RDAP Implementation Experience

Gavin Brown, Chief Innovation Officer
<gavin.brown@centralnic.com>

ROW #8, Bangkok, May 2019
CentralNic’s RDAP Server Implementation

- We use mod_whois for port 43, runs inside Apache

- It was trivial to add an extra `<VirtualHost>` for RDAP, deployed on the same infrastructure. TLS certificate via Let’s Encrypt

- We use PHP and an object-oriented foundation library based on an ORM

- RDAP implementation = 1,285 lines of PHP

- Deployed on all TLDs by early April
CentralNic’s RDAP Server Implementation

CentralNic is part of CentralNic Group plc, which also includes two other PHP-based ICANN-accredited registrars (Instra Corporation and Internet Domain Services BS).

We built a generic RDAP server framework in PHP to be used across the group.

RDAP server framework = 187 lines of code
Client Implementation
RDAP Client Implementation

I like Perl 🐫

Created “rdapper” back in 2012

Then... nothing

In 2018: Net::RDAP!
Net::RDAP

Net::RDAP aims to be the reference implementation for RDAP clients

"a single unified interface to information about all unique Internet identifiers"

Aims to implement 100% of the specification:
- Basic query/response
- Search
- All object types
- All object properties
- Bootstrap
- Object tagging
- JSON values
- EPP status mapping
- ...and eventually RDAP extensions too!

Inspired by Net::DNS and LWP

Net::RDAP
- Net::RDAP::Base
  - Net::RDAP::Event
  - Net::RDAP::ID
  - Net::RDAP::Object
    - Net::RDAP::Error
    - Net::RDAP::Help
    - Net::RDAP::Object::Autnum
    - Net::RDAP::Object::Domain
    - Net::RDAP::Object::Entity
    - Net::RDAP::Object::IPNetwork
    - Net::RDAP::Object::Nameserver
    - Net::RDAP::Object::SearchResult
  - Net::RDAP::Remark
  - Net::RDAP::Notice
- Net::RDAP::EPPStatusMap
- Net::RDAP::Registry
- Net::RDAP::Registry::IANARegistry
- Net::RDAP::Registry::IANARegistry::Service
- Net::RDAP::Service
- Net::RDAP::Link
- Net::RDAP::UA
- Net::RDAP::Values
Net::RDAP

Install using cpan –i Net::RDAP

Easy to use:

```
use Net::RDAP;

my $rdap = Net::RDAP->new;
#
# traditional lookup:
#
# get domain info:
$object = $rdap->domain(Net::DNS::Domain->new('example.com'));

# get info about IP addresses/ranges:
$object = $rdap->ip(Net::IP->new('192.168.0.1'));
$object = $rdap->ip(Net::IP->new('2001:DB8::/32'));

# get info about AS numbers:
$object = $rdap->autnum(Net::ASN->new(65536));
#
# search functions:
#
my $server = Net::RDAP::Service->new("https://www.example.com/rdap");

# search for domains by name:
my $result = $server->domains('name' => 'example.com');

# search for entities by name:
my $result = $server->entities('fn' => 'Jane Doe');

# search for nameservers by IP address:
my $result = $server->nameservers('ip' => '192.168.56.101');
```
rdapper

Now just a wrapper around Net::RDAP

Install using cpan –i rdapper
RDAP Web Client

Try it out at https://client.rdap.org

Uses RDAP.org bootstrap service (for now)
Conclusions

RDAP is easy...

...but jCard is hard

Boo jCard!

RDAP 1.1 needs to sprinkle some MUSTs and MUST NOTs across the specification