

BGP Policy Control

ISP Workshops



These materials are licensed under the Creative Commons Attribution-NonCommercial 4.0 International license (<http://creativecommons.org/licenses/by-nc/4.0/>)

Last updated 19th November 2019

Acknowledgements

- This material originated from the Cisco ISP/IXP Workshop Programme developed by Philip Smith & Barry Greene
 - Acknowledgements to Patrick Okui for the JunOS examples

- Use of these materials is encouraged as long as the source is fully acknowledged and this notice remains in place

- Bug fixes and improvements are welcomed
 - Please email *workshop (at) bgp4all.com*

Philip Smith

Overview

- Organisations tend to have particular non-technical routing policies
 - A circuit may be preferred because it is cheaper
 - A circuit may be preferred because the traffic by regulation must stay within a certain jurisdiction or country
- BGP in this case is more of a policy tool than the typical routing protocol which just tries to find the best technical route

Overview: Applying Policy with BGP

- ❑ You can accept a prefix announcement, meaning that traffic to that destination will flow towards whoever advertised it to you
- ❑ You can reject a prefix announcement, meaning that traffic to that destination will not flow towards whoever advertised it to you
- ❑ Similarly for prefixes you announce, if they are accepted then traffic to those destinations will flow towards you

Overview: Applying Policy with BGP

- ❑ In addition to the prefix itself you can make similar filtering decisions based on the AS_PATH attribute or which communities have been applied to the prefix announcements
- ❑ Once you have decided to accept a prefix you can optionally set other BGP attributes that will affect how preferred the announcement will be in your network
- ❑ This can be complex or simple and the goal is to influence the router based on the BGP path selection algorithm

Overview: Applying Policy with BGP

- Tools to do this are:
 - Cisco's "prefix-list" for filtering BGP prefixes
 - Juniper also has prefix-lists but the direct equivalent would be the "route-filter"
 - Cisco's filter lists for filtering AS-PATHs
 - Juniper has AS-PATH regular expressions
- For more advanced policy requirements:
 - Route-maps for Cisco IOS
 - BGP Policy statements for Juniper

Policy Control – Prefix List

- ❑ Incremental configuration
- ❑ Applies Inbound or Outbound
- ❑ Based upon network numbers (using familiar IP address/mask format)
- ❑ Prefix-list ends with an implicit default deny

- ❑ Using access-lists in Cisco IOS for filtering prefixes was deprecated long ago
 - **Strongly discouraged!**

- ❑ Note: JunOS equivalent is called “route-filter”

Cisco Prefix Lists – Command Syntax

□ Syntax:

```
[no] ip[v6] prefix-list list-name [seq value] permit|deny  
    network/len [ge value] [le value]
```

network/len: The prefix and its length

ge value: “greater than or equal to”

le value: “less than or equal to”

□ Both “ge” and “le” are optional

- Used to specify the range of the prefix length to be matched for prefixes that are more specific than *network/len*

□ Sequence number is also optional

- `no ip[v6] prefix-list sequence-number` to disable display of sequence numbers

Juniper Route-lists – Command Syntax

- **route-filter *prefix match-type* { *action*; }**
 - *prefix* is the network and its length we'd like to match
 - *match-type* is a group of optional keywords that further match prefixes out of the described network
 - { *action*; } is an optional set of actions to apply if this route-list matches

Match Type	Match Condition
exact	Matches exactly
longer	Matches subnets only
orlonger	Matches prefix and subnets
prefix-length-range X Y	Matches subnet sizes X through Y
upto Y	Matches all subnet sizes up to Y

Cisco Prefix Lists – Examples

- ❑ Deny default route in IPv4

```
ip prefix-list EG deny 0.0.0.0/0
```

- ❑ Deny default route in IPv6

```
ipv6 prefix-list EG-v6 deny ::/0
```

- ❑ Permit the prefix 35.0.0.0/8

```
ip prefix-list EG permit 35.0.0.0/8
```

- ❑ Permit the IPv6 prefix 2001:DB8::/32

```
ipv6 prefix-list EG-v6 permit 2001:DB8::/32
```

Juniper Route-filter – Examples

- ❑ Deny default route in IPv4

```
route-filter 0.0.0.0/0 exact { reject; }
```

- ❑ Deny default route in IPv6

```
route-filter ::/0 exact { reject; }
```

- ❑ Permit the prefix 35.0.0.0/8

```
route-filter 35.0.0.0/8 exact { accept; }
```

- ❑ Permit the IPv6 prefix 2001:DB8::/32

```
route-filter 2001:DB8::/32 exact { accept; }
```

Cisco Prefix Lists – Examples

- ❑ Deny the prefix 172.16.0.0/12

```
ip prefix-list EG deny 172.16.0.0/12
```

- ❑ Deny the IPv6 prefix 3FFE::/16

```
ipv6 prefix-list EG-v6 deny 3FFE::/16
```

- ❑ In 192/8 allow up to /24

```
ip prefix-list EG permit 192.0.0.0/8 le 24
```

- This allows all prefix sizes in the 192.0.0.0/8 address block, apart from /25, /26, /27, /28, /29, /30, /31 and /32.

- ❑ In 2000::/3 allow up to /48

```
ipv6 prefix-list EG-v6 permit 2000::/3 le 48
```

Juniper Route-filter – Examples

- Deny the prefix 172.16.0.0/12

```
route-filter 172.16.0.0/12 exact { reject; }
```

- Deny the IPv6 prefix 3FFE::/16

```
route-filter 3FFE::/16 exact { reject; }
```

- In 192/8 allow up to /24

```
route-filter 192.0.0.0/8 upto 24 { accept; }
```

- This allows all prefix sizes in the 192.0.0.0/8 address block, apart from /25, /26, /27, /28, /29, /30, /31 and /32.

- In 2000::/3 allow up to /48

```
route-filter 2000::/3 upto 48 { accept; }
```

Cisco Prefix Lists – Examples

- In 192/8 deny /25 and above

```
ip prefix-list EG deny 192.0.0.0/8 ge 25
```

- This denies all prefix sizes /25, /26, /27, /28, /29, /30, /31 and /32 in the address block 192.0.0.0/8.
- It has the same effect as the previous example

- In 193/8 permit prefixes between /12 and /20

```
ip prefix-list EG permit 193.0.0.0/8 ge 12 le 20
```

- This denies all prefix sizes /8, /9, /10, /11, /21, /22, ... and higher in the address block 193.0.0.0/8.

- Permit all prefixes

```
ip prefix-list EG permit 0.0.0.0/0 le 32
```

- 0.0.0.0 matches all possible addresses, “0 le 32” matches all possible prefix lengths

Juniper Route-filter – Examples

- In 192/8 deny /25 and above

```
route-filter 192.0.0.0/8 prefix-length-range 25 32 { reject; }
```

- This denies all prefix sizes /25, /26, /27, /28, /29, /30, /31 and /32 in the address block 192.0.0.0/8.
- It has the same effect as the previous example

- In 193/8 permit prefixes between /12 and /20

```
route-filter 193.0.0.0/8 prefix-length-range 12 20 { reject; }
```

- This denies all prefix sizes /8, /9, /10, /11, /21, /22, ... and higher in the address block 193.0.0.0/8.

- Permit all prefixes

```
route-filter 0.0.0.0/0 orlonger { accept; }
```

- 0.0.0.0 matches all possible addresses, "/0 orlonger" matches all possible prefix lengths

Cisco Prefix Lists – Full Example

□ Example Configuration

```
router bgp 100
  address-family ipv4
    network 105.7.0.0 mask 255.255.0.0
    neighbor 102.10.1.1 remote-as 110
    neighbor 102.10.1.1 prefix-list AS110-IN in
    neighbor 102.10.1.1 prefix-list AS110-OUT out
  !
ip prefix-list AS110-IN deny 218.10.0.0/16
ip prefix-list AS110-IN permit 0.0.0.0/0 le 32
!
ip prefix-list AS110-OUT permit 105.7.0.0/16
ip prefix-list AS110-OUT deny 0.0.0.0/0 le 32
```


Policy Control – Cisco Filter List

- ❑ Filter routes based on AS path
 - Inbound or Outbound
- ❑ Referenced in BGP neighbour configuration as:

```
neighbor <addr> filter-list <N> [in|out]
```
- ❑ Referenced in main configuration as:

```
ip as-path access-list <N> [permit|deny] ...
```
- ❑ The as-path access-list finishes with an implicit default deny

Cisco Filter List – Example

□ Example Configuration:

```
router bgp 100
  address-family ipv4
    network 105.7.0.0 mask 255.255.0.0
    neighbor 102.10.1.1 filter-list 5 out
    neighbor 102.10.1.1 filter-list 6 in
  !
  ip as-path access-list 5 permit ^200$
  !
  ip as-path access-list 6 permit ^150$
```

Policy Control – Regular Expressions (IOS)

- Like Unix regular expressions

- . Match one character
- * Match any number of preceding expression
- + Match at least one of preceding expression
- ^ Beginning of line
- \$ End of line
- \ Escape a regular expression character
- _ Beginning, end, white-space, brace
- | Or
- () brackets to contain expression
- [] brackets to contain number ranges

Policy Control – Regular Expressions (JunOS)

- ❑ Juniper AS regular expressions are quite similar to IOS except that the entire AS number comprises one term
 - It is not possible to reference individual characters within the AS number, which differs from the POSIX 1003.2 definitions as used in IOS
 - This means:
 - ❑ The [] operator works in a different way
 - ❑ Some operators have different meanings
 - ❑ There are some extra operators

Policy Control – Regular Expressions (JunOS)

Operator	Match Definition
$\{m,n\}$	At least m and no more than n repetitions of the term. n must be greater than m .
$\{m\}$	Exactly m repetitions of a term
$\{m,\}$	m or more repetitions of a term
?	Zero or one repetition of a term, equivalent to $\{0,1\}$
[]	Set of AS numbers (rather than individual digits)
^	Character at the start of the regex. This is implicit as all regexes must match the entire AS path so isn't needed
\$	Character at the end of the regex. This is also implicit and isn't needed
_	Underscore is not used in JunOS AS regexes since each term is an AS

Policy Control – Regular Expressions (IOS)

□ Simple Examples

<code>.*</code>	match anything
<code>.+</code>	match at least one character
<code>^\$</code>	match routes local to this AS
<code>_1800\$</code>	originated by AS1800
<code>^1800_</code>	received from AS1800
<code>_1800_</code>	via AS1800
<code>_790_1800_</code>	via AS1800 and AS790
<code>_(1800_)+</code>	multiple AS1800 in sequence (used to match AS-PATH prepends)
<code>_\\(65530\\)_</code>	via AS65530 (confederations)

Policy Control – Regular Expressions (JunOS)

□ Simple Examples

<code>.*</code>	match anything
<code>.+</code>	match at least one character
<code>"()"</code>	match routes local to this AS
<code>.* 1800</code>	originated by AS1800
<code>1800 .*</code>	received from AS1800
<code>.* 1800 .*</code>	via AS1800
<code>.* 790 1800 .*</code>	via AS1800 and AS790
<code>.* 1800+ .*</code>	multiple AS1800 in sequence (used to match AS-PATH prepends)
<code>.* 65530 .*</code>	via AS65530 (confederations) – no way to match the `('

Policy Control – Regular Expressions (IOS)

□ Not so simple Examples

<code>^[0-9]+\$</code>	Match AS_PATH length of one
<code>^[0-9]+_[0-9]+\$</code>	Match AS_PATH length of two
<code>^[0-9]*_[0-9]+\$</code>	Match AS_PATH length of one or two
<code>^[0-9]*_[0-9]*\$</code>	Match AS_PATH length of one or two (will also match zero)
<code>^[0-9]+_[0-9]+_[0-9]+\$</code>	Match AS_PATH length of three
<code>_(701 1800)_</code>	Match anything which has gone through AS701 or AS1800
<code>_1849(_.+_)12163\$</code>	Match anything of origin AS12163 and passed through AS1849

Policy Control – Regular Expressions (JunOS)

□ Not so simple Examples

- . Match AS_PATH length of one
- .. Match AS_PATH length of two
- .? . Match AS_PATH length of one or two
- .? .? Match AS_PATH length of one or two (will also match zero)
- ... Match AS_PATH length of three
- .* (701|1800) .* Match anything which has gone through AS701 or AS1800
- .* 1849 .* 12163 Match anything of origin AS12163 and passed through AS1849

Policy Control – Cisco's Route Maps

- ❑ A route-map is like a “programme” for IOS
- ❑ Has “line” numbers, like programmes
- ❑ Each line is a separate condition/action
- ❑ Concept is basically:
 - if *match* then do *expression* and exit
 - else
 - if *match* then do *expression* and exit
 - else etc
- ❑ Route-map “continue” lets ISPs apply multiple conditions and actions in one route-map

Policy Control – JunOS Policy Framework

- ❑ The same general framework is used on Juniper for routing policy as well as firewall filtering
- ❑ Like Cisco IOS route-maps there are three components
 - Match condition that select advertisements
 - Actions performed if the criteria match
 - A **term** is the actual line/statement that contain the match conditions and actions – there can be many **terms**
 - ❑ Unlike IOS they are not numbered
 - ❑ The **term** does not define a “default action” as there isn’t a “permit” or a “deny” in the **term** line

Route Maps – Rules

- Lines can have multiple set statements
 - All set statements are implemented

```
route-map SAMPLE permit 10
  set community 300:1
  set local-preference 120
!
```

- Lines can have multiple match statements
 - All conditions must match

```
route-map SAMPLE permit 10
  match community MY-COMMUNITY
  match ip address prefix-list MY-LIST
  set local-preference 300
!
```

Route Maps – Rules

- A match statement can have multiple commands
 - At least one command must match

```
route-map SAMPLE permit 10
  match ip address prefix-list MY-LIST OTHER-LIST
  set community 300:10
!
```

- Route-map with only a match statement
 - Only prefixes matching go through, the rest are dropped

```
route-map SAMPLE permit 10
  match ip address prefix-list MY-LIST
!
```

Route Maps – Rules

- Line with only a set statement
 - All prefixes are matched and set
 - Any following lines are ignored

```
route-map SAMPLE permit 10
  set local-preference 120
!
route-map SAMPLE permit 20
  remark This line is ignored
  set community 300:5
!
```

Route Maps – Rules

- Line with a match/set statement and no following lines
 - Only prefixes matching the condition are set, the rest are dropped

```
route-map SAMPLE permit 10
  match ip address prefix-list MY-LIST
  set local-preference 120
!
```

Route Maps – Caveats

□ Example

- Omitting the third line below means that prefixes not matching list-one or list-two are dropped

```
route-map SAMPLE permit 10
  match ip address prefix-list LIST-ONE
  set local-preference 120
!
route-map SAMPLE permit 20
  match ip address prefix-list LIST-TWO
  set local-preference 80
!
route-map SAMPLE permit 30
  remark Don't forget this
!
```


Route Maps – Matching prefixes

□ Example Configuration:

```
router bgp 100
  address-family ipv4
    neighbor 1.1.1.1 route-map INFILTER in
  !
  route-map INFILTER permit 10
    match ip address prefix-list HIGH-PREF
    set local-preference 120
  !
  route-map INFILTER permit 20
    match ip address prefix-list LOW-PREF
    set local-preference 80
  !
  ip prefix-list HIGH-PREF permit 10.0.0.0/8
  ip prefix-list LOW-PREF permit 20.0.0.0/8
```

Route Maps – Matching prefixes

□ Commentary:

- If address matches HIGH-PREF set local-pref 120, and then exit
- Otherwise if address matches LOW-PREF, set local-pref 80, and then exit
- No other condition, so all other prefixes are dropped

Route Maps – AS-PATH filtering

□ Example Configuration

```
router bgp 100
  address-family ipv4
    neighbor 102.10.1.2 remote-as 200
    neighbor 102.10.1.2 route-map FILTER-ON-ASPATH in
  !
  route-map FILTER-ON-ASPATH permit 10
    match as-path 1
    set local-preference 80
  !
  route-map FILTER-ON-ASPATH permit 20
    match as-path 2
    set local-preference 200
  !
  ip as-path access-list 1 permit _150$
  ip as-path access-list 2 permit _210_
```

Route Maps – AS-PATH filtering

□ Commentary:

- If prefix originated from AS150, then set local-pref to 80, and exit
- Otherwise if prefix transited AS210 (ie AS210 appears in the path), then set local-pref to 200, and exit
- No other condition, so all other prefixes are dropped

Route Maps – AS-PATH prepends

- Example configuration of AS-PATH prepend

```
router bgp 100
  address-family ipv4
    network 105.7.0.0 mask 255.255.0.0
    neighbor 102.10.1.2 remote-as 300
    neighbor 102.10.1.2 route-map SETPATH out
  !
  route-map SETPATH permit 10
    set as-path prepend 100 100
  !
```

- Use your **own** AS number when prepending
 - Otherwise BGP loop detection may cause disconnects
 - Deliberate insertion of other ASNs is called "AS PATH poisoning"

Route Maps – Matching Communities

□ Example Configuration

```
router bgp 100
  address-family ipv4
    neighbor 102.10.1.2 remote-as 200
    neighbor 102.10.1.2 route-map FILTER-ON-COMMUNITY in
  !
route-map FILTER-ON-COMMUNITY permit 10
  match community MY1
  set local-preference 50
!
route-map FILTER-ON-COMMUNITY permit 20
  match community MY2 exact-match
  set local-preference 200
!
ip community-list standard MY1 permit 150:3 200:5
ip community-list standard MY2 permit 88:6
```

Route Maps – Matching Communities

□ Commentary:

- If prefix belongs to communities 150:3 **AND** 200:5, then set local-pref to 50, and exit
- Otherwise if prefix belongs to **ONLY** community 88:6, then set local-pref to 200, and exit
- No other condition, so all other prefixes are dropped

Community-List Processing

□ Note:

- When multiple values are configured in the same community list statement, a logical AND condition is created. All community values must match to satisfy an AND condition

```
ip community-list standard MY1 permit 150:3 200:5
```

- When multiple values are configured in separate community list statements, a logical OR condition is created. The first list that matches a condition is processed

```
ip community-list standard MY1 permit 150:3  
ip community-list standard MY1 permit 200:5
```


Route Maps – Setting Communities

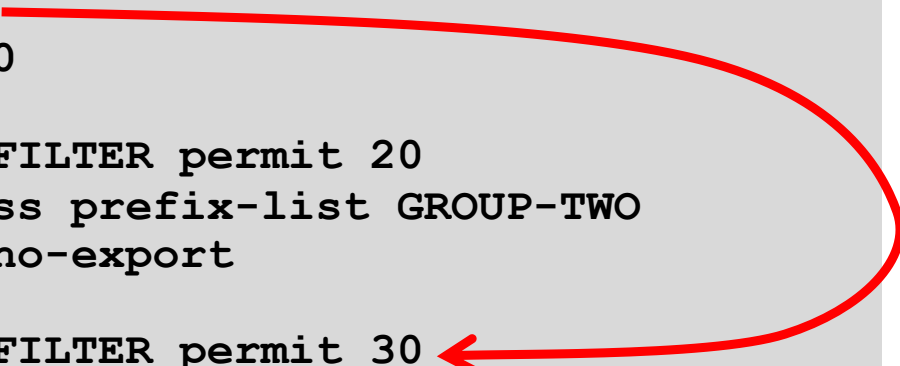
□ Example Configuration

```
router bgp 100
  address-family ipv4
    network 105.7.0.0 mask 255.255.0.0
    neighbor 102.10.1.1 remote-as 200
    neighbor 102.10.1.1 send-community
    neighbor 102.10.1.1 route-map SET-COMMUNITY out
  !
route-map SET-COMMUNITY permit 10
  match ip address prefix-list NO-ANNOUNCE
  set community no-export
!
route-map SET-COMMUNITY permit 20
  match ip address prefix-list AGGREGATE
!
ip prefix-list NO-ANNOUNCE permit 105.7.0.0/16 ge 17
ip prefix-list AGGREGATE permit 105.7.0.0/16
```

Route Map Continue

- Handling multiple conditions and actions in one route-map (for BGP neighbour relationships only)

```
route-map PEER-FILTER permit 10
  match ip address prefix-list GROUP-ONE
  continue 30
  set metric 2000
!
route-map PEER-FILTER permit 20
  match ip address prefix-list GROUP-TWO
  set community no-export
!
route-map PEER-FILTER permit 30
  match ip address prefix-list GROUP-THREE
  set as-path prepend 100 100
!
```



Juniper Policy Example Route Filter

```
policy-options {
  policy-statement import-example {
    term some-prefixes {
      from {
        route-filter 0.0.0.0/0 exact { reject; }
        route-filter 192.0.0.0/8 upto 24;
        route-filter 193.0.0.0/8 prefix-length-range 12 20;
      }
      then {
        preference 200;
        accept;
      }
    }
    term default-deny {
      then {
        reject;
      }
    }
  }
}
```

Juniper Policy Example AS-PATH regex

```
policy-options {
  as-path from1800 ``.* 1800``;
  policy-statement import-example {
    term filter-ases {
      from {
        as-path from1800;
      }
      then {
        preference 10;
      }
    }
  }
}
```

Juniper – applying to BGP session

```
protocols bgp {  
  export our-policy-out;  
  group upstreams {  
    type external;  
    export all-upstreams-out;  
    import incoming-upstreams;  
    neighbor 172.16.2.2 {  
      import import-example;  
    }  
    neighbor 172.20.3.1;  
  }  
}
```

Order of processing BGP policy in IOS

- For policies applied to a specific BGP neighbour, the following sequence is applied:
 - For inbound updates, the order is:
 1. Route-map
 2. Filter-list
 3. Prefix-list
 - For outbound updates, the order is:
 1. Prefix-list
 2. Filter-list
 3. Route-map

Managing Policy Changes in IOS

- ❑ New policies only apply to the updates going through the router **AFTER** the policy has been introduced or changed
- ❑ To facilitate policy changes on the entire BGP table the router handles the BGP peerings need to be “refreshed”
 - This is done by clearing the BGP session either in or out, for example:

```
clear ip bgp <neighbour-addr> in|out
```
- ❑ Do NOT forget **in** or **out** — forgetting results in a hard reset of the BGP session
- ❑ **Note:** Cisco IOS does not automatically apply policy changes after they are added to the configuration
 - Most other router operating systems will implement the route-refresh once the policy change has been committed

Managing Policy Changes in IOS

- Ability to clear the BGP sessions of groups of neighbours configured according to several criteria
- **clear ip bgp <addr> [in|out]**
<addr> may be any of the following:

x.x.x.x

IP address of a peer

all peers

ASN

all peers in an AS

external

all external peers

peer-group <name>

all peers in a peer-group

BGP Policy Control



ISP Workshops