Appendix – Routing Chains

Table of contents

The routing chains feature is offering a solution that enables one to configure different parts of the fabric and define a different routing engine to route each of them. The routings are done in a sequence (hence the name "chains") and any node in the fabric that is configured in more than one part is left with the last routing engine updated for it.

Configuring Routing Chains

The configuration for the routing chains feature consists of the following steps:

1. Define the port groups.
2. Define topologies based on previously defined port groups.
3. Define configuration files for each routing engine.
4. Define routing engine chains over defined topologies.

Defining Port Groups

The basic idea behind the port groups is the ability to divide the fabric into sub-groups and give each group an identifier that can be used to relate to all nodes in this group. The port groups are used to define the participants in each of the routing algorithms.

Defining Port Group Policy File

In order to define a port group policy file, set the parameter ‘pgrp_policy_file’ in the opensm configuration file, as follows:

```
/opt/ufm/files/conf/opensm/port_groups_policy_file.conf
```

Configuring Port Group Policy

The port groups policy file details the port groups in the fabric. The policy file should be composed of one or more paragraphs that define a group. Each paragraph should begin with the line ‘port-group’ and end with the line ‘end-port-group’.

For example:

```
port-group
...port group qualifiers...
end-port-group
```

Port Group Qualifiers

> **Note**
>
> Unlike the port group's begining and ending which do not require a colon, all qualifiers must end with a colon (':'). Also – a colon is a predefined mark that must not be used inside qualifier values. An inclusion of a colon in the name or the use of a port group, will result in the policy's failure.

Table 62: Port Group Qualifiers

Parameter	Description	Example
name	Each group must have a name. Without a name qualifier, the policy fails.	name: grp1
use	'use' is an optional qualifier that one can define in order to describe the usage of this port group (if undefined, an empty string is used as a default).	use: first port group

Rule Qualifiers

There are several qualifiers used to describe a rule that determines which ports will be added to the group. Each port group may contain one or more rules of the rule qualifiers in Table 63 (at least one rule shall be defined for each port group).

Table 63: Rule Qualifiers

Parameter	Description	Example
guid list	Comma separated list of guids to include in the group. If no specific physical ports were configured, all physical ports of the guid are chosen. However, for each guid, one can detail specific physical ports to be included in the group. This can be done using the following syntax: • Specify a specific port in a guid to be chosen port-guid: 0x283@3 • Specify a specific list of ports in a guid to be chosen port-guid: 0x286@1/5/7 • Specify a specific range of ports in a guid to be chosen port-guid: 0x289@2-5 • Specify a list of specific ports and ports ranges in a guid to be chosen port-guid: 0x289@2-5/7/9-13/18 • Complex rule port-guid: 0x283@5-8/12/14, 0x286, 0x289/6/8/12	port-guid: 0x283, 0x286, 0x289
port guid range	It is possible to configure a range of guids to be chosen to the group. However, while using the range qualifier, it is impossible to detail specific physical ports. Note: A list of ranges cannot be specified. The below example is invalid and will cause the policy to fail: port-guid-range: 0x283-0x289, 0x290-0x295	port-guid-range: 0x283-0x289

Parameter	Description	Example
port name	One can configure a list of hostnames as a rule. Hosts with a node description that is built out of these hostnames will be chosen. Since the node description contains the network card index as well, one might also specify a network card index and a physical port to be chosen. For example, the given configuration will cause only physical port 2 of a host with the node description 'kuku HCA-1' to be chosen. port and hca_idx parameters are optional. If the port is unspecified, all physical ports are chosen. If hca_idx is unspecified, all card numbers are chosen. Specifying a hostname is mandatory. One can configure a list of hostname/port/hca_idx sets in the same qualifier as follows: port-name: hostname=kuku; port=2; hca_idx=1 , hostname=host1; port=3, hostname=host2 Note: port-name qualifier is not relevant for switches, but for HCA's only.	port-name: hostname=kuku; port=2; hca_idx=1
port regexp	One can define a regular expression so that only nodes with a matching node description will be chosen to the group	port-regexp: SW.*
	It is possible to specify one physical port to be chosen for matching nodes (there is no option to define a list or a range of ports). The given example will cause only nodes that match physical port 3 to be added to the group.	port-regexp: SW.*:3
union rule	It is possible to define a rule that unites two different port groups. This means that all ports from both groups will be included in the united group.	union-rule: grp1, grp2
subtract rule	One can define a rule that subtracts one port group from another. The given rule, for example, will cause all the ports which are a part of grp1, but not included in grp2, to be chosen. In subtraction (unlike union), the order does matter, since the purpose is to subtract the second group from the first one. There is no option to define more than two groups for union/subtraction. However, one can unite/subtract groups which are a union or a subtraction themselves, as shown in the port groups policy file example.	subtract-rule: grp1, grp2

Predefined Port Groups

There are 3 predefined port groups that are available for use, yet cannot be defined in the policy file (if a group in the policy is configured with the name of one of these predefined groups, the policy fails) –

- ALL – a group that includes all nodes in the fabric
- ALL_SWITCHES – a group that includes all switches in the fabric.
- ALL_CAS – a group that includes all HCA's in the fabric.

Port Groups Policy Examples

```
port-group
name: grp3
use: Subtract of groups grp1 and grp2
subtract-rule: grp1, grp2
end-port-group

port-group
name: grp1
port-guid: 0x281, 0x282, 0x283
end-port-group

port-group
name: grp2
port-guid-range: 0x282-0x286
port-name: hostname=server1 port=1
end-port-group

port-group
```

```
name: grp4
port-name: hostname=kika port=1 hca_idx=1
end-port-group

port-group
name: grp3
union-rule: grp3, grp4
end-port-group
```

Defining Topologies Policy File

In order to define a port group policy file, set the parameter 'topo_policy_file' in the opensm configuration file.

```
/opt/ufm/files/conf/opensm/topo_policy_file.conf
```

Configuring Topology Policy

The topologies policy file details a list of topologies. The policy file should be composed of one or more paragraphs which define a topology. Each paragraph should begin with the line 'topology' and end with the line 'end-topology'.

For example:

```
topology
...topology qualifiers...
end-topology
```

Topology Qualifiers

Note

Unlike topology and end-topology which do not require a colon, all qualifiers must end with a colon (':'). Also – a colon is a predefined mark that must not be used inside qualifier values. An inclusion of a column in the qualifier values will result in the policy's failure.

All topology qualifiers are mandatory. Absence of any of the below qualifiers will cause the policy parsing to fail.

Parameter	Description	Example
id	Topology ID. Legal Values – any positive value. Must be unique.	id: 1
sw-grp	Name of the port group that includes all switches and switch ports to be used in this topology.	sw-grp: some_switches
hca-grp	Name of the port group that includes all HCA's to be used in this topology.	hca-grp: some_hosts

Configuration File per Routing Engine

Each engine in the routing chain can be provided by its own configuration file. Routing engine configuration file is the fraction of parameters defined in the main opensm configuration file.

Some rules should be applied when defining a particular configuration file for a routing engine:

- Parameters that are not specified in specific routing engine configuration file are inherited from the main opensm configuration file.
- The following configuration parameters are taking effect only in the main opensm configuration file:
- qos and qos_* settings like (vl_arb, sl2vl, etc.)

- lmc
- routing_engine

Defining Routing Chain Policy File

In order to define a port group policy file, set the parameter ‘rch_policy_file’ in the opensm configuration file, as follows:

```
/opt/ufm/files/conf/opensm/routing_chains_policy.conf
```

First Routing Engine in Chain

The first unicast engine in a routing chain must include all switches and HCA’s in the fabric (topology id must be 0). The path-bit parameter value is path-bit 0 and it cannot be changed.

Configuring Routing Chains Policy

The routing chains policy file details the routing engines (and their fallback engines) used for the fabric’s routing. The policy file should be composed of one or more paragraphs which defines an engine (or a fallback engine). Each paragraph should begin with the line ‘unicast-step’ and end with the line ‘end-unicast-step’.

For example:

```
unicast-step
...routing engine qualifiers...
end-unicast-step
```

Routing Engine Qualifiers

Note

Unlike unicast-step and end-unicast-step which do not require a colon, all qualifiers must end with a colon (':'). Also – a colon is a predefined mark that must not be used inside qualifier values. An inclusion of a colon in the qualifier values will result in the policy's failure.

Parameter	Description	Example
id	'id' is mandatory. Without an id qualifier for each engine, the policy fails. • Legal values – size_t value (0 is illegal). • The engines in the policy chain are set according to an ascending id order, so it is highly crucial to verify that the id that is given to the engines match the order in which you would like the engines to be set.	is: 1
engine	This is a mandatory qualifier that describes the routing algorithm used within this unicast step. Currently, on the first phase of routing chains, legal values are minhop/ftree/updn.	engine: minhop
use	This is an optional qualifier that enables one to describe the usage of this unicast step. If undefined, an empty string is used as a default.	use: ftree routing for cluster 1
config	This is an optional qualifier that enables one to define a separate opensm config file for a specific unicast step. If undefined, all parameters are taken from main opensm configuration file.	config: /etc/config/opensm2.cfg

Parameter	Description	Example
topology	Define the topology that this engine uses. • Legal value – id of an existing topology that is defined in topologies policy (or zero that represents the entire fabric and not a specific topology). • Default value – If unspecified, a routing engine will relate to the entire fabric (as if topology zero was defined). • Notice: The first routing engine (the engine with the lowest id) MUST be configured with topology: 0 (entire fabric) or else, the routing chain algorithm will fail.	topology: 1
fallback-to	This is an optional qualifier that enables one to define the current unicast step as a fallback to another unicast step. This can be done by defining the id of the unicast step that this step is a fallback to. • If undefined, the current unicast step is not a fallback. • If the value of this qualifier is a non-existent engine id, this step will be ignored. • A fallback step is meaningless if the step it is a fallback to did not fail. • It is impossible to define a fallback to a fallback step (such definition will be ignored)	-
path-bit	This is an optional qualifier that enables one to define a specific lid offset to be used by the current unicast step. Setting lmc > 0 in main opensm configuration file is a prerequisite for assigning specific path-bit for the routing engine. Default value is 0 (if path-bit is not specified)	Path-bit: 1

Dump Files per Routing Engine

Each routing engine on the chain will dump its own data files if the appropriate log_flags is set (for instance 0x43).

- The files that are dumped by each engine are:

    - opensm-lid-matrix.dump
    - opensm-lfts.dump
    - opensm.fdbs
    - opensm-subnet.lst

These files should contain the relevant data for each engine topology.

> **Note**
>
> sl2vl and mcfdbs files are dumped only once for the entire fabric and NOT by every routing engine.

- Each engine concatenates its ID and routing algorithm name in its dump files names, as follows:
    - opensm-lid-matrix.2.minhop.dump
    - opensm.fdbs.3.ftree
    - opensm-subnet.4.updn.lst
- If a fallback routing engine is used, both the routing engine that failed and the fallback engine that replaces it, dump their data.

If, for example, engine 2 runs ftree and it has a fallback engine with 3 as its id that runs minhop, one should expect to find 2 sets of dump files, one for each engine:

- opensm-lid-matrix.2.ftree.dump
- opensm-lid-matrix.3.minhop.dump
- opensm.fdbs.2.ftree
- opensm.fdbs.3.munhop