

---

Workgroup: Network Working Group  
Internet-Draft: draft-wullink-rpp-requirements-00  
Published: 24 March 2025  
Intended: Standards Track  
Status: 25 September 2025  
Expires: M. Wullink P. Kowalik  
Authors: *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 25 September 2025.

## Copyright Notice

Copyright (c) 2025 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. Terminology	3
3. Conventions Used in This Document	3
4. General	3
5. HTTP	3
6. REST	4
7. Data Model	4
8. Data Representation	4
9. Discoverability	5
10. EPP compatibility	5
11. Security	6
12. Extensibility	6
13. Scalability	7
14. Performance	7
15. Representation	7
16. New features	8
17. Other	8
18. IANA Considerations	8
19. Internationalization Considerations	9
20. Security Considerations	9
21. Normative References	9
Authors' Addresses	10

## 1. Introduction

This document describes the set of requirements for the RESTful Provisioning Protocol (RPP), an Application Programming Interface (API) for provisioning objects in a shared database. RPP is based on the HTTP [[RFC9110](#)] protocol and the architectural 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

A well defined architecture **MUST** be defined for RPP, including a description of the responsibilities of the defined protocol layers.

The API **MUST** provide a clear, clean, easy to use and self-explanatory interface that can easily be integrated into existing software systems and includes language bindings for the most popular programming languages.

RPP **MUST** leverage widely deployed web standards, tools, and infrastructure components such as HTTP, JSON, OpenAPI specification, API gateways, load balancing, caching and delegate functional responsibility to the HTTP layer when possible. For example, authentication is part of the HTTP layer and not part of the RPP application layer.

## 5. 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 identifying, accessing resources and enable request routing.

RPP MUST use standard HTTP status codes and MAY define new RPP status and map these to HTTP status code, RPP MUST NOT redefine HTTP status code semantics.

## 6. 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 RPP specification MUST specify all resource URLs used and therefore the URLs are already known and do not need to be dynamically discoverable, 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.

## 7. Data Model

The base data model structures MUST be data format agnostic, it MUST be possible to map the data model to multiple data formats (JSON, XML, YAML etc.)

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

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.

## 8. Data Representation

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

Validation of request and response message MUST be supported, 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 for supporting 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, using HTTP PATCH method and [JSON Merge Patch](#)

Contact information **MUST** be provided using the JSContact [[RFC9553](#)] format.

## 9. Discoverability

RPP **MAY** include a bootstrap mechanism to help clients locate RPP available services, possible solution include:

- IANA bootstrap Service Registry
- DNS TXT record

A discovery document **MUST** be made available in the well-known directory.

Server provided functionality, such as the set of supported extensions, **MUST** be discoverable using the discovery document.

The API version **MUST** be discoverable and **SHOULD** be added to the discovery document.

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.

## 10. EPP compatibility

RPP **MUST** 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)

The automatic or mechanical mapping or conversion between EPP and RPP **MUST** be possible. Compatibility definitions for RPP to EPP mappings **MAY** be defined in compatibility profiles.

The most commonly used EPP extensions **MAY** be included in the core RPP specification.

Only the string based EPP token defined in [[RFC5730](#)] **MUST** be supported, any EPP token extensions **MAY** be supported.

## 11. Security

RPP MUST support modern authentication and authorization schemes allowing for easy integration in modern HTTP infrastructure.

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 registrar 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, the following usecases MAY be supported:

- Domain transfers without first getting the "normal" transfertoken
- DNS providers use the API to update the NS records
- OpenID Connect to interactively allow for DNS provider to update NS records, directly at the registry or indirectly through a supporting registrar.
- Renewals

RPP MUST employ strong authentication and utilize encrypted transport (HTTPS) to protect sensitive data. 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.

RPP MAY include a mechanism for cryptographic verification of request and response messages as an additional security layer.

## 12. 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.

RPP MUST allow for flexibility in extending the 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/lock" MUST be supported.

EPP style command-response extensions MUST not be supported.

When a registry for RPP extensions is required, then IANA MUST be used for this function.

A namespace concept for JSON MUST be support, to prevent name collisions between the RPP core and a extension and between two or more extensions.

## 13. 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

Every request message MUST at most contain a single object for the server to operate on, with the exception of operations that are explicitly defined as a bulk operation, bulk operations MAY be processed asynchronously.

## 14. 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.

## 15. Representation

A client MAY want to request different depth of data representations, depending on the use case:

- Minimal representation (ID, or ID+name)
- Full representation (all data of the object)
- 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

Data representation in a responses MUST only contain the provisioning object itself, the transactional information MUST be represented in separate HTTP headers.

RPP MUST support internationalization, including for Contact objects, email addresses, and Internationalized Domain Names (IDNs).

Human-readable localized response messages MUST be supported.

## 16. New features

The server MAY support generating a representation of a historical overview for an object, e.g. show all events linked to the object (create, update ...). The historical time window is determined by server policy and MUST be included in the discovery service document.

## 17. 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)

## 18. IANA Considerations

TODO



## 19. Internationalization Considerations

TODO

## 20. Security Considerations

TODO

## 21. 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](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>>.
- [**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](mailto:maarten.wullink@sidn.nl)

URI: <https://sidn.nl/>

### **Pawel Kowalik**

DENIC

Email: [pawel.kowalik@denic.de](mailto:pawel.kowalik@denic.de)

URI: <https://denic.de/>