IETF 123 hackathon: experimenting RPP

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

First publication of this article on 25 August 2025

https://www.bortzmeyer.org/hackathon-ietf-123.html

July 2025 ? Among many other events in the world, this was the IETF meeting, the 123rd, in Madrid. Before the meeting, there was the now traditional hackathon. I worked on the future protocol RPP (RESTful Provisioning Protocol), which may be the successor of EPP.

EPP (RFC 5730 [1]) is the current standard for provisioning, specially domain names. When you register a domain name <https://www.afnic.fr/en/observatory-and-resources/expert-papers/what-happens-when-you-register-a-domain-name/>, the registrar you use probably talks to the registry with EPP. EPP has some limits and the IETF is working (in the rpp working group <https://datatracker.ietf.org/wg/rpp/>) on another protocol, RPP (for RESTful Provisioning Protocol) <https://www.afnic.fr/en/observatory-and-resources/expert-papers/rpp-a-future-protocol>. Such a REST API is already done, for instance at Afnic <https://www.afnic.fr/en/observatory-and-resource expert-papers/the-fr-api-for-registrars-1-4/> (the domain name registry of .fr).

At the time of the IETF hackathon, RPP was far from done, there was not even a consensus on some basic aspects. So, unlike many hackathon projects, there was no attempt to test interoperability of different implementations of a protocol. The idea was instead to explore various things <https://wiki.ietf.org/en/meeting/123/hackathon#restful-provisioning-protocol-rpp> related to a REST provisioning protocol.

OK, so my specific work (the code is on GitHub <https://github.com/bortzmeyer/RPP-Afnic>) :
— Develop a (very small, and very minimum don't be afraid) domain name registry, with a pseudo-RPP interface,
— Develop a client test suite.

Let's do a demo first :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5730.txt

```
% createdb registry
% psql -f ./create.sql registry
% ./test-server.py
```

Then let's get information about a domain with curl (one of the goals of a RESTful protocol like RPP is to be able to use regular HTTP clients) :

```
% curl --header @headers.txt http://localhost:8080/domains/nic.example
{"result": "Domain nic.example exists", "holder": 1, "tech_contact": 1, "admin_contact": 1, "registrar": 1,
```

And create a domain (note we have to authentify this time) :

```
% curl --header @headers.txt --request PUT --user 2:qwerty --data '{"holder": 2, "tech": 2, "admin": 2}'  ht
{"result": "durand.example created", "status_code": 201, "status_message": "Created"}
```

OK, you get the idea, the `README.md` file will give you some examples.

The registry is a PostgreSQL database. The server is written in Python, uses the WSGI framework, depends on several modules from the Python standard library and on psycopg2 <https://pypi.org/project/psycopg2/> and jsonschema <https://pypi.org/project/jsonschema/>. We use JSONschema <https://json-schema.org/> for the validation of the incoming JSON (remember that RPP is far from being standardized and no schema language have been chosen yet). The test client, as you saw, is regular curl.

I added a (very small) test suite written in Zig :

```
% cd tests
% zig build test
```

(I took the opportunity of the hackathon to post a sample HTTP client written in Zig to the Ziggit forum <https://ziggit.dev/t/yet-another-example-of-a-http-client-but-with-more-features->.)

The work done at the RPP table of the hackathon is presented here <https://datatracker.ietf.org/doc/slides-123-hackathon-sessd-rpp/>. All the live presentations are on YouTube <https://www.youtube.com/watch?v=ShjlUAN6FjI> and ours is at 1 :19 :08. Now, the rpp working group <https://datatracker.ietf.org/wg/rpp/> continues its work...(See the development on Github <https://github.com/ietf-wg-rpp>.)