Route-map continue
Route-map continue

Vincent Giles / Ravindra Kumar
Route-map continue

Agenda

1. Overview
2. Use Cases
3. Details / Caveats
4. Configuration
5. Best Practices
6. Troubleshooting
7. Demo
8. Resources
Overview

- Route-map is used to control and modify routing information for:
  1. Redistribution between routing protocols (connected, local, ospf, static)
  2. Locally originated network statement
  3. Inbound or outbound route filtering (per neighbor or per peer-group)

- When a route-map is processed, as soon as one condition is met, the process is stopped and any further route-map sequence is ignored. The parsing algorithm looks like:

  ```
  if condition then do action and exit
  else
    if condition then do action and exit
    else
      ...
  
  condition = match
  action = (deny) or (permit + set options)
  ```

- The route-map “continue” command allows to execute additional route-map entries for greater routing control flexibility.
Route-map operation

Reminder: Regular route-map processing

- The route-map policy are defined as a **sequence of rules**.
- A rule is defined as a **permit** or as a **deny**.
- Each sequence is identified by the **sequence number** and determines the order in which the rules are performed.
- Each sequence of rule contains a **set of match** and a **set of clauses**.
- A **empty** permit rule means **permit every routes**.
- The **match** clause is used to **select the routes** within the route-map policy.
- The **set clause** would determine the actions taken on these selected routes.
- The rules are performed starting from the **lowest sequence number until a matching rule is found** and the **remaining rules are ignored** thereafter.
- There is an **implicit deny at the end** of every route map (when no matching rule is found).
Route-map operation

New: route-map continue processing

- The “continue <sequence_number>” command specifies the route-map entry's sequence number that will be executed next if the current entry's match clause is successful.

- When a successful match occurs and continue command exists, the route map saves the set value first and then jumps to the specified route-map entry.

- Set clauses are saved during the match clause evaluation and are executed only after the route-map evaluation is completed.

- The set clauses are executed in the order in which they were configured. Set clauses can be accumulative/additive like "set as-path prepend" or it can be absolute like "set metric".

- For accumulative value, subsequent values are added in order in which they were configured.

- For absolute value, the values from the last instance will be applied.

- If the route-map entry specified in the continue command does not exist, the route-map processing will be terminated at the current sequence number if its clause is matched.

Example route-map with continue clause:

```
route-map test1 permit seq 10
   match ip address prefix-list P1
   set local-preference 200
   continue 30

route-map test1 permit seq 20
   match ip address prefix-list P2
   set local-preference 300
   continue 50

route-map test1 permit seq 30
   set metric 300

route-map test1 permit seq 40
   set local-preference 100
```
Use Cases
Use case #1
AND / OR MATCH clauses

Before 10.6
• Seq 20 will never match

Since 10.6
• “AND” logic is possible regardless the order in the prefix-list:

```plaintext
Use case #1
AND / OR MATCH clauses

Before 10.6
• Seq 20 will never match

Since 10.6
• “AND” logic is possible regardless the order in the prefix-list:
```
Use case #2

Execute a common set clause for all matched routes

- Continue clause can be used to process the same set clauses for multiple sequences

Before 10.6

```plaintext
ip prefix-list prefix-listA seq 10 permit 10.0.0.0/8 le 32
ip prefix-list prefix-listB seq 10 permit 10.1.0.0/16 le 32
route-map test1-in permit seq 10
  match ip address prefix-list prefix-listB
  set local-preference 110
  set metric 20
  set community 1:1
route-map test1-in permit seq 20
  match ip address prefix-list prefix-listA
  set local-preference 90
  set metric 20
  set community 1:1
```

Since 10.6

```plaintext
ip prefix-list prefix-listA seq 10 permit 10.0.0.0/8 le 32
ip prefix-list prefix-listB seq 10 permit 10.1.0.0/16 le 32
route-map test1-in permit seq 10
  match ip address prefix-list prefix-listB
  set local-preference 110
  continue 40
route-map test1-in permit seq 20
  match ip address prefix-list prefix-listA
  set local-preference 90
  continue 40
route-map test1-in deny seq 30
route-map test1-in permit seq 40
  set metric 20
  set community 1:1
```
Use case #3

More complex processing

- Such route-map logic would be difficult to construct without continue command

```
ip prefix-list prefix-listA seq 10 permit 192.168.0.0/16 le 32
ip prefix-list prefix-listB seq 10 permit 10.1.0.0/16 le 32
ip community-list standard comm1 seq 10 permit 65001:10
ip aspath-list as-list1 seq 10 permit ^65001$

route-map test-in permit seq 10
    match ip address prefix-list prefix-listA
    set local-preference 100
    set community "65001:10 65001:20"
    continue 50

route-map test-in permit seq 20
    match ip address prefix-list prefix-listB
    set local-preference 110
    set community 65001:10
    continue 50

route-map test-in permit seq 30
    match community comm1
    set local-preference 120
    continue 50

route-map test-in deny seq 40

route-map test-in permit seq 50
    match aspath-list as-list1
    set as-path prepend 65001

route-map test-in permit seq 60
```
Use Case #4

Troubleshooting a routing engineering issue

Nominal Situation

- Route-map used in production

During troubleshooting

- This can be useful to skip some rules for temporary support & troubleshooting tests.

```sh
route-map test-in permit seq 10
    match ip address prefix-list prefix-listA
    set local-preference 120
    set community 65001:10
! route-map test-in permit seq 20
    match ip address prefix-list prefix-listB
    set local-preference 110
    set community "65001:10 65001:20"
! route-map test-in deny seq 30
!
```

```sh
route-map test-in permit seq 10
    match ip address prefix-list prefix-listA
    set local-preference 120
    set community 65001:10
    continue 30
! route-map test-in permit seq 20
    match ip address prefix-list prefix-listB
    set local-preference 110
    set community "65001:10 65001:20"
! route-map test-in deny seq 30
!```

skipped
Route-map continue - Details

- Scale: 128 route-map entries in a single route-map
- A continue action is executed even if a route map entry does not contain any match criteria.

```
route-map test-in permit seq 10
  set local-preference 120
  set community "65001:10 65001:20"
  continue 20
!
route-map test-in permit seq 20
  match ip address prefix-list prefix-listB
  set local-preference 110
  set community 65001:10
!
route-map test-in deny seq 30
```

- Route-map continue applies to any possible route-map usage:
  - Redistribution: `switch(config-bgp-vrf-ipv4-uc)# redistribute <connected | local | ospf | static> route-map attribute_set1`
  - Inbound/outbound filtering: `switch(config-bgp-ipv4-uc)# neighbor 192.168.3.0 route-map filtering1 in`
  - Network statement: `switch(config-bgp-ipv4-uc)# network 10.21.0.0/16 route-map attribute_set1`
Route-map continue - REST API

Route-map json model for REST API

```
{
    "10": {
        "action": "permit",
        "call": null,
        "dampening": {},
        "description": null,
        "exitpolicy": null,
        "goto_target": null,
        "match": {},
        "match_as_path": null,
        "match_community_list": {},
        "match_extcommunity_list": {},
        "match_interface": {},
        "match_ipv4_next_hop_prefix_list": null,
        "match_ipv4_prefix_list": {
            "p1": "/rest/v10.04/system/prefix_lists/p1"
        },
        "match_ipv4_route_source_prefix_list": null,
        "match_ipv6_next_hop_prefix_list": null,
        "match_ipv6_prefix_list": null,
        "match_tag": [],
        "route_map_continue": 40,
        "set": {
            "as_path_prepend": "11",
            "metric": 11
        },
        "set_community_list_delete": {},
        "set_tag": []
    }
}
```
Route-map continue – REST API

- Route-map entry from REST API

```bash

{ "10": "/rest/v10.04/system/route_maps/route-in/route_map_entries/10", "20": "/rest/v10.04/system/route_maps/route-in/route_map_entries/20" }

https://172.25.0.8/rest/v10.04/system/route_maps/rm1/route_map_entries?attributes=&depth=2

{ "10": { "action": "permit", "call": null, "dampening": {}, "description": null, "exitpolicy": null, "goto_target": null, "match": {}, "match_as_path": null, "match_community_list": {}, "match_extcommunity_list": {}, "match_interface": {}, "match_ipv4_next_hop_prefix_list": null, "match_ipv4_prefix_list": { "p1": "/rest/v10.04/system/prefix_lists/p1" }, "match_ipv4_route_source_prefix_list": null, "match_ipv6_next_hop_prefix_list": null, "match_ipv6_prefix_list": null, "match_tag": [], "route_map_continue": 40, "set": { "as_path_prepend": "11", "metric": 11 }, "set_community_list_delete": {}, "set_tag": [] }, "20": { "action": "permit", "call": null, "dampening": {}, "description": null, "exitpolicy": null, "goto_target": null, "match": {}, "match_as_path": null, "match_community_list": {}, "match_extcommunity_list": {}, "match_interface": {}, "match_ipv4_next_hop_prefix_list": null, "match_ipv4_prefix_list": { "p1": "/rest/v10.04/system/prefix_lists/p1" }, "match_ipv4_route_source_prefix_list": null, "match_ipv6_next_hop_prefix_list": null, "match_ipv6_prefix_list": null, "match_tag": [], "route_map_continue": 40, "set": { "as_path_prepend": "11", "metric": 11 }, "set_community_list_delete": {}, "set_tag": [] }, }
Continue in REST
Preference with attribute and depth in route_map_entries

curl -X GET "https://172.25.0.8/rest/v10.04/system/route_maps/rm1/route_map_entries/10?attributes=route_map_continue&depth=1" -H "accept: application/json"

```json
{
   "route_map_continue": 40
}
```
Configuration
Route-map continue configuration

- Configure continue command in the route-map to jump to the route-map sequence 40

```
switch(config-route-map-route-in-10)# continue
<2-4294967295> Sequence number to be executed next
```

```
switch(config-route-map-route-in-10)# continue 40
```

```
switch(config-route-map-route-in-10)# no continue
<2-4294967295> Sequence number to be executed next
<cr>
```

- Continue sequence must be greater than the current entry's sequence number. If the continue sequence is lower than the current entry’s sequence number then it will not be applied. However the value will be saved in OVSDB and it can be seen as commented configuration in the show running-config output

```
switch(config-route-map-test1-in-10)# continue 5
Continue sequence must be greater than the current entry’s sequence number 10
```

```
switch# show run
route-map test1-in permit seq 10
  match ip address prefix-list prefix-listA
  set local-preference 90
! The below command has not been successfully configured, Continue sequence must be greater than the current entry's sequence number continue 5
```
Route-map continue configuration

Applying a route-map change

- A change in the route-map will only be applied on new BGP route updates.
- To enforce that the same routing policy is applied on the entire BGP table, the BGP session must be refreshed.
- BGP RIB can be refreshed with the following command:

  switch# clear bgp ipv4 unicast 192.168.1.1 soft in

  switch# clear bgp ipv4 unicast 192.168.1.1 soft in

- Route-refresh capability must be enabled in order to avoid resetting the BGP session (always enabled by default in AOS-CX).
- If route-refresh is not supported on BGP neighbor, inbound soft reconfiguration can be enabled, so that the stored updates are processed by the new policy configuration to create new inbound updates, without the need of BGP session reset. New outbound updates should be triggered from route-map change.

  Note: To perform inbound soft reconfiguration, the BGP speaker must store all received route updates, regardless of the current inbound policy.
Best Practices
Route-map continue - Best Practices

- Most of simple route-map use-cases do not need to use route-map continue.

- Continue command helps when a route matches multiple criteria’s that induce, for each, different set of actions from different sequence numbers.

- Continue command helps simplify the configuration, if same set clauses have to be added in multiple sequences.

```plaintext
route-map test-in permit seq 10
  match ip address prefix-list prefix-listA
  set local-preference 120
  set community "65001:10 65001:20"
  continue 50
!
route-map test-in permit seq 20
  match ip address prefix-list prefix-listB
  set local-preference 110
  set community 65001:10
  continue 50
!
route-map test-in permit seq 30
  match community comm1
  set local-preference 120
  continue 50
!
route-map test-in deny seq 40
!
route-map test-in permit seq 50
  match aspath-list as-list1
  set as-path prepend 65001
!
route-map test-in permit seq 60
```

```plaintext
ip prefix-list prefix-listA seq 10 permit 10.0.0.0/8 le 32
ip prefix-list prefix-listB seq 10 permit 10.1.0.0/16 le 32
!
route-map test1-in permit seq 10
  match ip address prefix-list prefix-listB
  set local-preference 110
  continue 50
route-map test1-in permit seq 20
  match ip address prefix-list prefix-listA
  set local-preference 90
  continue 50
!
route-map test1-in deny seq 30
!
route-map test1-in permit seq 40
  set metric 20
  set community 1:1
```
Troubleshooting
Route-map continue Troubleshooting

```
ip prefix-list prefix-listA seq 10 permit 10.0.0.0/8 le 32
ip prefix-list prefix-listB seq 10 permit 10.1.0.0/16 le 32

route-map test1-in permit seq 10
    match ip address prefix-list prefix-listA
    set local-preference 90
    continue 20

route-map test1-in permit seq 20
    match ip address prefix-list prefix-listB
    set local-preference 110
```

```
switch# show route-map test1-in
Route-map: test1-in
    Seq 10, permit,
        Match :
            ip prefix-list : prefix-listA
        Set :
            local-preference : 90
            continue : 20

    Seq 20, permit,
        Match :
            ip prefix-list : prefix-listB
        Set :
            local-preference : 110
```
Route-map Troubleshooting

- For troubleshooting purpose, it can be useful to know what is the sequence rule number that the route-map process stopped at. It can be a good practice to define and use a BGP community that won't have any impact in routing manipulation (using additive keyword).

```
route-map test1-in permit seq 10
match ip address prefix-list prefix-listA
set local-preference 90
set community 1:10 additive
continue 20
route-map test1-in permit seq 20
match ip address prefix-list prefix-listB
set local-preference 110
set community 1:20 additive
```

- When troubleshooting a route the community value will indicate the route-map exit sequence number.

```
Network           : 10.1.10.0/24
Nexthop           : 192.168.4.1
Peer              : 192.168.4.1               Origin              : IGP
Metric            : 0                         Local Pref         : 100
Weight            : 0                         Calc. Local Pref   : 100
Best              : Yes                       Valid               : Yes
Type              : external                  Stale               : No
Originator ID     : 0.0.0.0
Aggregator ID     :
Aggregator AS     :
Atomic Aggregate  :
RFD Flaps         : 0                         RFD Penalty         : 0
AS-Path           : 65001
Cluster List      :
Communities       : 1:10
Ext-Communities   :
```

In this example, only actions of sequence 10 were executed.
Route-map continue Troubleshooting

DIAG DUMP and FAST log

- Below MIB tables should be verified for the configured **continue sequence** value.
  These table are captured as part of “diag bgp dump mib” command.
  
  1) **BGP Neighbor Routemap:**
     - Table : bgpRouteMapTable, parameter: bgpRouteMapContinue
     - Command : “python /etc/mib.py get localhost bgpRouteMapTable | grep bgpRouteMapContinue”
  
  2) **BGP redistribute/Network Routemap:**
     - Table : rtmRouteMapTable, parameter: rtmRouteMapContinue
     - Command : “python /etc/mib.py get localhost rtmRouteMapTable | grep rtmRouteMapContinue

- **FAST log:**
  If the continue sequence number is not valid below error is logged in the fastlog.
  The configuration will be added in the DB but not in the MIB Tables.

30 Oct 20 10:28:23.743026243): Route map continue sequence is not valid
Demo
Feature/Solution References

Thank you

n-v-m-kishore.tanuku@hpe.com
ravindra.kumar@hpe.com
vincent.giles@hpe.com