LACNIC’s RDAP implementation

IETF 95 - Buenos Aires
April 2016
Agustín Formoso

@aguiformoso | agustin@lacnic.net
The implementation

- In Java
- JSON Response based on RIPE NCC RDAP Objects
- Underlying Queries based on LACNIC Database Objects
- LACNIC → RIPE NCC Object mapper
- Made following (and complying with) RDAP drafts a lot easier
- Out of beta on December 2014
The implementation (1)

- Simple Java Application
The mapper

- Easy way of implementing the Java mapper
- Based in Java Generics
So far...

- ~177 K monthly
- ~6 K daily (excluding Feb 2016)
- Daily minimums on the rise
So far...

- ~177 K monthly
- ~6 K daily
  (excluding Feb 2016)
Test suite

- APNIC’s test suite (GitHub) as a WEIRDS compliance check
- Jenkins Continuous Integration tests

https://github.com/APNIC-net/rdap-conformance
API Keys

- Additional query parameter
- Uses LACNIC API Key services
- API Key tokens used for rate limiting only
- Defaults to 2 rate limits: 10 per minute and 1000 per hour, whichever is violated first
- IP-based rate limits
- **Formal procedure** for requesting a new API Key (similar to bulk whois)
- Proxy flag

https://apikey.lacnic.net
Temporary Cache

- Additional query parameter
- URL-based
- 24-hour object cache
- `cache=false` bypasses cache, hits the database, and refreshes the cache
- Cache notification is added to the `notices` attribute
The implementation (2)
RDAP Bootstrap

• ARIN’s bootstrap ([GitHub](https://github.com/arineng/rdap_bootstrap_server))
• Out of the box Java application
• HTTP redirects are based on IANA and LACNIC json files
  – Regional NICs (NIC.br)
  – Other RIRs’ IP space

https://github.com/arineng/rdap_bootstrap_server
TODOs

• HEAD!!
• Better bootstrapping
THANKS!

RDAP  https://rdap.lacnic.net/rdap
RDAP Bootstrap https://rdap.lacnic.net/bootstrap