New .CL
NIC Chile

Registration of .CL 1987

1993 First .CL DNS server

1996 1000 .CL domain names

1997 Policies for .CL, Registration payment

2004 100,000 .CL domain names, Anycast in .CL DNS servers

2005 IDN support

2007 Support IPv6 NS glue record

2011 DNSSEC in .CL

New .CL 2013
Old system: 1997 - 2012

- Monolithic and Tightly coupled systems
  - 2002: from data files to database
- Built “on-demand”
- One registrar without user accounts
- Local rules:
  - RUT (National ID number), Comuna (district or prefecture)
- Technology tools:
  - Perl (CGI, shell scripts, cron), MySQL, Apache
Old System

NIC 2

- Zone Generation
- Dispute Resolution System
- Domain Names Operations
- Payments Invoicing
New .CL

- Main Goals
  - User accounts
  - Registry – Registrar model
  - Scalability (new architecture)
  - Online Dispute Resolution System
  - Transfer domain names from old system to new system
  - Stop registration in the old system
  - New website
New .CL: model, protocols

Diagram:
- Registry
- RCAL
- Arbitration System
- Registrar
  - User accounts
- NIC2

Connections:
- WS between Registry and RCAL
- WS between RCAL and Arbitration System
- <epp> from Registrar to Registry and NIC2
New .CL: model, users

Diagram:

- Registry
- RCAL
- Arbitration System
- Registrar
- User accounts
- NIC2
- Domain holder
- Arbitrator
- Complainant
- Client

- Create new domains
- Renew, update, delete
- Transfer request
- Transfer cancel
- Domain name complaint
New .CL: technology

- Web applications:
  - Java: Spring and Struts Frameworks, HTML5
- Scheduled tasks:
  - Java, Perl, Python
- Database: Percona server
- Message broker: Activemq
- Web Server: Apache
- Web App Server: Tomcat
New .CL: servers architecture
Did it work?
Domain Names: where are they?

Old system: 186.120  New .CL: 301.462
Why did we do it on our own?

Reason #1

External tools need a lot of customization:
- More than 400,000 domains working with .CL policies: RUT, Comuna, Local Arbitration System, Payment/Invoicing System, internal bureaucracy, etc
Why did we do it on our own?

Reason #2

Requirements evolve, currently:
- More than 209,000 lines of code in 15 new systems
- More than 350 new installs (only main systems)
Why did we do it on our own?

Reason #3

We have experienced engineers:
- developers, sysadmin, network admins, DNS admin
Lessons learned

• Coordination: this is an organization change
  – Direct line to customer service and other areas: legal, administrative, payments, invoicing
• Use the right technology for the right task
• If your pentest was OK the first time, don't trust it!
• Divide and conquer
  – Separate components and responsibilities
• It is useful to carry out stress tests and usability test
Thank you!

José Urzúa
jose@nic.cl