Workgroup:	Network Working Group
Internet-Draft:	draft-kowalik-rpp-data-objects-02
Published:	14 November 2025
Intended Status:	Standards Track
Expires:	18 May 2026
Authors:	P. Kowalik	M. Wullink
---	---	---
	DENIC	*SIDN Labs*

RESTful Provisioning Protocol (RPP) Data Objects

Abstract

This document defines data objects for the RESTful Provisioning Protocol (RPP) and sets up IANA RPP Data Object Registry to describe and catalogue them. Specifically, it details the logical structure, constraints, and protocol operations (including their inputs and outputs) for foundational resources: domain names, contacts, and hosts. In accordance with the RPP architecture [I-D.kowalik-rpp-architecture], these definitions focus entirely on the semantics, remaining independent of any specific data representation or media type (e.g., JSON or XML).

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 18 May 2026.

Copyright Notice

Table of Contents

1. Introduction

The RESTful Provisioning Protocol (RPP) defines a set of data objects used to represent and manage foundational registry resources, including domain names, contacts, and hosts. This initial list is not exhaustive; additional resource and component objects MAY be defined in future revisions or introduced via IANA registration to support new features and operational needs.

In accordance with the RPP architecture [I-D.kowalik-rpp-architecture], a core architectural principle is the clear distinction between the abstract data model and its concrete data representation. The data model defines the logical structure, relationships, and constraints of the objects, independent of formatting. The data representation defines how these abstract concepts are expressed in specific formats (e.g., JSON, XML, or YAML).

This document focuses on the data model of RPP objects and operations on them, including the data model of operation inputs and outputs. This separation of concerns ensures the protocol maintains a stable semantic foundation that can be consistently implemented across different media types and easily adapted to new representation formats. For instance, the model defines a contact's name as a required string type, but it remains agnostic as to whether that string is ultimately encoded as a JSON property or an XML element.

1.1. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Resource Definition Principles

2.1. Primitive Data Types

RPP data elements use strict typing, meaning that each element must conform exactly to its declared primitive data type, and type violations MUST be treated as errors by implementations. The exact specifications for these types, including allowed ranges, encoding, and formatting, are determined by the representation format used (e.g., JSON, XML, CBOR). New RPP non-primitive data types based on existing primitive data types MAY be defined to support additional features.

2.1.1. String

A String is a sequence of Unicode characters. Usages MAY impose additional constraints on string values, such as maximum length or allowed character sets, based on specific data element definitions. An example of a string is `"host.example"`.

2.1.2. Integer

An Integer is a whole number, positive or negative. Usages MAY impose additional constraints on integer values, such as minimum and maximum allowable values, based on specific data element definitions. An example of an integer is `42`.

2.1.3. Boolean

A Boolean represents a logical true or false value. An example of a boolean is `true` or `false`, but it MAY be different depending on the native encoding of boolean values in the representation.

2.1.4. Decimal

Decimal is a number providing an exact, base-10 representation of fractional values within a defined precision. Usage of this type MUST impose additional constraints on decimal values, such as precision or range, based on specific data element definitions. An example of a decimal is 3.14159.

2.1.5. Date

A Date is a full-date calendar date as described in [RFC3339], an example of a date is `2025-10-27`.

2.1.6. Timestamp

Timestamp (Date and time attribute) values MUST be represented in Universal Coordinated Time (UTC) using the Gregorian calendar using date-time form as defined in [RFC3339]. In EPP Compatibility Profile upper case "T" and "Z" characters MUST be used. An example of a timestamp is `2025-10-27T09:42:51Z`.

2.1.7. URL

A Uniform Resource Locator (URL) as defined in [RFC1738]. An example of a URL is `"https://host.example"`.

2.1.8. Binary

Raw binary data, implementations MAY choose how to encode the binary data, for example as base64 or hexadecimal string. An example of binary data encoded as base64 is `"UlBQIFNheXMgSGk="`.

2.2. Data Element Abstraction

Each data object is composed of logical data elements. A data element is a logical unit of information identified by a stable name, independent of its representation in any given media type. The definition for each element specifies its logical name, purpose, cardinality, data type, and constraints. The data type of a data element may also be a reference to another data object, using the target object's stable name.

2.3. Extensibility

The set of data elements for a given data object is extensible. New data elements, associations or operations MAY be defined and registered with IANA in order for the data object to support new features.

2.4. Data Element Semantics

The definition of each data element within an object consists of the following attributes:

- Name: A human-readable name for the data element.
- Identifier: A machine-readable, unique identifier for the element, using camelCase notation.
- Cardinality: Specifies the number of times an element may appear. The notation is as follows:
  - `1` for exactly one
  - `0-1` for zero or one
  - `0+` for zero or more
  - and `1+` for one or more
- Data Type: Defines the element's data structure, which can be a primitive type, a Common Data Type or a reference to another component object.

- Description: Explains the purpose of the data element and any other relevant information.
- Constraints: Provides specific validation rules or limitations on top of the data type itself, such as value ranges.
- Mutability: Defines the lifecycle of the data element's value. It MUST be one of the following:
  - create-only: The element's value is provided during the object's creation and cannot be modified thereafter.
  - read-only: The element's value is managed by the server. It cannot be set or modified directly by the client, though it may change as a result of server-side operations.
  - read-write: The element's value can be set and modified by the client.

2.5. Operations

For each data object a set of possible operations is defined together with their respective input and output data.

2.5.1. Authorisation

For each operation authorisation requirements and operation behaviour is specified. Wherever "object authorisation" is mentioned, it means that an operation MAY accept or require additional authorisation data related to the object beyond default client-level authorisation, or that an operation MAY have different effect or response if such authorisation is provided. Typically it would be a value related to or derived from Authorisation Information Object attached to the object.

2.5.2. Uniform interface

For the typical set of Create, Read, Update and Delete operations the following set of input and output data model is specified on top of additional transient input data, unless an operation for the specific object tells otherwise.

2.5.2.1. Create

- Input: Object representation (create-only and read-write properties)
- Output: Object representation (read-write and read-only properties)

2.5.2.2. Read

- Input: Object identifier
- Output: Object representation (read-write and read-only properties)

The output representation MAY vary depending on the identity of the querying client, use of object authorisation information, and server policy towards unauthorized clients. If the querying client is the sponsoring client, all available information MUST be returned. If the querying client is not the sponsoring client but the client provides valid object authorisation information, all available information SHOULD be returned, however some optional elements MAY be reserved to the sponsoring client only. If the querying client is not the sponsoring client and the client does not provide valid object authorisation information, server policy determines which OPTIONAL elements are returned, if any, or whether the entire request is rejected.

2.5.2.3. Update

- Input: Object identifier, Object changes representation (read-write properties)
- Output: Object representation (read-write and read-only properties)

2.5.2.4. Delete

- Input: Object identifier
- Output: Object representation (read-write and read-only properties) or no representation

2.5.3. Operations beyond uniform interface

For all other operations both input and output representation have to be fully specified.

2.6. EPP Compatibility Profile

RPP is designed to coexist with the Extensible Provisioning Protocol (EPP), often operating in parallel against a common backend provisioning system. While RPP is not inherently constrained by all of EPP's requirements, a specific set of rules is necessary to ensure seamless interoperability in such mixed environments.

To address this, this document defines an "EPP Compatibility Profile". This profile specifies a set of additional constraints on RPP data objects and operations that a server MUST adhere to when supporting both RPP and EPP concurrently.

Throughout this document, all constraints that are part of this profile are explicitly marked with a reference to "EPP Compatibility Profile". Implementers of systems in a mixed EPP/RPP environment MUST follow these specific constraints in addition to the base RPP requirements.

3. Common Data Types

This section defines new shared data types and structures that are re-used across multiple data object definitions and are based on the existing Primitive Data Types.

3.1. Identifier

Identifiers are character strings with a specified minimum length, a specified maximum length, and a specified format outlined in Section 2.8 of [RFC5730]. Identifiers for certain object types MAY have additional constraints imposed either by server policy, object specific specifications or both.

3.2. Client Identifier

Client identifiers are character strings with a specified minimum length, a specified maximum length, and a specified format. Contact identifiers use the "clIDType" client identifier syntax described in [RFC5730].

> TBC: do we need this or is it a relation with an entity/RFC8543 organisation? If registrars modeled are as first class objects (organisations), then clID is nothing else but a reference to this organisation, so maybe no need to define syntax separately on identifier level (or in other words it would be defined on this object). R8.1 in the form of -02 RPP requirements includes RFC8543.

3.3. Phone Number

Telephone number syntax is derived from structures defined in [ITU.E164.2005]. Telephone numbers described in this specification are character strings that MUST begin with a plus sign ("+", ASCII value 0x002B), followed by a country code defined in [ITU.E164.2005], followed by a dot (".", ASCII value 0x002E), followed by a sequence of digits representing the telephone number. An optional "x" (ASCII value 0x0078) separator with additional digits representing extension information can be appended to the end of the value.

4. Associations

RPP allows for different types of associations (relationship) between the objects. The association may be added between 2 objects with own indpendent lifecycle (UML aggregation) or in the relation when one object's existance and lifecycle is bound to the other parent/owner object (UML composition). In both cases, especially if the relation allows for cardinality higher than one on either side, the association may be assigned additional attributes, not being part of an object on either side of relation. In many cases such relation would be attributed with a single text string label, describing a role or a type of relation. Depending on the context this value might be unique, which allows using such label as a key in a dictionary.

The following generic Association Types are defined for RPP:

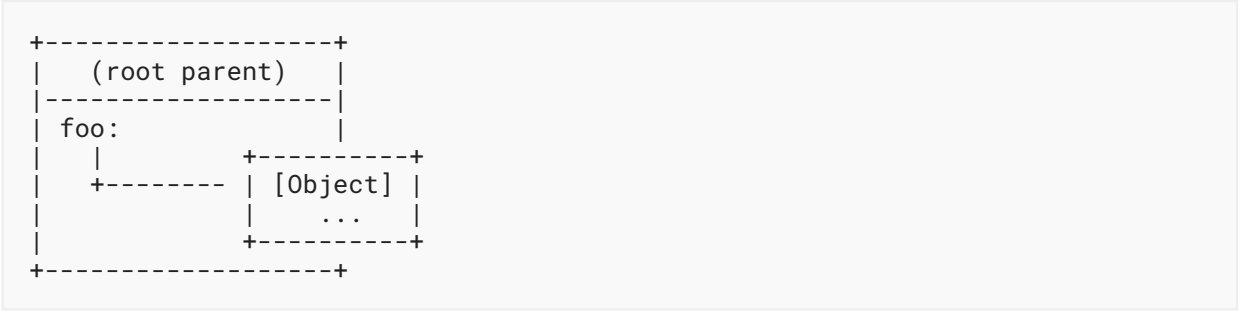
4.1. Aggregation

Notation: Aggregation[Type]

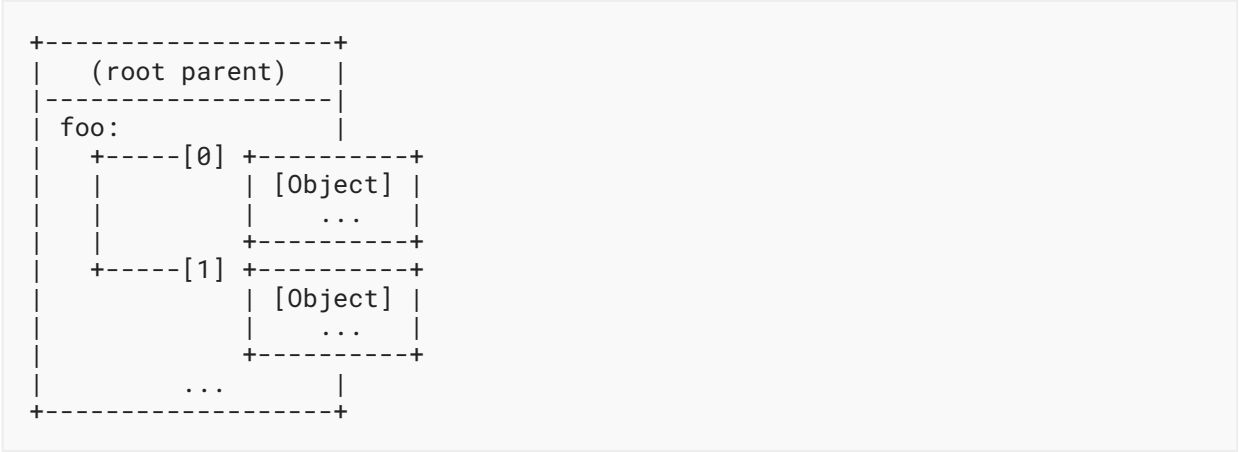
A relation between two independent objects.

If the cardinality of target object is more than 1, this represents an ordered array. It MUST assured that the same unchanged data is always inserted in the same order in order to allow stable reference by position to data elements. In case of data insertions, deletions or updates the remaining of the data SHALL preserve its order.

Example aggregation having cardinality 1:



Example aggregation having cardinality >1:



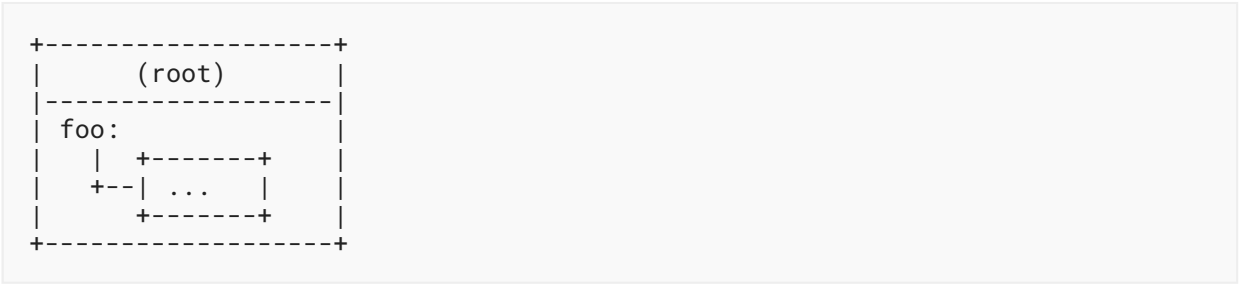
4.2. Composition

Notation: Composition[Type] or Type

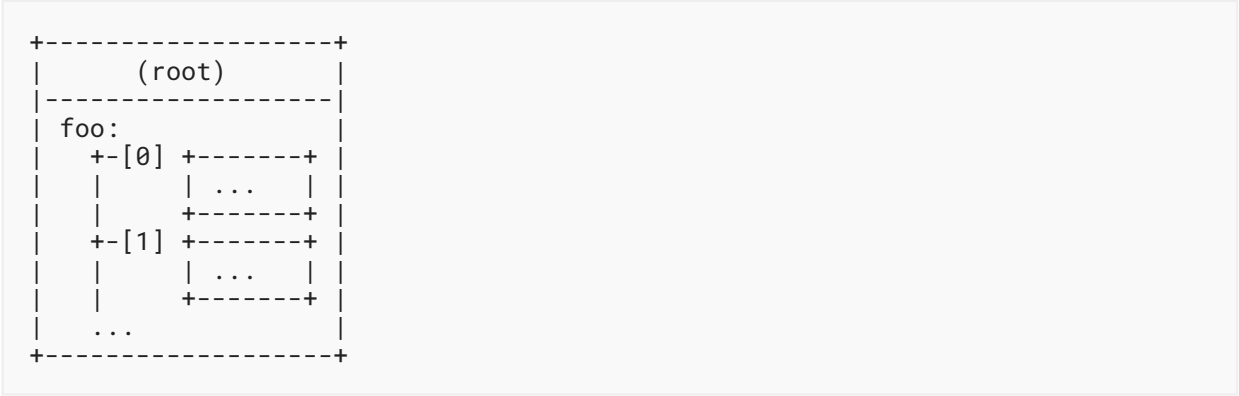
A relation between an independent parent object and 1 or more dependent child object(s).

If the cardinality of target object is more than 1, this represents an ordered array. It MUST assured that the same unchanged data is always inserted in the same order in order to allow stable reference by position to data elements. In case of data insertions, deletions or updates the remaining of the data SHALL preserve its order.

Example composition having cardinality 1:



Example composition having cardinality >1:



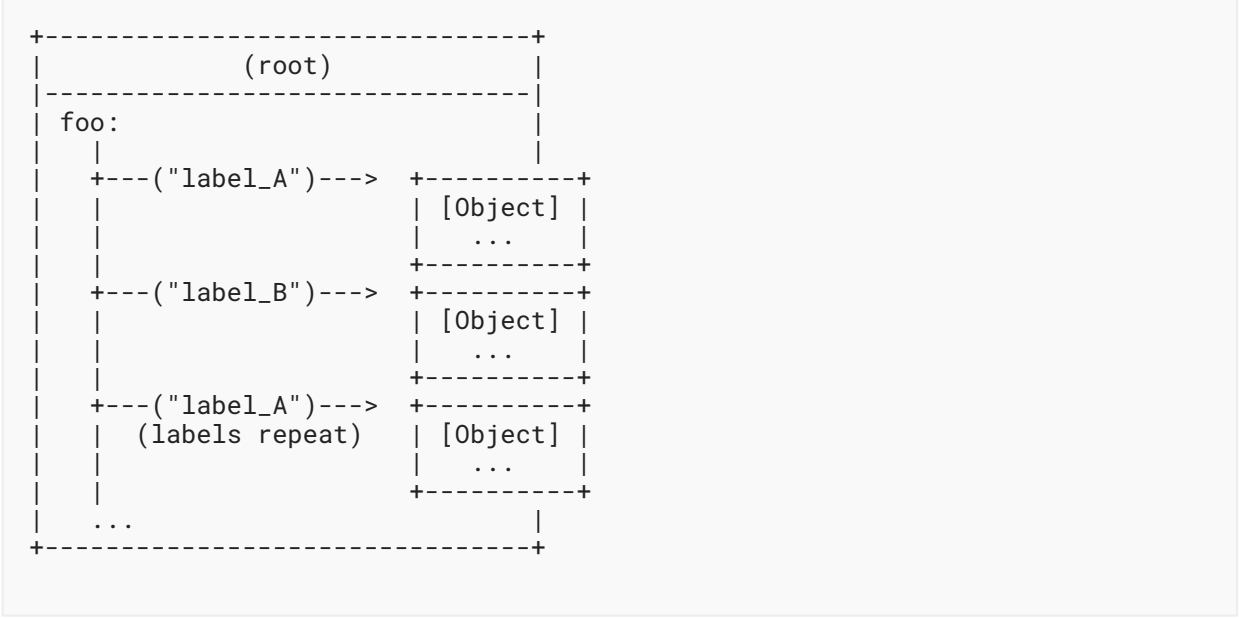
4.3. Labelled Aggregation

Notation: LabelledAggregation[Type]

A relation between two independent object with single text string attribute. Multiple associations with the same label are allowed and represent an unordered array.

A type defining such association MUST define Label Description with semantics of the label and Label Constraints with constraints related to the label.

Example labelled aggregation:



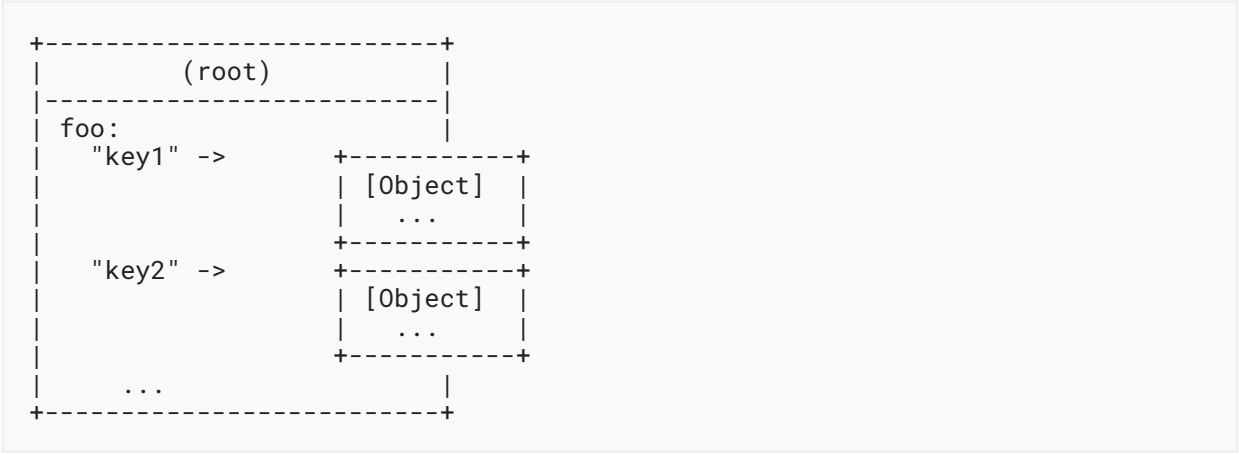
4.4. Aggregation Dictionary

Notation: AggregationDictionary[Type]

A relation between two independent object with single text string attribute. Association labels MUST be unique allowing it to be used as dictionary key.

A type defining such association MUST define Label Description with semantics of the label and Label Constraints with constraints related to the label.

Example Aggregation Dictionary:



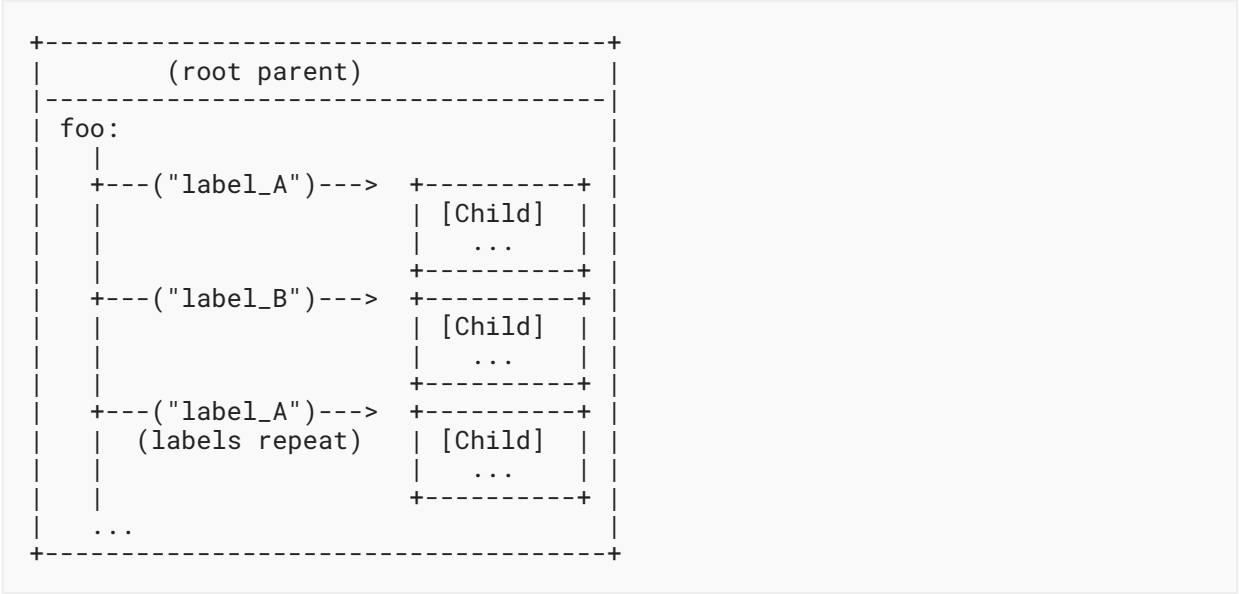
4.5. Labelled Composition

Notation: LabelledComposition[Type]

A relation between an independent parent object and a dependent child object with single text string attribute. Multiple associations with the same label are allowed.

A type defining such association MUST define Label Description with semantics of the label and Label Constraints with constraints related to the label.

Example Labelled Composition:



4.6. Composition Dictionary

Notation: CompositionDictionary[Type]

A relation between an independent parent object and a dependent child object with single text string attribute. Only single association with the same label is allowed allowing it to be used as dictionary key.

A type defining such association MUST define Label Description with semantics of the label and Label Constraints with constraints related to the label.

Example Composition Dictionary:

5. Component Objects

This section defines common objects that are re-used in the definitions of top-level data objects. Component objects carry only data but do not define any operations.

5.1. Period Object

- Name: Period Object
- Description: Represents a duration of time.
- Data Elements:
  - Value
    - Identifier: value
    - Cardinality: 1
    - Mutability: read-write
    - Data Type: Integer
    - Description: The numeric value of the period.
    - Constraints: The value MUST be from 1 to 99, inclusive.
  - Unit
    - Identifier: unit
    - Cardinality: 1
    - Mutability: read-write
    - Data Type: String
    - Description: The unit of the period.
    - Constraints: The value MUST be one of: "y" (years) or "m" (months).

5.2. Status Object

- Name: Status Object
- Description: Represents one of the status values associated with a provisioning object
- Data Elements:
  - Label
    - Identifier: label
    - Cardinality: 1
    - Mutability: create-only
    - Data Type: String
    - Description: machine-readible enum label of a status
    - Constraints:
      - Exact list of allowed status labels depends on the provisioning object type. This enumeration can be expanded by extensions.

        - The status labels MUST use camel case notation and use only ASCII alphabetic characters.
        - Statuses MAY be of three categories:

        1. those explicitly set by a server. Those MUST have "server" prefix
        2. those explicitly set by a client. Those MUST have "client" prefix
        3. those indirectly controlled by provisioning object lifecycle or business logic. Those MUST NOT use either "client" or "server" prefix. They MAY use another prefix or no prefix at all

    - Reason
        - Indentifier: reason
        - Cardinality: 0-1
        - Mutability: create-only
        - Data Type: String
        - Description: a human-readable text that describes the rationale for the status applied to the object.
        - Constraints: None

    - Due
        - Identifier: due
        - Cardinality: 0-1
        - Mutability: read-write
        - Data Type: Timestamp
        - Description: a timestamp, when this status is going to be removed automatically, or changed to other status. This field can be used to expresse lifecycle related information.
        - Constraints: servers MAY restrict possibility to set or update this value by the client.

> TBD: Idea - model status object as Labelled Composition using "Label"? Con: Generic Constraints for Label will be repeated.

5.3. Nameserver Object

- Name: Nameserver Object
- Description: Represents a single nameserver.
- Data Elements:
    - Host Name
        - Identifier: hostName
        - Cardinality: 1
        - Mutability: read-write
        - Data Type: String
        - Description: Fully qualified name of a host.

        - Constraints: The value MUST be a syntactically valid host name.

    - DNS Resource Records
        - Identifier: dns
        - Cardinality: 0+
        - Mutability: read-write
        - Data Type: Composition[DNS Resource Record]
        - Description: DNS Resource Records related to the host.
        - Constraints:
            - In EPP Compatibility Profile the entries MUST be limited to A and AAAA entries for IPv4 and IPv6 glue records respectively.
            - The labels of DNS entries MUST be subordinate to the Host Name of the Nameserver.

5.4. DNS Resource Record

- Name: DNS Resource Record
- Description: Represents a DNS Entry
- Data Elements:
    - Label
        - Identifier: hostNamelabel
        - Cardinality: 1
        - Mutability: read-write
        - Data Type: String.
        - Description: DNS entry label.
        - Constraints:
            - The value MUST be a syntactically valid DNS host name in Zone file string representation.
            - Absolute FQDNs and relative host names are allowed.

    - Type
        - Identifier: type
        - Cardinality: 1
        - Mutability: read-write
        - Data Type: String.
        - Description: DNS entry type.
        - Constraints:
            - Each value MUST be a valid string representation of resource record type as defined in [RFC1035].
            - Allowed values MAY be constrained by server policies. For domain provisioning typically the Type would be constrained to the allowed parent side entries.

- Data
  - Identifier: data
  - Cardinality: 1
  - Mutability: read-write
  - Data Type: String.
  - Description: DNS entry value.
  - Constraints: Each value MUST be a syntactically valid resource record data for a Type in zone file string representation.
- TTL
  - Identifier: ttl
  - Cardinality: 1
  - Mutability: read-write
  - Data Type: Number.
  - Description: TTL value of a resource record as defined in [RFC1035].
  - Constraints: The allowed value range MAY be constrained by server policy.

5.5. Authorisation Information Object

- Name: Authorisation Information
- Description: Contains information used to authorise operations on a data object. It may hold different kind of authorisation information.
- Data Elements:
  - Method
    - Identifier: method
    - Cardinality: 1
    - Mutability: create-only
    - Data Type: String
    - Description: The identifier of the RPP authorisation method.
    - Constraints:
      - The value MUST be one of the values registered at IANA as defined in [I-D.draft-wullink-rpp-core].
      - In EPP Compatibility Profile this value MUST be set to `authinfo` if standard password base authorisation is used
  - Authorisation Information
    - Identifier: authdata
    - Cardinality: 1
    - Mutability: create-only
    - Data Type: String

        - Description: The value of the authorisation information. It might be as simple as password string, but also more complex values like public key certificates or tokens encoded as string are possible.
        - Constraints:
            - Authorisation Information object is immutable. If the information changes (for example password is updated) a new instance MUST be created.
            - Depending on the method and server policy Authorisation Information MAY not be available for read or any other operation responding with this data element.

5.6. Postal Address Object

- Name: Postal Address Object
- Description: Contains the components of a postal address.
- Data Elements:
    - Street
        - Identifier: street
        - Cardinality: 0+
        - Mutability: read-write
        - Data Type: String
        - Description: The contact's street address.
        - Constraints: Implementations MAY limit the maximum length of entries or character set.
    - City
        - Identifier: city
        - Cardinality: 0-1
        - Mutability: read-write
        - Data Type: String
        - Description: The contact's city.
        - Constraints:
            - Implementations MAY limit the maximum length of entries or character set.
            - In EPP Compatibility Profile this data element MUST be provided.
    - State/Province
        - Identifier: sp
        - Cardinality: 0-1
        - Mutability: read-write
        - Data Type: String
        - Description: The contact's state or province.
        - Constraints: Implementations MAY limit the maximum length of entries or character set.

- Postal Code
  - Identifier: pc
  - Cardinality: 0-1
  - Mutability: read-write
  - Data Type: String
  - Description: The contact's postal code.
  - Constraints:
    - Implementation MAY limit the maximum length of entries or character set.
    - The limitations MAY differ depending on Country Code (`cc`) data element.
- Country Code
  - Identifier: cc
  - Cardinality: 0-1
  - Mutability: read-write
  - Data Type: String
  - Description: The contact's country code.
  - Constraints:
    - The value MUST be a two-character identifier from [ISO3166-1].
    - In EPP Compatibility Profile this data element MUST be provided.

5.7. Postal Info Object

- Name: Postal Info Object
- Description: Contains postal-address information in either internationalised or localised forms.
- Data Elements:

> TBC: Contact Type is not localised (shall be the same for PERSON and ORG). Moving it level up would however detach it from related/dependant fields Name/ Organisation

- Contact Type
  - Identifier: type
  - Cardinality: 0-1
  - Mutability: read-write
  - Data Type: String
  - Description: Specifies whether the contact is and individual or an organisation.
  - Constraints: The value MUST be one of: "PERSON" (individual) or "ORG" (organisation).
- Name
  - Identifier: name
  - Cardinality: 0-1

    - Mutability: read-write
    - Data Type: String
    - Description: The name of the individual or role.
    - Constraints:
        - Implementations MAY limit the maximum length of entries or character set.
        - In EPP Compatibility Profile this data element MUST be provided.
        - The implementations MAY require this field if Contact Type (`type`) is set to "PERSON".

- Organisation
    - Identifier: org
    - Cardinality: 0-1
    - Mutability: read-write
    - Data Type: String
    - Description: The name of the organisation.
    - Constraints:
        - Implementations MAY limit the maximum length of entries or character set.
        - The implementations MAY require this field if Contact Type (`type`) is set to "ORG".

- Address
    - Identifier: addr
    - Cardinality: 0-1
    - Mutability: read-write
    - Data Type: Postal Address Object
    - Description: The detailed postal address.
    - Constraints: In EPP Compatibility Profile this data element MUST be provided.

5.8. Disclose Object

> TODO: Model Disclose in universal (extendible) way

- Name: Disclose
- Identifier: disclose
- Description: TBD

6. Domain Name Data Object

6.1. Object Description

- Name: Domain Name Data Object
- Identifier: domainName

- Description: A Domain Name data object represents a domain name and contains the data required for its provisioning and management in the registry.

6.2. Data Elements

The following data elements are defined for the Domain Name Data Object.

- Name
  - Identifier: name
  - Cardinality: 1
  - Mutability: create-only
  - Data Type: String
  - Description: The fully qualified name of the domain object.
  - Constraints:
    - The value MUST be a fully qualified domain name that conforms to the syntax described in [RFC1035].
    - A server MAY restrict allowable domain names to a particular top-level domain, second-level domain, or other domain for which the server is authoritative.
    - The trailing dot required when these names are stored in a DNS zone is implicit and MUST NOT be provided when exchanging host and domain names.
- Repository ID
  - Identifier: repositoryId
  - Cardinality: 0-1
  - Mutability: read-only
  - Data Type: Identifier
  - Description: A server-assigned unique identifier for the object. In EPP Compatibility Profile this data element MUST be provided.
  - Constraints: (None)
- Status
  - Identifier: status
  - Cardinality: 0+
  - Mutability: read-only
  - Data Type: Status Object
  - Description: The current status descriptors associated with the domain.
  - Constraints: Possible combinations of Status Object Labels is specified in Section 2.3 of [RFC5731] and [RFC3915]

> TBC: IANA registry for statuses?

- Registrant
  - Identifier: registrant
  - Cardinality: 0-1
  - Mutability: read-write
  - Data Type: Contact Object.
  - Description: The contact object associated with the domain as the registrant.
  - Constraints:
    - The relation MUST correspond to a valid Contact Data Object known to the server.
    - Servers MAY restrict association of a Contact Object of a different sponsoring client.

> TBC: leave registrant here or move it to contacts with a type?

- Contacts
  - Identifier: contacts
  - Cardinality: 0+
  - Mutability: read-write
  - Data Type: LabelledAggregation[Contact Object]
    - Label Description: The role of the associated contact.
    - Label Constraints:
      - List of supported roles is defined by server policy
      - In the EPP Compatibility Profile, the value MUST be one of: "admin", "billing", or "tech"
  - Description: A collection of other contact objects associated with the domain object.
  - Constraints:
    - Maximum number of associated contacts (per role) MAY be restricted by server policy

> TBC: IANA registry for contact role label?

- Nameservers
  - Identifier: nameservers
  - Cardinality: 0+
  - Mutability: read-write
  - Data Type: Composition[Host Data Object] or Aggregation[Host Data Object]
  - Description: A collection of nameservers associated with the domain.
  - Constraints: (None)

- DNS
  - Identifier: dns
  - Cardinality: 0+
  - Mutability: read-write
  - Data Type: Composition[DNS Resource Record]
  - Description: A collection of DNS entries related to the domain name.
  - Constraints:
    - The Type of the entries MAY be constrained by the server policy. Typically the values would be limited to allowed parent side resource record types.
    - In EPP Compatibility Profide with DNSSEC Extension allowed values MUST be DS and DNSKEY.
    - The labels of DNS entries MUST be subordinate to the domain name and MUST NOT be below zone cut in case of present delegation.
- Subordinate Hosts
  - Identifier: subordinateHosts
  - Cardinality: 0+
  - Mutability: read-only
  - Data Type: Aggregation[Host Object]
  - Description: A collection of subordinate host objects that exist under this domain.
  - Constraints: (None)
- Sponsoring Client ID
  - Identifier: sponsoringClientId
  - Cardinality: 1
  - Mutability: read-only
  - Data Type: Client Identifier.
  - Description: The identifier of the client that is the current sponsor of the domain object.
  - Constraints: (None)
- Creating Client ID
  - Identifier: creatingClientId
  - Cardinality: 0-1
  - Mutability: read-only
  - Data Type: Client Identifier.
  - Description: The identifier of the client that created the domain object.
  - Constraints: (None)
- Creation Date
  - Identifier: creationDate

- Cardinality: 0-1
- Mutability: read-only
- Data Type: Timestamp.
- Description: The date and time of domain object creation.
- Constraints: The value is set by the server and cannot be specified by the client.

• Updating Client ID

- Identifier: updatingClientId
- Cardinality: 0-1
- Mutability: read-only
- Data Type: Client Identifier.
- Description: The identifier of the client that last updated the domain object.
- Constraints: This element MUST NOT be present if the domain has never been modified.

• Update Date

- Identifier: updateDate
- Cardinality: 0-1
- Mutability: read-only
- Data Type: Timestamp.
- Description: The date and time of the most recent domain object modification.
- Constraints: This element MUST NOT be present if the domain object has never been modified.

• Expiry Date

- Identifier: expiryDate
- Cardinality: 0-1
- Mutability: read-only
- Data Type: Timestamp.
- Description: The date and time identifying the end of the domain object's registration period.
- Constraints: The value is set by the server and cannot be specified by the client.

• Transfer Date

- Identifier: transferDate
- Cardinality: 0-1
- Mutability: read-only
- Data Type: Timestamp
- Description: The date and time of the most recent successful domain object transfer.
- Constraints: This element MUST NOT be provided if the domain object has never been transferred.

- Authorisation Information
  - Identifier: authInfo
  - Cardinality: 0-1
  - Mutability: read-write
  - Data Type: Authorisation Information Object
  - Description: Authorisation information associated with the domain object.
  - Constraints: (None)

6.3. Operations

6.3.1. Create Operation

The Create operation allows a client to provision a new Domain Name resource. The operation accepts as input all create-only and read-write data elements defined for the Domain Name Data Object.

- Authorisation:
  - Generally each client is authorised to create new domain objects becoming a sponsoring client. This can be however constrained by the server policy in many ways, i.e. by applying rate limiting, billing related constraints or compliance locks.

In addition, the following transient data element is defined for this operation:

- Registration Period
  - Identifier: period
  - Cardinality: 0-1
  - Data Type: Period Object.
  - Description: The initial registration period for the domain name. This value is used by the server to calculate the initial `expiryDate` of the object. This element is not persisted as part of the object's state.

6.3.2. Read Operation

The Read operation allows a client to retrieve the data elements of a Domain Name resource. The server's response MAY vary depending on client authorisation and server policy.

- Authorisation:
  - Sponsoring client:
    - Full object representation
  - Other client:
    - Without object authorisation:
      - Limited (non-confidential) object representation or operation denied
    - With object authorisation:
      - Full object representation, however some properties only authorised to the sponsoring client MAY be redacted according to server policy

The following transient data elements are defined for this operation:

- Hosts Filter
  - Identifier: hostsFilter
  - Cardinality: 0-1
  - Data Type: String
  - Description: Controls which host information is returned with the object.
  - Constraints: The value MUST be one of "all", "del" (delegated), "sub" (subordinate), or "none". The default value is "all".

6.3.3. Delete Operation

The Delete operation allows a client to remove an existing Domain Name resource. The operation targets a specific data object identified by its name.

- Authorisation:
  - Only sponsoring client is authorised to perform this operation

The server SHOULD reject a delete request if subordinate host objects are associated with the domain name.

The error response SHOULD indicate the related subordinate host objects.

6.3.4. Renew Operation

The Renew operation allows a client to extend the validity period of an existing Domain Name resource. The operation targets a specific data object identified by its name.

- Authorisation:
  - Only sponsoring client is authorised to perform this operation
- Input: Domain Name
- Output: Full object representation (read-write and read-only properties), or a minimum representation of properties affected by the operation (Expiry Date).

The following transient data elements are defined for this operation:

- Current Expiry Date
  - Identifier: currentExpiryDate
  - Cardinality: 1
  - Data Type: Timestamp
  - Description: The current expiry date of the domain name. The server MUST validate this against the object's current `expiryDate` to prevent unintended duplicate renewals.
- Renewal Period
  - Identifier: renewalPeriod
  - Cardinality: 0-1

- Data Type: Period Object
- Description: The duration to be added to the object's registration period. This value is used by the server to calculate the new `expiryDate`. The default value MAY be defined by server policy. The number of units available MAY be subject to limits imposed by the server.

6.3.5. Transfer operation

> TODO: define transfer operation https://github.com/pawel-kow/draft-kowalik-rpp-data-objects/issues/23

7. Contact Data Object

7.1. Object Description

- Name: Contact Data Object
- Identifier: contact
- Description: A Contact Data Object represents the social information for an individual or organisation associated with other objects.

7.2. Data Elements

The following data elements are defined for the Domain Name Data Object.

- Handle ID
  - Identifier: id
  - Cardinality: 1
  - Mutability: create-only
  - Data Type: Identifier.
  - Description: External unique identifier of the contact object.
  - Constraints:
    - This value MUST be supported to be provided by the client.
    - Servers MAY support server-side generation of this value.
- Repository ID
  - Identifier: repositoryId
  - Cardinality: 0-1
  - Mutability: read-only
  - Data Type: Identifier
  - Description: A server-assigned unique identifier for the object.
  - Constraints: In EPP Compatibility Profile this data element MUST be provided.

- Postal Information
  - Identifier: postalInfo
  - Cardinality: 1-2
  - Mutability: read-write
  - Data Type: AggregationDictionary[Postal Info Object]
    - Label Description: type of contact data localisation
    - Label Constraints: Allowed values: "int" for "internationalised" all-ASCII version of an address and "loc" for localised forms with possible non-ASCII character sets.
  - Description: Contains postal-address information.
  - Constraints: There MUST be no more that 1 element of type "int" and one element of type "loc".
- Voice Phone Number
  - Identifier: voice
  - Cardinality: 0+
  - Mutability: read-write
  - Data Type: Phone Number
  - Description: Voice phone number associated with the contact
  - Constraints: (None)
- Fax Phone Number
  - Identifier: fax
  - Cardinality: 0+
  - Mutability: read-write
  - Data Type: Phone Number
  - Description: Fax number associated with the contact
  - Constraints: (None)
- E-mail
  - Identifier: email
  - Cardinality: 0+
  - Mutability: read-write
  - Data Type: String.
  - Description: The contact's email address.
  - Constraints: Email address syntax is defined in [RFC5322].
- Status
  - Identifier: status
  - Cardinality: 0+
  - Mutability: read-only

- Data Type: Status Object
- Description: The current status descriptors associated with the contact.
- Constraints:
  - Description: The current status descriptors associated with the contact.
  - Constraints: Possible combinations of Domain Status Labels is specified in Section 2.2 of [RFC5733]
  - The value MUST be one of the status tokens defined in the IANA registry for domain statuses.
  - The initial value list MAY be as defined in [RFC5733]. In this case the values MUST have the same semantics.

- Sponsoring Client ID
  - Identifier: sponsoringClientId
  - Cardinality: 1
  - Mutability: read-only
  - Data Type: Client Identifier.
  - Description: The identifier of the client that is the current sponsor of the domain object.
  - Constraints: (None)

- Creating Client ID
  - Identifier: creatingClientId
  - Cardinality: 0-1
  - Mutability: read-only
  - Data Type: Client Identifier.
  - Description: The identifier of the client that created the contact object.
  - Constraints: (None)

- Creation Date
  - Identifier: creationDate
  - Cardinality: 0-1
  - Mutability: read-only
  - Data Type: Timestamp.
  - Description: The date and time of contact object creation.
  - Constraints: The value is set by the server and cannot be specified by the client.

- Updating Client ID
  - Identifier: updatingClientId
  - Cardinality: 0-1
  - Mutability: read-only
  - Data Type: Client Identifier.

- Description: The identifier of the client that last updated the contact object.
- Constraints: This element MUST NOT be present if the contact has never been modified.

• Update Date

- Identifier: updateDate
- Cardinality: 0-1
- Mutability: read-only
- Data Type: Timestamp.
- Description: The date and time of the most recent contact object modification.
- Constraints: This element MUST NOT be present if the contact object has never been modified.

• Transfer Date

- Identifier: transferDate
- Cardinality: 0-1
- Mutability: read-only
- Data Type: Timestamp.
- Description: The date and time of the most recent successful contact object transfer.
- Constraints: This element MUST NOT be provided if the contact object has never been transferred.

• Authorisation Information

- Identifier: authInfo
- Cardinality: 0-1
- Mutability: read-write
- Data Type: Authorisation Information
- Description: Authorisation information associated with the contact object.
- Constraints: (None)

• Disclose

- Identifier: disclose
- Cardinality: 0-1
- Mutability: read-write
- Data Type: Disclose Object.
- Description: Identifies elements that require exceptional server-operator handling to allow or restrict disclosure to third parties.

> TBC: IANA registry for statuses?

7.3. Operations

> TODO: Describe operations for contacts https://github.com/pawel-kow/draft-kowalik-rpp-data-objects/issues/15

8. Host Data Object

8.1. Object Description

A Host Data Object represents a name server that provides DNS services for a a domain name.

8.2. Data Elements

The following data elements are defined for the Host Data Object.

> TBC: hostName/dns properties are identical to Nameserver Object. Shall we define something like "Extends"?

- Host Name
  - Identifier: hostName
  - Cardinality: 1
  - Mutability: read-write
  - Data Type: String
  - Description: Fully qualified name of a host.
  - Constraints: The value MUST be a syntactically valid host name.
- DNS Resource Records
  - Identifier: dns
  - Cardinality: 0+
  - Mutability: read-write
  - Data Type: Composition[DNS Resource Record]
  - Description: DNS Resource Records related to the host.
  - Constraints:
    - The labels of DNS entries MUST be subordinate to the Host Name of the Nameserver.
    - In EPP Compatibility Profile the entries MUST be limited to A and AAAA entries for IPv4 and IPv6 glue records respectively.
- Status
  - Identifier: status
  - Cardinality: 0+
  - Mutability: read-only

- Data Type: Status Object
- Description: The current status descriptors associated with the domain.
- Constraints: Possible combinations of Domain Status Labels is specified in Section 2.3 of [RFC5732]

> TBD: this block repositoryId/sponsoringClientId/creationDate/updatingClientId/ updateDate/transferDate is the same as for Domain Name. Shall it be abstracted to a new component object like "Provisioning Metadata"?

- Repository ID
  - Identifier: repositoryId
  - Cardinality: 0-1
  - Mutability: read-only
  - Data Type: Identifier
  - Description: A server-assigned unique identifier for the object. For EPP compatibility this data element is obligatory.
  - Constraints: (None)
- Sponsoring Client ID
  - Identifier: sponsoringClientId
  - Cardinality: 1
  - Mutability: read-only
  - Data Type: Client Identifier.
  - Description: The identifier of the client that is the current sponsor of the host object.
  - Constraints: (None)
- Creating Client ID
  - Identifier: creatingClientId
  - Cardinality: 0-1
  - Mutability: read-only
  - Data Type: Client Identifier.
  - Description: The identifier of the client that created the host object.
  - Constraints: (None)
- Creation Date
  - Identifier: creationDate
  - Cardinality: 0-1
  - Mutability: read-only
  - Data Type: Timestamp.
  - Description: The date and time of host object creation.
  - Constraints: The value is set by the server and cannot be specified by the client.

- Updating Client ID
  - Identifier: updatingClientId
  - Cardinality: 0-1
  - Mutability: read-only
  - Data Type: Client Identifier.
  - Description: The identifier of the client that last updated the host object.
  - Constraints: This element MUST NOT be present if the domain has never been modified.
- Update Date
  - Identifier: updateDate
  - Cardinality: 0-1
  - Mutability: read-only
  - Data Type: Timestamp.
  - Description: The date and time of the most recent host object modification.
  - Constraints: This element MUST NOT be present if the host object has never been modified.
- Transfer Date
  - Identifier: transferDate
  - Cardinality: 0-1
  - Mutability: read-only
  - Data Type: Timestamp
  - Description: The date and time of the most recent successful host object transfer.
  - Constraints: This element MUST NOT be provided if the host object has never been transferred.

8.3. Operations

> TODO: Describe operations for hosts https://github.com/pawel-kow/draft-kowalik-rpp-data-objects/issues/16

9. IANA Considerations

9.1. RPP Data Object Registry

This document establishes the "RESTful Provisioning Protocol (RPP) Data Object Registry". This registry serves as a catalogue of all data objects, component objects, data elements, and operations used within RPP.

9.1.1. Registration Policy

The policy for adding new objects, data elements, or operations to this registry is "Specification Required" [RFC8126].

9.1.2. Registry Structure

The registry is organised as a collection of Object definitions. Each Object definition MUST include:

- A header containing the Object Identifier, Object Name, Object Type (Resource or Component), a brief description, and a reference to its defining specification.
- A "Data Elements" table listing all persisted data elements associated with the object. Each entry MUST specify the element's Identifier, Name, Cardinality, Mutability, Data Type, and description.
- If applicable, an "Operations" section. For each operation, the registry MUST provide:
  - The Operation's Name and a description.
  - A "Parameters" table listing all data elements that are provided as input to the operation but are not persisted as part of the object's state. Each entry MUST specify the parameter's Identifier, Name, Cardinality, Data Type, and a description.

9.1.3. Initial Registrations

The initial contents of the RPP Data Object Registry are defined below.

Object: period

Object Name: Period Object

Object Type: Component

Description: Represents a duration of time.

Reference: [This-ID]

Data Elements

Element Identifier	Element Name	Card.	Mutability	Data Type	Description
value	Value	1	read-write	Integer	The numeric value of the period.
unit	Unit	1	read-write	String	The unit of the period.

Table 1

Object: nameserver

Object Name: Nameserver Object

Object Type: Component

Description: Represents a single nameserver.

Reference: [This-ID]

Data Elements

Element Identifier	Element Name	Card.	Mutability	Data Type	Description
hostName	Host Name	1	read-write	String	The name of the host.
dns	DNS Resource Records	0+	read-write	Composition[DNS Resource Record]	DNS Resource Records related to the host.

Table 2

Object: dnsrr

Object Name: DNS Resource Record

Object Type: Component

Description: Represents a DNS Entry.

Reference: [This-ID]

Data Elements

Element Identifier	Element Name	Card.	Mutability	Data Type	Description
hostNamelabel	Label	1	read-write	String	DNS entry label.
type	Type	1	read-write	String	DNS entry type.
data	Data	1	read-write	String	DNS entry value.
ttl	TTL	1	read-write	Number	TTL value for a reource record.

Table 3

Object: authInfo

Object Name: Authorisation Information

Object Type: Component

Description: Contains authorisation credentials for an operation.

Reference: [This-ID]

Data Elements

Element Identifier	Element Name	Card.	Mutability	Data Type	Description
method	Method	1	create-only	String	The identifier of the RPP authorisation method.
authdata	Authorisation Information	1	create-only	String	The value of the authorisation information. It might be as simple as password string, but also more complex values like public key certificates or tokens encoded as string are possible.

Table 4

Object: domainStatus

Object Name: Status Object

Object Type: Component

Description: Represents one of the status values associated with the provisioning object.

Reference: [This-ID]

Data Elements

Element Identifier	Element Name	Card.	Mutability	Data Type	Description
label	Label	1	create-only	String	machine-reasible enum label of a status
reason	Reason	0-1	create-only	String	a human-readable text that describes the rationale for the status applied to the object.

Element Identifier	Element Name	Card.	Mutability	Data Type	Description
due	Due	0-1	read-write	Timestamp	a timestamp, when this status is going to be removed automatically, or changed to other status. This field can be used to expresse lifecycle related information

Table 5

> TODO: IANA table: Postal Address Object TODO: IANA table: Postal Info Object
> TODO: IANA table: Disclose Object

Object: domainName

Object Name: Domain Name Data Object

Object Type: Resource

Description: Represents a domain name and its associated data.

Reference: [This-ID]

Data Elements

Identifier	Name	Card.	Mutability	Data Type	Description
name	Name	1	create-only	String	The fully qualified name of the domain object.
repositoryId	Repository ID	0-1	read-only	Identifier	A server-assigned unique identifier for the object.
status	Status	0+	read-only	Status Object	The current status descriptors for the domain.

Identifier	Name	Card.	Mutability	Data Type	Description
registrant	Registrant	0-1	read-write	Contact Object	The registrant contact ID.
contacts	Contacts	0+	read-write	LabelledAggregation [Contact Object]	Associated contact objects.
nameservers	Nameservers	0+	read-write	Composition[Host Data Object] or Aggregation[Host Data Object]	A collection of nameservers associated with the domain.
dns	DNS	0+	read-write	Composition[DNS Resource Record]	A collection of DNS entries related to the domain name.
subordinateHosts	Subordinate Hosts	0+	read-only	Aggregation [Host Object]	Subordinate host names.
sponsoringClientId	Sponsoring Client ID	1	read-only	Client Identifier	The current sponsoring client ID.
creatingClientId	Creating Client ID	0-1	read-only	Client Identifier	The client ID that created the object.
creationDate	Creation Date	0-1	read-only	Timestamp	Creation timestamp.
updatingClientId	Updating Client ID	0-1	read-only	Client Identifier	The client ID that last updated the object.
updateDate	Update Date	0-1	read-only	Timestamp	The timestamp of the last update.

Identifier	Name	Card.	Mutability	Data Type	Description
expiryDate	Expiry Date	0-1	read-only	Timestamp	Expiry timestamp.
transferDate	Transfer Date	0-1	read-only	Timestamp	The timestamp of the last successful transfer.
authInfo	Authorisation Info	0-1	read-write	authInfo	Authorisation information for the object.

Table 6

Operations

Operation: Create

Description: Provisions a new Domain Name resource.

Parameters

Identifier	Name	Card.	Data Type	Description
period	Registration Period	0-1	period	The initial registration period for the domain name.

Table 7

Operation: Read

Description: Retrieves the data elements of a Domain Name resource.

Parameters

Identifier	Name	Card.	Data Type	Description
hostsFilter	Hosts Filter	0-1	String	Controls which host information is returned.
queryAuthInfo	Query Authorisation Information	0-1	authInfo	Credentials to authorise access to full object data.

Table 8

Operation: Delete

Description: Removes an existing Domain Name resource.

Parameters: (None)

Operation: Renew

Description: Extends the validity period of a Domain Name resource.

Parameters

Identifier	Name	Card.	Data Type	Description
currentExpiryDate	Current Expiry Date	1	Timestamp	The expected current expiry date, for validation.
renewalPeriod	Renewal Period	0-1	period	The duration to add to the registration period.

Table 9

> TODO: IANA table: Contact Data Object TODO: IANA table: Host Data Object

10. Security Considerations

> TODO: write security considerations, if any

11. Changes History

draft-kowalik-rpp-data-objects -01 - -02

- ontology of allowed basic datatypes #4
- add examples of associations #31

12. Normative References

[I-D.kowalik-rpp-architecture] Kowalik, P. and M. Wullink, "RPP Architecture", Work in Progress, Internet-Draft, draft-kowalik-rpp-architecture-03, 10 October 2025, <https://datatracker.ietf.org/doc/html/draft-kowalik-rpp-architecture-03>.

[ISO3166-1] International Organization for Standardization, "Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes", ISO Standard 3166, November 2000.

[ITU.E164.2005] International Telecommunication Union, "The international public telecommunication numbering plan", ITU-T Recommendation E.164, February 2005.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC1738] Berners-Lee, T., Masinter, L., and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, DOI 10.17487/RFC1738, December 1994, <https://www.rfc-editor.org/info/rfc1738>.

[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>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[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>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

Authors' Addresses

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

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