Route Selection and Filters

Route Filtering

Filter Syntax

The routing filter rule implements script-like syntax. The example below is a quick demonstration of a routing filter that matches prefixes with a prefix length greater than 24 from subnet 192.168.1.0/24 and increments the default distance by 1. If there is no match then subtract the default distance by one.

```
/routing filter rule
  add chain=myChain
  rule="if (dst in 192.168.1.0/24 && dst-len>24) {set distance +1; accept} else {set distance -1; accept}"
```

Filter rule may consist of multiple matchers and actions:

```
if ( [matchers] ) { [actions] } else { [actions] }
```

There are two types of properties:

- only readable - ones that value is only readable and cannot be rewritten, these properties can be used only by matchers
- readable/writable - ones that value is readable and writeable, used by filter actions, and also can be used by matchers

Readable properties can be matched by other readable properties (for numeric properties only) or constant values using boolean operators.

```
[matchers]:
[prop readable] [bool operator] [prop readable]

[actions]:
[action] [prop writeable] [value]
```

The boolean operator is not used if there is only one possible operation.

Example without boolean operator:

```
if ( protocol connected ) { accept }
```

Example with boolean operator:

```
if ( bgp-med < 30 ) { accept }
```

With readable flag properties, matcher is used without specified boolean operator and without value

```
if ( ospf-dn ) { reject }
```

> ⚠ Be aware that the default action of the routing filter chain is "reject"

Only Readable Properties

Property	Type	Description
Numeric properties		
dst-len		Destination prefix length
bgp-path-len		The current length of the BGP AS-PATH
bgp-input-local-as		AS number of the local peer to which the prefix was sent
bgp-input-remote-as		AS number of the remote peer from which the prefix was received
bgp-output-local-as		AS number of the peer that will advertise the prefix
bgp-output-remote-as		AS number of the peer to which the prefix will be advertised
ospf-metric		Current OSPF metric
ospf-tag		Current OSPF tag
rip-metric		Current RIP metric
rip-tag		Current RIP tag
Flag properties		
active		indicates whether the route is active
bgp-atomic-aggregate		
bgp-communities-empty		indicates if the BGP Communities attribute is empty
bgp-ext-communities-empty		indicates if the BGP Extended Communities attribute is empty
bgp-large-communities-empty		indicates if the BGP Large Communities attribute is empty
bgp-network		Indicates if the prefix is originated from BGP networks

ospf-dn		Indicates if the OSPF route has DN bit set.
Prefix properties		
dst		Destination
ospf-fwd		Current OSPF forwarding address
bgp-input-local-addr		The IP address of the local peer to which the prefix was sent
bgp-input-remote-addr		The IP address of the remote peer from which the prefix was received
bgp-output-local-addr		The IP address of the peer that will advertise the prefix
bgp-output-remote-addr		The IP address of the peer to which the prefix will be advertised
Other Properties		
afi	`ipv4 | ipv6 | l2vpn | l2vpn-cisco | vpnv4 | vpnv6`	The address family of the route.
bgp-as-path	numeric_regexp	AS path matching, read more>>
bgp-as-path-slow-legacy	string_regexp	**Deprecated**. Extremely slow old-style AS path matching. This parameter should be used only as a temporary matcher while migrating from an old ROS v6 config. Read more>>
chain	chain_name	
ospf-type	`ext1 | ext2 | inter | intra | nssa1 | nssa2`	Type of the OSPF route: • ext1 - external (Type 5 LSA) with type1 metric • ext2 - external (Type 5 LSA) with type2 metric • inter - inter-area-route (Type 3 LSA) • intra - intra-area-route (Type 4 LSA) • nssa1 - Type 7 LSA with type1 metric • nssa2 - Type 7 LSA with type1 metric
protocol	bgp | connected | dhcp | fantasy | modem | ospf | rip | static | vpn	Protocol type from which the route was imported.
rpki	invalid | unknown | valid | unverified	RPKI validation status of the prefix
rtab	routing_table_name	Name of the routing table the route was imported from
vrf	vrf_name	Name of the VRF the route was imported from

Writeable Properties

Property	Type	Description
Numeric properties		
distance		route distance
scope		
scope-target		target scope
bgp-weight		BGP WEIGHT attribute
bgp-med		BGP MED attribute is local to the router. It is also used in the output of iBGP peers.

bgp-out-med		BGP MED attribute to be sent to a remote peer. Should be used in the output chain of eBGP peers.
bgp-local-pref		BGP LOCALPREF attribute
bgp-igp-metric		BGP IGP METRIC
bgp-path-peer-prepend		Prepend last received remote peers ASN. If the prefix is originated from the router, then this parameter will not do anything on the router's output, because ASN does not exist yet. If used as a matcher in BGP input, it is possible to filter prefixes exceeding a certain number of prepends. For example, if a remote peer prepends its ASN 5 times, but we want to allow max 4 times prepended ASN, then we can use: "`if (bgp-path-peer-prepend > 4) {reject}`" This parameter also overrides any prepends received from the remote peer, for example, if the remote peer prepended it's AS 3 times, we can remove this prepend by setting "`bgp-path-peer-prepend 1`" in BGP input
bgp-path-prepend		Prepend routers ASN, should be used in BGP output.
ospf-ext-metric		OSPF External route metric
ospf-ext-tag		OSPF external route tag
rip-ext-metric		RIP External route metric
rip-ext-tag		RIP External route tag
Flag properties		
ospf-ext-dn		DN bit for external OSPF routes
blackhole		
suppress-hw-offload		Whether to suppress L3 HW offloading
use-te-nexthop		
Other properties		
gw	ipv4/6 address	IPv4/IPv6 address or interface name. In the case of BGP output, a gateway can be adjusted in the following setups: • is BGP reflector • nexthop-choice is set to propagate • is not eBGP and nexthop-choice=force-self is not set.
gw-interface	interface_name	Interface part of the gateway. Should be used if it is required to attach a specific interface for next-hop, like (1.2.3.4%ether1)
gw-check	*`none|arp|icmp|bfd|bfd-mh`*	
pref-src	ipv4/6 address	

bgp-origin	*igp|egp|incomplete*	
ospf-ext-fwd	ipv4/6 address	
ospf-ext-type	*type1|type2*	
comment	string	
bgp-communities	inline_community_set | set_name	BGP Communities attribute is defined in RFC 1997. Each community is 32-bit in size.
bgp-ext-communities		BGP Extended Communities attribute is defined in RFC 4360. RouterOS parses site-of-origin (prefixed with soo:) and route-target (prefixed with rt:) extended communities. For example, "set bgp-ext-communities rt:1111:2.3.4.5;". It is possible to set /match RAW extended communities value in 64-bit hex, for example, "set bgp-ext-community 0x.....;"
bgp-large-communities		BGP Large Communities attribute is defined in RFC 8092. Suitable for use with all ASNs including 32-bit ASNs. Each community is 12-bytes in length and consists of 3 parts: "global_admin:locap_part_1:local_part_2".

Commands

Command	Params	Description
accept		accept matched prefix
reject		reject matched prefix, the prefix will be stored in the memory as "filtered" and will not be the candidate to be selected as the best path.
return		return to the parent chain
jump	*jump chain_name*	jump to a specified chain
unset	*unset prop_name*	used to unset the value of the following properties: `pref-src|bgp-med|bgp-out-med|bgp-local-pref`
append		append at the end of the list or string. Following property values can be appended: `bgp-communities, bgp-ext-communities, bgp-large-communities, comment`
filter		Values of the following properties can be filtered: `bgp-communities, bgp-ext-communities, bgp-large-communities`
delete		Delete the value of the specified property. Values of the following properties can be deleted: `bgp-communities, bgp-ext-communities, bgp-large-communities`
set	*set prop_writeable value*	The command is used to set a new value to writeable properties. Value can be set from other readable properties of matching types. For numeric properties, it is possible to prefix the value with +/- which will increment or decrement the current property value by a given amount. For example, "`set pref-src +1`" will increment current pref-src by one, or extract value from other readable num property, "`set distance +ospf-ext-metric`"
rpki-verify	*rpki-verify rpki_group_name*	Enable RPKI verification in the current chain from the specified RPKI group.

Operators

Matcher Operators

Operator	Description	Example
&&	Logical AND operator	`if (dst in 192.168.0.0/16 && dst-len in 16-32) {reject;}`
||	Logical OR operator	
not	Logical NOT operator	`if ( not bgp-network) {reject; }`

Num Prop Operators

Operator	Description
in	return true if the value is in provided numeric range. Numeric range can be written in following formats: {int..int}, {int-int}
==	return true if numeric values are equal
!=	return true if numeric values are not equal
>	return true if the left numeric value is greater than the right numeric value
<	return true if the left numeric value is less than the right numeric value
>=	return true if the left numeric value is greater than or equal to the right numeric value
<=	return true if the left numeric value is less than or equal to the right numeric value

Prefix Operators

Operator	Description
in	Return true if the prefix is the subnet of the provided network. If an operator is used to match prefixes from the address list (e.g "`dst in list_name`"), then it will match only the exact prefix.
!=	Return true if the prefix is not equal to the provided value
==	Return true if the prefix is equal to the provided value

BGP Community Operators

Operator	Description	Example
equal	return true if provided communities are equal to the routes property value	
equal-list	return true if communities from provided community-list are equal to the route's property value	
any	returns true if the route's property value contains at least one of provided communities	
any-list	returns true if the route's property value contains at least one community from the provided list	
includes	returns true if the route's property value includes specified communities	
includes-list	returns true if the route's property value includes all communities from the specified communities-list	
subset		
subset-list		
any-regexp		
subset-regexp		

String Operators

Operator	Description

find	Check if provided substring is part of the property value
regexp	Match string regexp of the property value

AS-PATH Regexp Matching

AS Path is the sequence of autonomous system numbers (ASNs), for example AS Path 123 456 789 would indicate, that route originated from AS with the number 789, and to reach the destination, the packet would need to travel through two autonomous systems: 456 and 789. To apply specific routing policies administrator might want to match specific AS numbers or set of numbers in the AS Path (for example, reject prefixes that travel through AS 456), which can be achieved using regular expression (regexp).

There are two common ways how to operate with AS Path data:

- convert whole AS path to string and let regexp operate on the string (ROS v6 or Cisco style)
- let regexp operate on each entry in the AS path as a number (ROS v7, Juniper style)

Basically, the first method is performing the match per character, the second method is performing the match per whole AS number. As you would imagine the latter method is much faster and less resource-intensive than the string matching approach.

This change would require administrators to implement new Regex strategies. Old Regex patterns from RouterOS v6 cannot be directly copied/pasted as they will result either in syntax errors or unexpected results.

Let us take a very basic AS Path filter rule.

```
/routing/filter/rule/add
chain=myChain rule="if (bgp-as-path .1234.) {accept}"
```

In ROS v7 this Regex pattern will match ASN 1234 anywhere in the middle of the AS-path, the same pattern in ROS v6 would match any AS path that contains ASN consisting of at least 6 characters and contains a string of "1234". Obviously, if we directly copy/paste the Regex pattern from one implementation to another it will lead to unexpected/dangerous results. An equivalent pattern in ROS v6 would look something like this: "._1234_.".

Let's take another example from ROS v6, say we have a pattern "1234[5-9]" what it does is it matches 12345 to 12349 anywhere in the string, which means that valid matches are AS-path "12345 3434", "11 9123467 22" and so on. If you enter the same pattern in ROS v7 it will match AS path containing exact ASN 1234 followed by ASN in a range from 5 to 9 (matching AS-paths would be "1234 7 111", "111 1234 5 222" etc., it will not match "12345 3434").

> Do not copy Regex patterns directly from ROS v6 or Cisco configurations, they are not directly compatible. It can lead to unexpected or even dangerous configurations in some scenarios.

Regex Testing Tool

RouterOS now has a built-in regex checking tool to simplify the hard life of the administrators. This tool supports also num-list so now exact regex can be tested against any as-path before applying it to the routing filters.

```
/routing/filter/num-list add list=test range=100-1500

/routing/filter/test-as-path-regexp regexp="[[:test:]]5678\$" as-path="1234,5678"
```

Supported Operators

Operator	Description	Example	Example Explained	Example Matches
^	Represents the beginning of the path	^1234	will math AS-path starting with ASN 1234	
$	Represents the end of the path	1234$	will match AS-path of origin ASN 1234	

*	Zero or more occurrences of the listed ASN	^1234*$	will math Null as-path or as-path where ASN 1234 may or may not appear multiple times	**Math:** 1234 1234 1234 1234 Null path **No Match:** 1234 5678
+	One or more occurrences of the listed ASN	1234+	will match AS-path where ASN 1234 appears at least once	**Match:** 1234 3 1234 6 **No match:** 12345 678
?	Zero or one occurrence of the listed ASN	^1234? 5678	will match AS-path that may or may not start with ASN 1234 appearing once.	**Match**: 5678 1234 5678 **No match:** 1234 1234 5678 12345 5678
.	One occurrence of any ASN	^.$	will match any AS-path with the length of one.	**Match:** 12345 45678 **No match:** 1234 5678
|	Match one of two ASNs on each side	^ (1234|5678)	will match AS-path starting with ASN 1234 or 5678	**Match**: 1234 5678 1234 5678 **No Match:** 91011
[ ] [^]	Represents the set of AS numbers where one AS number from the list must match. Use ^ after opening the bracket to negate the set. It is also possible to reference the pre-defined num-lists from num-list with [[:numset_name:]]	^[1234 5678 1-100]	will match the AS-path that starts with 1234 or 5678 or from the range of 1 to 100	**Match:** 1234 99 5678 **No Match:** 101
()	Group of regexp terms to match	^ (1234$|5678)	will match AS-path that starts and ends with 1234 or AS-path that starts with 5678	**Match:** 1234 5678 9999 **No Match:** 1234 5678

⚠ Repetition ranges {} are not supported.

Community and Num Lists

A list of commonly used numbers can be configured from the `/routing/filter/num-list` menu. These lists of numbers can be used in the filter rules to simplify the filter setup process.

In a similar manner, you are allowed to define also community, extended community, and large community lists. Community sets can be used for matching, appending, and setting.

`/routing/filter/community-list`

Property	Description
comment (*string*; Default:)	
communities (*list of communities*; Default:)	List of communities expressed either as **well-known** name or in the following format: "**as:number**", where each section can be integer [0..65535]. Accepted well known names: accept-own graceful-shutdown no-advertise no-llgr route-filter-6 accept-own-nh internet no-export no-peer route-filter-xlate-4 blackhole llgr-stale local-as route-filter-4 route-filter-xlate-6
disabled (yes / *no*)	
name (*integer [string*; Default:)	Reference name.
regexp (string)	Regexp matcher to match communities. The community set with only the regexp parameter cannot be used to append communities.

`/routing/filter/community-ext-list`

Property	Description
comment (*string*; Default:)	
communities (*list of ext communities*; Default:)	List of extended communities expressed as **raw** integer value or in the typed format: "**type:value**", where type can be: • **rt** - route-target • **soo** - site of origin Value depends on the type, for more info on RT and SoO values ask google.
disabled (yes / *no*)	
name (*integer [string*; Default:)	Reference name.
regexp (string)	Regexp matcher to match communities. The community set with only the regexp parameter cannot be used to append communities.

`/routing/filter/community-large-list`

Property	Description
comment (*string*; Default:)	
communities (*list of large communities*; Default:)	List of large communities expressed in following format: "**admin:value1:value2**", where each section can be integer [0..4294967295].
disabled (yes / *no*)	

name (*integer [string*; Default:)	Reference name.
regexp (string)	Regexp matcher to match communities. The community set with only the regexp parameter cannot be used to append communities.

Route Selection

Route selection rules allow controlling how output routes are selected from available candidate routes. By default, (if no selection rules are set) output always picks the best route.

For example, if we look at the routing table below, we can see that there are 2 candidate routes and one best route. By default when BGP selects which route to send out, it will pick the active route.

```
[admin@4] /routing/route> print where dst-address=1.0.0.0/24
Flags: A - ACTIVE; b, y - COPY
Columns: DST-ADDRESS, GATEWAY, AFI, DISTANCE, SCOPE, TARGET-SCOPE, IMMEDIATE-GW
   DST-ADDRESS  GATEWAY         AFI  DISTANCE  SCOPE  TARGET-SCOPE  IMMEDIATE-GW
 b 1.0.0.0/24   10.155.101.217  ip4        19     40            30  10.155.109.254%ether1
Ab 1.0.0.0/24   10.155.101.232  ip4        20     40            30  10.155.109.254%ether1
 b 1.0.0.0/24   10.155.101.231  ip4        20     40            30  10.155.109.254%ether1
```

But there might be cases where you would want preference for other routes, not the active ones, and here come in-play selection rules.

Selection rules in RouterOS are configured from `/routing/filter/select-rule` menu.

Select rules can also call routing filters where routes get selected based on filter rules. For example, to mimic default output selection we can set up the following rule sets:

```
/routing filter rule
add chain=get_active rule="if (active) {accept}"

/routing filter select-rule
add chain=my_select_chain do-where=get_active
```

Property Reference

/routing/filter/chain

Dynamic list of filter rule chains that can be referenced in BGP/OSPF configuration.

Read-only properties:

Property	Description
dynamic (*yes* | *no*)	
inactive (yes | *no*)	
name (*string*)	

/routing/filter/select-chain

Dynamic list of filter select chains that can be referenced in BGP/OSPF configuration.

Read-only properties:

Property	Description
dynamic (*yes* | *no*)	
inactive (yes | *no*)	
name (*string*)	