Workgroup:	Network Working Group
Internet-Draft:	draft-wullink-rpp-requirements-00
Published:	20 March 2025
Intended Status:	Standards Track
Expires:	21 September 2025
Authors:	M. Wullink	P. Kowalik
---	---	---
	SIDN Labs	*DENIC*

RESTful Provisioning Protocol (RPP) - Requirements

Abstract

This document describes the requirement for the development of the RESTful Provisioning Protocol (RPP).

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 September 2025.

Copyright Notice

Table of Contents

1. Introduction

This document describes the set of requirements RESTful Provisioning Protocol (RPP) an Application Programming Interface (API) API, for provisioning objects in a shared database, based on the HTTP protocol [RFC2616] and the principles of [REST].

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].

4. General

RPP MUST be a client server protocol

provide a clear, clean, easy to use and self-explanatory interface that can easily be integrated into existing software systems and has good support for multiple programming languages.

RPP SHOULD leverage widely deployed web standards, tools, and infrastructure components such as HTTP, JSON, OpenAPI specification, API gateways, and load balancing, caching and delegate responsibility to the HTTP layer where possible.

5. Transport

5.1. HTTP

The Hypertext Transfer Protocol (HTTP) [RFC9110] MUST be used as the transport mechanism for RPP messages.

RPP SHOULD use the common best practices for designing a HTTP based application described in [BCP56].

Consistent and meaningful URL structures MUST be used for for identifying, accessing resources and enable request routing.

RPP MUSRT use standard HTTP response and error handling functiality, e.g. status codes and response headers.

5.2. REST

The RPP architecture MUST use the principles of the [REST] architectural style. A RPP server MUST conform to at least level 2 of the [RICHARDSON] Maturity Model (RMM). The third level of the RMM describes a method for dynamically discovery of application resources. The RPP specification MUST specify the URLs used for all data model resources and therefore the URLs are fixed and do not need to be dynamically discovered and therefore designing for RMM level 3 is not recommended.

When the semantics of a resource URL and HTTP method do not require a request message, the use of a request message MUST be optional.

RPP specifications SHOULD include an OpenAPI specification to facilitate documentation, testing, and code generation, and provide implementer-friendly extension descriptions.

Every RPP request MUST be atomic and idempotent when possible.

6. Data Model

The base data model structures MUST be data format agnostic and can be mapped to multiple data formats (JSON, XML, YAML etc.)

Commonly used EPP extensions MAY be added to the RPP core data model (example: DNSSEC)

The data model MUST have support for internationalization, including for Contact objects, email addresses, and Internationalized Domain Names (IDNs).

RPP MUST support human-readable localized responses.

RPP SHOULD provide mechanisms for registrars to signal data omission, indicating data collected but not transmitted to the registry.

RPP MUST allow for the use of different profiles to indicate required parts of the data model, mapping definitions, or functional subsets for compatibility.

The server MAY choose to let the client decide how strict the data validation must be. Use Prefer HTTP header "handling=strict" vs. "handling=lenient” to make the server behave strictly about unknown attributes vs. ignoring unknown attributes. Another way would be with a more fine-granular approach like the “crit” claim in JWT.

7. Data Representation

RPP MUST use JSON as the default data format, but support for multiple data formats (e.g. XML, YAML) MAY be included.

Support validation of request and response message, in order to determine if the content is valid and no required attributes are missing.

A server MAY choose to include support for multiple media types.

A client MUST be able to signal to the server what media type the server should expect for the request content and to use for the response content.

Allow for the use of server profiles, indicating required parts for the data model and/or mapping definitions.

RPP SHOULD consider mechanisms to support data formats outside of core RPP domain. Especially formats, which lose their properties if transformed, like Verifiable Credentials for contacts which are digitally signed.

Partially updating an object MAY be supported, ussing HTTP PATCH method and JSON Merge Patch

Contact information MUST be provided using the JSContact [RFC9553] format.

8. Discoverability

RPP MUST include a bootstrap mechanism te help clients locate RPP available services, possible solution include:

- IANA bootstrap Service Registry
- DNS TXT record

The API version MUST be discoverable and SHOULD be added to the discovery document in the well-known directory.

Notices related to scheduled server maintenance timeslots MAY be included in the discovery document

A RPP service MAY choose to only support a subset of EPP functionality, this MUST be discoverable by the client.

9. EPP compatibility

RPP SHOULD provide functional equivalents for core EPP functionalities related to domain names, hosts, and contacts as defined in RFC5731, RFC5732 and RFC5733 mappings for core objects (domain, contact, host) and a selection of commonly used EPP extensions will be provided in separate specifications.

RPP MUST support automatic/mechanical mapping/conversion between EPP and RPP. Compatibility definitions for RPP to EPP mappings MAY be defined in compatibility profiles.

10. Security

RPP MUST support modern authentication and authorization schemes that allow for easy integration in modern HTTP infrastructure, and may enable support for new functionality and or protocol features that are not (easily) possible using EPP.

RPP MUST support modern authorization standards (OAuth, OpenId Connect)

Support for an easier and faster object transfer process MAY be included, where approval from the losing registar can be obtained interactively by the registrant during the transfer process

The authorisation model MUST support granular authorizations, using framework such as OAuth, beyond current auth-code based authorisation for transfers only:

- Domain transfers without first getting the "normal" transfertoken should be possible
- DNS providers should be able to use the API to update the NS records
- OpenID Connect to interactively allow for DNS provider to update NS records, directly at theregistry of indirectly through a supporting registar.
- Renewals

RPP MUST employ strong authentication and utilize encrypted transport (HTTPS) to protect sensitive data and authentication material. Security mechanisms SHOULD be flexible to allow operators to choose appropriate methods and support federated authentication scenarios. RPP authorization models are intended to be fine-grained and go beyond simple auth-code based models, allowing for control at the operation and potentially attribute level, supporting use cases like domain transfers, DNS provider authorizations, and renewals.

RPP MAY include support for DNS hosters to update NS records directly in registry when approved by sponsoring registrar.

11. Extensibility

The protocol MUST be extensible to accommodate new functionalities, data objects, and operations beyond the initial scope.

The RPP data model SHOULD aim for easy and natural extensibility to richer models compared to EPP, including attributes for VAT numbers, company numbers etc.

Allow for flexibility in extending data model (EPP object extension) e.g. adding new objects or a new attribute to an existing object MUST be possible.

Extensions for new operations (EPP protocol extension) on resources, e.g. registry-lock “/domains/example.nl/extensions/lock” MUST be supported.

The extension name/definition MAY need to include an IANA registration.

EPP style command-response extensions MUST not be supported.

When a registry of extensions is required then IANA MUST be used.

The process of publishing extensions MUST be lightweight.

Every extension MUST include an implementer-friendly description, preferred is OpenAPI,

12. Scalability

RPP MUST be stateless and MUST NOT maintain application state on the server required for processing future RPP requests. Every client request needs to provide all the information required for the server to be able to successfully process the request. The client MAY maintain application session state, for example by using a JWT token.

Server responses that are cacheable MUST not include RPP transaction related identifiers and values.

RPP MUST support load balancing at the level of request messages

13. Performance

RPP MUST allow optional or no request/response message when this is not required, improving performance and network bandwidth requirements for both client and server. Fewer messages have to be created, marshalled, and transmitted.

RPP MAY allow for common bulk operations, resource listing, and filtering capabilities where this does not impact scalability negatively.

RPP MAY support compound object create request having embedded contact/host vs. request serialization (client waiting for contact/host creation to succeed before sending a domain request). Return complete representation (similar to object info in EPP) after compound request completed or return redirect to newly created object location.

14. Representation

Regarding to the Depth of data representation The client MAY want to request different depth of data representations, depending of its use-case:

- Minimal representation (like ID, or ID+name)
- Full representation (all data of object itself)
- Full representation + dereferenced referrals (for example domain with contact and host details)

Different representations may be requested in different contexts:

- GET request to the resource itself
- GET request to get a collection of objects
- responses to PUT/POST/PATCH requests

Consider using Prefer HTTP header “return” tag to distinguish between full and minimal data representation in the responses (for example if client is not interested in the full response for bulk use-cases)

Representation of the data vs. transaction information

The data representation in responses to transactions MUST only contain the provisioning object itself, the transaction information MUST be represented in HTTP headers.

15. Other

The items below have been mentioned on the mailinglist and may need to be added as an requirement.

- Data Omission - what requirements will there be around a registrar's ability to signal that it has collected some data but has not transmitted it to the registry?
- Registration attribution - will there be requirements for attribution of registration actions (who did what), and will cryptography be used?
- Registrant verification - will there be requirements to support registrant verification (NIS2)?
- Linking - will the protocol support linking to RDAP objects, other RPP objects, etc...
- Include identification of legal vs. natural contacts.
- Expanded common models (when compared to EPP), maybe there should be much more attributes then it is in EPP (Vatnumber / Company Number/ properties for Identity papers) Trademark information and more There are a lot of epp extensions in the wild that try to handle this.
- include also DNS provisioning as potential use-case
- possibility to seamlessly compose the API with other registry use-cases to have uniform API layer from the client perspective
- investigate possible multi-party authorisation schemas. Use case: DNS operator would get authorisation to update DS or NS record through RPP.
  - This may refer to roles not yet considered in EPP processes, e.g. a third-party DNSSEC signing authority. For a potential use case, see the discussion on the DD mailing list on that topic, which discusses potential approaches for DNS, some of which would require modelling relationships between parent/child/signer zone authorities in a provisioning protocol.
- possibility of mobile app or direct browser integration (use case for registries which directly authenticate their domain holders and allow operations on a domain and/or if RPP would be exposed by a registrar)

16. IANA Considerations

TODO

17. Internationalization Considerations

TODO

18. Security Considerations

TODO

19. Normative References

[BCP56] Nottingham, M., "Building Protocols with HTTP", BCP 56, RFC 9205, June 2022.

<https://www.rfc-editor.org/info/bcp56>

[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>.

[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>.

[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>.

[RFC9553] Stepanek, R. and M. Loffredo, "JSContact: A JSON Representation of Contact Data", RFC 9553, DOI 10.17487/RFC9553, May 2024, <https://www.rfc-editor.org/info/rfc9553>.

[RICHARDSON] Fowler, M., "Richardson Maturity Model", 2010, <https://martinfowler.com/articles/richardsonMaturityModel.html>.

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/