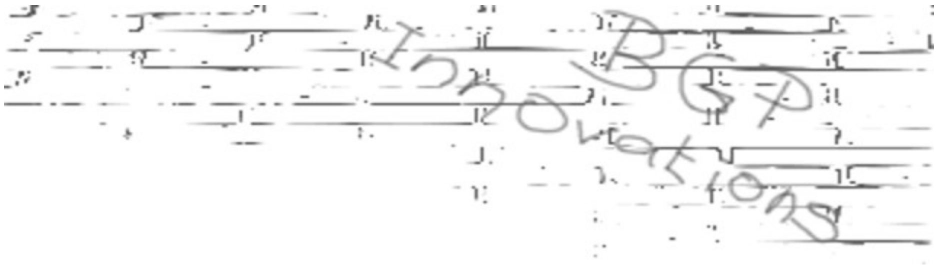
RDL: A programmatic approach to generating router configurations



Per Gregers Bilse

Benno Overeinder

RDL: The background

- ENGRIT: Extensible Next Generation Routing Information Toolset
- Improve Internet routing security and stability
- Multi-pronged approach, RDL is one aspect
- Other aspects will focus on authentication, etc
- NLnetLabs has done much work with DNS

RDL: The rationale

- Global turnover $dozens of millions per hour
- Even small problems can be very costly
- Router configuration is inherently low level
- Large number of only moderately related detail
- Limited or no verification tools
- Limited scope for inter-ISP routing management

RDL: The idea

- A high level Routing Documentation Language
- Dual purpose:
- 1) Architecture independent generation of BGP config:
  - RDL->Cisco, RDL->Juniper, RDL->BIRD
  - C->68k, C->x86_64, C->ARM
- 2) Description and publication of routing policies:
  - Enable automated verification and proofing
  - Improve exchange of information between peers

RDL: Not RPSL NG NG

- RDL intended to reuse parts of RPSL:
  - Some objects
  - Publication/repository means, where feasible
- But, more importantly:
  - RDL to describe BGP topology
  - RDL to cover both iBGP and eBGP peerings
  - RDL to fully qualify and identify routing policies

RDL: What is a policy?

- Much confusion between **Policy** and **Enforcement Action**
- A policy is **Thieves will be prosecuted**
- An enforcement action is **Arrest Nosey Parker**
- Existing tools and approaches focus on enforcement actions
- Quickly degenerate into route filter mechanics

RDL: Policies in 3D

- A routing policy as seen by RDL has three dimensions to it:
  - Where it applies: topological location
  - When it applies: NLRI attributes
  - What to do: filtering and attribute manipulation
- Think of it as similar to a piece of legislation, eg speed limits: Where, When, What
- These three aspects jointly describe a given policy in its entirety

RDL: A policy example

- Policy: My AS will not announce bogons
- RDL's 3D approach:
  - Where: all peerings with foreign ASs
  - When: prefix is in list of bogons
  - What: block it
- RDL's BGP topology description is the key to specifying the **Where** of a policy
- the **Where** is statically analysed and applied when generating configurations
- The **When** and the **What** are done by the routers

RDL: The language

- Designed specifically for the purpose of describing BGP topologies simply and intuitively
- Free form curly brace, recursive, and concatenative syntax, allowing quick and easy specification of objects and their location
- Borrows inadvertently and disrespectfully from several unusual languages
- Fully dynamically typed and declaration free

RDL: BGP topology

- RDL describes BGP topology by way of three objects:
  - Zones – may contain other zones, and routers
  - Routers – may contain one or more eBGP peers
  - Peers
- Structure similar to file system directories
- Each object has a number of attributes
- Attributes may be inherited from lexical scope

RDL: Topology example

```
hibernia = new(zone) . {
  asn = 5580;
  EU = new(zone) . {
    NL = new(zone) . {
      ams1 = new(router) . {
        address = 134.222.1.1;
        ripe = new(peer) . { 1.2.3.4, 3333 };
      };
    };
  };
  US = new(zone) . { .... };
  APAC = new(zone) . { ... };
};
```

RDL: What's in a zone

- Zones are containers for similar policies
  - often significant geographical correlation
  - should be chosen to reflect the reality of your network, not the other way around (your network is the ground, the zone map is the map)
  - you decide what your zone map should be, it is there to help you
  - again: RDL is all about BGP topology
  - the zone map identifies reference points for policies

RDL: Policy example

- Policy descriptions follow the topology format

```
nobogons = new(policy) . {
  where = export peer.asn != peer.remote.asn;
  when = nlri.prefix & bogons;
  what = reject;
};
bogons = { 0.0.0.0/8^+, 10.0.0.0/8^+, 100.64.0.0/10^+, ... };
```

- Policy syntax is experimental/undecided
- Probably a good idea to stick to general syntax of RDL

RDL: Unusual Example I

```
hibernia = new(zone) . {
  asn = 5580;
  RR1 = new(router) . { 134.222.12.1, RR };
  EU = new(zone) . {
    ibgp = { RR1, localmesh };
    NL = new(zone) . {
      ams1 = new(router) . { 134.222.1.1 } . { ... };
    };
  };
  US = new(zone) . { ibgp = { RR1, localmesh }; ... };
};
```

RDL: Unusual Example II

- Policy: de-prioritise all EU routes in US
- RDL to the rescue:

```
EUexport = new(policy) . {
  where = export peer.zone <= EU && peer.remote.zone <= US;
  when = ;
  what = local-preference = 90;
};
```

- Because RR1 is a route reflector it is transparent

RDL: Nirvana

RDL is all about not **configuring routers**, but **programming the AS**.

ENGRIT + admin: benno@nlnetlabs.nl

RDL: pgb@bgpinnovations.com