing the client to select from a set of representation media types supported by the server, such as JSON [RFC8259], XML or [YAML].

2. Terminology

In this document the following terminology is used.

REST - Representational State Transfer ([REST]). An architectural style.

RESTful - A RESTful web service is a web service or API implemented using HTTP and the principles of [REST].

EPP RFCs - This is a reference to the EPP version 1.0 specifications [RFC5730], [RFC5731], [RFC5732] and [RFC5733].

RESTful Provisioning Protocol or RPP - The protocol described in this document.

URL - A Uniform Resource Locator as defined in [RFC3986].

Resource - An object having a type, data, and possible relationship to other resources, identified by a URL.

RPP client - An HTTP user agent performing an RPP request

RPP server - An HTTP server responsible for processing requests and returning results in any supported media type.

3. 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].

In examples, lines starting with "C:" represent data sent by a RPP client and lines starting with "S:" represent data returned by a RPP server. Indentation and white space in examples are provided only to illustrate element relationships and are not REQUIRED features of the protocol.

All example requests assume a RPP server using HTTP version 2 is listening on the standard HTTPS port on host rppp.example.nl. An authorization token has been provided by an out of band process and MUST be used by the client to authenticate each request.

4. Request Headers

A RPP request does not always require a request message body. The information conveyed by the HTTP method, URL, and request headers may be sufficient for the server to be able to successfully processes a request. However, the client MUST include a request message body when the server requires additional attributes to be present in the request message. The RPP HTTP headers listed below use the "RPP-" prefix, following the recommendations in [RFC6648].

- RPP-Cltrid: The client transaction identifier is the equivalent of the clTRID element defined in [RFC5730] and MUST be used accordingly, when the HTTP message body does not contain an EPP request that includes a cltrid.
- RPP-Authorization: The client MAY use this header to send authorization information in the format <method> <authorization information>, similar to the HTTP Authorization header. The <method> indicates the type of authorization being used. The authorization information MUST be defined as a structured field (dictionary) as described in [RFC8941]. For the EPP Authorization Information using plain password the following method is defined and MUST be used: <method> = AuthInfo The <authorization information> for AuthInfo method defines the following fields:
  - AuthInfo (REQUIRED): base64 encoded EPP authorization information. Base64 encoding is used to prevent potential problems when non-ascii characters or other characters are present that may conflict with the format rules for structured field.
  - Roid (OPTIONAL): A Roid as defined in [RFC5731], [RFC5733], and [RFC5730]. This field MAY be REQUIRED in the usage context as defined in [RFC5731], [RFC5733], and [RFC5730].
- RPP-Authorization: The client MAY use this header to send authorization information in the format <method> <authorization information>, similar to the HTTP Authorization header. The <method> indicates the type of authorization being used. The authorization information MUST be defined as a structured field (dictionary) as described in [RFC8941]. For the EPP AuthInfo method this format MUST be used: AuthInfo=<Base64(AuthInfo)>; Roid=<Roid>, where AuthInfo is REQUIRED and MUST be encoded using base64, to prevent potential problems when non-ascii characters or

other characters are present that may conflict with the format rules for structured field. The Roid is OPTIONAL unless required by the context (as described in [RFC5731], [RFC5733], and [RFC5730]). For other methods, the authorization information format is method-specific and may not use key/value pairs unless otherwise specified.

5. Response Headers

The server HTTP response contains a status code, headers, and MAY contain an RPP response message in the message body. HTTP headers are used to transmit additional data to the client and MAY be used to send RPP process related data to the client. HTTP headers used by RPP MUST use the "RPP-" prefix, the following response headers have been defined for RPP.

- RPP-Svtrid: This header is the equivalent of the "svTRID" element defined in [RFC5730] and MUST be used accordingly when the RPP response does not contain an EPP response in the HTTP message body. If an HTTP message body with the EPP XML equivalent "svTRID" exists, both values MUST be consistent.
- RPP-Cltrid: This header is the equivalent of the "clTRID" element defined in [RFC5730] and MUST be used accordingly when the RPP response does not contain an EPP response in the HTTP message body. If the contents of the HTTP message body contains a "clTRID" value, then both values MUST be consistent.
- RPP-Code: This header is the equivalent of the EPP result code defined in [RFC5730] and MUST be used accordingly. This header MUST be added to all responses and MAY be used by the client for easy access to the result code, without having to parse the HTTP response message body.

For the EPP codes related to session management (1500, 2500, 2501 and 2502) there are no corresponding RPP codes.

In order for RPP to be backwards compatible with EPP, RPP will use 5-digit coding of the result codes, where first digit will denote origin specification of the result codes.

For [RFC5730] Result Codes the leading digit MUST be "0". For RPP result codes the leading digit MUST be "1". For avoidance of confusion RPP MUST not define new codes with the same semantic meaning as already defined in EPP.

For RPP codes the remaining 4 digits MUST keep the same semantics as [RFC5730] Result Codes.

- RPP-Queue-Size: Return the number of unacknowledged messages in the client message queue. The server MAY include this header in all RPP responses.

6. Error handling and relation between HTTP status codes and RPP codes

RPP leverages standard HTTP status codes to reflect the outcome of RPP operations. The RPP result codes are based on the EPP result codes defined in [RFC5730]. This allows clients to handle responses generically using common HTTP patterns. While the HTTP status code provides the primary, high-level outcome, the specific RPP result code MUST still be provided in the RPP-Code HTTP header for detailed diagnostics.

The mapping strategy is to use the most specific HTTP code that accurately reflects the operation's result.

For common and well-defined outcomes, a specific HTTP status code is used. For example, an attempt to access a non-existent resource (EPP code 2302) MUST return 404 Not Found, and an attempt to create a resource that already exists (EPP code 2303) MUST return 409 Conflict. This allows a client to handle these common situations based on the HTTP code alone.

For all other failures, a generic HTTP status code is used. Client-side errors (e.g., syntax, parameter, or policy violations) MUST return 400 Bad Request. Server-side failures MUST return 500 Internal Server Error.

The server MUST return HTTP status codes, following the mapping rules in Table 1.

Table 1: RPP result code and HTTP Status-Code mapping.

HTTP Status-Code	**Description**	**Corresponding RPP result code(s)**
Success (2xx)		
200 OK	The request was successful (e.g., for GET or UPDATE).	01000 (in all cases not specified otherwise),01300,01301
201 Created	The resource was created successfully.	01000 for resource creating requests (POST/PUT)
202 Accepted	The request was accepted for asynchronous processing.	01001
204 No Content	The resource was deleted successfully.	01000 for DELETE
Client Errors (4xx)		
400 Bad Request	Generic client-side error (syntax, parameters, policy).	02000-02005,02104-02106,02300-02301,02304-02308

HTTP Status-Code	Description	Corresponding RPP result code(s)
403 Forbidden	Authentication or authorization failed.	02200-02202
404 Not Found	The requested resource does not exist.	02303
409 Conflict	The resource could not be created because it already exists.	02302
Server Errors (5xx)		
500 Internal Server Error	Generic server-side error; command failed.	02400
501 Not Implemented	The requested command or feature is not implemented.	02100-02103

Table 1

Some EPP result codes, like 01500, 02500, 02501 and 02502 are related to session management and therefore not applicable to a sessionless RPP protocol.

7. Endpoints

subsequent sections provide details for each endpoint. URLs are assumed to be using the prefix: "/{context-root}/{version}/". Some RPP endpoints do not require a request and/or response message.

{c}: An abbreviation for {collection}: this MUST be substituted with "domains", "hosts", "entities" or any other collection of objects. {i}: An abbreviation for an object id, this MUST be substituted with the value of a domain name, hostname, contact-id or a message-id or any other defined object.

A RPP client MAY use the HTTP GET method for executing informational request only when no request data has to be added to the HTTP message body. Sending content using an HTTP GET request is discouraged in [RFC9110], there exists no generally defined semantics for content received in a GET request. When an RPP object requires additional information, the client MUST use the HTTP POST method and add the query command content to the HTTP message body.

7.1. Availability for Creation

The Availability for Creation endpoint is used to check whether an object can be successfully provisioned. Two distinct methods are defined for checking the availability of provisioning of an object, the first method uses the HEAD method for a quick check to find out if the object can be provisioned. The second method uses the GET method to retrieve additional information about the object's availability for provisioning, for example about pricing or additional requirements to be able to provision the requested object.

When the client uses the HTTP HEAD method, the server MUST respond with an HTTP status code 200 (OK) if the object can be provisioned or with an HTTP status code 404 (Not Found) if the object cannot be provisioned.

When the client uses the HTTP GET method, the server MUST respond with an HTTP status code 200 (OK) if the object can be provisioned. The server MUST include a message body containing more detailed availability information, for example about pricing or additional requirements to be able to provision the requested object. The message body MAY be and empty JSON object if no additional information is applicable.

If the object cannot be provisioned then the server MUST return an HTTP status code 404 (Not Found) and include a problem statement in the message body.

As an extension point the server MAY define and the client MAY use additional HTTP query parameters to further specify the check operation or the kind of response information that shall be returned. For example Registry Fee Extension [RFC8748] defines a possibility to request certain currency, only certain commands or periods. Such functionality would add query parameters, which could be used with GET request to receive additional pricing information with the response. HEAD request would not be affected in this case.

The server MUST respond with the same HTTP status code if the same URL is requested with HEAD and with GET.

```
- Request: HEAD|GET /{collection}/{id}/availability
- Request message: None
- Response message: Optional availability response
```

Example request for a domain name that is not available for provisioning:

```
HEAD /rpp/v1/domains/example.nl HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept-Language: en
RPP-Cltrid: ABC-12345
```

Example response:

```
HTTP/2 404 Not Found
Date: Wed, 24 Jan 2024 12:00:00 UTC
Server: Example RPP server v1.0
RPP-Cltrid: ABC-12345
RPP-Svtrid: XYZ-12345
RPP-code: 01000
Content-Length: 0
```

7.2. Resource Information

The Object Info request MUST use the HTTP GET method on a resource identifying an object instance. If the object has authorization information attached then the client MUST use an empty message body and include the RPP-AuthInfo HTTP header. If the authorization is linked to a database object the client MUST also include the RPP-Roid header. The client MAY also use a message body that includes the authorization information, the client MUST then not use the RPP-AuthInfo and RPP-Roid headers.

- Request: GET /{collection}/{id}
- Request message: Optional
- Response message: Info response

Example request for an object not using authorization information.

```
GET /rpp/v1/domains/example.nl HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Accept-Language: en
RPP-Cltrid: ABC-12345
```

Example request using RPP-AuthInfo and RPP-Roid headers for an object that has attached authorization information.

```
GET /rpp/v1/domains/example.nl HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Accept-Language: en
RPP-Cltrid: ABC-12345
RPP-AuthInfo: secret-token
RPP-Roid: REG-XYZ-12345
```

Example Info response:

```
HTTP/2 200 OK
Date: Wed, 24 Jan 2024 12:00:00 UTC
Server: Example RPP server v1.0
Content-Length: 424
Content-Type: application/rpp+json
Content-Language: en
RPP-code: 01000

TODO: JSON message here
```

7.3. Poll for Messages

The messages endpoint is used for retrieving messages stored on the server for the client to process.

- Request: GET /messages
- Request message: None
- Response message: Poll response

The client MUST use the HTTP GET method on the messages resource collection to request the message at the head of the queue.

Example request:

```
GET /rpp/v1/messages HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Accept-Language: en
RPP-Cltrid: ABC-12345
```

Example response:

```
HTTP/2 200 OK
Date: Wed, 24 Jan 2024 12:00:00 UTC
Server: Example RPP server v1.0
Content-Length: 312
Content-Type: application/rpp+json
Content-Language: en
RPP-code: 01301

TODO
```

7.4. Delete Message

- Request: DELETE /messages/{id}
- Request message: None
- Response message: Poll Ack response

The client MUST use the HTTP DELETE method to acknowledge receipt of a message from the queue. The "msgID" attribute of a received RPP Poll message MUST be included in the message resource URL, using the {id} path element. The server MUST use RPP headers to return the RPP result code and the number of messages left in the queue. The server MUST NOT add content to the HTTP message body of a successful response, the server may add content to the message body of an error response.

Example request:

```
DELETE /rpp/v1/messages/12345 HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Accept-Language: en
RPP-Cltrid: ABC-12345
```

Example response:

```
HTTP/2 200 OK
Date: Wed, 24 Jan 2024 12:00:00 UTC
Server: Example RPP server v1.0
Content-Language: en
RPP-code: 01000
RPP-Queue-Size: 0
RPP-Svtrid: XYZ-12345
RPP-Cltrid: ABC-12345
Content-Length: 145

TODO
```

7.5. Create Resource

- Request: POST /{collection}
- Request message: Object Create request
- Response message: Object Create response

The client MUST use the HTTP POST method to create a new object resource. If the RPP request results in a newly created object, then the server MUST return HTTP status code 200 (OK). The server MUST add the "Location" header to the response, the value of this header MUST be the URL for the newly created resource.

Example Domain Create request:

```
POST /rpp/v1/domains HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Content-Type: application/rpp+json
Accept-Language: en
Content-Length: 220

TODO
```

Example Domain Create response:

```
HTTP/2 200
Date: Wed, 24 Jan 2024 12:00:00 UTC
Server: Example RPP server v1.0
Content-Language: en
Content-Length: 642
Content-Type: application/rpp+json
Location: https://rpp.example.nl/rpp/v1/domains/example.nl
RPP-code: 01000

TODO
```

7.6. Delete Resource

- Request: DELETE /{collection}/{id}
- Request message: Optional
- Response message: Status

The client MUST the HTTP DELETE method and a resource identifying a unique object instance. The server MUST return HTTP status code 200 (OK) if the resource was deleted successfully.

Example Domain Delete request:

```
DELETE /rpp/v1/domains/example.nl HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Accept-Language: en
RPP-Cltrid: ABC-12345
```

Example Domain Delete response:

```
HTTP/2 200 OK
Date: Wed, 24 Jan 2024 12:00:00 UTC
Server: Example RPP server v1.0
Content-Length: 80
RPP-Svtrid: XYZ-12345
RPP-Cltrid: ABC-12345
RPP-code: 01000

TODO
```

7.7. Renew Resource

- Request: POST /{collection}/{id}/renewal
- Request message: Optional
- Response message: Renew response

Not all EPP object types include support for the renew command. The current-date query parameter MAY be used for date on which the current validity period ends, as described in Section 3.2.3 of [RFC5731]. The new period MAY be added to the request using the unit and value request parameters. The response MUST include the Location header for the renewed object.

TODO:: current-date: can also be a HTTP header?

Example Domain Renew request:

```
POST /rpp/v1/domains/example.nl/renewal?current-date=2024-01-01 HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Content-Type: application/rpp+json
Accept-Language: en
Content-Length: 0
```

Example Domain Renew request, using 1 year period:

```
POST /rpp/v1/domains/example.nl/renewal?current-date=2024-01-01?unit=y&value=1
HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Content-Type: application/rpp+json
Accept-Language: en
Content-Length: 0
```

Example Renew response:

```
HTTP/2 200 OK
Date: Wed, 24 Jan 2024 12:00:00 UTC
Server: Example RPP server v1.0
Content-Language: en
Content-Length: 205
Location: https://rpp.example.nl/rpp/v1/domains/example.nl
Content-Type: application/rpp+json
RPP-code: 01000

TODO
```

7.8. Transfer Resource

The Transfer command is mapped to a nested resource, named "transfer". The semantics of the HTTP DELETE method are determined by the role of the client executing the DELETE method. The DELETE method is defined as "reject transfer" for the current sponsoring client of the object. For the new sponsoring client the DELETE method is defined as "cancel transfer".

7.8.1. Start

- Request: POST /{collection}/{id}/transfer
- Request message: Optional
- Response message: Status

In order to initiate a new object transfer process, the client MUST use the HTTP POST method on a unique resource to create a new transfer resource object. Not all RPP objects support the Transfer command.

If the transfer request is successful, then the response MUST include the Location header for the object being transferred.

Example request not using object authorization:

```
POST /rpp/v1/domains/example.nl/transfer HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Accept-Language: en
RPP-Cltrid: ABC-12345
Content-Length: 0
```

Example request using object authorization:

```
POST /rpp/v1/domains/example.nl/transfer HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
RPP-Cltrid: ABC-12345
RPP-AuthInfo: secret-token
Accept-Language: en
Content-Length: 0
```

Example request using 1 year renewal period, using the unit and value query parameters:

```
POST /rpp/v1/domains/example.nl/transfer?unit=y&value=1 HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Accept-Language: en
RPP-Cltrid: ABC-12345
Content-Length: 0
```

Example Transfer response:

```
HTTP/2 200 OK
Date: Wed, 24 Jan 2024 12:00:00 UTC
Server: Example RPP server v1.0
Content-Language: en
Content-Length: 328
Content-Type: application/rpp+json
Location: https://rpp.example.nl/rpp/v1/domains/example.nl/transfer
RPP-code: 01001

TODO
```

7.8.2. Status

A transfer object may not exist, when no transfer has been initiated for the specified object. The client MUST use the HTTP GET method and MUST NOT add content to the HTTP message body.

- Request: GET {collection}/{id}/transfer
- Request message: Optional
- Response message: Transfer Status response

Example domain name Transfer Status request without authorization information required:

```
GET /rpp/v1/domains/example.nl/transfer HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Accept-Language: en
RPP-Cltrid: ABC-12345
```

If the requested transfer object has associated authorization information that is not linked to another database object, then the HTTP GET method MUST be used and the authorization information MUST be included using the RPP-AuthInfo header.

Example domain name Transfer Query request using RPP-AuthInfo header:

```
GET /rpp/v1/domains/example.nl/transfer HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Accept-Language: en
RPP-Cltrid: ABC-12345
RPP-AuthInfo: secret-token
```

If the requested object has associated authorization information linked to another database object, then the HTTP GET method MUST be used and both the RPP-AuthInfo and the RPP-Roid header MUST be included.

Example domain name Transfer Query request and authorization using RPP-AuthInfo and the RPP-Roid header:

```
GET /rpp/v1/domains/example.nl/transfer HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Accept-Language: en
RPP-AuthInfo: secret-token
RPP-Roid: REG-XYZ-12345
Content-Length: 0
```

Example Transfer Query response:

```
HTTP/2 200 OK
Date: Wed, 24 Jan 2024 12:00:00 UTC
Server: Example RPP server v1.0
Content-Length: 230
Content-Type: application/rpp+json
Content-Language: en
RPP-code: 01000

TODO
```

7.8.3. Cancel

- Request: POST /{collection}/{id}/transfer/cancelation
- Request message: Optional
- Response message: Status

The new sponsoring client MUST use the HTTP POST method to cancel a requested transfer.

Example request:

```
POST /rpp/v1/domains/example.nl/transfer/cancelation HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Accept-Language: en
RPP-Cltrid: ABC-12345
```

Example response:

```
HTTP/2 200 OK
Date: Wed, 24 Jan 2024 12:00:00 UTC
Server: Example RPP server v1.0
Content-Length: 80
RPP-Svtrid: XYZ-12345
RPP-Cltrid: ABC-12345
RPP-code: 01000

TODO
```

7.8.4. Reject

- Request: POST /{collection}/{id}/transfer/rejection
- Request message: None
- Response message: Status

The currently sponsoring client of the object MUST use the HTTP POST method to reject a started transfer process.

Example request:

```
POST /rpp/v1/domains/example.nl/transfer/rejection HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Accept-Language: en
RPP-Cltrid: ABC-12345
```

Example Reject response:

```
HTTP/2 200 OK
Date: Wed, 24 Jan 2024 12:00:00 UTC
Server: Example RPP server v1.0
Content-Length: 80
RPP-Svtrid: XYZ-12345
RPP-Cltrid: ABC-12345
RPP-code: 01000

TODO
```

7.8.5. Approve

- Request: POST /{collection}/{id}/transfer/approval
- Request message: Optional
- Response message: Status

The currently sponsoring client MUST use the HTTP POST method to approve a transfer requested by the new sponsoring client.

Example Approve request:

```
POST /rpp/v1/domains/example.nl/transfer/approval HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Accept-Language: en
RPP-Cltrid: ABC-12345
Content-Length: 0
```

Example Approve response:

```
HTTP/2 200 OK
Date: Wed, 24 Jan 2024 12:00:00 UTC
Server: Example RPP server v1.0
Content-Length: 80
RPP-Svtrid: XYZ-12345
RPP-Cltrid: ABC-12345
RPP-code: 01000

TODO
```

7.9. Update Resource

- Request: PATCH /{collection}/{id}
- Request message: Object Update message
- Response message: Status

An object Update request MUST be performed using the HTTP PATCH method. The request message body MUST contain an Update message.

TODO: when using JSON, also allow for JSON patch so client can send partial update data only?

Example request:

```
PATCH /rpp/v1/domains/example.nl HTTP/2
Host: rpp.example.nl
Authorization: Bearer <token>
Accept: application/rpp+json
Content-Type: application/rpp+json
Accept-Language: en
Content-Length: 252

TODO
```

Example response:

```
HTTP/2 200 OK
Date: Wed, 24 Jan 2024 12:00:00 UTC
Server: Example RPP server v1.0
Content-Length: 80
RPP-Svtrid: XYZ-12345
RPP-Cltrid: ABC-12345
RPP-code: 01000

TODO
```

8. Extension Framework

TODO

9. IANA Considerations

TODO

10. Internationalization Considerations

TODO

11. Security Considerations

RPP relies on the security of the underlying HTTP [RFC9110] transport, hence the best common practices for securing HTTP also apply to RPP. It is RECOMMENDED to follow them closely.

Data confidentiality and integrity MUST be enforced, all data transport between a client and server MUST be encrypted using TLS [RFC5246]. Section 9 describes the level of security that is REQUIRED for all RPP endpoints.

Due to the stateless nature of RPP, the client MUST include the authentication credentials in each HTTP request. This MAY be done by using JSON Web Tokens (JWT) [RFC7519] or Basic authentication [RFC7617].

12. Change History

12.1. Version 01 to 02

- Merged the RPP-EPP-Code and RPP-Code headers into a single RPP-Code header

12.2. Version 00 to 01

- Updated "Request Headers" and "Response Headers" section
- Changed transfer resource URL and HTTP method for reject, approve and cancel, in order to make the API easier to use

12.3. Version 00 (draft-rpp-core) to 00 (draft-wullink-rpp-core)

- Renamed the document name to "draft-wullink-rpp-core"
- Removed sections: Design Considerations, Resource Naming Convention, Session Management, HTTP Layer, Content Negotiation, Object Filtering, Error Handling
- Renamed Commands section to Endpoints
- Removed text about extensions
- Changed naming to be less EPP like and more RDAP like

13. References

13.1. Normative References

[REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000, <http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>.

[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>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[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>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[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>.

[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>.

[RFC6648] Saint-Andre, P., Crocker, D., and M. Nottingham, "Deprecating the "X-" Prefix and Similar Constructs in Application Protocols", BCP 178, RFC 6648, DOI 10.17487/RFC6648, June 2012, <https://www.rfc-editor.org/info/rfc6648>.

[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>.

[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>.

[RFC8941] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", RFC 8941, DOI 10.17487/RFC8941, February 2021, <https://www.rfc-editor.org/info/rfc8941>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[YAML] YAML Language Development Team, "YAML: YAML Ain't Markup Language", 2000, <https://yaml.org/spec/1.2.2/>.

13.2. Informative References

[RFC8748] Carney, R., Brown, G., and J. Frakes, "Registry Fee Extension for the Extensible Provisioning Protocol (EPP)", RFC 8748, DOI 10.17487/RFC8748, March 2020, <https://www.rfc-editor.org/info/rfc8748>.

Authors' Addresses

Maarten Wullink
SIDN Labs
Email: maarten.wullink@sidn.nl
URI: https://sidn.nl/

Pawel Kowalik
DENIC
Email: pawel.kowalik@denic.de
URI: https://denic.de/