Workgroup:	Network Working Group
Internet-Draft:	draft-ietf-rpp-requirements-03
Published:	13 October 2025
Intended Status:	Standards Track
Expires:	16 April 2026
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 16 April 2026.

Copyright Notice

Table of Contents

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.

Thin registry - A registry model in which the registry stores and serves only the minimal data necessary for delegation and identification of an object (for example, domain name, registrar identifier, status, and nameserver delegation). Registrant and contact data are maintained by the registrar and are not held by the registry.

Thick registry - A registry model in which the registry stores and serves the complete data set for an object, including registrant, administrative, and technical contact information, and other relevant attributes required for provisioning.

Sponsoring Client - The RPP client that currently has sponsorship of the object.

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

R1.1 A well defined architecture MUST be defined for RPP, including a description of the responsibilities of the defined protocol layers.

R1.2 *Removed*

R1.3 Wherever applicable RPP SHOULD leverage existing best practices and well adopted standards for building and documenting RESTful APIs. There MUST be a clear justification when deviating from this.

R1.4 RPP MUST include support for application level status codes, and MAY reuse the EPP status codes defined in [RFC5730].

R1.5 RPP MUST include support for providing detailed information about application status codes, for example as described in [RFC7807].

R1.6 RPP MUST support additional information about a successful operation (information or warning) to convey additional information to the client for example about deprecation or partial success.

R1.7 RPP MUST support both Thin and Thick registry models, and MUST allow for flexibility in how much data and what type of data is stored and returned by the server, according to the chosen registry model. The data elements returned is based on what the client is authorized to receive.

5. HTTP

R2.1 The Hypertext Transfer Protocol (HTTP) [RFC9110] MUST be used as the transport mechanism for RPP.

R2.2 RPP SHOULD use the best common practices for designing HTTP based applications, described in [BCP56]. There MUST be a clear justification when deviating from this.

R2.3 Consistent, predictable and meaningful URL structures MUST be used for identifying, accessing object resources and enable request routing.

R2.4 RPP MUST use the existing HTTP status codes and MUST define application level status codes and map these to HTTP status codes. RPP MUST NOT redefine existing HTTP status code semantics and when overloading (generic) HTTP status codes with multiple RPP status codes, the provided RPP status code MUST be used by the client to determine the exact nature of the problem.

6. REST

R3.1 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).

R3.2 RPP architecture MUST follow Resource-Oriented Architecture [ROI].

R3.3 RPP specification MUST strive to minimise round trips between client and server. Approaches, where client would need to make multiple requests each time to discover resource URL or server capabilities in order to perform operation SHOULD be used sparingly and be always well justified.

R3.4 *Merged with R12.1*

R3.5 RPP specifications SHOULD incorporate a machine-readable and well-established API specification, such as [!@OpenAPI] or [RAML]. This will facilitate documentation, testing, code generation, and user-friendly extension descriptions. RPP MUST NOT require what API specification technology is to be used. The RPP core documents and extension documents may also choose different API specification solutions, this choice is left to the document authors.

R3.6 RPP architecture MUST define a common pattern to allow a single resource to be addressable via multiple, alternative identifiers in its URL. The protocol MUST specify that one address is canonical, and any alternative addresses for the same resource MUST be treated as aliases resolving to the canonical resource.

7. Data Model

R4.1 The base data model structures MUST be data format agnostic. It MUST be possible to map the base data model to multiple data formats such as JSON, XML or YAML.

R4.2 Commonly used EPP extensions SHOULD be added to the RPP core data model. An example of this is the DNSSEC extension.

R4.3 RPP MUST allow an extension mechanism that allows clients to signal data omission or redaction, indicating data collected but not transmitted to the registry or redacted.

> TODO: Issue #34

R4.4 RPP MUST have mechanisms to define profiles to indicate:

- Required parts of the data model
- Mapping definition
- Functional subsets for compatibility.

> TODO: Issue #15

R4.5 RPP architecture MUST include loose coupling between the server and the client, allowing for non-coordinated introduction of non-breaking version changes on both sides.

R4.6 RPP MUST enforce the use of strict validation, where unknown properties, query parameters, url segments and RPP specific headers are treated as an error.

R4.7 RPP MUST support linking of objects of the same or different types, with flexible cardinality (one-to-one, one-to-many, many-to-many). The links MUST also support to have attributes of their own (e.g., a link between a domain and a contact object with a different contact role).

R4.8 RPP MUST allow a client to reference a shared object (e.g., a host or contact) sponsored by a different client, while ensuring the sponsoring client retains full administrative control over the shared object.

R4.9 RPP MUST support the definition of first-class, read-only resources whose state is managed by the server. These resources MAY expose logically associated information as sub-resources.

R4.10 RPP data model MUST support composition by defining a pattern where a resource can contain child resources of same or different type, with flexible cardinality (one-to-one, one-to-many, many-to-many),that are integral to it. Such resources MAY be embedded directly in the parent resource representation or exposed as sub-resources within the parent's URL structure.

8. Data Representation

R5.1 RPP MUST use JSON as the default data format.

R5.2 It MUST be possible to extended RPP to include support other data formats (e.g. XML, YAML).

R5.3 Validation of request and response message MUST be supported for both clients and the servers, in order to determine if the content is valid and no required attributes are missing.

> TODO: Issue #36

R5.4 RPP MUST define a default media type however the protocol SHALL be extensible to enable support for other media types.

R5.5 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.

R5.6 *Removed*

R5.7 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.

R5.8 RPP MUST support partial update of data objects.

R5.9 RPP MUST support full update of data objects.

R5.10 A generated RPP response representation that includes an object identifier (for example a contact handle) MUST also include a URL reference to the location of the object representation.

R5.11 RPP MUST support representation of a collections of resources.

R5.12 The representation of the links (see R4.7) MUST be expressed using the RPP server's internal identifiers. The RPP SHOULD also consider using URIs and/or [RFC6570] URI templates for uniform addressing of link targets, both internal and external to the RPP server.

9. Operations and responses

R6.1 RPP MUST include support for a client requesting 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)

R6.2 RPP MAY return different representations of the same object in different contexts:

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

R6.3 The data representation in a RPP response MUST only contain data related to the object, transactional information MUST be represented as one or more separate HTTP headers.

R6.4 RPP MUST support search for resource collections and SHOULD support filtering (e.g., by name prefix, status, registrar) and pagination.

> TODO: Issue #56

R6.5 RPP operations that modify repository state MUST be atomic. A single request MUST either succeed completely or fail completely, leaving the repository in its original state.

R6.6 RPP MUST provide services for the client to assure a re-tried operation changing resource state is executed only once if a request has been terminated or timed out before complete response has been received by the client (idempotency).

R6.7 The protocol specification MUST define the expected server state for a request that times or terminates out before a response is fully sent out the the client.

R6.8 For every request the server MUST generate a permanent, server-unique transaction identifier. This identifier MUST be returned to the client in the response.

R6.9 RPP MUST support informational and validation functions that are not directly tied to a persistent, provisioned object. These operations SHOULD be exposed as read-only resources that represent the result of a query or check.

R6.10 RPP MUST support a functional equivalent of the EPP Poll command described in EPP RFC [RFC5730], allowing for clients discovering and retrieving service messages available on the server. The RPP equivalent MAY contain additional options or features for discovering and retrieving service messages, such as:

- Allowing clients to subscribe to specific types of service messages.
- Allowing clients to receive multiple service messages in a single request.
- Allowing clients to use multiple concurrent readers.
- Support for streaming service messages to clients.

R6.11 RPP MUST include a functional equivalent of [RFC9038] to allow clients retrieve service messages including information it may not understand due to missing extension support.

10. Discoverability

R7.1 RPP MAY include a bootstrap mechanism designed to allow clients to locate the network identifier for the RPP service of a registry operator, e.g. rpp.sidn.nl for the registry operator for the .nl ccTLD.

Solutions may include:

- IANA bootstrap Service Registry
- DNS TXT records

R7.2 An RPP server MUST publish a service discovery document in the well-known directory, described in [RFC5785]. This document contains structured machine-readable information that is required or useful for the client to be able to generate valid RPP requests. The information may contain, but is not limited to:

- Available services,
- Used Extensions
- Versions used for services and extensions
- Environment name (production, test etc.)
- Server datetime
- Maintenance notices
- Supported profiles

R7.3 Server provided functionality, such as the set of supported profiles, languages or extensions, MUST discoverable using the discovery document.

R7.4 RPP MUST support versioning of:

- The protocol itself
- Data object types
- Representations
- Operations
- Profiles
- Extensions

R7.5 Versioning schema MUST carry information about breaking vs. non-breaking changes and allow clients to decide whether it is able to interact with the server. The versioning scheme SHOULD be like the scheme used for HTTP where minor version changes do not break compatibility.

R7.6 Notices related to scheduled server maintenance timeslots MAY be included in the discovery document, this could be a human-readable, non machine parsable character string.

R7.7 RPP MAY only support a subset of EPP functionality, the supported functionality MUST be discoverable by the client

R7.8 *Removed*

R7.9 An RPP response that includes unique object identifiers, MAY also include URL references for these objects.

R7.10 Versions used by the RPP and used extensions MUST be discoverable by the client.

11. EPP compatibility

R8.1 RPP MUST provide functional equivalents for core EPP functionalities related to domain name, host, contact, and organisation objects as defined in [RFC5731], [RFC5732], [RFC5733] and [RFC8543].

R8.2 The automatic or mechanical mapping or conversion between EPP and RPP data model MUST be possible.

R8.3 Compatibility definitions for a RPP to EPP mapping MAY be defined in compatibility profiles (see: R4.4).

> TODO: Issue #15

R8.4 RPP MUST include an extension framework able to define equivalents of most commonly used EPP extensions, which are not a part of core protocol (see: R4.2)

R8.5 EPP password based Authorisation Information defined in [RFC5731] and [RFC5733] MUST be supported in RPP. RPP MUST by default support the requirements for Secure Authorization Information for Transfer [RFC9154] operational practise.

R8.6 RPP SHOULD support client_id/password authentication to match EPP client authentication.

R8.7 Where applicable RPP MUST support longer authorisation information compared to EPP, metainformation about client software and operational environment as well security related information (events) in the responses providing a functional equivalence to Login Security Extension for the Extensible Provisioning Protocol [RFC8807].

12. Security

R9.1 RPP MUST support state-of-the-art authentication and authorisation schemes allowing for easy integration in modern HTTP infrastructure.

R9.2 RPP MUST support robust authentication and authorisation mechanisms, such as OAuth 2.0 and OpenID Connect, to ensure that only authorised clients and users can access or modify resources.

R9.3 Support for a simplified and quicker object transfer process MAY be included, where approval from the losing registrar is to be obtained interactively by the registrant during the transfer process.

R9.4 RPP MUST include an authorisation model/framework that goes beyond the current EPP password based Authorisation Information (AuthInfo) used for object transfers. The following use cases MAY be supported:

- Object transfers without using an EPP password based Authorisation Information
- Registrants using OpenID Connect can interactively allow DNS operator to update their NS records, directly in the registry database or indirectly using a registrar.

R9.5 All RPP communications MUST use HTTPS (TLS) to protect data in transit from eavesdropping and man-in-the-middle attacks.

R9.6 Security mechanisms SHOULD be flexible to allow operators to choose appropriate methods and support federated authentication scenarios.

R9.7 RPP MAY include a mechanism for cryptographic verification of request and response messages as an additional security layer.

R9.8 RPP MUST allow for multiple user accounts linked to a single registrar, registrar user management MAY be delegated to an administrator account linked to a registrar, allowing for self service account management by the registrar.

R9.9 RPP MUST support a granular authorisation matrix, where one or more permissions are coupled to a user account. Allowing for the creation of different types of user accounts, such a readonly users only allowed to fetch data about existing objects, and power users allowed to create and modify objects.

R9.10 RPP MUST allow users to update their credentials and enforce strong passwords and limited lifetime for passwords and other tokens.

R9.11 RPP must support the Least Privilege Principle, to allow server operators to ensure that clients have only the permissions necessary.

R9.12 RPP MUST support secure credentials management, ensuring that credentials are protected against replay and theft, and have limited lifetimes.

R9.13 Any protocol extensions MUST be subject to the same security review and requirements as the core protocol.

R9.14 There MUST be mechanisms to revoke or deprecate credentials, tokens, or permissions when no longer needed or if compromised.

R9.15 RPP must support mechanisms to prevent Denial-of-Service attacks, whether from malicious actors or misbehaving clients. These mechanisms can include rate limiting and throttling of requests with related protocol signalling.

13. Extensibility

R10.1 The protocol MUST be extensible to accommodate new functionalities, data elements, read-only resources, operations, informational and validation functions, alternative addressing of resources, resource linking, and resource composition beyond the initial definitions in RPP core.

R10.2 RPP MUST support the extension of the data model by enabling the definition and provisioning of entirely new object types, and by providing a standardised mechanism for adding new persistent properties to any existing object type.

R10.3 RPP SHOULD promote standardisation of commonly used extension attributes.

R10.4 Extensions for new operations on existing resources MUST be supported.

> TODO: Issue #47

R10.5 RPP MUST support extensions that define new status codes not already defined in the core RPP RFCs. Extension designers MAY add new status codes. If a newly created status code is generic enough to be useful for the wider RPP community, the designer SHOULD register it in the appropriate IANA registry.

R10.6 RPP MUST support extensions adding new HTTP headers.

R10.7 RPP SHALL have mechanisms to assure conflict avoidance when extending the protocol, including but not limited to data model, representations, operations, parameters, error codes and signalling. There MUST be a mechanism of conflict-free, non-coordinated extending in private/vendor discretion as well as a coordinated process for core, generic or shared elements.

R10.8 When a public registry for RPP extensions is required, then IANA MUST be used for this function.

R10.9 RPP extensions MUST include support for versioning, the version of the extension supported by the server MUST be included in the discovery document.

R10.10 *Removed*

R10.11 The protocol MUST support mechanisms for extending standard processes, such like delete or create, with additional transient parameters or non-persistent data (e.g. intended premium price or tier).

R10.12 The protocol MUST support mechanisms for extending results of an operation with additional transient, non-persistent information not defined in the RPP core (e.g. information about discount applied to a create request).

R10.13 The protocol MUST allow extensions to add additional information to object statuses (e.g. due date of a status).

R10.14 Data model for DNS (see R4.7) MUST be extensible to future DNS record types as well as future ways of delegation over DNS (e.g. DELEG).

14. Scalability

R11.1 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.

R11.2 RPP MUST support cacheability of responses, if applicable to the operation semantics and MUST not include transaction related identifiers and values.

> TODO: Issue #50

R11.3 RPP MUST support load balancing at the level of request messages (URL) and load balancing MUST be possible without processing HTTP body.

R11.4 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.

R11.5 RPP MUST support asynchronous processing for operations on multiple objects, otherwise resource intensive or involving manual steps. The client request results in a confirmation of receipt and a means for retrieving the final completed processing result at a later time.

15. Performance

R12.1 In order to minimise message sizes and needed processing RPP SHOULD be designed not to include a HTTP message body in the request or response when this is not necessary, for example when the required data can be transmitted using the URL and/or HTTP headers.

R12.2 RPP MAY allow for common bulk operations, resource listing, and filtering capabilities. RPP MUST NOT mandate such functionalities where this may impact scalability or performance negatively.

R12.3 *Removed*

R12.4 The protocol MUST be usable in both high volume and low volume operating environments.

16. Internationalisation

R13.1 RPP MUST support internationalisation, for object types and messages defined in the core protocol and extensions

R13.2 RPP MUST support human-readable localised response messages.

17. Clients

R14.1 RPP MUST support server applications as clients. This will be a primary use-case of registry/registrar integration.

R14.2 RPP MUST support interaction from command-line tools or desktop applications capable of sending HTTP requests. These can be generic clients such as `curl` or `Postman` but also specialised RPP command line tools or scripts.

R14.3 RPP SHOULD support web browsers as clients, such as SPA (single page applications) without any proxy backend between web browser and the RPP server.

R14.4 RPP SHOULD support mobile applications as clients, also here through direct integration without any proxy backend.

18. Requirements for object types

18.1. Common

O1.1 RPP MUST define a single, structured data model for representing DNS resource records. This model MUST be used consistently for all object types that require DNS representation (e.g., Host, Domain). The model MUST support common DNS record types (such as A, AAAA, CNAME, MX, NS, DS, TXT) and their standard attributes, like TTL. The model SHOULD be designed to be extensible for future, experimental or less common record types.

18.1.1. Object Transfers

For the purposes of requirements related to transfers, the following specific terms are used:

- Gaining Client: The client seeking to gain sponsorship of the object.
- Initiating Client: The client that starts the transfer request.

O2.1 RPP MUST support two types of object transfer operations:

- Pull Transfer: Initiated by the Gaining Client.
- Push Transfer: Initiated by the Sponsoring Client, who designates a Gaining Client.

O2.2 A Gaining Client MUST provide valid authorisation information to initiate a Pull Transfer request.

O2.3 For Pull Transfers, the RPP MUST provide operations for the Sponsoring Client to explicitly approve or reject a pending transfer request. The RPP MUST reject any approval or rejection attempts not initiated by the Sponsoring Client.

O2.4 For Push Transfers, the RPP MUST provide operations for the Gaining Client to explicitly approve or reject a pending transfer request. The RPP MUST reject any approval or rejection attempts not initiated by the designated Gaining Client.

O2.5 RPP MUST provide an operation for the Initiating Client to cancel its own pending transfer request. The RPP MUST reject any cancellation attempts not initiated by the Initiating Client.

O2.6 RPP MUST provide an operation to query the status of a pending or recently completed transfer request. This operation MUST be accessible to the Sponsoring Client and the Gaining Client.

O2.7 The response to a successful object transfer MUST include a representation of the transferred object and a list of any associated objects that were also transferred.

18.2. Domain Object Type

D1.1 RPP domain object data model MUST include, at a minimum, the attributes defined in RFC5731: the fully qualified domain name, repository object identifier, object status, the current sponsoring client identifier, creating registrar client ID, creation timestamp, last update client ID, last update timestamp, expiration timestamp, last transfer timestamp, name servers, subordinate hosts, the registrant, and other associated contacts.

D1.2 RPP MUST only accept valid domain names, which MUST be valid FQDNs.

D1.3 RPP MUST support internationalised domain names (IDN) and accept A-labels and U-labels, also know as IDNA2008 and defined in [RFC5890].

D1.4 RPP MUST apply the rules of Label Equivalence as defined in Section 2.3.2.4 of [RFC5890] when processing domain names in requests and responses by both clients and servers.

D1.5 RPP domain object data model MUST allow for the association of zero, one or more objects representing the DNS configuration of the domain including name servers. These may be defined by a reference to separate repository objects (equivalent of EPP host objects) or aggregate object (equivalent of EPP host attribute).

D1.6 RPP MUST support domains that have linkage to at minimum registrant, administrative, technical, and billing contacts. In thin registries, only identifiers MAY be stored; in thick registries, contact data MAY be included per (privacy) policy. The list of contact link types MUST be extensible.

D1.7 RPP domain objects MUST support EPP password-based Authorisation Information (authInfo) for transfer operations (see R8.5).

D1.8 RPP MUST provide functional equivalents for EPP domain status values (e.g., ok, inactive, client/server<command>Prohibited, pending<command>) and define their mapping to RPP responses and HTTP status codes. The protocol MUST define which statuses can be set by the server and which can be set by the sponsoring client.

D1.9 RPP MUST enforce referential integrity. the parent domain name for a subordinate host object MUST not be deleted. RPP MUST return a conflict error when deletion is disallowed and the domain representation MAY include an attribute with information about linked objects.

D1.10 RPP MUST provide a mechanism for clients to discover the server's supported Internationalized Domain Name (IDN) policies. Information such as the identifiers for supported IDN tables, applicable language tags, and variant disposition policies MUST be discoverable via the service discovery document (see R7.2).

18.2.1. Operations

D2.1 RPP MUST provide operations to check, create, read, update, transfer, renew and delete domain name objects as defined in [RFC5731].

D2.2 When creating or renewing a domain object, RPP MUST allow a client to specify a registration period. The protocol MUST provide a mechanism for the server to confirm the resulting registration expiration date in the response.

D2.3 RPP MUST support the Domain Registry Grace Period Mapping as defined by [RFC3915], the restore report MAY be ignored or included in the initial restore request, making this a 1-step process vs the 2-step process in EPP.

D2.4 RPP MAY support searching and listing domains filtered by name (exact/prefix), and sponsoring client.

D2.5 Only the sponsoring client (or an authorised server administrator) MAY modify or delete a domain object; servers MUST enforce authorisation.

D2.6 RPP MUST prevent creation of duplicate domain objects within a registry namespace (TLD) and return a HTTP 409 (Conflict) status on collision.

D2.7 The transfer of a domain object MUST also result in the transfer of any subordinate host objects (ns.foo.example when foo.example is transferred).

D2.8 RPP domain transfer operation MUST allow for an implicit renewal. If a transfer results in an extension of the registration period, the response to the successful transfer MUST include the new expiration date of the domain object.

D2.9 RPP domain transfer operation SHOULD support optional update of DNSSEC and delegation information directly with the transfer request. This feature, as not standard EPP feature, SHALL be optional to be supported by server policy.

18.2.2. Data Representation

D3.1 The JSON representation MUST include the canonical domain name and any U-label/A-label when IDN is used by the server.

D3.2 RPP domain object representation MUST include link relations to related objects, for example: self, hosts and contacts.

D3.3 The representation MUST adapt to the registry model. In thin mode, only identifiers (e.g., contact IDs) MUST be returned; in thick mode, full contact data MUST be returned.

18.2.3. Embedding of EPP extensions

D4.1 RPP core protocol MUST incorporate functional equivalent of the following list of widely used and considered essential EPP extensions as recommended in [TigerTeamRecc]:

- Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol [RFC5910]
- Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol [RFC3915]
- Extensible Provisioning Protocol (EPP) mapping for DNS Time-To-Live (TTL) values [RFC9803]
- Organization Extension for the Extensible Provisioning Protocol [RFC8544]

18.3. Host Object Type

H1.1 RPP host object data model MUST include, at a minimum: fully qualified host name, all associated IP addresses (see O1.1), repository object identifier, object status, current sponsoring client identifier, creating client identifier, creation timestamp, last updating client identifier, last update timestamp, the last transfer timestamp.

H1.2 RPP MUST map the EPP host attribute model to the generic JSON DNS model defined by RPP (O1.1).

H1.3 Host names MUST be valid FQDNs.

H1.4 RPP MUST support internationalized domain names (IDN) for host names and accept A-labels and U-labels, also know as IDNA2008 and defined in [RFC5890].

H1.5 RPP MUST apply the rules of Label Equivalence as defined in Section 2.3.2.4 of [RFC5890] when processing host names in requests and responses by both clients and servers.

H1.6 When used for linking a name server to a domain name, RPP MUST support both in-bailiwick and out-of-bailiwick host objects.

H1.7 RPP MUST support zero or more IP addresses (IPv4 or IPv6) for host objects, when the host object is used for linking a name server to a domain name. Addresses MUST be syntactically valid, normalised, and unique within the host. The maximum number of allowed addressess and any disallowed ranges (e.g., [RFC1918]) are server policy. The server MAY require, make optional or disallow IP addresses depending on whether host is in-bailiwick or out-of-bailiwick. This is also server policy.

H1.8 RPP MUST provide functional equivalents for EPP host status values (e.g., ok, linked, client/server<command>Prohibited, pending<command>) and define their mapping to RPP responses and HTTP status codes.

18.3.1. Operations

H2.1 RPP MUST RPP MUST provide operations to check, create, read, update and delete host objects as defined in [RFC5732], with partial update semantics available to allow for efficient updates. Transfer operation of subordinate host object MUST be implicit with the transfer operation of parent domain name.

H2.2 RPP SHOULD support searching and listing hosts filtered by name (exact/prefix), IP address, and sponsoring client, with pagination, the server MAY use a maximum limit on results.

H2.3 Only the sponsoring client (or an authorised server administrator) MAY modify or delete a host; servers MUST enforce authorisation.

H2.4 RPP MUST assure that in-bailiwick host objects are created and managed by the same sponsoring client as the parent domain name.

H2.5 RPP MUST enforce referential integrity. A host referenced by any domain (linked) MUST NOT be deleted. Servers MUST return a conflict error when deletion is disallowed and the host representation MAY include an attribute with information about linked objects. RPP MUST allow for safe deletion of referenced hosts - with grace period, restore and prior removal of references as recommended in [RFC9874].

H2.6 RPP MUST prevent creation of duplicate hosts within a registry namespace (TLD) and return a conflict on collision.

18.3.2. Data Representation

H3.1 RPP MUST support a JSON representation for both Host objects and for Host attributes as defined in the EPP RFCs.

H3.2 The JSON representation MUST include the canonical host name and any U-label/A-label when IDN is used.

H3.3 The representation SHOULD include link relations to related objects, for example: self, and parent domain for in-bailiwick hosts.

18.4. Contact Object Type

C1.1 RPP contact object data model MUST include, at a minimum an equivalent of RFC5733 contact data model: a unique identifier, repository object ID, current status, name, organisation, full postal address, voice and fax numbers, email addresses,the sponsoring client identifier, the creating client identifier, creation timestamp, the last updating client identifier, last update timestamp, last transfer timestamp, and authorisation information.

C1.2 RPP MUST support server-generated opaque IDs, support for client-supplied IDs is OPTIONAL.

C1.3 RPP SHOULD support an explicit indication of entity type (person or organisation) in the contact model.

C1.4 When RPP is used with thick registries, full contact data MAY be returned, for thin registries only the contact identifier MUST be returned.

C1.5 RPP MUST support disclosure and privacy preferences equivalent to EPP “disclose”.

C1.6 RPP MUST support contact object to refer to external identity provider (e.g. when digital identity schemes are used), where the personal data would not be persisted within RPP server. RPP SHOULD allow to only store a stable identifier, reference or credential for future verification (see Privacy Considerations).

C1.7 RPP MUST enforce referential integrity. A contact MUST not be deleted when it is referenced by other objects. RPP MUST return a conflict error when deletion is disallowed and the contact representation MAY include an attribute with information about linked objects.

C1.8 RPP SHOULD consider renaming the EPP contact object type to "entity" to better align with the RDAP data model, defined in [RFC9083].

18.4.1. Operations

C2.1 RPP contact object type is mapped to the EPP equivalent and MUST support all operations (commands) defined for the contact object in [RFC5733], such as check, create, read, update, and delete with the possible exception of transfer command, and include support for partial update semantics available to allow for efficient updates.

C2.2 RPP MAY support the contact transfer command from EPP.

C2.3 RPP MAY support searching and listing contacts filtered by name (exact/prefix), and sponsoring client, with pagination, the server MAY use a maximum limit on results.

C2.4 Functional equivalents for EPP contact statuses (e.g., ok, linked, client/serverUpdateProhibited, client/serverDeleteProhibited, pendingTransfer) MUST be supported, with clear mapping to HTTP/RPP responses. The protocol MUST define which statuses can be set by the server and which can be set by the sponsoring client.

C2.5 RPP MUST prevent creation of contacts with duplicate ids within registry namespace (TLD) and return a HTTP 409 (Conflict) status on collision.

C2.6 The protocol MUST provide an operation to retrieve full contact representation. An authorisation mechanism MUST ensure that sensitive data, such as authorisation information, is only returned to the current sponsoring client.

C2.7 The protocol MUST provide an operation to retrieve an appropriate contact representation to non-sponsoring clients. The representation MAY vary depending if the authorisation information is provided - depending on server policy.

18.4.2. Data Representation

C3.1 RPP SHOULD consider using JSContact [RFC9553] format for contact representation.

C3.2 RPP MUST support contact attribute disclosure preferences per field (or field group) and this MUST be mapped to the EPP disclosure preferences described in [RFC5733].

18.4.3. Internationalisation

C4.1 RPP MUST support internationalisation (character encoding) for contact objects in the following areas:

- name
- address data
- any other contact-related data containing human provided or readable text

C4.2 RPP MUST support both the localized and internationalised version of the EPP postalInfo element from [RFC5733].

C4.3 RPP MUST support internationalised Email addresses [RFC6530] in Contact objects. Functional equivalent of [RFC9873] MUST be assured in EPP compatibility mode, however RPP MAY remove requirement for at least one all-ASCII Email address.

C4.4 RPP MUST support multiple localised expressions of the same data, e.g. fields mentioned in C4.1 having both international and localised variants.

C4.5 All future RPP contact object extensions MUST be able to handle internationalisation and localisation requirements.

18.5. Organisation Object Type

G1.1 RPP MUST support data model and operations defined for Organisations - functional equivalent of [RFC8543].

19. IANA Considerations

This document has several requirements for the RESTful Provisioning Protocol (RPP) that create considerations for IANA. Future architecture and design documents may identify additional needs for IANA registries.

Therefore, the core RPP specifications MUST include "IANA Considerations" sections that formally request the creation of any necessary IANA registries. These sections MUST also provide the initial registration of values defined within those core documents.

20. Security Considerations

The security section of this document defines the security related requirements for RPP, these requirements MUST be addressed in the design and implementation of RPP. Implementations MUST follow best practices, described in [BCP56] for HTTP API design.

RRP core specifications MUST include appropriate Security Considerations sections, specifying implementation and operational security requirements for both RPP clients and servers.

21. Privacy Considerations

DP.1 The protocol MUST provide mechanisms to support the implementation of data privacy principles, such as those found in modern data protection frameworks (e.g., GDPR). These mechanisms MUST support, at a minimum, the principles of data minimisation and purpose limitation.

DP.2 To support data minimisation, the protocol MUST allow clients to provide and manage only the data that is strictly necessary for a specific purpose. The protocol MUST also allow for different representations of an object, so that a client can request a representation containing only the data it needs and server can return the data a client is authorised to access (See also R4.3 and R6.1).

DP.3 The protocol's operations and data models MUST be sufficiently flexible to allow an operator to implement workflows for exercising data subject rights, such as access, rectification, and erasure of personal data, in a manner consistent with the operational and policy constraints of the provisioning environment.

DP.4 The protocol MUST provide services to identify data collection policies and privacy practices. Information about data collection, retention, and privacy policies MUST be included in the service discovery document, enabling clients to understand how personal and sensitive data is handled.

22. Changes History

This section is to be removed before publishing as an RFC.

22.1. Version -02 to -03

22.2. Version -01 to -02

- Added requirement for support of both thick and thin registry models (R1.7)
- Added requirements for Host Object Type
- Added R10.12 on future ways of delegation
- Added requirements for Domain Object Type
- Added relevant and not yet covered requirements from [RFC3375] (R6.5-R6.8, R12.4, Section 18.1.1)
- Added R6.4 RPP MUST include a functional equivalent of the EPP Poll command.
- Added requirements for the contact object type
- The security considerations section has been restructured and expanded to provide more detailed guidance on security best practices for RPP implementations.
- Added additional security requirements.
- R1.2 removed

- added essential and optional extensions sections in Appendix A
- Added generic IANA considerations
- Added requirement for support of both thick and thin registry models (R1.7)
- Added requirements related to extensibility following the recommendation from the Tiger Team [TigerTeamRecc]
- Added requirements related to embedding of EPP extensions following the recommendation from the Tiger Team [TigerTeamRecc]
- Updated R4.6 to require the use of strict data validation

22.3. Version -00 to -01

- Added Privacy Considerations section
- R1.5 has been changed to MUST instead of MAY.
- R1.6 has been changed to MUST instead of SHOULD.
- Updated the entire text to make consistent use of the British spelling style.
- stripped down version history of pre-WG -00 to -01

22.4. Version -01 to -00 (WG)

- The document has been adopted by the working group, the version number has been reset from -01 to -00.

22.5. Version -00 to -01

- Structurally reorganised the document, renumbering all requirements and adding new sections for Operations, Clients, and Internationalisation.
- Replaced specific technology mandates with a general requirement to leverage RESTful best practices.
- Introduced a formal framework for extensions, including versioning and discoverability.
- Significantly expanded the security model to support granular, multi-user authorisation.
- Mandated support for partial object updates and asynchronous processing for long-running operations.
- Detailed the requirements for a machine-readable discovery document (/.well-known).
- Added mandatory support for internationalisation (i18n) and specified considering JSContact for contact objects.

23. References

23.1. Normative References

[BCP56] Best Current Practice 56, <https://www.rfc-editor.org/info/bcp56>.
At the time of writing, this BCP comprises the following:

Nottingham, M., "Building Protocols with HTTP", BCP 56, RFC 9205, DOI 10.17487/RFC9205, June 2022, <https://www.rfc-editor.org/info/rfc9205>.

[RAML] raml.org, "RESTful API Modeling Language", 2025, <https://raml.org/>.

[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>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <https://www.rfc-editor.org/info/rfc1918>.

[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>.

[RFC3915] Hollenbeck, S., "Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP)", RFC 3915, DOI 10.17487/RFC3915, September 2004, <https://www.rfc-editor.org/info/rfc3915>.

[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>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <https://www.rfc-editor.org/info/rfc5785>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC5910] Gould, J. and S. Hollenbeck, "Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)", RFC 5910, DOI 10.17487/RFC5910, May 2010, <https://www.rfc-editor.org/info/rfc5910>.

[RFC6530] Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, DOI 10.17487/RFC6530, February 2012, <https://www.rfc-editor.org/info/rfc6530>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <https://www.rfc-editor.org/info/rfc6570>.

[RFC7807] Nottingham, M. and E. Wilde, "Problem Details for HTTP APIs", RFC 7807, DOI 10.17487/RFC7807, March 2016, <https://www.rfc-editor.org/info/rfc7807>.

[RFC8543] Zhou, L., Kong, N., Yao, J., Gould, J., and G. Zhou, "Extensible Provisioning Protocol (EPP) Organization Mapping", RFC 8543, DOI 10.17487/RFC8543, March 2019, <https://www.rfc-editor.org/info/rfc8543>.

[RFC8544] Zhou, L., Kong, N., Wei, J., Yao, J., and J. Gould, "Organization Extension for the Extensible Provisioning Protocol (EPP)", RFC 8544, DOI 10.17487/RFC8544, April 2019, <https://www.rfc-editor.org/info/rfc8544>.

[RFC8807] Gould, J. and M. Pozun, "Login Security Extension for the Extensible Provisioning Protocol (EPP)", RFC 8807, DOI 10.17487/RFC8807, August 2020, <https://www.rfc-editor.org/info/rfc8807>.

[RFC9038] Gould, J. and M. Casanova, "Extensible Provisioning Protocol (EPP) Unhandled Namespaces", RFC 9038, DOI 10.17487/RFC9038, May 2021, <https://www.rfc-editor.org/info/rfc9038>.

[RFC9083] Hollenbeck, S. and A. Newton, "JSON Responses for the Registration Data Access Protocol (RDAP)", STD 95, RFC 9083, DOI 10.17487/RFC9083, June 2021, <https://www.rfc-editor.org/info/rfc9083>.

[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>.

[RFC9154] Gould, J. and R. Wilhelm, "Extensible Provisioning Protocol (EPP) Secure Authorization Information for Transfer", RFC 9154, DOI 10.17487/RFC9154, December 2021, <https://www.rfc-editor.org/info/rfc9154>.

[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>.

[RFC9803] Brown, G., "Extensible Provisioning Protocol (EPP) Mapping for DNS Time-to-Live (TTL) Values", RFC 9803, DOI 10.17487/RFC9803, June 2025, <https://www.rfc-editor.org/info/rfc9803>.

[RFC9873] Belyavsky, D., Gould, J., and S. Hollenbeck, "Additional Email Address Extension for the Extensible Provisioning Protocol (EPP)", RFC 9873, DOI 10.17487/RFC9873, October 2025, <https://www.rfc-editor.org/info/rfc9873>.

[RFC9874] Hollenbeck, S., Carroll, W., and G. Akiwate, "Best Practices for Deletion of Domain and Host Objects in the Extensible Provisioning Protocol (EPP)", BCP 244, RFC 9874, DOI 10.17487/RFC9874, September 2025, <https://www.rfc-editor.org/info/rfc9874>.

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

[ROI] Richardson, L. and S. Ruby, "RESTful Web Services, Chapter 4", 2007, <https://www.oreilly.com/library/view/restful-web-services/9780596529260/ch04.html>.

23.2. Informative References

[RFC3375] Hollenbeck, S., "Generic Registry-Registrar Protocol Requirements", RFC 3375, DOI 10.17487/RFC3375, September 2002, <https://www.rfc-editor.org/info/rfc3375>.

[TigerTeamRecc] RPP WG, Gould, J., Kolker, J., Kowalik, P., Skoglund, E., and M. Wullink, "EPP Extensibility and Extension Analysis", 2025, <https://docs.google.com/document/d/1WR00oB43XZCDqD0zvRvRajuWAq_9wQ3c0RrFKlGC3So>.

Appendix A. Extensions

A.1. Essential extensions

> TODO: These lists are far from being complete -> input from Tiger Team on EPP Extensibility will fill these lists

The following list of extensions is considered essential for the completeness of RPP as provisioning protocol for domain names. The core RPP and its extensibility framework MUST enable creation of those extensions.

A.1 *Moved to Appendix A.2 as A2.2*

A1.1 An extension that allows a DNS operator to update the DNSSEC key material for a domain object. This extension MAY be used by the DNS operator to update the DNSSEC key material for a domain object, without the need for the registrar to be involved in this process.

A.2. Optional extensions

The following list of extensions is considered as possible need for certain deployments of RPP, however other solutions outside of RPP would be possible. Therefore RPP and its extensibility framework MAY enable creation of those extensions, however it is not a MUST criteria.

A2.1 An extension that allows 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 ist included in the discovery service document.

> TODO: Issue #57

A2.2 An extension for a Search API to allow for searching for objects in the registry database. Includes advanced search capabilities for object info request.

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/