# APRICOT 2005: Network Management Workshop

Gaurab Raj Upadhaya
Dhurba Raj Bhandari
Tom Vest

# RPSL / IRRd / IRRToolSET

Gaurab Raj Upadhaya
Packet Clearing House
gaurab@pch.net

## RPSL – Routing Policy Specification Language

RPSL is Defined in RFC 2622

The best reference for RPSL is RFC 2650

# RPSL Tutorial

Mark Prior
Australia

## Agenda

- Routing Policy
  - What is Routing Policy?
  - Why define one?
- RPSL
  - What is RPSL?
  - Benefits of using RPSL
  - How to use RPSL.
- Questions anytime!

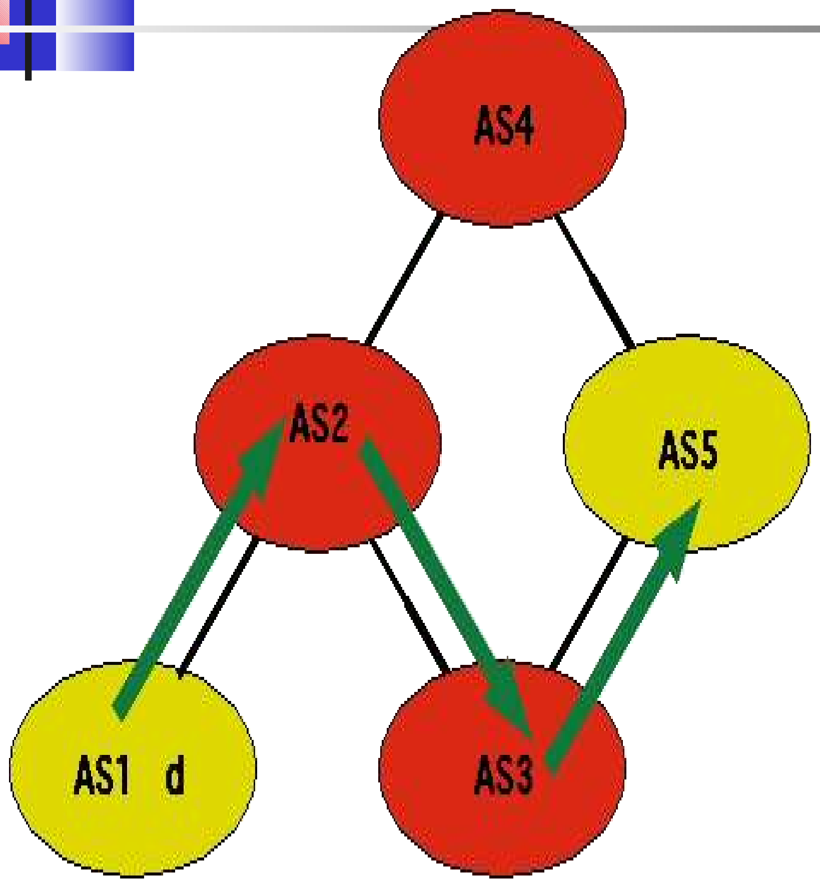
## What is Routing Policy

- Public description of the relationship between external BGP peers
- Can also describe internal BGP peer relationship

## Routing Policy

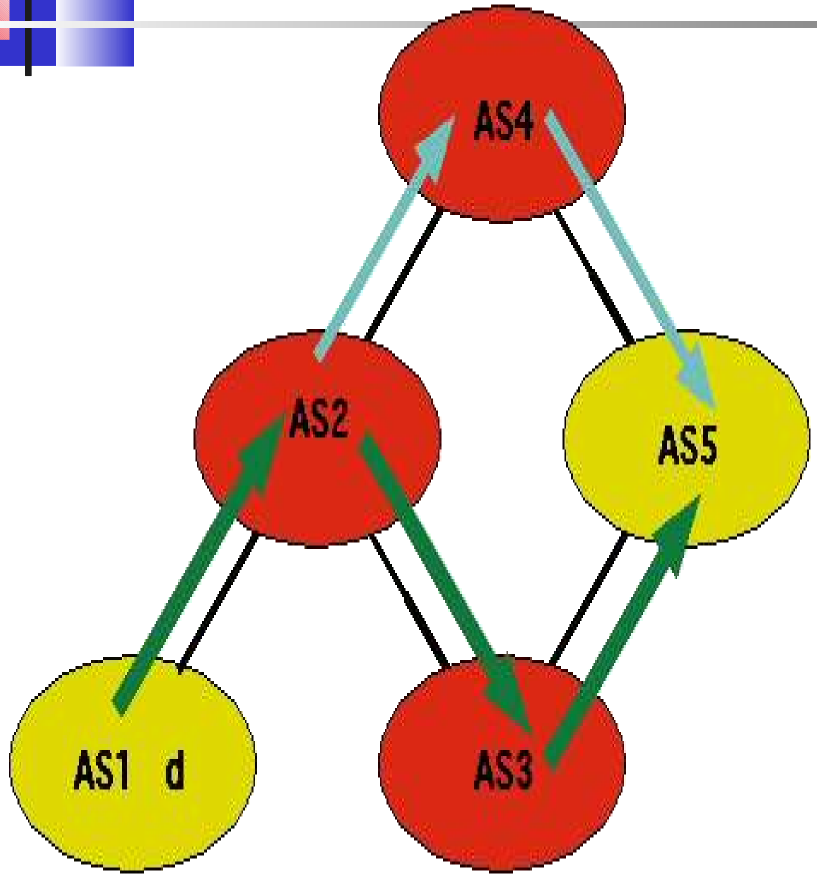
- Who are my BGP peers
- What routes are
  - Originated by a peer
  - Imported from each peer
  - Exported to each peer
  - Preferred when multiple routes exist
- What to do if no route exists

# Routing Policy Example



- AS1 originates prefix “d”
- AS1 exports “d” to AS2, AS2 imports
- AS2 exports “d” to AS3, AS3 imports
- AS3 exports “d” to AS5, AS5 imports

# Routing Policy Example (cont)



- AS5 also imports "d" from AS4
- Which route does it prefer?
  - Does it matter?
  - Consider case where
    - AS3 = Commercial Internet
    - AS4 = Internet2

## Why define a Routing Policy?

- Documentation
- Provides routing security
  - Can peer originate the route?
  - Can peer act as transit for the route?
- Allows automatic generation of router configurations
- Provides a debugging aid
  - Compare policy versus reality

## What is RPSL?

- Object oriented language
- Development of RIPE 181
- Structured whois objects
- Describes things interesting to routing policy
  - Routes
  - AS Numbers
  - Relationships between BGP peers
  - Management responsibility

**FOR MORE INFO...**

RFC 2622 - “Routing Policy Specification Language (RPSL)”

# Person, Role & Maintainer Objects

- Maintainer objects used for authentication
- Person and role objects are for contact info

```
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]
```

## Maintainer Object Example

```
mntner:        MAINT-AS2764
descr:         Maintainer for AS 2764
admin-c:       MP151
upd-to:        routing@connect.com.au
mnt-nfy:       routing@connect.com.au
auth:          PGPKEY-81E92D91
auth:          PGPKEY-562C2749
auth:          PGPKEY-8C1EEB21
mnt-by:        MAINT-AS2764
changed:       mrp@connect.com.au 20000725
source:        RADB
```

# Route Object

- Use CIDR length format
- Specifies origin AS for a route
- Can indicate membership of a route set

```
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 Object Example

```
route:         203.63.0.0/16
descr:         connect.com.au pty ltd
origin:        AS2764
notify:        routing@connect.com.au
mnt-by:        MAINT-AS2764
changed:       mrp@connect.com.au 19971027
source:        RADB
```

# AS Set

- Collect together Autonomous Systems with shared properties
- Can be used in policy in place of AS
- RPSL has hierarchical names

```
as-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]
```

## AS Set Object Example

```
as-set:        AS2764:AS-CUSTOMERS:AS3409
descr:         connect.com.au AS set
members:       AS7632, AS9324
remarks:       Autonomous systems that transit through AS3409
admin-c:       CC89
tech-c:        MP151
mnt-by:        MAINT-AS2764
changed:       mrp@connect.com.au 20001214
source:        RADB
```

## Route Set

- Collects routes together with similar properties

```
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:      AS2764:RS-PROVIDER
descr:          Connect's provider blocks
members:        202.21.8.0/21, 203.8.176.0/21, 203.63.0.0/16,
210.8.0.0/15, 210.10.0.0/16
admin-c:        CC89
tech-c:         MP151
notify:         routing@connect.com.au
mnt-by:         MAINT-AS2764
changed:        mrp@connect.com.au 20000604
source:         RADB
```

## Autonomous System Object

- Routing Policy Description object
- Most important components are
    - import
    - export
- These define the incoming and outgoing routing announcement relationships

# Autonomous System Object (cont)

```
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]
```

# Simple “Documentation” Policy

- The simplest policy is strict customer/provider relationship
  - Customer accepts everything the provider sends
  - Customer sends its routes to provider

```
aut-num:      AS2
as-name:      EXAMPLE-NET
descr:        RPSL Example
import:       from AS1 accept ANY
export:       to AS1 announce AS2
admin-c:      MANAGEMENT
tech-c:       OPERATIONS
mnt-by:       MAINT-AS2
changed:       noc@example.net  20010101
source:       TEST
```

## Why use (RPSL) Policy?

- Consistent configuration between BGP peers (peers & customers)
- Expertise encoded in the tools that generate the policy rather than engineer configuring peering session
- Automatic, manageable solution for filter generation

## Use of RPSL

- Use RtConfig  to generate filters based on information stored in our routing registry
  - Avoid filter errors (typos)
  - Filters consistent with documented policy (need to get policy correct though)
  - Engineers don’t need to understand filter rules (it just works :-)

## References

- RPSL - RFC 2622
  - ftp://munnari.oz.au/rfc/rfc2622.Z
- Using RPSL in Practice - RFC 2650
  - ftp://munnari.oz.au/rfc/rfc2650.Z
- RAToolSet
  - ftp://ftp.isi.edu/ra/RAToolSet
- RPSL Training Page
  - http://www.isi.edu/ra/rps/training
- RADB
  - http://www.merit.edu/radb

# RPSL / IRRd / IRRToolSET

Gaurab Raj Upadhaya
Packet Clearing House
gaurab@pch.net

# IRRD

- IRRd is a complete Internet Routing Registry Server supporting indexing, mirroring, whois queries, and email/TCP updates.
- Developed by Merit and used for RADB
- Download from www.irrd.net

# IRRd

- Source code and documentation for IRRd is available online at:
- http://www.irrd.net, current version is 2.2.3

- Also, a user guide is included as part of the distribution as irrd-user.pdf.
- IRRd software is used to run Merit's RADB routing registry which can be queried at whois.radb.net. For more info on the RADB, see www.radb.net.

- To build and install the distribution, execute the following commands:

```
cd src
./configure
make
make install
```

- Binaries are installed in /usr/local/sbin by default.

## Configuration

- Irrd has two ports
  - 5674 for user interface
  - 43 for listening (whois port)
- Configuration file is /etc/irrd.conf
  - New installs don’t have any config
- Configuration language is similar to Cisco CLI

## Setting it up

- root@ktm # irrd &
- root@ktm # telnet localhost 5674

- … do the lab.. ☺

(if you put in the appropriate line in /etc/services you can telnet differently)

## IRRd Databases

- You can mirror an Internet Routing Registry by contacting one of them

```
irr_database radb.db mirror_host 198.108.0.18 43
```

- You set up your own locally authorative routing registry

```
irr_database local.db authoritative
```

- In either case, your database is in the RPSL Format

# /etc/irrd.conf

```
!
! Sample config file
!
! The cache directory
irr_directory /var/irr/databases/
debug server file-name /var/log/irrd.log
debug server syslog
debug submission file-name /var/log/irr-email.log
!
! The port of whois and IRRToolset connections
irr_port 43
!
! Make sure we don't get overwhelmed
irr_max_connections 42
irr_database radb.db mirror_host 198.108.0.18 43
irr_database radb.db clean 172800
irr_database local.db authoritative
irr_database local.db clean 172800
```

# IRRToolSET ?

- The "Internet Routing Registry Toolset" (IRRToolSet) project was a activity of the RIPE NCC. This project had been migrated from the USC Information Sciences Institute, where it was developed in 1997-2001 as the "Routing Arbiter ToolSet" (RAToolSet) project. As the RAToolSet was no longer developed by ISI but was used worldwide, the RIPE NCC proposed to migrate this project to the RIPE NCC in order to continue its development and support. The original name of the project was preserved during the transition process, but wa finally changed to IRRToolSet.

## IRRToolSet

- Recently
  - The work on IRRToolSet has migrated to ISC (internet systems consortium), the same people who also maintain BIND and DHCPd
  - Website is www.isc.org/irrtoolset/
  - Hopefully we'll see some more development

## Functionality

The project consists of the following tools:

- RtConfig

  analyzes the routing policies registered in the Internet Routing Registry (IRR) and produces router configuration files;

- CIDRAdvisor

  suggests safe cidr aggregates (i.e. those that do not violate any policy constraints) that an Autonomous System (AS) can advertise to each of its neighbour ASes;

- peval

  low level policy evaluation tool that can be used to write router configuration generators;
- prtraceroute

  prints the route and policy information packets take to a network host;
- prpath

  enumerates a list of paths between Autonomous System and specified destination;
- aoe

  C++/Tcl/Tk program that displays the aut-num object for the specified Autonomous System;
- roe

  C++/Tcl/Tk program that lists the routes registered by the specified autonomous system;
- rpslcheck(prcheck)

  syntax-checks the aut-num object for Autonomous System registered in the

  Internet Routing Registry (IRR).

## How these stack up

- You can use the rtconfig to query the irrd or any other Routing Registry on the Internet

- You can create an network wide Routing Registry for your network or mirror a copy from one of the existing IRR operator

- **IRRd Synopsis**
- **irrd** [–a] [–d *database_dir*] [–f *conf_file*] [–g *group_name*]
- [–l *user_name*] [–n] [–s *password*] [–u] [–v] [–w *irr_port*] [–x]
- **Options**
- `-a` Enable atomic transactions for database updates
- `-d <path>` Set database directory
- 5
- Chapter 3. Using IRRd
- `-f <conf file>` Specify the configuration file to use (default: /etc/irrd.conf)
- `-g <group name>` Drop priveleges to given group name
- `-l <user name>` Drop priveleges to given user name
- `-n` Do not daemonize
- `-s <password>` Set the UII password
- `-u` Don’t allow privileged commands
- `-v` Verbose logging, debug mode

## IRRD Interface

- **Interactive Interface**

IRRd provides an interactive user interface that can be used to control various and operational aspects of IRRd and show the current status of the daemon. The port number can be specified in the configuration file. The default is TCP port 5673, or the number associated with "irrd" in /etc/services. If a password is specified in the configuration file, it must be supplied on login.

Unix shell-like redirection (or filename) is available for output. To edit a line, emacslike line editing including ^a, ^b, ^e, ^f, ^d, ^k, ^u and ^c is available. To reuse a previous line, tcsh-line history function is available by typing ^p and ^n.

# IRRD Command Syntax

The IRRd command language shares many similarities with the language used on Cisco Systems routers. Commands include:

- **show config** -- view the configuration file
- **show version** -- show the current version
- **show threads** -- show the status of application threads
- **show connections** -- show current TCP tool queries
- **reboot** -- restart the daemon
- **help** -- shows all commands available
- **exit** -- leave the UII interface
- **mirror** -- synchronize database with remote server
- **reload** -- reload an IRR database file
- **show database** -- show database status
- **dbclean** -- synchronize IRR diskfiles with memory

***Demonstration of the syntax on IRRD installed earlier.***

# Updating your RR

» When using IRRd to run an authoritative database registry (as opposed to simply mirroring other registries), it will be necessary to configure the irr_rpsl_submit program to accept e-mail and/or TCP based object submisssions. This program performs RPSL syntax checking and maintainer authorization verification and acts as a frontend for IRRd.

» The irr_rpsl_submit command is configured by command line flag values, by setting configuration commands in the IRRd configuration file, or by a combination of both.Command line options override options set in the IRRd configuration file.

## irr_rpsl_submit

- irr_rpsl_submit accepts e-mail updates and controls the process of entering and modifyingdatabase data. irr_rpsl_submit can perform PGP authentication, the standard authentication mechanisms of encrypted password and mail-from, syntax checking, and standard RIPE/RPSL notifications.
- /etc/irrd.conf is the default. ’-l’ specifies the location for the acknowledgement and transaction logs. The default is the ’irr_directory’ value from /etc/irrd.conf. ’-r’ gives the PGP ring files location. The default is ~/.pgp in the user’s home directory. ’-s’ specifies authoritative databases.
- irr_rpsl_submit will only allow updates to authoritative databases and will signal an error for all others. The ’-s’ option may appear multiple times as necessary. ’-x’ stops notifications from being sent. ’filename’ is the name of the input file. The irr_rpsl_submit flag options override options in the IRRd configuration file.
- These options enable irr_rpsl_submit to reside on a remote machine from IRRd and to operate without an IRRd configuration file
- To enable the irr_rpsl_submit to receive TCP submission, it is run through the inetd daemon

# IRRD Lab Work

- Creating a configuration for the first time
- Getting status reports from IRRD
- Setting up authentication
- Setting up administrative e-mail
- Learing the user interface

# Updating RR using e-mail

Step One - Register One or More Maintainers

Step Two - Register AS and Policy Information

Step Three – Register Routes

Step Four – error checking and Overrides

(You can Refer to your RIR IRR examples)
(Now let's go and create some of our own)

## Creating configuration

- You can use the information from the RR to create configurations for your Cisco, Juniper and other routers.
- You use the IRRToolSet and RtConfig to achieve this functionality
- (Let's do the lab on creating the configs)

# RtConfig

- Version > 4.0 supports RPSL
- Generates cisco configurations
- Contributed support for Bay’s BCC, Juniper’s Junos and Gated/RSd
- Creates route and AS path filters.
- Can also create ingress/egress filters

## Using RtConfig for static route importation into BGP

- We use policy to filter static routes into BGP
  - Allows for martian filtering
  - Tagging routes with special communities
  - Other filtering, such as filter host routes

```
import:          protocol STATIC into BGP4
                  from AS2170
                  action community.append(2170:1);
                 accept AS2170
```

- @RtConfig import/export <ASN-1> <rtr-1> <ASN-2> <rtr-2>
- <ASN-1> and <ASN-2> are AS numbers preceded with string
- "AS". For example, AS number 1 is specified as "AS1".
- <rtr-1> and <rtr-2> are ip addresses in prefix notation.
- For example, the router with address 128.9.128.9 is spec-
- ified as "128.9.128.9". This command instructs RtConfig
- to generate import filters where <rtr-1> in <ASN-1> is
- importing routes from <rtr-2> in <ASN-2>. The appropri-
- ate filters are generated by considering the import lines
- for <ASN-2>-<rtr-1>-<rtr-2> in the aut-num object for
- <ASN-1>.
-

## RtConfig commands for static import

```
RtConfig> @RtConfig set cisco_map_name = "STATIC-EXPORT"
RtConfig> @RtConfig static2bgp AS2170 0.0.0.0
!
no access-list 100
access-list 100 permit ip 203.17.185.0   0.0.0.0   255.255.255.0   0.0.0.0
access-list 100 permit ip 205.191.168.0   0.0.0.0   255.255.255.0   0.0.0.0
access-list 100 permit ip 210.8.207.176   0.0.0.0   255.255.255.240   0.0.0.0
access-list 100 deny ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
!
no route-map STATIC-EXPORT
!
route-map STATIC-EXPORT permit 1
 match ip address 100
 set community  2170:1 additive
!
router bgp 2170
 redistribute static route-map STATIC-EXPORT
```

# Customer Import Policy

```
import: {
     from AS-ANY
          action med=0;
          accept ANY AND NOT { 0.0.0.0/0 };
     } refine {
     from AS-ANY
          action community.append(2764:65408); pref=25;
          accept community.contains(2764:3) AND NOT AS2764:RS-PROVIDER^-;
     from AS-ANY
          action community.append(2764:65408); pref=15;
          accept community.contains(2764:4) AND NOT AS2764:RS-PROVIDER^-;
     from AS-ANY
          action community.append(2764:65408); pref=5;
          accept community.contains(2764:5);
     from AS-ANY
          action community.append(2764:65408); pref=0;
          accept ANY;
     } refine {
     from AS2764:AS-CUSTOMERS
          accept PeerAS AND <^PeerAS+$>;
     from AS2764:AS-TRANSIT
          accept AS2764:AS-CUSTOMERS:PeerAS AND <^PeerAS+ AS2764:AS-CUSTOMERS:PeerAS+$>;
     }
```

# RtConfig Configuration Template

```
@RtConfig set cisco_map_first_no = 10
@RtConfig set cisco_map_increment_by = 10
@RtConfig set cisco_prefix_acl_no = 130
@RtConfig set cisco_aspath_acl_no = 130
@RtConfig set cisco_pktfilter_acl_no = 130
@RtConfig set cisco_community_acl_no = 30
@RtConfig set cisco_max_preference = 100
!
router bgp 2764
neighbor 203.63.122.193 remote-as 9313
neighbor 203.63.122.193 description On The Net
@RtConfig set cisco_map_name = "AS9313-EXPORT"
@RtConfig export AS2764 203.63.80.230 AS9313 203.63.122.193
@RtConfig set cisco_map_name = "AS9313-IMPORT"
@RtConfig import AS2764 203.63.80.230 AS9313 203.63.122.193
!
end
```

## cisco Configuration

```
! access-list 135 – customer routes
!
no ip as-path access-list  130
ip as-path access-list 130 permit ^(_9313)+$
!
no route-map AS9313-IMPORT
!
no ip community-list 32
ip community-list 32 permit 2764:3
!
route-map AS9313-IMPORT permit 20
 match as-path 130
 match community 32
 match ip address 135
 set local-preference 75
!
no ip community-list 33
ip community-list 33 permit 2764:4
!
route-map AS9313-IMPORT permit 30
 match as-path 130
 match community 33
 match ip address 135
 set local-preference 85
no ip community-list 34
ip community-list 34 permit 2764:5
!
route-map AS9313-IMPORT permit 40
 match as-path 130
 match community 34
 match ip address 135
 set local-preference 95
!
route-map AS9313-IMPORT permit 50
 match as-path 130
 match ip address 135
 set local-preference 100
!
router bgp 2764
neighbor 203.63.122.193 route-map AS9313-IMPORT in
!
end
```

## Problems?

- Policy can easily get very complex and result in even more complex router configuration
- Line limit on cisco AS path filters (need to be careful when using as-sets)
- ISI/Qwest whois server doesn’t cope with the RPSL v2 community format

# References

- **RFC-1786: RIPE-181**
- RIPE-181 (RIPE-81++) started it all. This document describes the original database formats used by the RIPE NCC for the storage of routing policy in its database.
- **RFC-2622: Routing Policy Specification Language**
- The current routing language used by IRRd.
- **RFC-2650: Using RPSL in Practice**
- A tutorial that gives many examples of common policies in RPSL.
- **RFC-2726: PGP Authentication for RIPE Database Updates**
- How to store PGP public keys within the RIPE database format,and by extension, the RPSL database
- **RFC-2725: Routing Policy System Security**
- The RPSL-Security specification provides a mechanism for delegating objects and providing a rooted (top-down) delegation and authentication model for objects such as AS numbers, address space and routes. Status: IRRd does not yet support this RFC.
- **RFC-2769: Routing Policy System Replication**
- This mechanism provides for a more robust and authenticated mechanism of distributing data from registry to registry.Status: IRRd does not yet support this RFC.

# RPSL / IRRd / IRRToolSET

Gaurab Raj Upadhaya
Packet Clearing House
gaurab@pch.net

# Using IRR for BGP

- http://www.ripe.net/db/rpsl/bgp-conf-irr.html