RPP

Motivation and current work in the area

Pawel Kowalik - DENIC
IETF 121 - RPP BoF
Nov 6th, 2024 Dublin

Motivation

History – Before 2004

- Registries develop their own unique domain name provisioning interface
- No standard, registrar needs custom integration for each registry
  - RRP RFC 2832 published as Informational

History - 2004

Extensible **P**rovisioning **P**rotocol (EPP)

- Published as IETF standard in 2004
- XML-based
- TCP transport only
- Support for protocol extensions
- Deployed at many registries

Present Day

RFC5730: “The original motivation for this protocol was to provide a standard Internet domain name registration protocol ”

- Registries are again developing non-standardized provisioning APIs on top of HTTP
- Creating the risk of ending up in pre-EPP state
- EPP was developed when the use of REST and APIs was not yet common

RESTful Provisioning Protocol

A new proposed standard for domain names provisioning over a RESTful API based protocol

- Based on HTTP following the REST principles
- Data model (at least one way) compatible with existing EPP standards
- Mapping of objects and operations to REST endpoints
- JSON as native data format

Why a RESTful API?

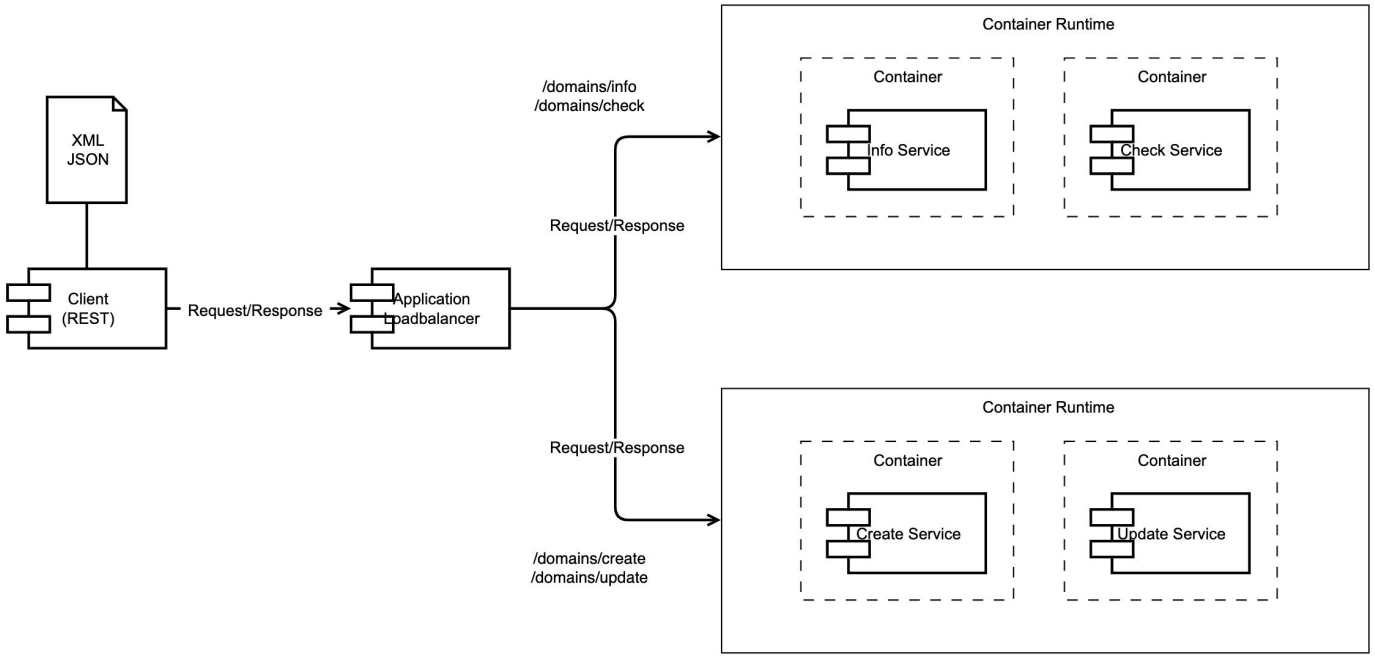
Deployment challenges in modern (cloud) environments

- Load balancing (L3/4 vs L7)
- Rate limiting (session vs request)
- Identity and Access Management (IAM)
- Leverage existing HTTP traffic management solution

Ease of use

- REST is the defacto standard for API development
- Simplifies development of registrar-registry integrations
- Simpler and cheaper for (smaller) registrar

Example: Resource allocation & request based routing



Reasons for HTTP-based protocol as per BCP 56

- familiarity by implementers, specifiers, administrators, developers, and users ✔
- availability of a variety of client, server, and proxy implementations ✔
- ease of use ✔
- availability of Web browsers ✔/✘
- reuse of existing mechanisms like authentication and encryption ✔
- presence of HTTP servers and clients in target deployments ✔
- its ability to traverse firewalls. ✔/✘

Fix some pain points in extensibility of EPP data model

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
 <command>
   <create>
     <contact:create
      xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
       <contact:id>sh8013</contact:id>
       ...
       <contact:email>jdoe@example.com</contact:email>
       ...
   </contact:create>
   </create>
   <extension>
    <addlEmail:addlEmail
     xmlns:addlEmail="urn:ietf:params:xml:ns:epp:addlEmail-1.0">
      <addlEmail:email>jdoe-alt@example.net</addlEmail:email>
    </addlEmail:addlEmail>
   </extension>
   ...
</command>
</epp>
```

Extension needed to change the cardinality of email address from 1 to many. Are these email addresses equally applicable?

https://datatracker.ietf.org/doc/html/draft-ietf-regext-epp-eai-22

More than just mapping XML to JSON

- OpenAPI - enable the ecosystem of tools to foster documentation, implementation and testing of clients and servers
- Provide same developer experience beyond just provisioning of registry objects
- Compact the data model - incorporate very frequently used extensions into the core data model
- Review operations available in STD 69, remove those not needed and used
- Define more flexibility in extending the data models and operations and loose coupling between clients and servers
- Take advantage from 20 years of EPP and several years of RDAP, but also from existing experiences of current work in RESTful provisioning of domains

Current work in the area

Afnic Phoenix API - Domain Create

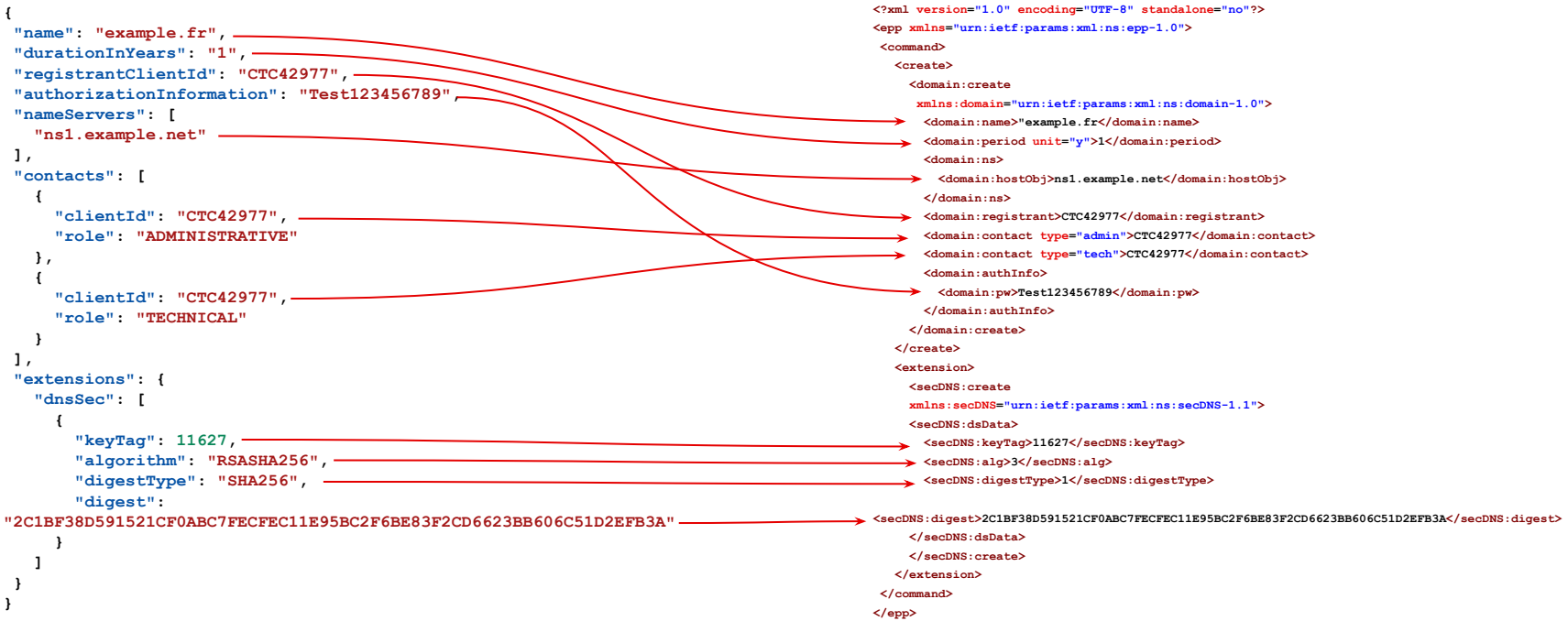
POST /v1/domains

```
{
 "name": "example.fr",
 "durationInYears": "1",
 "registrantClientId": "CTC42977",
 "authorizationInformation": "Test123456789",
 "nameServers": [
   "ns1.example.net"
 ],
 "contacts": [
   {
     "clientId": "CTC42977",
     "role": "ADMINISTRATIVE"
   },
   {
     "clientId": "CTC42977",
     "role": "TECHNICAL"
   }
 ],
 "extensions": {
   "dnsSec": [
     {
       "keyTag": 11627,
       "algorithm": "RSASHA256",
       "digestType": "SHA256",
       "digest":
"2C1BF38D591521CF0ABC7FECFEC11E95BC2F6BE83F2CD6623BB606C51D2EFB3A"
     }
   ]
 }
}
```

https://api-sandbox.nic.fr/api-docs/

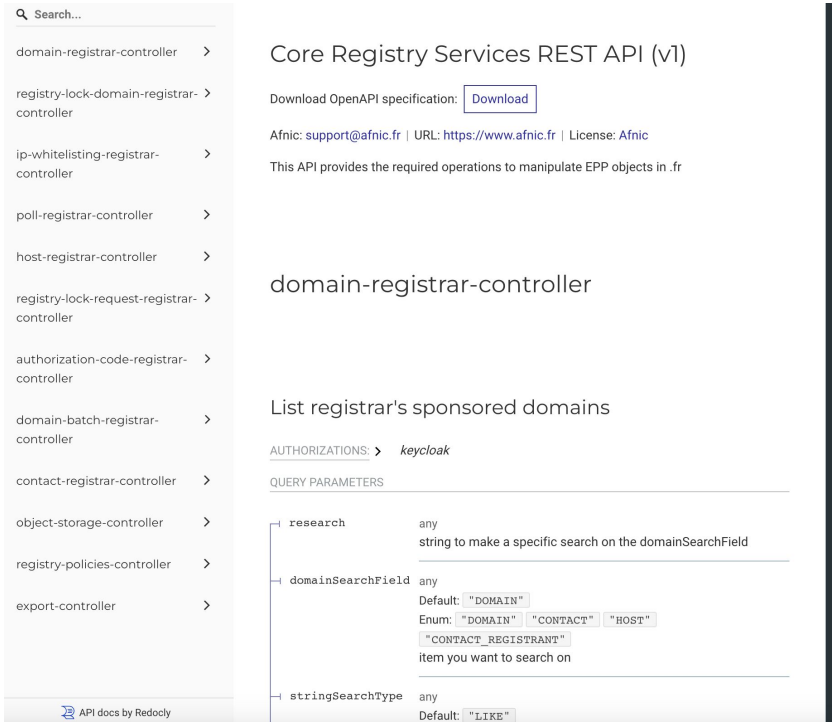
Afnic Phoenix API - Domain Create - transformation to EPP

POST /v1/domains



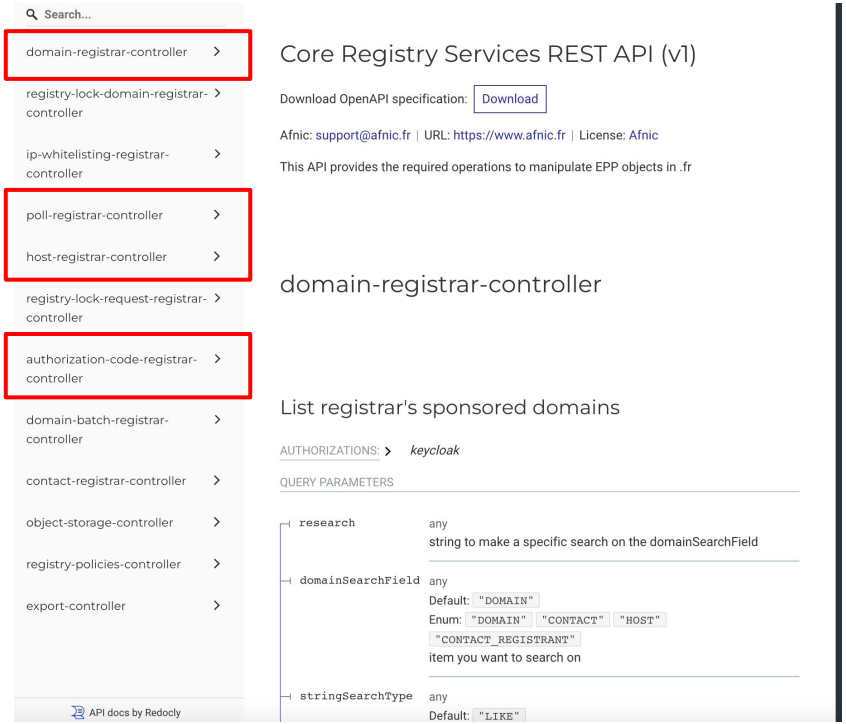
https://api-sandbox.nic.fr/api-docs/

Afnic Phoenix API - Scope



- Added use-cases and functionalities
- Common authentication and authorization
- Common developer experience

Afnic Phoenix API - Scope



- Added use-cases and functionalities
- Common authentication and authorization
- Common developer experience
- Only part of it covered in EPP

.ee registry system REPP API - Domain Create

POST POST /repp/v1/domains

```
{
   "domain": {
       "name": "kasskoer.ee",
       "registrant": "ATSAA:KARL",
       "period": 1,
       "period_unit": "y",
       "admin_contacts": [
           "ATSAA:KARL"
       ],
       "tech_contacts": [
           "ATSAA:KARL"
       ],
       "nameservers_attributes": [
           {
               "hostname": "ns1.kreative.ee"
           },
           {
               "hostname": "ns2.kreative.ee"
           }
       ]
   }
}
```

https://internetee.github.io/repp-apidoc/

GoDaddy API - Domain Create

POST /v1/domains/purchase

```
{
 "domain": "example.com",
 "period": 1,
 "nameServers": [
   "string"
 ],
 "contactAdmin": {
   [...]
 },
 "contactBilling": {
   [...]
 },
 "contactRegistrant": {
   [...]
 },
 "contactTech": {
   [...]
 },
 "privacy": false,
 "renewAuto": true,
 "consent": {
   "agreedAt": "XXX",
   "agreedBy": "YYY",
   "agreementKeys": [
     "string"
   ]
 },
}
```

Contact objects embedded in the domain representation

Additional attributes

https://developer.godaddy.com/doc/endpoint/domains#/v1/purchase

GoDaddy - API landscape

GoDaddy has a number of APIs that you can explore below.

Click on a resource below to view and explore the available endpoints and operations. For any API that needs a `customerId`, you can get it here.

Concept nic.br

```
Domain object
{
        "fqdn": "example.example"
        "Entity" array : ["Fulano de Tal", entity ID]
        "regPeriod": 1
        "dsdata": [ [
                "keyTag": "string"
                "alg": integer
                "digestType": integer
                "digest": "string" ] , (...) ]

        "status": "string" (domain GET/domain PUT response only)
        "contacts": [ "registrant": [
                        "fullname": "string"
                        "email": "string"
                        "streetAddress": "string"
                        "addressComplement": "string"
                        "city": "string"
                        "stateProvince": "string"
                        "country": "string"
                        "fullPhoneNumber": "string" (E.164 format) ]
        "Nameservers": [        { "dnsservername": "string" } or
                                { "dnsservername": "string/IPv4" } or
                                { "dnsservername": "string//IPv6" } or
                                { "dnsservername": "string/IPv4/IPv6" , {...} ]
}
```

DNSSEC embedded in the main model

Optional attributes for GET

https://docs.google.com/document/d/1eT7BqVJUwsXXsfgPuvGMNde-u5V_AaqQ-cmZG3NWBI0

And many more…

- https://docs.openprovider.com/doc/all
- https://developer.hosting.ionos.com/docs/domains
- https://help.internetx.com/pages/viewpage.action?pageId=14878531
- https://github.com/james-stevens/epp-restapi
- …

And even more considering…

- SIDN/CIRA - new registration system, also BERO for other TLDs
- .si
- DENIC
- cz.nic - FRED also used by several other ccTLDs
- Punktum dk

… this is likely by far not an exhaustive list

Some   in the room

Why not just use EPP over HTTPS (EoH)?

- Not RESTful, just pushing XML payload over one HTTP endpoint
  - URL based routing is not possible
  - Limited use of existing HTTP tooling and frameworks
- Stateful sessions
- Lack of JSON support
- Extending EPP beyond what it does well today is not a first choice for operators - they do RESTful instead
- Not what people (developers, operators) really want

Why spend time on RPP if EPP is fit for the job?

- Stateless service architecture
- Possible transaction (request) based authorisation with IETF standard solutions for HTTP
- URL based routing and load balancing
- RPP can be combined with other HTTP APIs to offer a consistent API plane to the clients
- HTTP API support in the standard infrastructure elements, server and client implementation frameworks, development, testing and monitoring tools
- RPP is technology-wise closely aligned to RDAP
- RPPs-like APIs are being created and used - so this is what people seem to want more. Starting the work on RPP now will limit future fragmentation.

Thank you