The fractal horror of ancient crufty DNS APIs

Tony Finch
<fanf2@cam.ac.uk>
<hostmaster@cam.ac.uk>

Tuesday 26th July 2016

Abstract

A group therapy session in which I share my traumatic experience last year writing some software to help manage our DNS registrations, integrating with four external suppliers.

- What is EPP and why does it matter?
- XML is an awesome basis for a protocol, honest!
- Aren't open standards great?
- When an API can't quite paper over the cracks...
- How to make JSON as bad as XML
- What to do when they don't give you an API

1 Introduction

- I work in the Network Systems team in room 1N25
- I am the University's Hostmaster, responsible for our DNS services
- Title refers to Dave Langford's story "What Happened at Cambridge IV" and the Langford Basilisk
- Slight misnomer because this is actually about domain registration APIs rather than DNS APIs

Tractatus Logico-Philosophicus

Proposition 7

> Wovon man nicht sprechen kann, darüber muß man schweigen.

— Ludwig Wittgenstein

Translated into English, this roughly means,

> When you can't speak to the DNS servers, the whole network falls silent

2 DNS in Cambridge

IP Register

`https://jackdaw.cam.ac.uk/ipreg/`

- `cam.ac.uk`
- `111.131.in-addr.arpa`
- `195.18.192.in-addr.arpa`
- `5.84.192.in-addr.arpa`
- `213.153.192.in-addr.arpa`
- `252.63.193.in-addr.arpa`
- `253.63.193.in-addr.arpa`
- $80 \cdots 95$ `.60.193.in-addr.arpa`
- `1.2.0.0.3.6.0.1.0.0.2.ip6.arpa`
- `232.128.in-addr.arpa` *(shared with CL)*
- Contents of the DNS in Cambridge come from two databases
- Main one is the IP Register database, hosted on Jackdaw
- Forward DNS for cam.ac.uk
- IP address space and corresponding reverse DNS
- IP address allocation mainly managed by others on Network Systems

Managed Zone Service

`https://www.mzs.csx.cam.ac.uk/`

cambridge-university.ac.uk cambridge-university.biz cambridge-university.net.uk cambridge.ac.uk cambridge.net.uk cambridgebioresource.org.uk cambridgeconservation.org cambridgeinterfaithprogramme.com cambridgeinterfaithprogramme.net cambridgeinterfaithprogramme.org cambridgeinterfaithprogramme.org.uk cambridgephysics.org cambridgeuniversity.ac.uk cambridgeuniversity.biz cambridgeuniversity.net.uk cantab.ac.uk carriagestudy.org.uk cchsr.ac.uk cfas.ac.uk churchill.ac.uk cmpcp.ac.uk conflictincities.org cudos.ac.uk culturefinder.co.uk culturefinder.org.uk darwinendlessforms.org doitpoms.ac.uk eclipsestudy.eu esas.org eu-eppidem.eu exoplanets.uk extend-nmr.eu gaia-eso.eu genet.ac.uk idpatcued.org.uk intervalstudy.org.uk melatools.org memoryatwar.org miquit.co.uk multikultura.org.uk newton.ac.uk niees.ac.uk nihrbioresource.org.uk nri.org.uk pads.ac.uk parisiansoundscapes.org pembroke1347.net phagedisplay.org qumeshs.org regionalvisions.ac.uk researchers14.ac.uk rmdq.org romeyka.org scopic.ac.uk shaare.org.uk silentpartners.org.uk ska-sdp.org sots.ac.uk suburbansolutions.ac.uk superspintronics.org terrahunting.org thedecoratedschool.org.uk thresholds.org.uk tombtreasuresofhanchina.org tombtreasuresofhanchina.org.uk ucam.ac.uk ucam.biz ukca.ac.uk ukindemand.ac.uk university-of-cambridge.ac.uk university-of-cambridge.biz university-of-cambridge.net university-of-cambridge.net.uk university-of-cambridge.org university-of-cambridge.org.uk universityofcambridge.ac.uk universityofcambridge.biz universityofcambridge.net.uk uxsup.org.uk vist.org.uk woodssupermarket.co.uk

- Other database is the Managed Zone Service
- Hosted on the Managed Web Service
- Multi-instituion domains
- e.g. `researchers14.ac.uk`
- "vanity domains"

3 Subject of this talk

Last year ...

- Talk is based on work I did last year
- Wrote some software called "superglue"
- Automate management of DNS delegations in our parent domains
- Two reasons –
  - Short term: add an off-site secondary
  - `sns.isc.org`
  - Long term: automatic DNSSEC key rollovers
- Updating >100 delegations tedious and error-prone

Delegation – old

```
cam.ac.uk.  NS  authdns0.csx.cam.ac.uk.
cam.ac.uk.  NS  authdns1.csx.cam.ac.uk.
cam.ac.uk.  NS  dns0.cl.cam.ac.uk.
cam.ac.uk.  NS  dns1.cl.cam.ac.uk.
cam.ac.uk.  NS  dns0.eng.cam.ac.uk.
cam.ac.uk.  NS  ns2.ic.ac.uk.
```

- What does a delegation look like?
- Inside the parent zone, `.ac.uk`
- One off-site secondary at Imperial College
- (We provide secondary DNS to Imperial, Salford, the Sanger, and York)

Delegation – `sns.isc.org`

```
cam.ac.uk.  NS  authdns0.csx.cam.ac.uk.
cam.ac.uk.  NS  authdns1.csx.cam.ac.uk.
cam.ac.uk.  NS  dns0.cl.cam.ac.uk.
cam.ac.uk.  NS  dns1.cl.cam.ac.uk.
cam.ac.uk.  NS  dns0.eng.cam.ac.uk.
cam.ac.uk.  NS  ns2.ic.ac.uk.
cam.ac.uk.  NS  sns-pb.isc.org.
```

- Add our off-site secondary
- Belated replacement for the secondary at MIT that we lost years ago
- `isc.org` produce the DNS software BIND
- and run the F root DNS server
- `sns.isc.org` is used by nearly 50 top-level domains

Delegation – pruned

```
cam.ac.uk.  NS  authdns0.csx.cam.ac.uk.

cam.ac.uk.  NS  dns0.cl.cam.ac.uk.

cam.ac.uk.  NS  dns0.eng.cam.ac.uk.
cam.ac.uk.  NS  ns2.ic.ac.uk.
cam.ac.uk.  NS  sns-pb.isc.org.
```

- No need for so much redundancy on-site
- Increase probability of off-site clients choosing an off-site server

Delegation – glue

```
cam.ac.uk.                NS    authdns0.csx.cam.ac.uk.
cam.ac.uk.                NS    dns0.cl.cam.ac.uk.
cam.ac.uk.                NS    dns0.eng.cam.ac.uk.
cam.ac.uk.                NS    ns2.ic.ac.uk.
cam.ac.uk.                NS    sns-pb.isc.org.

authdns0.csx.cam.ac.uk.   A     131.111.8.37
authdns0.csx.cam.ac.uk.   AAAA  2001:630:212:8::d:a0
dns0.cl.cam.ac.uk.        A     128.232.0.19
dns0.cl.cam.ac.uk.        AAAA  2001:630:212:200::d:a0
dns0.eng.cam.ac.uk.       A     129.169.8.8
```

- In order to find the address of a name in `cam.ac.uk`, talk to `authdns0.csx.cam.ac.uk`
- But you need to find the address of `authdns0.csx.cam.ac.uk` which is a name in `cam.ac.uk`
- So the parent domain also has copies of address records of nameservers in child domains.
- These address records are called "glue"

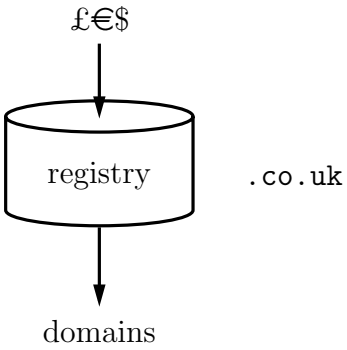
Delegation – DNSSEC

```
cam.ac.uk.                NS    authdns0.csx.cam.ac.uk.
cam.ac.uk.                NS    dns0.cl.cam.ac.uk.
cam.ac.uk.                NS    dns0.eng.cam.ac.uk.
cam.ac.uk.                NS    ns2.ic.ac.uk.
cam.ac.uk.                NS    sns-pb.isc.org.

authdns0.csx.cam.ac.uk.   A     131.111.8.37
authdns0.csx.cam.ac.uk.   AAAA  2001:630:212:8::d:a0
dns0.cl.cam.ac.uk.        A     128.232.0.19
dns0.cl.cam.ac.uk.        AAAA  2001:630:212:200::d:a0
dns0.eng.cam.ac.uk.       A     129.169.8.8

cam.ac.uk.                DS    52543 5 2 F3D339A8602D464
                                63CBDCDC33769165B58489FAE
                                38FC5DB84AE0136A8874FCB2
```

- DNSSEC adds another part to a delegation
- A DS record is a cryptographic digest of a child zone's key
- Creates a link in the chain of trust from the DNS root to the child zone
- Ultimate goal is to make DNSSEC keys relatively short-lived
- e.g. change them every month or quarter
- Cannot be done without automation

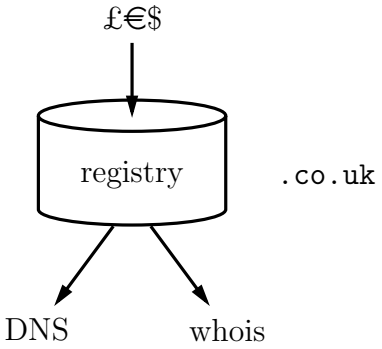
4 Domain registries

A domain registry



- In order to update our DNS delegations we need to make a change in the relevant domain registry
- A registry is a system that consumes money and produces domain names
- It consists of a database, some operational and support staff, and lots of lawyers

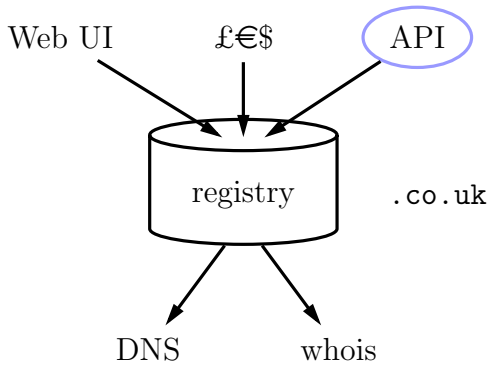
A domain registry



- Actually a registry database is exported via two protocols

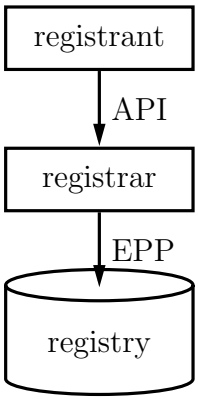
- The DNS is the useful part that we care about
- But `whois` has much more complete information
- Who is the domain owner, their contact details, when it was registered and when it will expire

A domain registry



- In the 1990s the typical user interface for making changes to a registry was to email `whois` records to the registry administrators
- Nowadays they pretty much all provide a web user interface
- And for automated updates there is usually an application programming interface of some kind
- I am going to talk about some interesting API features I discovered last year
- Tiny subset of each API
- Actually I am mostly going to moan about horrible things that you should avoid inflicting on your users if you ever create an API

The RRR model



- In many cases we can't actually talk directly to the registry
- We have to go via a reseller
- Known as a "registrar"
- There is an open standard protocol between registries and registrars called "EPP"
- The "Extensible Provisioning Protocol"
- The interface between registrars and their customers is usually proprietary

5 Digressions

5.1 Digression – origin of EPP

It's worth going on a digression for a moment to explain the origin of EPP, because some of its ugliness makes more sense if you know a bit about the circumstances of its birth.

The Network Solutions saga

- 1990 – ARPANET decommissioned
- NSFNET becomes the Internet backbone
- rapid commercialization and privatization
- decentralization too – CIDR and BGP

- 1972–1990 NIC at SRI
- 1990 DCA contracts to GSI subcontracted to NSI
- 1993 NSF contracts non-military InterNIC to NSI
- 1995 NSF allows NSI to charge fees
- 1997 anti-trust lawsuit
- 1998 `shitakemushrooms.com` censored

- monopoly over `.com`, `.net`, `.org`
- Fees initially $100 for two years
- Lawsuit reduced fee to $70
- Inconsistent censorship – `shit.com` had existed since 1996

- 1996–1997 International Ad-Hoc Committee
- 1998 Internet Corporation for Assigned Names and Numbers
- more gTLDs
- break up monopoly over existing gTLDs
- many gTLDs for competition between registries
- competition between registrars within TLDs

RFC 1591 – March 1994

> There are a set of what are called "top-level domain names" (TLDs).
>
> These are the generic TLDs (EDU, COM, NET, ORG, GOV, MIL, and INT), and the two letter country codes from ISO-3166.
>
> **It is extremely unlikely that any other TLDs will be created.**

— Jon Postel

5.2 Digression – Digression – protocol architecture and economics

Tussle in Cyberspace

I'm going to go on a digression within a digression now, because there's a really cool concept that I want to share.

I think I got the idea from David D. Clark's paper called "tussle in cyberspace", which talks about the interaction between protocol design and conflicting priorities.

`http://conferences.sigcomm.org/sigcomm/2002/papers/tussle.pdf`

5.2.1 Public APIs

The basic idea is that if you create a public API, you are implicitly creating a market.

For example, when Twitter was young they created a public API because that was the cool web-2.0 thing to do.

They can control the market of businesses that build on top of Twitter by changing the API.

For example, they adjusted the rate limits to discourage alternative user interfaces.

5.2.2 Key observation

As well as being a programming interface, the API is also an interface between different businesses.

And if this interface doesn't match the economic incentives then your API will not be successful.

5.2.3 Open standard protocols

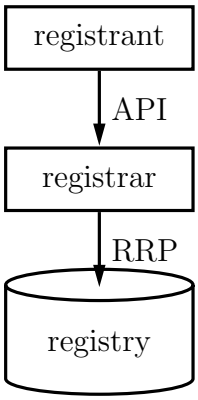
It is much more difficult to get this right in an open standard protocol, rather than a proprietary API.

For instance, if you design a client-server protocol, you are implicitly saying that:

- clients should have free choice of servers
- servers have to compete for clients without being able to control the user interface

5.3 Digression – RRR and EPP

The RRR model



So, going back to the first digression,

To break up the Network Solutions monopoly over `.com`, the United States Department of Commerce changed their contract to force a split between the back-end registry database business, and the customer-facing registrar business.

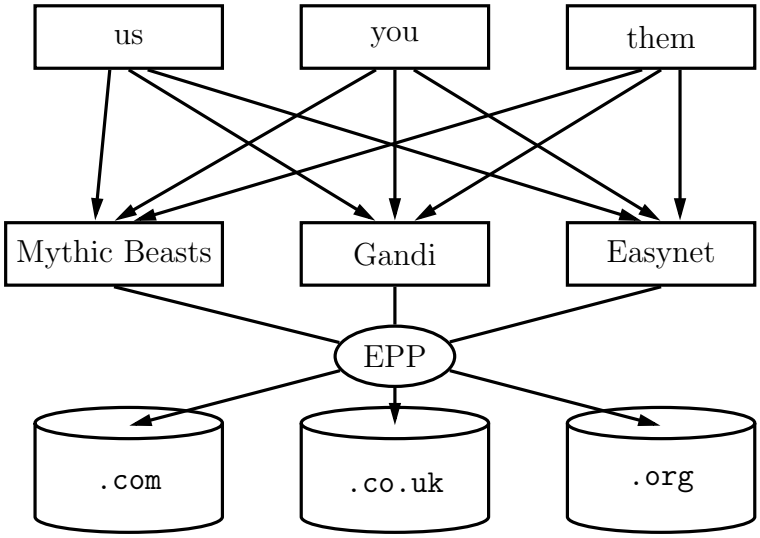
There is still only one `.com` registry, but it is not allowed to deal directly with registrants.

Registrants have to deal with registrars, who can compete with each other.

To support this contractual structure, the "Registry-Registrar Protocol" was created to be the technical interface that corresponds to the business interface between registrars and registries.

EPP replaced RRP about 10 years ago.

The RRR model



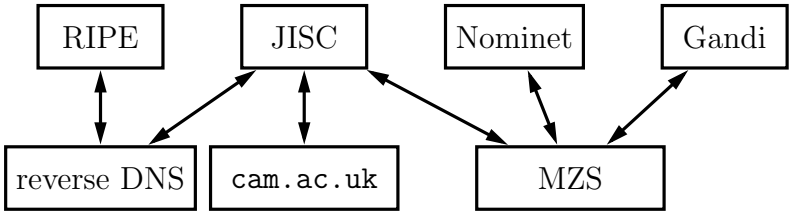
When you expand the diagrem to include multiple registrants, registrars, and registries, the RRR structure looks like this.

EPP allows a registrar to be a reseller for many registries with minimal technical effort.

There is not an equivalent standard for the registrant-registrar interface, so it is not so easy for registrants to switch registrars.

- This nice three-tier diagram is a big simplification
- It does not apply to the reverse DNS
- It only vaguely applies to ccTLDs like `.uk`

Registries, registrars, and our registrations



- Our suppliers at the time I was working on "superglue" last year

  - (I am moving all Nominet and Gandi domains to Mythic Beasts)

- Nominet is a registry
  - ccTLD not bound by ICANN rules
  - We have a registrar account for managing our own domains
- Gandi is a registrar
  - we use them for non-`.uk` domains
- JISC is our ISP
  - domain registry for `.ac.uk`
  - LIR – "local internet registry"
  - used for PA address space
- RIPE is an RIR
  - "regional internet registry"
  - used for PI address space

6 Nominet

Nominet

This is the blurb on Nominet's web site.

I like how it doesn't explicitly mention the DNS or domain names, and it obliquely talks about their Internet of Things research as if it is equally important.

Actually, I hate Nominet's web site more and more.

Registrar user interface: `https://secure.nominet.org.uk/`

Registrant user interface: `https://registrants.nominet.org.uk/app/`

Especially designed for primary school children. Ugh.

Nominet Automaton – RIP

The old Nominet API was based sending commands and whois objects by PGP signed email

20 years ago it was a straightforward automation of their old manual processes

It was finally retired in February this year.

Nominet Automaton – example

```
To: auto-co@nic.uk
Subject: Modify

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

key: uxsup.org.uk
nserver: authdns0.csx.cam.ac.uk 131.111.8.37
nserver: authdns1.csx.cam.ac.uk 131.111.12.37
nserver: sns-pb.isc.org
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1

iD4DBQFXlzOT/8DxTITHG24RAt5kAJ9nIT+P7DGTF+03xAVpvv/UJq08bQCVHNfy
uHUvLyilGIiANha8LybVeg==
=UB0O
-----END PGP SIGNATURE-----
```

Nominet's supported API is now just EPP

7 EPP

Registry-Registrar Protocol – RFC 2382

```
C: add
C: EntityName:Domain
C: DomainName:example2.com
C: -Period:10
C: NameServer:ns1.example.com
C: NameServer:ns2.example.com
C: .
S: 200 Command completed successfully
S: registration expiration date:2000-09-22 10:27:00.0
S: status:ACTIVE
S: .
```

- RRP was a very traditional Internet protocol
- 3-digit response codes, like SMTP and HTTP
- dot-delimited data, like SMTP
- transfers plaintext whois objects

Extensible Provisioning Protocol – RFC 5731 – request

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:period unit="y">2</domain:period>
C:        <domain:ns>
C:          <domain:hostObj>ns1.example.net</domain:hostObj>
C:          <domain:hostObj>ns2.example.net</domain:hostObj>
C:        </domain:ns>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

- EPP dates from a time when XML was taking over the world
- Heavily uses XML namespaces for extensibility

Extensible Provisioning Protocol – RFC 5731 – response

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:creData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:exDate>2001-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:creData>
```

```
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

- Absolutely revolutionary improvement to response codes
- Replace 3 digit codes with 4 digit codes!
- (Why not URNs?)

Extensible Provisioning Protocol

- `<login>`
- `<logout>`
- `<check>`
- `<info>`
- `<poll>`
- `<create>`
- `<delete>`
- `<renew>`
- `<transfer>`
- `<update>`
- domain
- host
- contact
- Basic CRUD protocol
- Unusual `<renew>` and `<transfer>` actions
- For billing and changes of ownership or registrar

Extensible Provisioning Protocol Problems

- Abstraction layer over the underlying transport
- Would be nice to use HTTP instead of TCP
- But! stateful authentication
- (BEEP might have been an alternative transport)

- Thick registry vs thin registry
  - contact details held by registrar or registry?
  - most registries are thick, except `.com` and `.net`
- DNSKEY vs DS records for DNSSEC
  - if you don't know which the registry supports, you have to spot the error code and retry with the alternative
- separate host objects, or glue in domain objects
  - in this case you can tell what kind of registry you are talking to from the server capability list
  - (why not the same as DNSSEC?)

Cancelling a domain with dependencies

- At registrar A –

```
deadbeat.com.  NS  ns0.deadbeat.com.
deadbeat.com.  NS  ns1.deadbeat.com.
```

- At registrar B –

```
bystander.com.  NS  ns0.deadbeat.com.
bystander.com.  NS  ns1.deadbeat.com.
```

- the owner of `deadbeat.com` isn't paying the bills
- Registrar A wants to cancel their domain
- Can't do it becaus there are subordinate host objects used by another domain!
- Must rename the hosts to a different domain first

8 Gandi

Recall, we use Gandi for non-`.uk` domain registrations.

Gandi

Gandi are a French registrar and web hosting provider.

The Gandi API is amazingly comprehensive.

It covers things like virtual servers, web sites, X.509 certificates, as well as domain registration. It is based on XML-RPC.

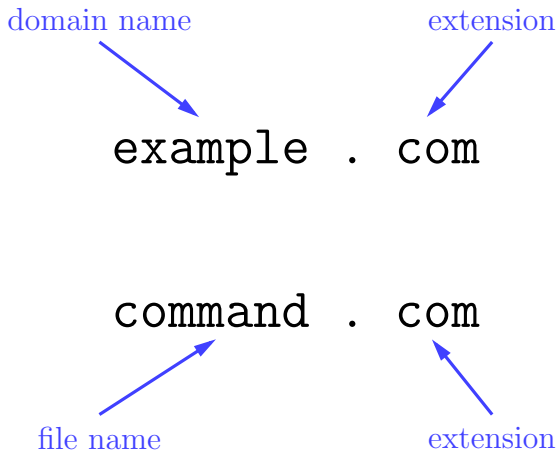
It looks like their web user interface is layered on their public API.

In the documentation there are parallel examples in Python, PHP, node.js, Perl, Ruby, and even C.

It has flowcharts for the more complicated processes.

Some oddities, such as, email-related methods bundled with the domain registration methods, and methods appear to be documented in order of creation rather than alphabetical order.

"extensions"



I am not a fan of applying the term "extension" to domain names because they are obviously completely different thins from file names.

Gandi – domain delegation methods

- `domain.info(apikey, domain)`
- `domain.nameservers.set(apikey, domain, nameservers[, options=nil])`
- `domain.host.list(apikey, domain[, opts=nil])`

- `domain.host.create(apikey, fqdn, ips)`
- `domain.host.info(apikey, fqdn)`
- `domain.host.update(apikey, fqdn, ips)`
- `domain.host.delete(apikey, fqdn)`

- `domain.dnssec.create(apikey, domain, params)`
- `domain.dnssec.list(apikey, domain)`
- `domain.dnssec.delete(apikey, keyid)`

Summary of the tiny part of the Gandi API used by Superglue

At first, it looks like a domain update involves creating any host objects (i.e. glue) that might be missing, then updating the domain's nameservers, then deleting any host objects that are superfluous.

But! There is a `nameservers_ips` option to `domain.nameservers.set` which allows you to specify glue and nameservers in one go!

However can use it to create glue, but you can't use it to update glue, and you can't use it to delete glue.

Also, the API appears to be based on an asynchronous EPP implementation:

An API call can return success before the underlying EPP transaction has completed, so a subsequent API call which depends on the first one can fail if you do it too soon.

There are even circumstances when an API call can return success, but later fail. In this case the error is sent to you via email!

The API doesn't seem to have a good way to find out when an operation completed other than polling the query methods.

The final process is —

- use `domain.nameservers.set` to change the NS records, and create any glue that is necessary
- then use `domain.host.update` to fix the rest of the glue
- then use `domain.host.delete` to clean up superfluous glue (which might fail because of the asynchronous API)

But there are still gotchas to do with a delegation that has glue pointing to a host under a different delegation in the same domain. Different registries have different rules about glue in this case.

9 RIPE

Recall, we use RIPE for our provider-independent IP address space and corresponding reverse DNS

RIPE

RIPE is a very different beast from the other providers in this survey.

Its database has a much greater variety of objects in it.

The RIPE database documentation table of contents gives you a good idea of how complicated it is.

- section 4 – object types
- section 6 – methods of updating the database
- section 10 – authorization

So what update method to choose?

- email updates
  - similar to Nominet Automaton
  - again, too asynchronous for a command-line tool
  - as well as PGP, allows S/MIME and even plaintext password authentication
- syncupdates
  - basically the same as email updates, except over `https`
  - nice and simple!
  - BUT no secure query interface!
- webupdates
  - Friendly user interface layered on top of REST API
- REST API
  - query and update interface over `https`

So the REST API it is.

RIPE object – text

```
organisation:   ORG-UCAM1-RIPE
org-name:       University of Cambridge
org-type:       OTHER
address:        Roger Needham Building
address:        7 JJ Thomson Avenue
address:        CAMBRIDGE
address:        CB3 0RB
address:        UK
admin-c:        UCSD1-RIPE
tech-c:         UCSN1-RIPE
abuse-c:        UISA1-RIPE
mnt-ref:        CAM-AC-UK-MNT
mnt-ref:        JANET-HOSTMASTER
mnt-by:         CAM-AC-UK-MNT
created:        2015-03-12T16:50:06Z
last-modified:  2015-06-15T08:37:35Z
```

- used by whois and syncupdates etc.
- basically a list of key–value pairs
- ordered
- multiple values for a key

RIPE REST API allows you to choose XML or JSON

Uses a common internal data representation serialized in a simplistic way to each format

```
<objects>
  <object type="organisation">
    <!-- ... -->
    <attributes>
      <attribute name="organisation" value="ORG-UCAM1-RIPE"/>
      <attribute name="org-name" value="University of Cambridge"/>
      <attribute name="org-type" value="OTHER"/>
      <attribute name="address" value="Roger Needham Building"/>
      <attribute name="address" value="7 JJ Thomson Avenue"/>
      <attribute name="address" value="CAMBRIDGE"/>
      <attribute name="address" value="CB3 0RB"/>
      <attribute name="address" value="UK"/>
      <attribute name="admin-c" value="UCSD1-RIPE" referenced-type="role">
        <link xlink:type="locator" xlink:href="http://rest.db.ripe.net/ripe/role/UCSD
      </attribute>
```

```
      <attribute name="tech-c" value="UCSN1-RIPE" referenced-type="role">
        <link xlink:type="locator" xlink:href="http://rest.db.ripe.net/ripe/role/UCSN
      </attribute>
```

```
  { "objects": {
    "object": [
      { // ...
        "attributes": {
          "attribute": [
            { "name":  "organisation",
              "value": "ORG-UCAM1-RIPE" },
            { "name":  "org-name",
              "value": "University of Cambridge" },
            { "name":  "org-type",
              "value": "OTHER" },
            { "name":  "address",
              "value": "Roger Needham Building" },
            { "name":  "address",
              "value": "7 JJ Thomson Avenue" },
            { "name":  "address",
              "value": "CAMBRIDGE" },
            { "name":  "address",
              "value": "CB3 0RB" },
```

```
  my $r = $ua->get($url, Accept => "application/json");
  # error checking ...
  my $j = decode_json $r->content;
  my @obj = @{ $j->{objects}->{object}->[0]
                 ->{attributes}->{attribute} };
```

FIVE levels of indirection just to get to the data we want!

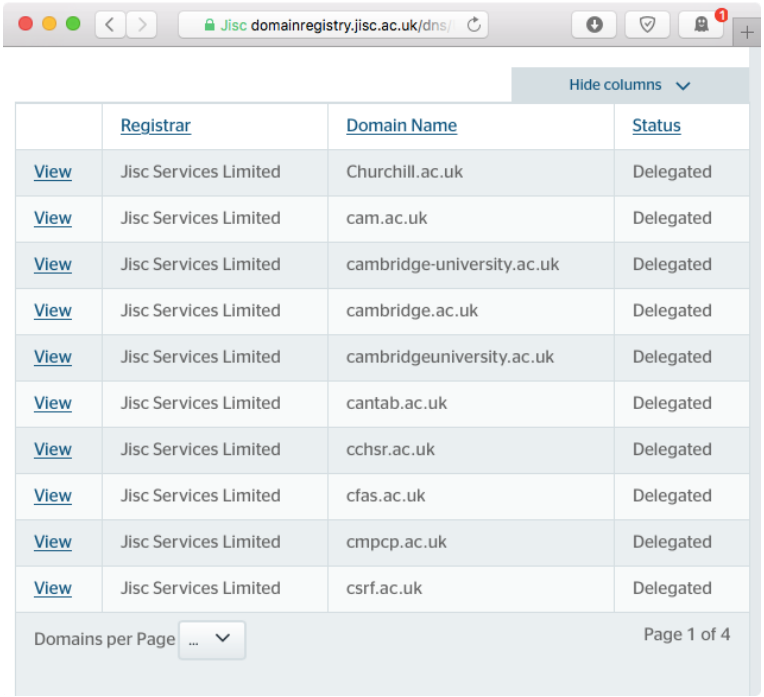
```
<plist version="1.0">
  <dict>
    <key>AlwaysShowFavoritesBarInFullScreen</key>
    <false/>
    <key>AlwaysShowTabBar</key>
    <false/>
    <key>AlwaysShowTabBarInFullScreen</key>
    <false/>
    <key>AutoFillCreditCardData</key>
    <false/>
    <key>AutoFillPasswords</key>
    <false/>
    <key>AutoShowToolbarInFullScreen</key>
```

```
    <true/>
    <key>BlockStoragePolicy</key>
    <integer>3</integer>

$ plutil -convert json -o - \
> Library/Preferences/com.apple.Safari.plist
Library/Preferences/com.apple.Safari.plist:
  invalid object in plist for destination format
$ _
```

10 JISC

Recall, we use JISC for `.ac.uk` domain registrations and for some of our IP adddress allocations.

JISC



The JISC domain registration documentation still talks about filling in whois templates and emailing them to the registry staff. And this method is still supported.

(There is no authentication. They mostly rely on knowing their customers and everyone behaving reasonably.)

But JISC also have a reasonably plausible domain registration web user interface

It has a few minor irritations

- Low information density disease

- Only two DNS updates per day
- Their IPv6 registry only allows /48 allocations
  - We have a /44 allocation, which is special
  - and so we have 16 permanently open tickets, one for each /48

The major irritation is that they do not have an API.

So I thought I would automate their web site.

Three options —

- Reverse engineer the web pages and talk to the server with an HTTP library
- Use a web scraping library such as "Beautiful Soup"
- Drive the web site with browser automation software

Unfortunately the JISC domain registry web site uses a very strange framework which wraps the whole of every page in an HTML `<form>` which is dynamically adjusted with Javascript.

So the first two options were ruled out.

PhantomJS

PhantomJS is a web browser without a user interface.

You can script it with Javascript.

Its web page includes the blurb —

> PhantomJS is an optimal solution for page automation.
> Access and manipulate webpages with the standard DOM API, or with usual libraries like jQuery.

CasperJS

CasperJS is a JavaScript library that makes scripting PhantomJS really quick and easy.

This combination turned out to work well.

I was lucky that when JANET rebranded to JISC and revamped the website towards the end of last year, I just had to change the URL and the code continued to work!

11 IP Register

The punchline

Ἰατρέ, θεράπευσον σεαυτόν

- Biblical proverb “Physician, heal thyself”
- Attend to your own defects rather than criticizing others

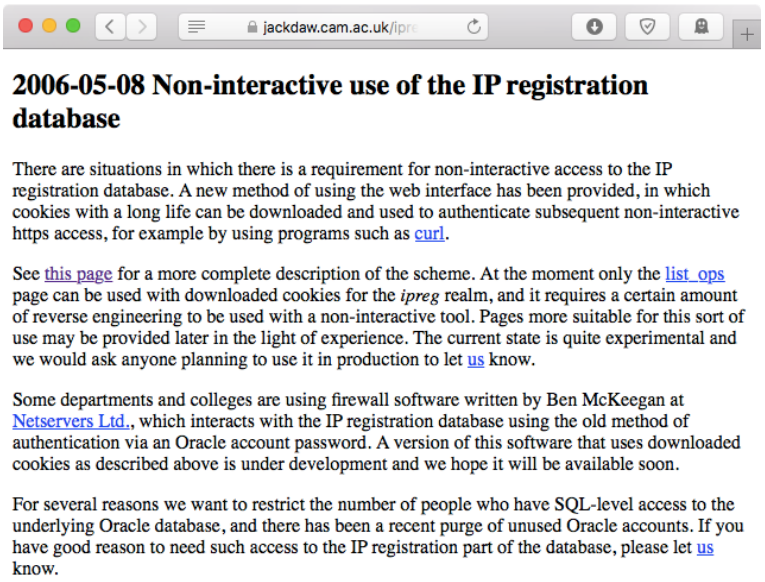
Almost the worst API is IP Register’s own API.

- Based on HTML forms
- Coupled to the user interface
- Widely used by college Firerack setups
- Extremely limited
- Requires re-enrolment every 4 months

IP Register – `list_ops`

As well as an example of IP Register’s delightful user interface and attractive visual style, this page is also its public API.

IP Register – API documentation



The documentation is mostly buried in a 10-year-old news item.

Questions?