INTERNET-DRAFT
File name: draft-ardas-rpcl-00.txt

Ardas Cilingiroglu
FORE Systems
Tony Przygienda
Bell Labs, Lucent Technologies
Expires : February 1999

Routing Policy Configuration Language (RPCL)

Status Of This Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as "work in progress".

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet- Drafts Shadow Directories on ftp.ietf.org (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

Table Of Contents

1. Introduction

Routing policies define a set of rules and actions to be applied by protocols when they send and receive routing updates. When a protocol receives an updates, routing policies are used to determine if, and with what information a route gets added to the Routing Information Base (RIB). When preparing to send a routing update, routing policy is consulted to see if the route is to be sent and if so, what information is to be included in the update. The Routing Policy Configuration Language (RPCL) specifies a language for defining routing policies in a router.

RPCL is derived from the Routing Policy Specification Language (RPSL) [1, 2], which is the evolving standard for Internet's policy specification. RPSL specifies the routing policies, inter-domain routes, border routers and some administrative objects of an Autonomous System (AS). Autonomous Systems (AS) in the Internet registers its RPSL specification with a cooperatively maintained distributed database, called Internet Routing Registry (IRR) [3,4]. IRR has the global view of the Internet and its policies. The IRR is used to verify the integrity of the Internet's routing, and to protect the Internet against accidental and malicious distribution of inaccurate route information, such as non-existing routes, incorrect aggregation or aggregation boundaries.

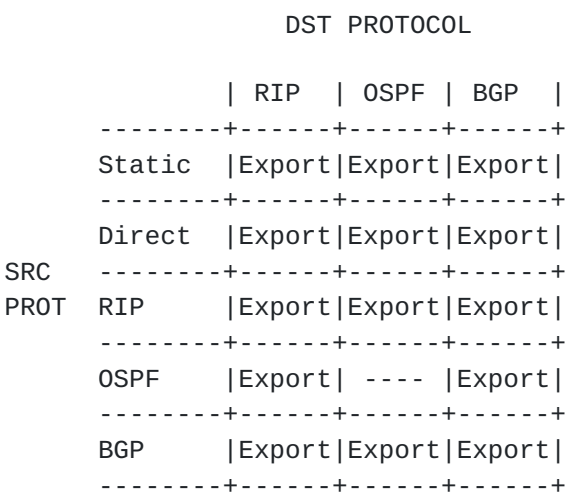
Unlike RPSL, which specifies the policies and other internal objects of an AS, RPCL is used to configure the policies of an individual router. Its syntax and semantics for the policy language is, however, derived from the RPSL. This consistency in policy specification language at the AS level and the router level facilitates the conversion from one to the other.

RPCL is intended to be an extendable language. It provides a uniform syntax and semantics for policy configuration across various policies and protocols. RPCL is designed to be able to add support for new protocols and specifications for new policies. For example, RPCL currently defines policies for unicast protocols, but it can be extended to support multicast protocols as well.

RPCL currently supports the configuration and evaluation of the following policies:

- Import Policy: used by protocols to update the content of the RIB based on the routes received. When a protocol receives a route, it uses the import policy to determine whether to accept or block the route, and to set the route's attributes (e.g., the route's preference) if the route is accepted. RIP, OSPF and BGP are example unicast protocols using import policies.

- Export Policy: used by protocols to determine what routes and associated attributes to send to neighbors based on the contents of the RIB. There is a separate policy configured for each protocol pairs (SRC-PR, DST-PR), where DST-PR is the exporting protocol and the SRC-PR is the owner protocol of the route considered. Some example protocol pairs are show below:



- Aggregation Policy: An aggregate route is used to summarize the component routes of an address range. By default when an aggregate route is exported to a neighbor, all of its component routes (i.e. routes that are more specific than the aggregate) are suppressed. This reduces the amount of route propagation and controls the growth of the IP routing and forwarding tables. Some component routes, however, may need to be exported along with the aggregate to satisfy some topology constraints (such as multi-homed components in BGP). Aggregation policy is used by a protocol to determine the aggregate routes that a router generates and which component routes to export along with the aggregate. BGP is an example protocol using aggregation policy.

2. Basic RPCL Types; Filter, Action and Peering Specifications

Routes are composed of a set of attributes. Some attributes are common to all routing protocols, and some are unique to a single protocol. For example, an address prefix and the preference are common to all routing protocols, whereas the tag for RIP and OSPF routes and the aspath for BGP routes are protocol specific.

Policies are configured using policy rules. All policy rules are composed of the following basic constructs:

- Filter specification: a logical expression over the route attributes. Filter specifications are used by the policy rules to select a set of routes for a particular operation. For example, the RIP import rule "accept (128/8^+ and tag==100)" uses the filter "(128/8^+ and tag==100)" to allow the import of the RIP routes that are more specific than 128/8 and have the tag value of 100.

- Action specification: an ordered list of commands to set or modify the attributes of a route. For example, a RIP export rule with the action specification "{tag=10; metric=100}" will set the tag and metric attributes of a route to 10 and 100, respectively.

- Peering specification: used by an External Gateway Protocols (EGP), such as BGP, to restrict the policy evaluation to a subset of peers. For example, a BGP import rule with a peering "{128.10.5.5, AS11}" will only be evaluated for the routes coming from the peer 128.10.5.5 or from a peer in AS11.

2.1 Route Attribute Data Types

Every route attribute has a specific data type. The route attributes of the unicast routing protocols and their corresponding data types are shown in Figure-1. The basic data types are explained below:

<ipv4-addr> An IPv4 address is a 32 bit number represented by four 8 bit integers (standard "dotted quad" format). For example 128.10.9.9 represents a valid address.

<prefix> An address prefix is composed of an IPv4 address followed by its length, an integer between 0 to 32 which represents the number of significant leftmost contiguous bits (CIDR-based addressing). Only the significant part of the IPv4 address needs to be specified. 0/0, 128.9/16, 150.10.0/16, and 169.144.100/24 are examples of valid prefixes, whereas 128.9/24 is invalid (the IPv4 address must include three integers when the mask length is 24).

<as> An autonomous system (AS) x is denoted by ASx. For example, AS312 represents the AS 312.

<as-path> AS-path is a BGP attribute that keeps track of the sequence of ASes traversed by the route. It is represented by a list of AS numbers in braces, separated by commas. For example, {AS23, AS247, AS11, AS66} is an AS-path.

<ospf-type> OSPF routes can be of type intra-area, inter-area, external-1 or external-2. They are represented by the symbols in the set {INTRA-AREA, INTER-AREA, EXTERNAL-1, EXTERNAL-2}.

<comm-list> Community list is a BGP attribute represented by a list of communities in braces, separated by commas. Each community is either a 4-byte unsigned integer, or one of the keywords NO-EXPORT or NO-ADVERTISE, or a list of two 2-byte unsigned integers. For example, {100, NO-EXPORT, {312, 10}, 200} is a community list.

<peer> A BGP peer is represented by the peer AS and the peer address. For example AS312 10.9.9.1 represents the peering with the router 10.9.9.1 in the AS 312.

2.2 Filter Types

Unicast protocol filters are shown in Figure-1. The following describes some of the arguments used in these filters:

ANY-ROUTE is a constant that always evaluates to TRUE.

<prefix-range> is a prefix followed by one of the following range operators:

^- is the exclusive more specific operator. It represents the more specifics of an address prefix excluding the prefix itself. For example, 128.9/16^- includes all of the more specifics such as 128.9.10/24, 128.9.30.48/28, and 128.9.50.99/32, but not 128.9/16.

^+ is the inclusive more specifics operator. It represents the more specifics of an address prefix including the prefix itself. For example 128.9/16^+ will match all of the more specifics and also 128.9/16 itself.

^n represents all length n specifics of an address prefix. For example, 169.144.128/17^24 will match 169.144.128/24 and 169.144.132/24 but not 169.144.132/22 since the mask is not 24, and not 169.144.96/24 since it is not a more specific (17th bit does not match).

^n-m represents any more specifics of an address prefix with the mask length between n to m. For example, 128.9.0/20^24-32, will match 128.9.10/24 and 128.9.0.22/31, but not 128.9.10/23 since the mask is not in the range of 24-32, and not 129.9.34.24/32 since it is not a more specific (19th bit does not match).

<prefix-range-list> is a list of address prefix ranges in braces separated by commas. It is a filter expression and it matches the address prefixes that match one of the prefix ranges specified in the list. For example,

{10.9.1/24}

will match the prefix 10.9.1/24, and

{10.1/16, 10.122/15^-, 11/8^+, 128.10/16^20-32}

will match
- route 10.1/16,
- more specifics of 10.122/15 excluding 10.122/15 itself, such as 10.123/16,
- more specifics of 11/8 including 11/8 itself, such as 11.144.10/24, and
- more specifics of 128.10/16 with the mask length between 20 to 32, such as 128.10.200/22.

<as-set> is a set of AS numbers in braces separated by commas, such as {AS12, AS47, AS111}.

<aspath-regex> is a regular expression on ASes, enclosed in '<' and '>'. It is used as a policy filter for the BGP AS-path attribute.

The regular expression constructs are as follows:

ASx	matches ASx
<as-set>	matches any one of the ASes in the set
.	matches any AS number
^	matches to the empty string at the begining of an AS-path
$	matches to the empty string at the end of an AS-path

Regular expression operators are as follows:

Unary postfix operators (left associative)

*	zero or more occurances
+	one or more occurances
?	zero or one occurance
(m)	m occurances
(m,)	m or more occurances
(m,n)	m to n occurances

Binary concetanation operator (lower precedence, left associative)

is an implicit operator. If A and B are regular expressions, then A B matches an AS-path if A matches some prefix of the AS-path and B matches the rest of the AS-path.

In the following, examples of <aspath-regex> expression are given along with the description of the aspaths that will match them:

<AS5>	aspaths that contain AS5
<AS5 AS7>	aspaths with the subsequence AS5 AS7
<AS5 .* AS7$>	aspaths containing AS5 and ending with AS7
<^AS5 {AS3 AS7}+ AS9$>	aspaths starting with AS5 ending with AS9 and having at least one occurances of AS3 and/or AS7 in the middle
<^. AS5 .?$>	aspaths of length 2 or 3 with AS5 as the 2nd AS
<{AS7 AS9 AS11}(2,4)$>	aspaths with the last 2 to 4 ASes being from the set {AS7 AS9 AS11}

2.3 Peering Types

The basic peering expressions are as follows:

ANY-PEER is a constant that will identify all peerings with the local router.

<peer-addr> An IPv4 address that uniquely identifies the peering between the local router and the peer router with the given address.

<peer-as> A Peer AS identifies all the peerings between the local router and any of its peer routers in the given AS.

<peer-set> is a set of Peer addresses and Peer ASes. It identifies all the peerings between the local router and any of its peer routers whose address or AS is in the given set.

Some example peering expressions are given below:

128.10.12.5 : matches the peering between the local router and its peer router 128.10.12.5.

AS17 : matches all peerings between the local router and any of its peer routers in AS17.

{128.10.12.5, 10.9.64.10}: matches two peerings of the local router, one with the peer router 128.10.12.5 and the other with 10.9.64.10.

{AS17, AS99} : matches all peerings between the local router and any of its peer routers in AS17 or AS99.

{128.10.12.5, 10.9.64.10, AS17, AS99}: matches all peerings that matches with the peer sets {128.10.12.5,

10.9.64.10} or {AS17, AS99}.

PROT	Attribute	Type	Filter	Action
STATIC	prefix	<prefix>	<prefix-range-list>	---
DIRECT	prefix	<prefix>	<prefix-range-list>	---
RIP	prefix	<prefix>	<prefix-range-list>	---
	src-gateway	<prefix>	src-gtw==<prefix-range-list>	---
	preference	<integer>	---	pref=<integer>
	metric	[1-15]	---	metric=<integer>
	tag	<integer>	tag==<integer>	tag=<integer>
OSPF	prefix	<prefix>	<prefix-range-list>	---
	preference	<integer>	---	pref=<integer>
	metric	[1-65535]	---	metric=<integer>
	tag	<integer>	tag==<integer>	tag=<integer>
	type	<ospf-type>	type==<ospf-type>	type=<ospf-type>
BGP	prefix	<prefix>	<prefix-range-list>	---
	origin	<as>	<as>	---
	source-peer	<peer>	src-peer==<peering-spec>	---
	preference	<integer>	---	pref=<integer>
	med	<integer>	---	med=<integer>
	dpa	<integer>	---	dpa=<integer>
	as-path	<as-path>	<aspath-regex>	aspath.prepend<as-path>
	community	<comm-list>	comm==<comm-list>	comm=<comm-list>
			comm.contains<comm-list>	comm.=<comm>
				comm.append<comm-list>
				comm.delete<comm-list>

Figure-1: route attributes, with their corresponding data types, filter expressions and action commands.

2.4 Filter, Action and Peering Specifications

For a protocol PR, filter, action and peering specifications are defined as follows:

<PR-filter> is a logical expression using the connectives AND, OR and NOT, and the filter expressions over the attributes of the PR routes. Filter expressions for several protocols are listed in Figure-1. More details on the syntax and semantics are given through examples in the following sections.

<PR-action> is an ordered list of action commands over the attributes of the PR routes. Action command types for several protocols are listed in Figure-1.

<PR-import-action> is an ordered list of action commands to be used at import time. The command types are a subset of the command types available in <PR-action>. In Figure-1, commands associated with the tag, metric and type attributes of RIP and OSPF are used at export time only. All other command types can be used during import time.

<PR-export-action> is an ordered list of action commands to be used at export time. The command types are a subset of the command types available in <PR-action>. In Figure-1, commands associated with the preference attribute in RIP and OSPF are used at import time only. All other command types can be used during export time.

<peering> is either ANY-PEER or of type <peer-addr> or <peer-AS> or <peer-set>.

3. Import Policy Rules

A routing protocol imports some of the routes that it receives from its neighbor by inserting them into the RIB. Import policy is configured for each protocol to determine:

- which routes should be imported and which ones should be blocked by that protocol, and
- what should be the route attributes of the routes that are imported.

Import policy is configured using import rules. The basic syntax of an import rule for a protocol PR is the following:

```
 <PR-Import-Rule>:
    [from <peering>] [action <PR-import-action>] accept <PR-filter> |
    [from <peering>] block <PR-filter>
```

where [from <peering>] is used by External Gateway Protocols (EGP), such as BGP, but not by Internal Gateway Protocols (IGP), such as RIP and OSPF.

Evaluation of an import rule for the routes received by the protocol PR is as follows:

- A route matches the import rule, if it matches the <PR-filter> specification and, in case PR is an EGP protocol, the peer it is received from matches the <peering> specification.

- A matching rule is applied by taking the action suggested by the keyword preceding the <PR-filter>, i.e. importing the route if the keyword is "accept", and rejecting it if it is "block". The attributes of an accepted route are updated by the <action> specification. It is optional and if not specified the default values are used for the attributes.

In Figure-2, some of the properties of import policies are listed for protocols RIP, OSPF and BGP. They can be summarized as follows:

- All three protocols need policy configuration for import decisions.

- Only RIP is allowed to have a block rule, i.e. a rule with the "block" keyword. OSPF and I-BGP (Internal BGP) cannot block routes, because all routers running OSPF or I-BGP should have the similar view of the network. E-BGP (External BGP), on the other hand, does not need block rules simply because block is the default action to take.

- If no rule matches a received route, then the default action is taken. It is to accept the route with default attributes if the protocol is RIP, OSPF or I-BGP, and to block it if the protocol is E-BGP.

	RIP	OSPF	I-BGP	E-BGP
Configuration	YES	YES	YES	YES
Block Rule	YES	---	---	---
Default-Action	ACCEPT	ACCEPT	ACCEPT	BLK

Figure-2: Properties of Import policies for several protocols

We now give examples of import rules for RIP, OSPF, and BGP. The following are examples of RIP import rules:

```
block {128.9/16^+} and tag==10
```

blocks all routes that are more specifics of 128.9/16 (including 128.9/16) and have the tag value of 10.

```
action pref=10 accept src-gw=={192.100.100/24^+}
```

imports all routes that are received from a gateway where the gateway address is a more specific of 192.100.100/24. It sets the preference of all such routes to 10.

Example of OSPF import rules are as follows:

```
action pref=5 accept ANY-ROUTE
```

imports all OSPF routes with a preference of 5.

```
   action pref=10
   accept {10.3/16^-, 20/8^+} and (type==INTER-AREA or tag==100)
```

imports any more specifics of 10.3/16 (excluding 10.3/16) and more specifics of 20/8 (including 20/8) that are of type INTER-AREA or have the tag value 100. All imported routes will be assigned the preference 10.

Examples of BGP Import rules are as follows:

```
   from 10.9.9.6 action pref=1 accept AS11 or {128.9/16^+}
```

imports all routes received from the peer router 10.9.9.6, that are either originated from AS11 or match 128.9/16^+. All imported routes will get the preference 1.

```
   from {10.9.9.1, 10.9.9.5, AS17, AS25}
   action med=0; community.append(3261,NO-EXPORT); aspath.append(AS5)
   accept {169.9/16^24-32} and <^AS11 AS26 .* AS7> and
          community.contains(10250)
```

imports all routes that:

- are received from the peer router 10.9.9.1 or 10.9.9.5, or from a peer router that is in AS17 or AS25, and
- are more specifics of 169.9/16 with a length between 24 to 32, and
- have an aspath that starts with AS11, AS26 and contains AS7, and
- have a community list that contains the community 10250.

All routes that are imported will have the following changes in their attributes:

- multi-exit-discriminator set to 0
- community-list appended with the community 3261 and NO-EXPORT
- aspath appended with AS5.

So far only the basic forms of import rules are described. They can also be structured. The two types of structured rules, compound rules and refine rules, are described below.

A compound rule is of the form:

```
   <PR-Import-Compound-Rule>: {<PR-Import-Rule>; <PR-Import-Rule>; ...}
```

It is an ordered list of import rules with the following semantics:

- A route matches a compound rule if it matches at least one of the rules in its list, and
- if there are matching rules, then the first such rule is applied to the route.

As an example, consider the following BGP rule:

```
{
  from 10.9.9.5 action pref=1;  accept ANY-ROUTE;
  from AS5      action pref=5;  accept AS5;
  from AS5      action pref=10; accept ANY-ROUTE;
}
```

Assume that the router 10.9.9.5 is in AS 5. The above rule imports any route coming from AS 5, but with different preferences. Routes coming from the peer router 10.9.9.5 are given the preference 1. If they are coming from another peer router in AS 5, then the ones that are originated by AS 5 are given the preference 5, and all the others are given the preference 10.

The other type of structured rule is a refine rule with the following syntax:

```
<PR-Import-Refine-Rule>: <PR-Import-Rule> refine <PR-Import-Rule>
```

It is used to refine, i.e. extend, one rule by another one. Its semantics is as follows:

-A route matches a refine rule if it matches the component rules on both sides.
-If it matches, then the rule on the left is applied followed by the rule on the right. One restriction in refine rules is that rules on both sides should be composed of either all accept rules or all block rules. Otherwise, the semantics become ambigious if an accept rule matches on one side and a block rule on the other.

Consider the following examples of BGP import rules:

```
from AS5 action pref=1; accept <^AS12>
refine
   from 10.9.9.5
   action community.append(1210);
   accept {10/8^+} and <AS39>
```

Assume that the router 10.9.9.5 is in AS 5. Then the above rule is equivalent to

```
from 10.9.9.5
action pref=1; community.append(1210);
accept {10/8^+} and <^AS12 .* AS39>
```

Notice that the two rules on both sides of the refine are combined by anding their peering and the filter specifications and appending their action specifications.

```
from ANY-PEER accept AS5;
   refine
     {
       from ANY-PEER action pref=1;  accept community.contains(1210);
       from ANY-PEER action pref=5;  accept community.contains(1410);
       from ANY-PEER action pref=10; accept ANY-ROUTE;
     }
     refine
       {
         from 10.9.9.5 action med=100; accept ANY-ROUTE;
         from AS5      action med=50;  accept ANY-ROUTE;
         from ANY-PEER action med=0;   accept ANY-ROUTE;
       }
```

imports all routes that are originated by AS5, but with different preference and med values based on the content of their community strings and the peers that they are received from, respectively. Specifically, the routes that have the community 1210 are given the preference 1, the ones that have the community 1410 are given the preference 5, and all the others are given the preference 10. Similarly, the routes that are received from the peer router 10.9.9.5 are given the med value 100, the ones that are received from another peer router in AS 5 are given the med value of 50, and the ones that are received from elsewhere are given the med value of 0.

Finally, the full syntax of an import rule for protocol PR is:

```
 <PR-Import-Rule>:
    [from <peering>] [action <PR-import-action>] accept <PR-filter> |
    [from <peering>] block <PR-filter>                               |
    {<PR-Import-Rule>; <PR-Import-Rule>; ...}                        |
    <PR-Import-Rule> refine <PR-Import-Rule>
```

4. Export Policy Rules

A routing protocol exports (i.e. advertizes) some of the routes in the RIB to its neighbors. The exported route might belong to the exporting protocol or to another protocol. Export policy is configured from a protocol SRC_PR (the owner protocol) to a protocol DST_PR (the exporting protocol) to determine:

-which SRC_PR routes should be exported by DST_PR protocol to its neighbors, which ones should be blocked, and
-with what attributes should the routes be exported. Notice that the SRC_PR routes will be exported as DST_PR routes, therefore they should be exported with DST_PR attributes.

Export policy is configured using export rules. The syntax of an export rule from a protocol SRC_PR to a protocol DST_PR is as follows:

```
<SRC_PR-to-DST_PR-export-rule>:
  [to <peering>] [action <DST_PR-export-action>] announce <SRC_PR-filter> |
  [to <peering>] block <SRC_PR-filter>
```

where [to <peering>] is used by External Gateway Protocols, such as BGP, but not by Internal Gateway Protocols, such as RIP and OSPF.

The evaluation of an export policy rule is similar to the import rules. For a rule from protocol SRC_PR to protocol DST_PR, the rule evaluation is as follows:

-A SRC_PR route matches the export rule, if it matches the <SRC_PR-filter> and, in case DST_PR is an EGP protocol, the peer that the route is sent to matches the <peering> specification.

-Matching routes are exported if the <SRC_PR-filter> is preceded by the keyword "announce", and they are blocked otherwise. An exported route will be a DST_PR route, and some its attributes will be set by the optional <DST_PR-export-action>, and the others will take its default values.

SRC PROTOCOL

DST PROT	RIP	OSPF	BGP
Static	YES	YES	YES
Direct	YES	YES	YES
RIP	YES	YES	YES
OSPF	YES	---	YES
BGP	YES	YES	YES
Block Rule	YES	---	---

(a) Configuration and Block Rule

SRC PROTOCOL

DST PROT	RIP	OSPF	BGP
Static	BLK	BLK	BLK
Direct	BLK	BLK	BLK
RIP	ACCEPT	BLK	BLK
OSPF	BLK	ACCEPT	BLK
I-BGP	BLK	BLK	ACCEPT
E-BGP	BLK	BLK	BLK

(b) Default-Action

Figure-3: Properties of Export policies for several protocols

In Figure-3, some properties of export policies are listed for some of the protocol pairs. They can be summarized as follows:

-Export policy need not be configured for every protocol pair. For example, OSPF has to export all OSPF internal routes to all of its neighbors without any attribute change. Therefore, there is no policy configuration needed for an export from OSPF to OSPF.

-Only RIP is allowed to have a block rule. The other protocols do not have it either because they cannot block any route from export, or because their default actions are block and they don't need it.

-The default action is taken in case no rule matches the route. It is to export the route with default attributes, in case RIP, OSPF or I-BGP is exporting its own route, and it is to block for all other cases.

Following are examples of export policy rules:

The following rule from RIP to RIP

```
action tag=100; metric=1210
announce tgt-gw=={120.100/16^+} and src-gw=={196.122.44/24^+}
```

is used by RIP to export all RIP routes received from a gateway that matches 196.122.44/24^+, to a gateway that matches 120.100/16^+. All such routes are exported with the tag value of 100 and the metric value of 1210.

The rule from BGP to RIP

```
action tag=200;
announce src-peer==AS11 or AS11
```

is used by RIP to export BGP routes that are either originated by AS 11 or received from a peer in AS 11. All such routes are exported with the tag value of 200.

The rule from BGP to OSPF

```
action tag=100; type=EXTERNAL-2;
announce community.contains(3210)
```

is used by OSPF to export BGP routes that has the community 3210, with the tag value of 100 and the OSPF type EXTERNAL 2.

The rule from RIP to BGP

```
to AS11
action med=10; aspath.append(AS9);
announce {120.100/16^+} and src-gw=={196.122.44.12}
```

is used by BGP to export RIP routes to peers in AS 11, that are received

from the gateway 196.122.44.12 and match 120.100/16^+. All such routes are exported with a med value of 10 and with AS9 appended to their aspath.

The rule from BGP to BGP

```
to {12.10.10.3, 12.10.10.7, AS11}
action community.append(NO-EXPORT)
announce <^AS19 .* AS54>
```

is used by BGP to export BGP routes that have an aspath starting with AS 19 and including AS 54, to peers 12.10.10.3, 12.10.10.7 or the ones in AS 11. Routes are exported with NO-EXPORT community appended to their community string.

Export rules can be compounded and refined just like import rules. The full syntax for an export rule is the following:

```
<SRC_PR-to-DST_PR-export-rule>:
   [to <peering>] [action <DST_PR-export-action>] announce <SRC_PR-filter> |
   [to <peering>] block <SRC_PR-filter>                                     |
   {<SRC_PR-to-DST_PR-Export-Rule>; <SRC_PR-to-DST_PR-Export-Rule>; ...}    |
   <SRC_PR-to-DST_PR-Export-Rule> refine <SRC_PR-to-DST_PR-Export-Rule>
```

5. Aggregation Policy Rules

An aggregate route is used to summarize the component routes of an address range. By default when an aggregate route is exported to a neighbor, all of its component routes (i.e. routes that are more specific than the aggregate) are suppressed. This reduces the amount of route propagation and controls the growth of the IP routing and forwarding tables. Some component routes, however, may need to be exported along with the aggregate to satisfy some topology constraints (such as multi-homed components in BGP). Aggregation policy is used by a protocol to determine the aggregate routes that a router generates and which component routes to export along with the aggregate.

Some protocols, such as BGP, perform aggregation via a configured policy. Based on the current state of the RIB, the configured aggregation rules of a protocol PR determine:

- what aggregates to generate and the route attributes of those aggregates (aggregate activation evaluation)
- in case PR exports an aggregate route to a neighbor, what components (i.e. more specifics) to export along with the aggregate and what components to suppress (aggregate export evaluation)

The syntax of an aggregation rule for a protocol PR is as follows:

```
<PR-aggregation-rule>:
   aggregate into     <prefix>
   triggered by       AND/OR of HC{<prefix-range-list>} and
                                EX{<prefix-range-list>}
   components         ATOMIC | [protocol <PR_1>] <PR_1-filter>
                              [[protocol <PR_2>] <PR_2-filter>] ...
   action             <PR-action>
   export components  from <PR_1> <PR_1-to-PR-export-rule>
                     [from <PR_2> <PR_2-to-PR-export-rule>] ...
   holes              <prefix-range-list>
```

The semantics of an aggregation rule for a protocol PR is best described by explaining each of its attributes as follows:

aggregate: specifies the prefix of the aggregate for which the rule is defined.

trigger: determines, along with the components attribute, the conditions needed to generate the aggregate. The trigger requires certain prefix ranges to be present and certain others to be absent in the RIB as a precondition for generation.

The trigger is specified by a logical expression using the connectives AND and OR, and the constructs HC (Have-Components) and EX (EXclude) defined as follows:

HC{<prefix-range-list>}: true iff there is a matching route in the RIB for every prefix-range in the list.

EX{<prefix-range-list>}: true iff there is no matching route in the RIB for any of the prefix-ranges in the list.

The aggregate route is generated and put into the RIB when the trigger condition becomes true, and removed from the RIB when it becomes false. For example, the rule

```
aggregate into 10/8
triggered by   HC{10.1/16^24-32, 10.2.2/24^+} AND EX{11/8^+}
```

specifies that the aggregate 10/8 will be added to the RIB only if there are routes matching 10.1/16^24-32 and 10.2.2/24^+, and no routes matching 11/8^+ in the RIB.

components: determines the route attributes of the aggregate, if it is generated. This attribute has the following syntax:

```
components ATOMIC | [protocol <PR_1>] <PR_1-filter>
                  [[protocol <PR_2>] <PR_2-filter>] ...
```

The components attribute is either specified by the keyword ATOMIC or by a list of filters and their corresponding protocols. In the former case attributes of the aggregate are assigned to their default values, and in the latter they are computed from a subset of the component routes, called attribute-components.

Attribute-components consist of the more specifics of the aggregate that match one of the filters and are learned from the corresponding protocol. If the protocol is omitted in front of a filter then it defaults to any protocol.

The properties of the attribute computation are as follows:

- the computation might result in a conflict of some of the component attributes, in which case the aggregate will not be generated. An example conflict is the case where the routes in attribute-components do not have the same next-hop.
- the attributes of the aggregate have to be re-computed every time the state of the attribute-components change (i.e. a new component passes the components filter), when a component no longer passes the components filter, or when a component still passes the components filter but had a change in its route attributes.
- if no trigger attribute is specified, then the components attribute is used to decide whether to generate the aggregate or not. It will be generated only if the attribute-components is not empty (i.e. at least one of the more specifics in the RIB matches the components filter).

For example,

```
aggregate into 128.64/10
components
   protocol BGP  {128/8^+}
   protocol OSPF {128.70/15^16-24} and type==INTER-AREA
```

uses BGP routes that are more specifics of 128.64/10 and inter-area OSPF routes that match 128.70/15^16-24 for computing the route attributes of the aggregate 128.64/10. Notice that since the more specifics are exclusively used in the computation, all filters are implicitly ANDed with {128.64/10^-}.

action: a list of PR actions applied each time the route attributes of the aggregate are (re-)evaluated. The action attribute is used to modify some of the attributes that are either set to their default values (i.e. ATOMIC) or computed from the more specifics.

For example,

```
action  {comm.delete(NO-EXPORT); med=100}
```

if present, removes the NO-EXPORT community from the community list of the aggregate and set its med to 100.

export components: allows some of the more specific components of an aggregate to be exported. In general, when a set of routes are aggregated, the intent is to export only the aggregate and suppress all the more specifics. However, to satisfy some topology constraints such as a multi-homed component, this attribute allows some of the components to be exported. Using the export components attribute a list of export policy rules from a number of protocols to the protocol PR (the aggregator protocol) is specified. The export components attribute is of the form:

```
export components  from <PR_1> <PR_1-to-PR-export-rule>
                   [from <PR_2> <PR_2-to-PR-export-rule>] ...
```

The following specifies the evaluation of an aggregate and its more specifics for export:

- Every time the aggregate is generated or has a change in its attributes:
  - it is (re)evaluated for export to all the neighbors.
  - to the neighbors that the aggregate was exported before but blocked now, the export of more specifics are evaluated using the configured export policy.
  - to the neighbors that the aggregate was not exported before but exported now, the export of more specifics are evaluated using the export components of the corresponding aggregate rule.

- When the aggregate is removed from the RIB (either because the trigger became false or the component attributes conflict):
  - to the neighbors that the aggregate was exported before, the export of more specifics are evaluated using the configured export policy.

The following are some examples of the export components attribute:

```
aggregate into     10/8
export components  from ANY-PROTOCOL announce {10.1/16^-}
```

specifies that when the aggregate 10/8 is exported to a neighbor, all the more specifics are suppressed from export to that neighbor except the ones that are also more specifics of 10.1/16.

```
aggregate into 128.100/16
export components
   from OSPF
      to {AS17, AS126} action dpa=50 announce ANY-ROUTE
```

is a BGP aggregate rule where OSPF more specifics of 128.100/16 are exported to the peers in AS17 or AS126 along with the aggregate, and in all other cases the more specifics are suppressed.

holes: a prefix range list that specifies the holes in the aggregate, i.e. the more specific address ranges not reachable through the aggregate route (perhaps they are not allocated). Holes attribute has no functional role, it is for diagnostic purposes only.

Here is an example BGP aggregation rule that uses all the rule attributes:

```
aggregate into     10/8
triggered by       HC{10.1/16^24-32, 10.2.2/24^+} AND EX{11/8^+}
components         protocol RIP  {10.3/16^+} AND
                                 src-gw=={50.10/16^+}
                   protocol BGP  AS5
                   protocol OSPF ANY-ROUTE
action             {med=100; community.delete(NO-EXPORT);}
export components  from OSPF
                      to {AS10, 128.10.9.9} announce ANY-ROUTE
                   from BGP
                      to {AS10} announce AS5
holes              {10.100.100/24^+}
```

In case of overlapping aggregation rules, where the aggregate of one rule is a lesser specific of another one, the semantics of rule evaluation is extended as follows:

- activation evaluation: is performed starting from the rule for the most specific aggregate to the rule for the least specific one. This ensures that, the activation evaluation for an aggregate will consider the most current activation state of the more specific aggregates (i.e. whether they are present in the RIB or not and their route attributes if they are in the RIB).

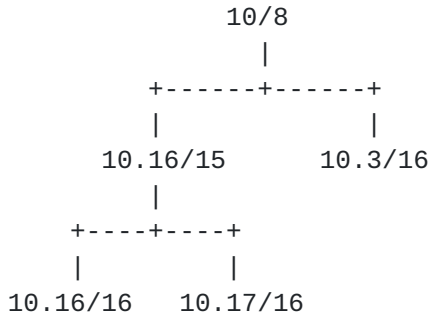
- export evaluation: a route is exported to a neighbor iff:
  - it is the least specific aggregate in the RIB that passes the export policy, or
  - it passes the export components attribute of an aggregation rule whose aggregate is exported, or
  - there are no aggregates of this route that passes the export policy, but the route itself passes it.

  Notice that the above conditions describe chains of routes, A1,A2,...,An,Rn+1, where:
  - all Ai are active aggregates, and
  - A1 is the least specific aggregate in the RIB that passes the export policy, and
  - each Ai is a more specific of Ai-1 and Ai passes the export components of Aj for some j<i, and
  - Rn+1 is a non-aggregate route which passes the export components of Aj for some j<n+1.

Consider the following example of overlapping aggregate rules for BGP:

```
aggregate into 10/8
triggered by
   HC{10.3/16, 10.16/15, 10.72.3/24}
export components
   from BGP to {AS2} announce {10.3/16}
   from BGP to {AS1, AS2} announce {10.16/12^12-15}

aggregate into 10.16/15
triggered by
   HC{10.16/16}
export components
   from BGP to {AS1, AS2} announce {10.16/15^16}

aggregate into 10.16/16
triggered by
   HC{10.16.0/18^+, 10.16.128/18^+}
export components
   from RIP to ANY-PEER announce {10.16.100/24^+}

aggregate into 10.17/16
triggered by
   HC{10.17/16^-}
export components
   from OSPF to {AS2} announce {10.17/16^24}

aggregate into 10.3/16
triggered by
   HC{10.3/16^-}
```

The aggregate rules form the following tree:



Activation evaluation for the aggregates proceed bottom up (i.e., from the leaf nodes to the root node). For example, consider the activation evaluation of 10/8 (assuming no conflicts in route attributes):

- 10/8 will be generated only if 10.3/16, 10.16/15, and 10.72.3/24 are all in the RIB. Notice that 10.3/16 and 10.16/15 can themselves be active aggregates or non-aggregate routes exported from a neighbor.
- if 10.3/16 is an active aggregate then the generation of 10/8 also depends on the existance of a route in the RIB that matches 10.3/16^-.
- similarly, if 10.16/15 is an active aggregate then the generation of 10/8 depends on the existance of a route in the RIB that matches 10.16.0/18^+ and another one that matches 10.16.128/18^+.

Export evaluation, on the other hand, is performed top down (i.e. from the least specific active aggregate to the most specific). For example, in the above configured policy, the following routes will be exported to AS1 (assuming that all aggregates are active and 10/8 passes the export policy for AS1):

- 10/8 is the least specific aggregate that passes the export policy, so it will be exported.
- 10.16/15 will be exported, since it passes the export components of the rule for 10/8
- Both 10.16/16 and 10.17/16 will be exported, since they pass the export components of the rule for 10.16/15.
- Any RIP route that matches 10.16.100/24^+ will also be exported, since it matches the export components of the rule for 10.16/16.

6. Acknowledgments

We would like to thank Rob Coltun and Sanjay Wadhwa for their contributions, valuable comments and suggestions.

7. Authors' Addresses

Ardas Cilingiroglu
FORE Systems
1595 Spring Hill rd. 5th floor,
Vienna, VA 22182
acilingi@fore.com

Tony Przygienda
Bell Labs, Lucent Technologies
101 Crawfords Corner rd.
Holmdel, NJ 07733-3030
prz@dnrc.bell-labs.com

8. References

[1] C. Alaettinoglu, T. Bates, E. Gerich, D. Karrenberg, D. Meyer, M. Terpstra, and C. Villamizar. "Routing Policy Specification Language (RPSL)". Request for Comment RFC-2280, January 1998.

[2] Alaettinouglu, C., Meyer, D., and J. Schmitz. "Application of Routing Policy Specification Language (RPSL) on the Internet", Work in Progress.

[3] Internet Routing Registry. Procedures. http://www.ra.net/RADB.tools.docs/, http://www.ripe.net/db/doc.html.

[4] A. M. R. Magee. "RIPE NCC Database Documentation". Technical Report RIPE-157, RIPE, RIPE NCC, Amsterdam, Netherlands, May 1997.

APPENDIX A. A Policy Editor for RPCL

Previous sections have described the syntax and semantics of a language to specify routing policies. This section describes an editor to to be used for configuring RPCL policies on a router.

On a physical router device, there can be multiple IP instances, where each instance identifies the implementation of a logical router with its own RIB running its own instances of routing protocols and policies. The command

(1) set ip-instance <ip-inst-id>

is used to set the current ip instance on a physical router device. The commands following (1) are used to configure the protocols and policies of the logical router specified by the given IP instance.

Each IP instance stores its routing policies in a database, where each policy of a protocol is configured on a separate global policy table. The names and contents of these tables are listed below:

IMPORT-<PR>: defines the ordered list of import rules used by the protocol <PR> to decide which of the received routes to import and with what route attributes. This list is of the form:

```
IMPORT-<PR> :
     {<PR-Import-Rule>; <PR-Import-Rule>; ... }
```

EXPORT-<SRC_PR>-<DST_PR>: defines the ordered list of export rules used by the protocol <DST_PR> to decide which <SRC_PR> routes to export to a neighbor and with what route attributes. This list is of the form:

```
EXPORT-<SRC_PR>-<DST_PR> :
     {<SRC_PR-to-DST_PR-Export-Rule>;
      <SRC_PR-to-DST_PR-Export-Rule>; ... }
```

AGGREGATE-<PR>: defines the unordered set of aggregation rules used by the protocol <PR> to decide what <PR> aggregates to generate and which more specifics of these aggregates to suppress from export. This set is of the form:

```
AGGREGATE-<PR>:
    {<PR-Aggregation-Rule>; <PR-Aggregation-Rule>; ... }
```

Rules in an import or export table are ordered. When a protocol evaluates its import or export policy for a route, the decision is taken by applying the first matching rule in the corresponding global policy table. If no rule in the table matches, then the appropriate default action is taken. Default actions depend on the policy and the protocols involved, and they are shown in Figures 2 and 3.

Rules in an aggregation policy table, on the other hand, are not ordered. Every time the state of the RIB changes (i.e. some route R gets added, removed or changed), all aggregation rules that define an aggregate for R are evaluated in the order of the most specific aggregate to the least specific one.

Initially, when the router is started, all global tables are empty. This means that the default actions will be taken for all import and export decisions, and no aggregate will be generated. Later, policies are configured (and re-configured) by changing the contents of their corresponding global tables.

Parts of global tables that are repeatedly used can be defined as a macro. The macros are explained in section A.1. Configuration commands to insert, remove and reorder policy rules in a global policy table are given in sections A.2 and A.3.

Changing the contents of the policy database is an expensive operation. All current policy decisions that might get affected by the policy changes have to be re-evaluated by the protocols. Therefore, changes in the database (as requested by the configuration commands that are issued) are not committed after each configuration command but delayed until an explicit commit command is specified. The details of commiting is explained in A.4.

A.1 Policy Macros

Some policy pieces that are used repeatedly during policy configuration can be defined as a macro. The generic syntax of a macro is:

```
define <macro-name> <object>
```

All macro references, <macro-name>, in the configuration are replaced by its object definition, <object>.

Naming of macros are restricted based on the type of the policy piece they define. Each macro name is made up of two parts separated by the character hyphen "-". The first part is a constant string specifying the type of the policy piece, and the second part is a free string of 8 characters assigning a unique name to the macro. For example "peer-internal" is a valid macro name where "peer" specifies that it defines a peer set and "internal" results in a unique name for the macro.

Some macros define attributes of a policy rule. Their syntax are as follows:

```
(2) define peer-<str8>  <peer-set>

(3) define fltr-<str8>  <filter>

(4) define act-<str8>   <action>
```

The following is an example of their use:

```
define peer-nghbors {as17, 128,166.9.9, as57}

define fltr-for128  {128.176/20^+, 128.0/9^20-24} and
                    (as12 or <as12>)

define act-for128   {pref=10; aspath.prepend(as99)}
```

Using the above macros one can define the following BGP import rule:

```
from peer-nghbors
action act-for128
accept (fltr-for128 and {as217, as61, as126})
```

which expands to

```
from {as17, 128,166.9.9, as57}
action {pref=10; aspath.prepend(as99)}
accept ({128.176/20^+, 128.0/9^20-24} and
        (as12 or <as12>) and
        {as217, as61, as126})
```

Macros may also be used for defining list of policy rules. The following is the syntax for import, export and aggregation rule lists, respectively:

```
(5) define imp-<str8>
       protocol <PR>
       {<PR-Import-Rule>; <PR-Import-Rule>; ... }

(6) define exp-<str8>
       protocol <SRC_PR> into <DST_PR>
       {<SRC_PR-to-DST_PR-Export-Rule>;
        <SRC_PR-to-DST_PR-Export-Rule>; ... }

(7) define aggr-<str8>
       protocol <PR>
       {<PR-Aggregation-Rule>; <PR-Aggregation-Rule>; ... }
```

As explained before, rules in an import or export policy list are ordered. When defining a macro, this ordering is explicitly specified by listing the rules. However, macros can also be modified by inserting/removing rules to/from them, in which case the ordering has to be specified implicitly. An integer, called a rule number, is assigned to individual rules of a macro to impose this ordering. A rule with a smaller rule number comes before a rule with a larger one. Rule numbers are also used to refer to the rules of an import or export policy macro. The syntax for rule referrals can take the following forms:

<macro-name>: refers to all the rules of that macro.

<macro-name> <rule-num>: refers to the rule in <macro-name> with the rule number <rule-num>.

<macro-name> <rule-num-1>-<rule-num-2>: refers to all the rules in <macro-name> with the rule number in between <rule-num-1> and <rule-num-2>, both inclusive.

For example, "imp-mypol 5-7" refers to the rules in imp-mypol with the rule numbers 5, 6 or 7. In rule referrals, the character "$" denotes the maximum rule number used in a macro. For example "exp-pol5 $" refers to the last rule in exp-pol5.

Initially, when an import or export policy macro is defined, rules are assigned consecutive rule numbers starting from 1. Later the contents and the rule numbers of macros can be modified using the following commands:

(8) insert imp-<str8>
{<PR-Import-Rule>; <PR-Import-Rule>; ... }
[<rule-num>]

appends the specified import rules after imp-<str8> [<rule-num>]. If <rule-num> is not specified it defaults to $, i.e. rules are appended to the end. The inserted rules are assigned consecutive rule numbers starting from the <rule-num> plus one. If any of the rule numbers are already assigned to another rule, then the insertion is aborted. In such cases, the move command has to be used first to open up the space for insertion.

(9) remove imp-<str8>
<rule-num>[-<rule-num>]

removes the rules that match imp-<str8> <rule-num>[-<rule-num>] from the macro imp-<str8>.

(10) move imp-<str8>
<rule-num>[-<rule-num>] up | down <num>

assigns new rule numbers to the rules that match imp-<str8> <rule-num>[-<rule-num>]. If "up" is specified in the command, the rules are assigned their current rule numbers plus <num>. Similarly, if "down" is specified, they are assigned their current rule numbers minus <num>. If any of the new rule numbers are already assigned to another rule, then the move operation is aborted. Notice that move does not change the relative ordering of the rules in a macro. It is used to open up the rule number space for inserting new rules.

(11) compact imp-<str8>

is useful when rule numbers are too sparse in a macro. Without changing the current rule ordering, they assign new rule numbers to the rules incrementally starting from 1.

The commands that modify export macros have the same semantics as that of import macros. They are listed below:

```
(12) insert exp-<str8>
       {<SRC_PR-to-DST_PR-Export-Rule>;
        <SRC_PR-to-DST_PR-Export-Rule>; ... }
       [<rule-num>]

(13) remove exp-<str8>
       <rule-num>[-<rule-num>]

(14) move exp-<str8>
       <rule-num>[-<rule-num>]  up | down  <num>

(15) compact exp-<str8>
```

An example import/export macro definition and modification is given below. Text following the character "#" in a line are considered comments and they are used to explain the affects of the example commands.

```
 # Rules are not specified explicitly, but represented by identifiers
 # such as R1, R2 etc. Numbers shown before each rule denotes its rule
 # number. The content of the macro is shown after each configuration
 # command as if they are committed right away.

define imp-as57pol  protocol bgp  {R1; R2; R3; R4}

 #   Initially the content of imp-as57pol is
 #
 #     imp-as57pol = {
 #                     (1) R1; (2) R2; (3) R3; (4) R4
 #                   }
```

```
insert imp-as57pol  {Y1; Y2}  2

 #     rejected because rule numbers 3 and 4 are in use

move imp-as57pol  3-$  down 3

 #     imp-as57pol = {
 #                     (1) R1; (2) R2; (6) R3; (7) R4
 #                   }

insert imp-as57pol  {Y1; Y2}  2

 #     imp-as57pol = {
 #                     (1) R1; (2) R2; (3) Y1; (4) Y2; (6) R3; (7) R4
 #                   }

remove imp-as57pol  4-6

 #     imp-as57pol = {
 #                     (1) R1; (2) R2; (3) Y1; (7) R4
 #                   }

compact imp-as57pol

 #     imp-as57pol = {
 #                     (1) R1; (2) R2; (3) Y1; (4) R4
 #                   }
```

Since aggregation policy rules are not ordered, there is no rule number assigned to the individual rules. Instead, rules can be referred by the aggregation prefix that it defines. The syntax for an aggregation policy macro definition is already given in (7). The commands to modify an aggregation macro are described below:

(16) insert aggr-<str8>
{<PR-Aggregation-Rule>; <PR-Aggregation-Rule>; ... }

adds the specified aggregation rules to aggr-<str8>.

(17) remove aggr-<str8>
<prefix-range-list>

removes the aggregation rules from aggr-<str8> that define an aggregate matching <prefix-range-list>.

An example aggregation macro definition and modification is given below.

```
 # For brevity, the rules are not completely specified. The content of
 # the macro is shown after some of the commands as if changes are
 # committed right away.

define aggr-net10
   protocol bgp
   {
     aggregate 10/8  ...;
     aggregate 10.9.9/24  ...;
     aggregate 10.10.9/24  ...;
   }

insert aggr-net10
   {
     aggregate 10.10/16  ...;
     aggregate 10.10.9.128/25  ...;
     aggregate 10.20/16  ...;
   }

 #       aggr-net10 = {
 #                       aggregate 10/8  ...;
 #                       aggregate 10.10/16  ...;
 #                       aggregate 10.20/16  ...;
 #                       aggregate 10.9.9/24  ...;
 #                       aggregate 10.10.9/24  ...;
 #                       aggregate 10.10.9.128/25  ...;
 #                     }

remove aggr-net10  {10.0/12^16-24}

 #       aggr-net10 = {
 #                       aggregate 10/8  ...;
 #                       aggregate 10.10.9.128/25  ...;
 #                       aggregate 10.20/16  ...;
 #                     }
```

A.2 Import and Export Policy Configuration

Import policy for a protocol <PR> is configured in the global table IMPORT-<PR>. Similarly, export policy from a protocol <SRC_PR> to a protocol <DST_PR> is configured in the global table EXPORT-<SRC_PR>-<DST_PR>. These global tables are made up of policy macros, where the ordering of the rules is specified hierarchically as follows:

- the ordering within the rules of a member macro is determined by the rule numbers
- the ordering of the member macros is determined by unique integers assigned to individual macros, named macro numbers. In the global table,

the rules of a member macro with a smaller macro number comes before the rules of a member macro with a larger one.

Assigning unique macro numbers to member macros allow another way to refer to them, in addition to using macro names. The syntax is as follows:

import-<PR> <macro-num>: refers to the import macro for protocol <PR> that is inserted into import-<PR> with the macro number <macro-num>.

export-<SRC_PR>-<DST_PR> <macro-num>: refers to the export macro for the protocol pair (<SRC_PR>, <DST_PR>) that is inserted into export-<SRC_PR>-<DST_PR> with the macro number <macro-num>.

A member macro in a global table can be assigned a new macro number in order to reorder its rules relative to the rest of the global table. When this happens, referring to the macro from the global table changes, using the new macro number.

Not all macros that are used in a global table have to be defined previously. Some macros are defined on the fly by listing their rules and inserting them into a global table with a unique macro number. Such macros are called on-the-fly macros and they do not have macro names. They are temporary in the sense that when such a macro is removed from the global table, its definition is lost. To prevent that, an on-the-fly macro can be changed into a permenant one by assigning a macro name to it.

Notice that global tables for import and export policies are tightly coupled with their member macros. Any change in a macro changes the global policy tables that it is a member of, which in turn changes the corresponding protocol policies. To decouple a global table from one of its member macros, the member macro can be replaced by an exact copy of it and can be given a new macro name or can be made an on-the-fly macro. This operation does not change the rules that are in the global table, but prevents future changes in the decoupled macro to be reflected to the global table and vice versa.

Import and Export policies are configured using the following commands:

(18) insert-macro import-<PR>
imp-<str8> <macro-num>

inserts the macro imp-<str8> into the table import-<PR> with a unique macro number <macro-num>. If the protocol in the imp-<str8> definition does not match with <PR>, or the <macro-num> is already assigned to another macro, the insertion operation is rejected. Notice that after this operation, imp-<str8> can also be referred as import-<PR> <macro-num>.

(19) insert-macro import-<PR>
{<PR-Import-Rule>; ...} <macro-num>

inserts an on-the-fly macro into the table import-<PR> with a unique macro number <macro-num>. The rules of the macro are specified explicitly in the command. Like (18), the operation is rejected if <macro-num> is already in use. Notice that the inserted on-the-fly macro can only be referred as import-<PR> <macro-num>.

(20) remove-macro import-<PR>
<macro-num>

removes the macro referred by import-<PR> <macro-num> from the table import-<PR>. If the removed macro is defined on-the-fly, then its definition is lost after this operation. To prevent this, link operation can be used as described below.

(21) move-macro import-<PR>
<src-macro-num> to <dst-macro-num>

relocates the macro referred by import-<PR> <src-macro-num> by assigning a new macro number, <dst-macro-num>, to it. This changes the ordering of the rules in this macro relative to the rest of the rules in the table import-<PR>. It also changes the way this macro is referred, from import-<PR> <src-macro-num> to import-<PR> <dst-macro-num>.

(22) insert import-<PR>
{<PR-Import-Rule>; ...} <macro-num>[.<rule-num>]
(23) remove import-<PR>
<macro-num>.<rule-num>[-<rule-num>]
(24) move import-<PR>
<macro-num>.<rule-num>[-<rule-num>] up | down <num>
(25) compact import-<PR> <macro-num>

commands (22)-(25) are used to change the contents of import macros. Their semantics are the same as the commands (8)-(11), except that here the macros are referred by their macro numbers instead of their macro names.

(26) link import-<PR> <macro-num> to imp-<str8>

replaces the macro that is inserted in import-<PR> <macro-num> with a fresh copy of it, and assigns the name imp-<str8> to the new copy. If the replaced macro was already assigned a name (i.e. was not an on-the-fly macro), then this operation decouples the global table import-<PR> from the replaced macro, meaning no

changes in one will affect the other. If the replaced macro was an on-the-fly macro, then the operation makes it permanent, i.e. the macro definition will not be lost after it is removed from the global table import-<PR>.

(27) unlink import-<PR> <macro-num> from imp-<str8>

replaces the macro that is inserted in import-<PR> <macro-num> with a fresh on-the-fly copy of it. It decouples the import-<PR> from the replaced macro. Notice that since the new copy is an on-the-fly macro, its definition will be lost after it is removed from the global table import-<PR>.

The commands to configure export policies have the same semantics as that of import policies. They are listed below:

(28) insert-macro export-<SRC_PR>-<DST_PR>
exp-<str8> <macro-num>

(29) insert-macro export-<SRC_PR>-<DST_PR>
{<SRC_PR-to-DST_PR-Export-Rule>; ...} <macro-num>

(30) remove-macro export-<SRC_PR>-<DST_PR>
<macro-num>

(31) move-macro export-<SRC_PR>-<DST_PR>
<src-macro-num> to <dst-macro-num>

(32) insert export-<SRC_PR>-<DST_PR>
{<SRC_PR-to-DST_PR-Export-Rule>; ...} <macro-num>[.<rule-num>]

(33) remove export-<SRC_PR>-<DST_PR>
<macro-num>.<rule-num>[-<rule-num>]

(34) move export-<SRC_PR>-<DST_PR>
<macro-num>.<rule-num>[-<rule-num>] up | down <num>

(35) compact export-<SRC_PR>-<DST_PR> <macro-num>

(36) link export-<SRC_PR>-<DST_PR> <macro-num> to exp-<str8>

(37) unlink export-<SRC_PR>-<DST_PR> <macro-num> from exp-<str8>

An example is shown below for an import policy configuration of BGP:

```
       # For brevity, the rules are not explicitly specified but represented
       # by identifiers such as R1, R2 etc.
       # The example assumes that initially the IMPORT-BGP is empty.
       # The contents of the global table and some of the macros are shown
after
       # some of the commands as if the changes are committed right away.

      define imp-rippol  protocol rip  {M1; M2}

      define imp-bgppol  protocol bgp  {X1; X2; X3}

      insert-macro import-bgp  imp-rippol  100

       #     rejected because of the mismatch of protocols

      insert-macro import-bgp  imp-bgppol  100

      insert-macro import-bgp  {Y1; Y2}  100

       #     rejected because macro is 100 is already in use

      insert import-bgp  {Y1; Y2} 100

       #     IMPORT-BGP =                imp-bgppol =
       #        {                           {
       #          (100 = imp-bgppol)          (1) X1;
       #              (1) X1;                 (2) X2;
       #              (2) X2;                 (3) X3;
       #              (3) X3;                 (4) Y1;
       #              (4) Y1;                 (5) Y2;
       #              (5) Y2;                }
       #        }

      move import-bgp  3-$  down 1

      insert import-bgp  {Z1}  2

      insert-macro import-bgp  {R1; R2}  200

       #     IMPORT-BGP =                imp-bgppol =
       #        {                           {
       #          (100 = imp-bgppol)          (1) X1;
       #              (1) X1;                 (2) X2;
       #              (2) X2;                 (3) Z1;
       #              (3) Z1;                 (4) X3;
       #              (4) X3;                 (5) Y1;
       #              (5) Y1;                 (6) Y2;
       #              (6) Y2;                }
       #
```

```
 #          (200 = ON-THE-FLY)
 #              (1) R1;
 #              (2) R2;
 #        }

move-macro import-bgp  100 to 500

unlink import-bgp  500 from imp-bgppol

link import-bgp  200 to imp-newrules

 #     IMPORT-BGP =                imp-bgppol =      imp-newrules =
 #        {                           {                 {
 #          (200 = imp-newrules)        (1) X1;           (1) R1;
 #              (1) R1;                 (2) X2;           (2) R2;
 #              (2) R2;                 (3) Z1;         }
 #                                      (4) X3;
 #          (500 = ON-THE-FLY)          (5) Y1;
 #              (1) X1;                 (6) Y2;
 #              (2) X2;               }
 #              (3) Z1;
 #              (4) X3;
 #              (5) Y1;
 #              (6) Y2;
 #        }

remove import-bgp  500  3-6

remove imp-bgppol  1-3

remove-macro import-bgp  200

 #     IMPORT-BGP =                imp-bgppol =      imp-newrules =
 #        {                           {                 {
 #          (500 = ON-THE-FLY)          (4) X3;           (1) R1;
 #              (1) X1;                 (5) Y1;           (2) R2;
 #              (2) X2;                 (6) Y2;         }
 #        }                           }
```

A.3 Aggregation Policy Configuration

The aggregation policy for a protocol <PR> is configured in the global table AGGREGATE-<PR>. Aggregation tables are different than the import and export tables in the following ways:

- Rules in a global aggregation table are not ordered.
- There is no rule number assigned to individual rules. Instead, the rules are referred by the aggregate prefix that they define.

- Global aggregation tables are not coupled with aggregation macros. When a macro is inserted into a global aggregation table, the current contents of the macro is copied into the global table. Future changes made in the macro are not reflected into the global table.

The following are the commands to configure an aggregation policy:

(38) insert aggregate-<PR>
{<PR-Aggregation-Rule>; <PR-Aggregation-Rule>; ... }

adds the specified aggregation rules to the global table aggregate-<PR>.

(39) insert aggregate-<PR>
aggr-<str8>

creates a copy of the rules in aggr-<str8> and adds them to the global table aggregate-<PR>. Note that, the global table and the macro each have their own copies of the rules, and therefore there is no coupling between them.

(40) remove aggregate-<PR>
<prefix-range-list>

removes all the rules from the table aggregate-<PR> that define an aggregate matching <prefix-range-list>.

(41) remove aggregate-<PR>
aggr-<str8>

removes all the rules from the table aggregate-<PR> that was inserted from the macro aggr-<str8>.

A.4 Commit Operation

Any run-time policy change of a protocol results in an expensive series of operations. All current policy decisions that might get affected by the policy change have to be re-evaluated by the protocol. Therefore, changing the global policy tables after each command is very inefficient. Instead, all changes are committed simultaneously at well-defined points in time. The following command is used for this purpose:

(42) commit [changes]

changes all the policies of all the protocols according to the commands that are issued between the previous commit operation and now.

APPENDIX B. RPCL Reference

This appendix section provides the complete syntax of RPCL in BNF form. The first subsection, B.1, gives the syntax of RPCL policy rules, and the next subsection, B.2, gives the syntax of the RPCL policy editor.

B.1 RPCL Policy Syntax

B.1.1 Basic Types

```
<IPv4Addr>: [0-255] |
            [0-255].[0-255] |
            [0-255].[0-255].[0-255] |
            [0-255].[0-255].[0-255].[0-255]

<Mask>: [0-32]

<protocol>: STATIC | DIRECT | RIP | OSPF | BGP

<policy-protocol>: RIP | OSPF | BGP

<prefix>: <IPv4Addr>/<Mask>

<preference>: <integer>

<tag> : <integer>

<rip-metric>: [1-15]

<ospf-metric>: [1-65535]

<ospf-type>: INTRA-AREA | INTER-AREA | EXTERNAL-1 | EXTERNAL-2

<as>: AS<integer>

<as-set>: {<as>, <as>, ..., <as>}

<as-path>: {<as>, <as>, ..., <as>}

<community>: <integer>

<comm-list>: {<community>, <community>, ... }

<peer-AS>: <as>

<peer-IPv4Addr>: [0-255].[0-255].[0-255].[0-255]

<bgp-peer>: <peer-AS> <peer-IPv4Addr>
```

```
<peer-element>: <peer-AS> | <peer-IPv4Addr>

<peer-set>: {<peer-element>, <peer-element>, ... }
```

B.1.2 Filter Types

```
# m and n are [0-32]
<prefix-range>: <prefix> [ ^- | ^+ | ^m | ^m-n ]

<prefix-range-list>: {<prefix-range>, <prefix-range>, ... } |
                     <pfx-setname>

<aspath-regex-member>: <as> | <as-set> | <as-setname> | .

<aspath-regex-operator>: * | + | ? | (m) | (m,) | (m,n) | <empty>

<aspath-unit>: <aspath-regex-member> <aspath-regex-operator>

<aspath-regex>: '<' [^] <aspath-unit> <aspath-unit> ... [$] '>'
```

B.1.3 Policy Filters

```
<STATIC-filter>: <prefix-range-list>

<DIRECT-filter>: <prefix-range-list>

<RIP-filter>: <prefix-range-list> |
              src-gw==<prefix-range-list> |
              tag==<tag>

<OSPF-filter>: <prefix-range-list> |
               type==<ospf-type> |
               tag==<tag>

<BGP-filter>: <prefix-range-list> |
              <aspath-regex> |
              <as> |
              <as-set> |
              community == <comm-list> |
              community.contains <comm-list>
```

B.1.4 Policy Actions

```
<RIP-import-single-action>:
   pref=<preference>

<RIP-import-action>:
   {<RIP-import-single-action>; <RIP-import-single-action>; ...}
```

```
   <RIP-export-single-action>:
      tag=<tag> |
      metric=<rip-metric>

   <RIP-export-action>:
      {<RIP-export-single-action>; <RIP-export-single-action>; ...}

   <OSPF-import-single-action>:
      pref=<preference>

   <OSPF-import-action>:
      {<OSPF-import-single-action>; <OSPF-import-single-action>; ...}

   <OSPF-export-single-action>:
      tag=<tag> |
      metric=<ospf-metric> |
      type=<ospf-type>

   <OSPF-export-action>:
      {<OSPF-export-single-action>; <OSPF-export-single-action>; ...}

   <BGP-single-action>:
      pref=<preference> |
      med=<med> |
      dpa=<dpa> |
      aspath.prepend <peer-AS-list> |
      community=<comm-list> |
      community.=<comm> |             # append a single community
      community.append <comm-list> |  # append multiple communities
      community.delete <comm-list>    # delete the ones that exist

   <BGP-import-action>:
      {<BGP-single-action>, <BGP-single-action>, ...}

   <BGP-export-action>:
      {<BGP-single-action>, <BGP-single-action>, ...}
```

B.1.5 Policy Peering Specification

```
   <peering>: <peer-AS> | <peer-IPv4Addr> | <peer-set>
```

B.1.6 Policy Rules

```
   <PR-import-rule>:
      [from <peering>] [action <PR-import-action>] accept <PR-filter> |
      [from <peering>] block <PR-filter>                              |
      {<PR-Import-Rule>; <PR-Import-Rule>; ...}                       |
      <PR-Import-Rule> refine <PR-Import-Rule>
```

```
<SRC_PR-to-DST_PR-export-rule>:
   [to <peering>] [action <DST_PR-export-action>] announce <SRC_PR-filter> |
   [to <peering>] block <SRC_PR-filter>                                     |
   {<SRC_PR-to-DST_PR-Export-Rule>; <SRC_PR-to-DST_PR-Export-Rule>; ...}   |
   <SRC_PR-to-DST_PR-Export-Rule> refine <SRC_PR-to-DST_PR-Export-Rule>

<PR-aggregation-rule>:
   aggregate into     <prefix>

   triggered by       AND/OR of HC{<prefix-range-list>} and
                              EX{<prefix-range-list>}

   components         ATOMIC |
                     [protocol <PR_1>] <PR_1-filter>
                    [[protocol <PR_2>] <PR_2-filter>] ...

   action             <PR-action>

   export components  from <PR_1> <PR_1-to-PR-export-rule>
                     [from <PR_2> <PR_2-to-PR-export-rule>] ...

   holes              <prefix-range-list>
```

B.2 Policy Editor

```
<str8> : string of {[a-z], [A-Z], [0-9], "-", "_"} of length 8
```

B.2.1 Policy Attribute Macros

```
define peer-<str8>  <peer-set>

define fltr-<str8>  <filter>

define act-<str8>   <action>

undefine peer-<str8>

undefine fltr-<str8>

undefine act-<str8>

show peer[-<str8>]

show fltr[-<str8>]

show act[-<str8>]
```

B.2.2 Import Policy Macros

```
define imp-<str8>
    protocol <PR>
    {<PR-Import-Rule>; <PR-Import-Rule>; ... }

undefine imp-<str8>

insert imp-<str8>
    {<PR-Import-Rule>; <PR-Import-Rule>; ... }
    [<rule-num>]

remove imp-<str8>
    <rule-num>[-<rule-num>]

move imp-<str8>
    <rule-num>[-<rule-num>]  up | down  <num>

compact imp-<str8>

show imp[-<str8>]
```

B.2.3 Export Policy Macros

```
define exp-<str8>
    protocol <SRC_PR> into <DST_PR>
    {<SRC_PR-to-DST_PR-Export-Rule>; <SRC_PR-to-DST_PR-Export-Rule>;

undefine exp-<str8>

insert exp-<str8>
    {<SRC_PR-to-DST_PR-Export-Rule>; <SRC_PR-to-DST_PR-Export-Rule>; ... }
    [<rule-num>]

remove exp-<str8>
    <rule-num>[-<rule-num>]

move exp-<str8>
    <rule-num>[-<rule-num>]  up | down  <num>

compact exp-<str8>

show exp[-<str8>]
```

B.2.4 Aggregation Policy Macros

```
define aggr-<str8>
    protocol <PR>
    {<PR-Aggregation-Rule>; <PR-Aggregation-Rule>; ... }
```

```
undefine aggr-<str8>

insert aggr-<str8>
    {<PR-Aggregation-Rule>; <PR-Aggregation-Rule>; ... }

remove aggr-<str8>
    <prefix-range-list>

show aggr[-<str8>] [<prefix-range-list>] [forest]
```

B.2.5 Import Policy Configuration

```
insert-macro import-<PR>
    imp-<str8>  <macro-num>

insert-macro import-<PR>
    {<PR-Import-Rule>; ...}  <macro-num>

remove-macro import-<PR>
    <macro-num>

move-macro import-<PR>
    <src-macro-num> to <dst-macro-num>

insert import-<PR>
    {<PR-Import-Rule>; ...}  <macro-num>[.<rule-num>]

remove import-<PR>
    <macro-num>.<rule-num>[-<rule-num>]

move import-<PR>
    <macro-num>.<rule-num>[-<rule-num>]  up | down  <num>

compact import-<PR>  <macro-num>

link import-<PR>  <macro-num> to imp-<str8>

unlink import-<PR>  <macro-num> from imp-<str8>

show import-<PR> [<macro-num>]
```

B.2.6 Export Policy Configuration

```
insert-macro  export-<SRC_PR>-<DST_PR>
    exp-<str8>  <macro-num>

insert-macro  export-<SRC_PR>-<DST_PR>
    {<SRC_PR-to-DST_PR-Export-Rule>; ...}  <macro-num>
```

```
remove-macro  export-<SRC_PR>-<DST_PR>
    <macro-num>

move-macro  export-<SRC_PR>-<DST_PR>
    <src-macro-num> to <dst-macro-num>

insert  export-<SRC_PR>-<DST_PR>
    {<SRC_PR-to-DST_PR-Export-Rule>; ...}  <macro-num>[.<rule-num>]

remove  export-<SRC_PR>-<DST_PR>
    <macro-num>.<rule-num>[-<rule-num>]

move  export-<SRC_PR>-<DST_PR>
    <macro-num>.<rule-num>[-<rule-num>]  up | down  <num>

compact  export-<SRC_PR>-<DST_PR>  <macro-num>

link  export-<SRC_PR>-<DST_PR>  <macro-num> to exp-<str8>

unlink  export-<SRC_PR>-<DST_PR>  <macro-num> from exp-<str8>

show export-<SRC_PR>-<DST_PR> [<macro-num>]
```

B.2.7 Aggregation Policy Configuration

```
insert aggregate-<PR>
    {<PR-Aggregation-Rule>; <PR-Aggregation-Rule>; ... }

insert aggregate-<PR>
    aggr-<str8>

remove aggregate-<PR>
    <prefix-range-list>

remove aggregate-<PR>
    aggr-<str8>

show aggregate-<PR> [<prefix-range-list>] [forest]
```

B.2.8 Commit Operation

```
commit [changes]
```