RPSL with IRRToolSet

Muhammad Moinur Rahman

IRR Toolset, RPSL: Introduction

- Tutorial
  - Do not think of bypassing the RFC
- Target audience
  - Knowledge of Internet Routing(specially BGP)
  - Familiar with any IRR Database
  - No need to know Internet Routing Registry
- Layout
  - Theory
  - Handson Lab using IRR Power Tools, Net:IRR, rpsltools and IRRToolSet

Historical Context

- The basic concept of routing registries dates back to the 1980's and NSFNet
- A high-level policy based routing database (PRDB) was used to generate configs
- NSFNet regional networks were required to submit Network Announcement Change Requests (NACR) to update the PRDB
- NACR’s documented connected networks and their Autonomous System numbers

Historical Context (Early European Works)

- RIPE – Reseaux IP Europeens
- Formed in 1989 to coordinate and promote IP networking in Europe
- Developed a registry for allocation of IP addresses and Autonomous System numbers in Europe (first RIR)
- No routing policy support initially

Historical Context (RIPE)

- RIPE-81 document was published in Feb, 1993 - extended the RIPE address registry to include basic routing policy information
- Added ability to specify an Autonomous System number for an IP address allocation
- Also allowed the expression of Autonomous System relationships

Historical Context (RIPE-181)

- RIPE-181 (RIPE-81++) document was published in Oct, 1994
- Introduced concept of object classes
- Separated routing policy information from IP address allocation information with introduction of the “route” object
- Extended Autonomous System policy expression functionality
- Also adopted a mechanism for grouping Autonomous Systems with the “as-macro”

Historical Context (RPSL)

- In March 1995, the RIPE-181 standard was accepted as an IETF informational document – RFC-1786
- IETF created the Routing Policy System Working Group to revise and standardize the language under the auspices of the IETF
- Result was known as the Routing Policy Specification Language (RPSL)

Historical Context (RFC-2622)

- RFC 2622 was released in June, 1999 and formally defined RPSL standard
- Based on the RIPE-181 standard
  - Significantly extended the functionality of the aut-num object
  - route object also extended
  - as-macro became as-set object
  - Added a number of new object types
  - Included a dictionary based extension mechanism

Historical Context (RFC-2622 New Objects)

- as-set
- route-set
- filter-set
- rtr-set
- peering-set
- inet-rtr
- mntner, role, and person objects for authentication and contact information

Historical Context (RFC-4012 RPSLng)

- IPv6 and multicast support
- Address Family Identifier(afi i.e, ipv4 and ipv6)
- MPBGP added in protocol Dictionary
- RPSL types ipv6-address, ipv6-address-prefix and ipv6-address-prefix- range added
- Policy Attribute mp-import, mp-export and mp-default added
- Class route6 added
- route-set class now supports both IPv4 and IPv6 mp-members
- peering-set supports mp-peering attribute
- rtr-set class supports both IPv4 and IPv6

Routing Policy Specification Language(RPSL)

- Object-based language
  - route, autonomous system, router, contact and set objects
- Defines the syntax, semantics and format of data in IRR
- Vendor independent
- Extensible
- IETF Proposed Standard (RFC2622) later superseded by RPSLng (RFC4012)
- Based on RIPE-181 (RFC 1786)

RPSL Basics

- Each object type (class) contains mandatory and optional attributes
- All objects must have these attributes
  - mnt-by: identifies mntner object that controls the object
  - changed: lists email and time of change
  - source: identifies the registry name where the object is located

mntner Object

- Mntner is an abbreviation of maintainer
- Identifies accounts in the registry
- Maintainer objects used for authentication
- Specifies authentication mechanism in the “auth” attribute
  - CRYPT-PW or MD5-PW - password auth
  - PGP-KEY – PGP/GPG based auth
  - MAIL-FROM – email based auth
  - NONE

mntner Object

```
mntner:  [mandatory] [single] [primary/look-up key]
descr:   [mandatory] [multiple]
admin-c: [mandatory] [multiple] [inverse key]
tech-c:  [optional] [multiple] [inverse key]
upd-to:  [mandatory] [multiple] [inverse key]
mnt-nfy: [optional] [multiple] [inverse key]
auth:              [mandatory] [multiple]
remarks: [optional] [multiple]
notify:  [optional] [multiple] [inverse key]
mnt-by:  [mandatory] [multiple] [inverse key]
changed: [mandatory] [multiple]
source:  [mandatory] [single]
```

mntner Object Example

```
mntner:         MAINT-BD-1ASIAAHL
descr:          1Asia Alliance Communication Ltd
country:        BD
admin-c:        MMR13-AP
upd-to:         hostmaster@1asia-ahl.com
mnt-by:         MAINT-BD-1ASIAAHL
auth:           # Filtered
referral-by:    APNIC-HM
changed:        moin@1asia-ahl.com 20121127
source:         APNIC
```

route/route6 Object

- Defines a CIDR prefix and origin AS
- Most common type of object found in routing registries
- Used by a number of ISP's to generate filters on their customer BGP sessions
  - Customers must register all routes in order for their ISP to route them
  - Allows automation of adding new prefixes

route/route6 object and keys

- Every RPSL class has a primary “key”
- For most classes, it is simply the main class attribute value
- For example, the mntner class uses the mntner attribute value as the key
- However, route objects use both route and origin fields as the primary key
- There can be multiple objects for the same prefix with different origins
- This is by design
  - Multi-origin multi-homing
  - When changing to a new origin AS, want routes for both until switched
- However, also many cases of multiples due to stale routes not being cleaned

route/route6 Object Format

```
route: [mandatory] [single] [primary/look-up key]
descr: [mandatory] [multiple]
origin: [mandatory] [single] [primary/inverse key]
withdrawn: [optional] [single]
member-of: [optional] [single] [inverse key]
inject: [optional] [multiple]
components: [optional] [single]
aggr-bndry: [optional] [single] [inverse key]
aggr-mtd: [optional] [single]
export-comps: [optional] [single]
holes: [optional] [single]
remarks: [optional] [multiple]
cross-nfy: [optional] [multiple] [inverse key]
cross-mnt: [optional] [multiple] [inverse key]
notify: [optional] [multiple] [inverse key]
mnt-by: [mandatory] [multiple] [inverse key]
changed: [mandatory] [multiple]
source: [mandatory] [single]
```

route/route6 Object Example

```
route:          182.16.140.0/22
descr:          1Asia Communication Pte Ltd
origin:         AS10102
mnt-lower:      MAINT-BD-1ASIAAHL
mnt-routes:     MAINT-BD-1ASIAAHL
mnt-by:         MAINT-BD-1ASIAAHL
changed:        moin@1asia-ahl.com 20121209
source:         APNIC
```

aut-num Object

- Defines routing policy for an AS
- Uses mp-import: and mp-export: attributes to specify policy
- Can be used for highly detailed policy descriptions and automated config generation
- Can reference other registry objects such as
  - as-sets
  - route-sets
  - filter-sets

aut-num Object Format

```
aut-num:     [mandatory] [single] [primary/look-up key]
as-name:     [mandatory] [single]
descr:                   [mandatory] [multiple]
member-of:   [optional] [single] [inverse key]
import:      [optional] [multiple] [inverse key]
export:      [optional] [multiple] [inverse key]
default:     [optional] [multiple] [inverse key]
admin-c:     [mandatory] [multiple] [inverse key]
tech-c:      [mandatory] [multiple] [inverse key]
remarks:     [optional] [multiple]
cross-nfy:   [optional] [multiple] [inverse key]
cross-mnt:   [optional] [multiple] [inverse key]
notify:      [optional] [multiple] [inverse key]
mnt-by:      [mandatory] [multiple] [inverse key]
changed:     [mandatory] [multiple]
source:      [mandatory] [single]
```

aut-num Object Example

```
aut-num:    AS10102
as-name:    SG-1ASIACOM-AS-AP
descr:      1Asia Communication Pte Ltd
descr:      151 Chin Swee Road
descr:      14-01 Manhattan House
country:    SG
admin-c:    SHC12-AP
tech-c:     MMR13-AP
mnt-by:     MAINT-SG-1ASIACOM-SG
mnt-routes:MAINT-SG-1ASIACOM-SG
mnt-irt:    IRT-SG-1ASIACOM-SG
changed:    hm-changed@apnic.net 20100428
changed:    hm-changed@apnic.net 20121116
source:     APNIC
```

as-set Object

- Provides a way of grouping AS'es
- Name must begin with prefix “AS-” or in the format
  - AS<NUM>:AS-CUSTOMERS
  - AS<NUM>:AS-PEERS
- Frequently used to list downstream/customer AS numbers
- Maybe referenced in aut-num import/export policy expressions
- Can reference other as-set's

route-set Object

- Defines a set of routes prefixes
- Name must begin with prefix “RS-” or in the format ASNUM:RS-<ORGANIZATION>
- Can reference other route-sets
- Can also reference AS's or as-set's
  - In this case, the route-set will include all route object prefixes which have an origin which matches the AS numbers

route-set Object Format

```
route-set:      [mandatory] [single] [primary/look-up key]
descr:          [mandatory] [multiple]
members:        [optional] [single]
mbrs-by-ref:[optional] [single]
remarks:        [optional] [multiple]
tech-c:         [mandatory] [multiple] [inverse key]
admin-c:        [mandatory] [multiple] [inverse key]
notify:         [optional] [multiple] [inverse key]
mnt-by:         [mandatory] [multiple] [inverse key]
changed:        [mandatory] [multiple]
source:         [mandatory] [single]
```

route-set Object Example

```
route-set:      AS10102:RS-1ASIA
descr:              Routes announced across
Peers
members:
     103.4.108.0/22,182.16.140.0/22
tech-c:             MMR13-AP
admin-c:        MMR13-AP
mnt-by:             MAINT-BD-1ASIAAHL
changed:        moin@1asia-ahl.com 20140129
source:             APNIC
```

filter-set Object

- Defines a set of routes that are matched by a filter expression
- Similar in concept to route-set's
- Name must begin with prefix “fltr-”

The IRR(internet Routing Registry)

- Concept of “the” Internet Routing Registry system established in 1995
- Shares information regarding production Internet Routing Registries
- Web site at http://www.irr.net
- Initially RIPE-181 format, shifted to RPSL
- Mirror Routing Registry data in a common repository for simplified queries
- The IRR currently consists of roughly 35 operational registries
- Registries operators
  - Regional Internet Registers (RIR’s), such as ARIN, RIPE, and APNIC
  - ISP’s - SAVVIS, NTT, Level3
  - Non-affiliated public registries – RADB and ALTDB

RADB Routing Registry

- The RADB launched in 1995 as part of NSFNet funded Routing Arbiter project
- The Routing Arbiter project was intended to ease transition from the NSFNet to the commercial Internet
- Registry was used to configure Route Servers located at designated Network Access Points (NAP's) located in Chicago, Washington, New York, and San Francisco
- RADB transitioned from public NSFNet funding to fee-based model in 1999
- Re-branded Routing Assets Database in 2002 – http://www.radb.net
- The registry can be queried at website and via whois at whois.radb.net
- This server also mirrors the other registries in the IRR as documented at www.irr.net

Why Register?

- Document routing policy
- In particular, register route objects to associate network prefixes with origin AS
- A number of transit providers require their customers to register routes and filter customer route announcements based on registry contents
- Filters unauthorized announcements to prevent route hijacking, denial of service

Incidents

- BGP->RIP->BGP injection
- 128/7 leak
- bogon 0/0, 10/8 leaks
- Daily, someone is leaking somelse’s prefix.

Common IRR query flags

- IRR's support a number flag options
- -i flag performs inverse query
  - “-i origin AS10102” returns all route objects with an origin of AS10102
  - “-i mnt-by MAINT-AS10102” returns all routes maintained by MAINT-AS10102
- -M flag returns more specific route objects for a prefix
  - “-M 27.0.8.0/22” returns all more specific route objects in the 27.0.8.0/22 prefix
- -s flag limits number of sources queried
  - May not want to query all 30+ IRR db's
  - example, “-s RADB,RIPE”
- -K flag – return primary keys only
  - Useful for route object queries, excludes extraneous fields not needed for policy
  - Often used by tools

Advanced IRR queries

- IRRd provides the ability to perform server side set expansions (as-set and route-set)
- This is done with the “!i” query
  - “!iAS-ESNETUS” returns members of ASESNETUS as-set object
- Add a “,1” for a recursive expansions
  - “!iAS-ESNETUS,1” will recurse any as-set members and return individual as-members
  - Reduces number of queries to server

Advanced RPSL – aut-num

- The aut-num object can be used to express an Autonomous System's routing policy and peering information
- Powerful structured syntax allows for complex policy expressions
- Some operators drive their network configuration off of their RPSL data
- Others simply use it to document AS relationships in a public manner

RPSL Tools

- Several tools have been developed to facilitate the use of RPSL registry data in the configuration of networks
- Tools range from sophisticated and powerful to simple and limited
- Use the IRR by querying over the whois protocol
- Some ISP's use in-house developed tools which process RPSL database files directly

Tools of trade for RPSL

- IRRToolSet
- NET::IRR
  - Perl module supporting basic IRR queries
- IRR Power Tools
  - IRR based router configuration – PHP + CVS
- Rpsltool – generates cisco configs - Perl

Tools of trade for RPSL

- IRRToolSet
- NET::IRR
  - Perl module supporting basic IRR queries
- IRR Power Tools
  - IRR based router configuration – PHP + CVS
- Rpsltool – generates cisco configs - Perl

IRRToolSet

- Based on original RAToolSet used in NSF Routing Arbiter project
- Written in C++ and now maintainer by ISC
- rtconfig tool uses templates to generate router configs from IRR data
- Other provided tools include
  - peval – low level policy evaluation tools
  - rpslcheck – verfies RPSL syntax of objects
- Death of IRRToolSet??
- Revamped by ISC, yet complex to configure

Net::IRR

- Perl CPAN module
- Provide several useful Perl functions
  - get_routes_by_origin
  - get_ipv6_routes_by origin
  - get_as_set
  - get_route_set
  - route_search

IRR Power Tools

- PHP based toolset
  - http://sourceforge.net/projects/irrpt
- Allows ISP to easily track, manage and utilize IRR data
- Performs tracking with CVS
- Can email notifications of updates
- irrpt_pfxgen script can generate router configs in Cisco/ Foundry, Juniper, Extreme, and Force10 formats

Routing Registry Futures

- RPKI(Resource Public Key Infrastructure) work will likely have impact on routing registry usage
- APNIC along with RIPE has already designed the portal for RPKI usage
- Latest subset of IRRToolSet has added support for integrating RPKI along with RPSL

- Feeling sorry for being here .. ☹
  - Don’t be ..
    - Configuration part will make you think life is really easy .. ☺

Lets go for a Tea Break

IRR Toolset, RPSL: Installation

- Available in most Unix/Linux like OS
- Basic Requirements for IRRToolset are as of following
  - GNU Make
  - GCC
  - flex
  - bison
  - libtool
- Additional tools for autoconfiguration are as of following:
  - expect
  - cron

IRR Toolset, RPSL: Installation – Get Source

```
root@bofh:~ #wget ftp://ftp.isc.org/isc/
IRRToolSet/IRRToolSet-5.0.1/
irrtoolset-5.0.1.tar.gz
root@bofh:~ # tar -zxvf irrtoolset-5.0.1.tar.gz
root@bofh:~ # cd irrtoolset-5.0.1
```

IRR Toolset, RPSL: Installation – Build and Install

```
root@bofh:~irrtoolset-5.0.1# ./configure
root@bofh:~irrtoolset-5.0.1# make
root@bofh:~irrtoolset-5.0.1# make install
```

IRR Toolset, RPSL: RPSL Primer

```
root@bofh:~ whois –h whois.apnic.net AS131208
#####snipped######
mp-import: afi any.unicast {
                from AS-ANY accept ANY AND NOT RS-MARTIANS;
           } refine {
                from AS-ANY action pref = 50;
                        accept community.contains(131208:50);
                from AS-ANY action pref = 30;
                        accept community.contains(131208:70);
                from AS-ANY action pref = 10;
                        accept community.contains(131208:90);
                from AS-ANY action pref = 0; accept ANY;
           } refine afi ipv4.unicast {
```

IRR Toolset, RPSL: RPSL Primer(Contd)

```
from AS6453 66.110.0.126 at 103.4.109.254 action pref=10;
       community.append(131208:11000,131208:11010,131208:11011);
      accept ANY AND NOT RS-MARTIANS;
from AS58715 103.4.108.62 at 103.4.108.61 action
community.append(131208:41000,131208:41010,131208:41011); accept
AS-58715^24 AND <^AS58715+ AS-58715*$>;    from AS58656 103.4.108.94 at
103.4.108.93 action
community.append(131208:41000,131208:41010,131208:41011); accept AS-
BDHUB^24 AND <^AS58656+ AS-BDHUB*$>;
from AS58657 103.4.108.178 at 103.4.108.177 action
community.append(131208:41000,131208:41010,131208:41011); accept
AS58657^24 AND <^AS58657+$>;
from AS15169 27.0.9.10 at 27.0.9.9 action pref=5;
community.append(131208:31000,131208:31020,131208:31021); accept
AS15169^24 AND <^AS15169+ AS-GOOGLE*$>;
} refine afi ipv6.unicast {
```

IRR Toolset, RPSL: RPSL Primer(Contd)

```
from AS6453 2001:5a0:2300:100::55 at 2001:5a0:2300:100::56 action pref=10;
community.append(131208:11000,131208:11010,131208:11011);  accept ANY AND
NOT RS-MARTIANS;
from AS15169 2404:a100:2000::11 at 2404:a100:2000::12 action pref=5;
community.append(131208:31000,131208:31020,131208:31021);  accept AS15169
AND <^AS15169+ AS-GOOGLE*$>;
}
```

IRR Toolset, RPSL: rtconfig Caveats

- Hard to debug as debug message has no clue to original error
- By default uses irrd whois server which none of the RIR's uses except Merit RADB
- For using with APNIC, RIPE etc RIR's whois server we must change the protocol to bird(Original RIPE whois daemon)

IRR Toolset, RPSL: rtconfig

- Prompt based shell application
- root@bofh:~# rtconfig –h whois.apnic.net –protocol bird

```
rtconfig>
Takes any of the following commands:
@rtconfig import <ASN-1> <rtr-1> <ASN-2> <rtr-2>
@rtconfig export <ASN-1> <rtr-1> <ASN-2> <rtr-2>
@rtconfig configureRouter <inet-rtr-name>
@rtconfig importGroup <ASN-1> <peering-set-name>
@rtconfig exportGroup <ASN-1> <peering-set-name>
@rtconfig static2bgp <ASN-1> <rtr-1>
@rtconfig set sources = <source-list>
@rtconfig access_list filter <filter>
@rtconfig aspath_access_list filter <filter>
@rtconfig printPrefixes <format> filter <filter>
```

IRR Toolset, RPSL: rtconfig(Contd)

```
@rtconfig printPrefixRanges <format> filter <filter>
@rtconfig printSuperPrefixRanges <format> filter <filter>
```

IRR Toolset, RPSL: rtconfig(Contd)

```
Cisco Specific
@rtconfig set cisco_map_name = <map-name>
@rtconfig set cisco_map_first_no = <no>
@rtconfig set cisco_map_increment_by = <no>
@rtconfig set cisco_prefix_acl_no = <no>
@rtconfig set cisco_aspath_acl_no = <no>
@rtconfig set cisco_pktfilter_acl_no = <no>
@rtconfig set cisco_community_acl_no = <no>
@rtconfig set cisco_access_list_no = <no>
@rtconfig set cisco_max_preference = <no>
@rtconfig networks <ASN-1>
@rtconfig inbound_pkt_filter <if-name> <ASN-1> <rtr-1> <ASN-2> <rtr-2>
```

IRR Toolset, RPSL: rtconfig(Contd)

```
@rtconfig pkt_filter <if-name> <ASN-1> <rtr-1> <ASN-2> <rtr-2>
@rtconfig outbound_pkt_filter <if-name> <ASN-1> <rtr-1> <ASN-2> <rtr-2>
```

IRR Toolset, RPSL: rtconfig(Contd)

```
Junos Specific
@rtconfig set junos_policy_name = <policy-name>
@rtconfig networks <ASN-1>
```

IRR Toolset, RPSL: rtconfig Input File(Provision)

```
router bgp 131208
 neighbor 103.4.108.54 remote-as 58682
 neighbor 103.4.108.54 version 4
!
# Earth Communication Ltd
@RtConfig set cisco_access_list_no = 500
@RtConfig set cisco_map_name = "AS58715-IN"
@RtConfig import AS131208 103.4.108.62 AS58715 103.4.108.61
@RtConfig set cisco_access_list_no = 599
@RtConfig set cisco_map_name = "ANY"
@RtConfig export AS131208 103.4.108.62 AS58715 103.4.108.61
!
# BDHub Ltd
@RtConfig set cisco_access_list_no = 501
@RtConfig set cisco_map_name = "AS58656-IN"
@RtConfig import AS131208 103.4.108.94 AS58656 103.4.108.93
@RtConfig set cisco_access_list_no = 599
@RtConfig set cisco_map_name = "ANY"
@RtConfig export AS131208 103.4.108.94 AS58656 103.4.108.93
!
end
```

IRR Toolset, RPSL: rtconfig Input File(Output)

Live Demonstration. Output is attached as Provision1.txt

IRR Toolset, RPSL: Daily Changes

- For automated processing we concentrate on :
  - AS-SET

- Changes in AS-SET requires the following configuration changes:
  - Prefix-list
  - AS-PATH access list

IRR Toolset, RPSL: rtconfig Input File(Changes)

```
# Earth Communication Ltd
@RtConfig set cisco_access_list_no = 500
@RtConfig aspath_access_list filter <^AS58715+ AS-58715*$>
@RtConfig access_list filter AS-58715
# BDHub Ltd
@RtConfig set cisco_access_list_no = 501
@RtConfig aspath_access_list filter <^AS58656+ AS-BDHUB*$>
@RtConfig access_list filter AS-BDHUB
!
end
```

IRR Toolset, RPSL: rtconfig Input File(Output)

Live Demonstration. Output is attached as changes1.txt.

IRR Toolset, RPSL: Uploading Configuration

Various ways to upload configuration:

- SNMP Write
- NETCONF XML Based
- Automated Script using expect

IRR Toolset, RPSL: SNMP Write

Cons

- Secured only while SNMPv3 is used
- Uses UDP
- Long Running Process
- Non-Standard MIB
- Tough to integrate with rtconfig

IRR Toolset, RPSL: NETCONF

Cons

- Works good with so many routers
- Overkill for a small number of routers
- Needs detailed concept of XML and how it works
- Not for the faint hearted
- Need detailed idea of Yang too

IRR Toolset, RPSL: Expect

Expect is a tool for automating interactive applications such as telnet, ftp, passwd, fsck, rlogin, tip, etc.

Pros

- Good for automating tasks that prompts for information
- Easy to understand
- Used for automatic Testing

Cons

- Keeps login credentials inside script
- Wrong file permission can be fatal

IRR Toolset, RPSL: Script for Configuration

```
#!/usr/local/bin/expect
set timeout 500
set hostname "dhk-agg-rtr01.1asiacom.net"
set file [open changes1.txt r]
set username "rtconfig"
set password "yovHyWer@lijZashexyuefs7"

while {![eof $file]} {
        set buffer [read $file 10240000]
}
spawn ssh -2 -l $username $hostname

expect "assword:" {
        send "$password\n"
}
```

IRR Toolset, RPSL: Script for Configuration

```
expect "DHK-AGG-RTR01#" {
        send "conf t\n"
        expect "(config)#" {
                foreach line [split $buffer "\n"] {
                        send "$line\n"    }
                expect "(config)#" {
                        send "commit\n"
                        expect "(config)#" {
                                send "exit\n"        }
                }
        }
}
expect "DHK-AGG-RTR01#" {
        send "exit\n"
}
close $spawn_id
```

IRR Toolset, RPSL: Further Reading

- RFC-2622: Routing Policy Specification Language
- RFC-2725: Routing Policy System Security
- RFC-2650: Using RPSL in Practice
- RFC-4012: Routing Policy Specification Language next generation (RPSLng)
- RFC-2726: PGP Authentication for RIPE Database Updates
- RFC-2769: Routing Policy System Replication

IRR Toolset, RPSL: Questions

```
Contact
        person:         Muhammad Moinur Rahman
        address:        The Alliance Building. (6th Floor),
        address:        63 Pragati Sharani, Baridhara,
        country:        BD
        phone:          +8801977881132
        e-mail:         moin@1asia-ahl.com
        nic-hdl:        MMR13-AP
        notify:         moin@1asia-ahl.com
        mnt-by:         MAINT-BD-1ASIAAHL
        changed:        moin@1asia-ahl.com 20121128
        source:         APNIC
```