Hybrid OpenFlow Switch

In This Chapter

Alcatel-Lucent supports Hybrid OpenFlow Switch (H-OFS) functionality. The hybrid model allows operators to deploy Software Defined Network (SDN) traffic steering using OpenFlow (OF) atop of the existing routing/switching infrastructure.

Topics in this chapter include:

Hybrid OpenFlow Switching

The hybrid OpenFlow model allows operators to deploy Software Defined Network (SDN) traffic steering using OpenFlow atop of the existing routing/switching infrastructure. Some of the main benefits of the hybrid model include:

- Increased flexibility and speed for new service deployment—H-OFS implements flexible, policy-driven, standard-based Hybrid OpenFlow Switch traffic steering that allows deployment of new services and on-demand services through policy updates rather than service and infrastructure programming.
- Evolutionary capex/opex-optimized SDN deployment—The H-OFS functionality can be deployed on the existing hardware through software upgrade, realizing benefits of FlexPath programmability. The OpenFlow traffic placement is focused access only (i.e. flexible, fast, on-demand service deployment) while network infrastructure provides robustness, resiliency, scale and security.

In a basic mode of operation, a single OpenFlow Switch instance is configured on the router and controlled by a single OpenFlow controller.

The OF controller(s) and router exchange OpenFlow messages using the OpenFlow protocol (version 1.3.1) over the TCP/IPv4 control channel. Both out-of-band (default) and in-band management is supported for connectivity to the controller. An OpenFlow message is processed by the OpenFlow switch instance on the router that installs all supported H-OFS traffic steering rules in a flow table for the H-OFS instance. A single table per H-OFS instance is supported initially.

The H-OFS allows operators to:

- Steer IPv4/IPv6 unicast traffic arriving on a Layer 3 interface by programming the 7x50 L3 PBR ACL actions.
- Steer IPv4/IPv6 unicast traffic arriving on a Layer 2 interface by programing the 7x50 L2 PBF ACL actions.
- Drop traffic by programming ACL action drop.
- Forward traffic using regular processing by programming ACL action forward.

Steering actions programmed using OpenFlow are functionally equivalent to ACL actions. Please see later sections for more details on how OpenFlow standard messages are translated by the SR OS OpenFlow switch into SR OS ACL filter actions.

The router allows operators to control traffic using OF, as follows:

- Operator can select a subset of interfaces on the router to have OF rules enabled, by embedding given instance of H-OFS in filter policies used only by those interfaces.
- For the interfaces with a given H-OFS instance enabled, operator can:

→ Steer all traffic arriving on an interface by programming the flow table with match all entry. Steer a subset of traffic arriving on an interface with this H-OFS enabled by programming the flow table with match rules that select subset of traffic (OpenFlow match criteria are translated to ACL filter match criteria). Unless explicitly listed as a limitation, SR OS H-OFS supports any OpenFlow match criteria that can be translated to SR OS ACL IPv4/IPv6 filter policy match criteria.

To enable rules in a given H-OFS on an existing service router interface, an operator must:

1. Create one or more ingress line card policy
2. Assign those line card ingress filter policies to the 7x50 service/router interfaces
3. Embed H-OFS instance into those line card policies
4. Program OF rules as required

OpenFlow can be embedded in IPv4/IPv6 ACL filter policies deployed on:

- L3 IES service interfaces
- L3 Network interfaces in base router context
- L3 VPRN service interfaces
- L2 VPLS service interfaces
- System ACL filters

OpenFlow functionality is supported in addition to all existing functionality on a given interface and can be enabled with no impact on forwarding performance. Operators can move from CLI/SNMP programmed steering rules to OpenFlow operational model in service without service disruption.

Redundant Controllers and Multiple Switch Instances

The operator can configure one or more instances of an H-OFS (using SNMP/CLI interfaces) with each instance controlled by an OF-controller over a unique OF channel (using openflow protocol). One OF controller can control multiple H-OFS instances (using dedicated channels), or a dedicated OF controller per switch can be deployed. For each switch, up to two OF controllers can be deployed for redundancy. If two controllers are programmed, they can operate in either OFPCR_ROLE_EQUAL roles or in OFPCR_ROLE_MASTER and OFPCR_ROLE_SLAVE roles. Figure 25 depicts this architecture:

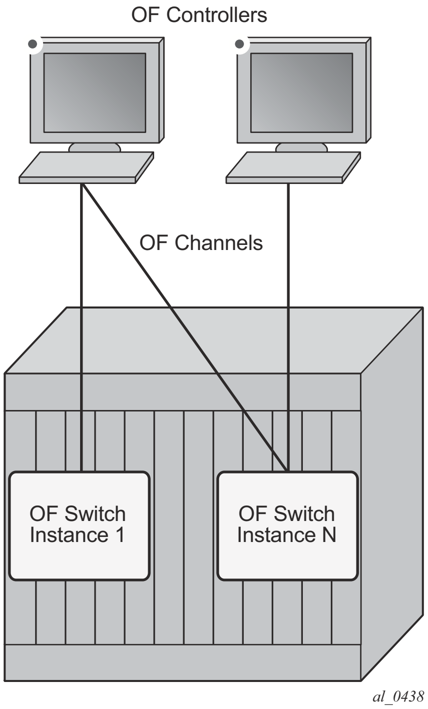

Figure 25: SROS Router/Switch OF Controller/Switch Architecture Overview

GRT-only and Multi-Service H-OFS Mode of Operations

SR OS supports two modes of operations for an H-OFS instance: GRT-only and multi-service. The mode of operations is operator-controlled per H-OFS instance by enabling or disabling **switch-defined-cookie** option (**configure**>**open-flow**>**of-switch**>**flowtable 0**). For backward compatibility, GRT-only mode of operation is default but, since multi-service mode is a functional superset, it is recommended to operate in multi-service mode whenever possible. The operator can change the mode in which an H-OFS instance operates but a shutdown is required first. This will purge all the rules forcing the OF controller to reprogram the switch instance once re-enabled in a new mode. An SROS router supports both H-OFS operational modes concurrently for different switch instances.

Multi-service operational mode uses part of the FlowTable cookie field (higher order 32 bits) to provide the enhanced functionality; the lower order FlowTable cookie bits are fully controlled by the OF controller. Table 11 depicts higher order bit Flow Table cookie encoding used when operating in the multi-service mode.

Table 11: Multi-Service Mode — Higher Order Bit Flow Table Cookie Encoding

sros-cookie Name	sros-cookie Type (Bits 63...60)	sros-cookie Value (Bits 59...32)	FlowTable Entry Interpretation Based on the sros-cookie
grt	0000	0	FlowTable rule is applicable to GRT instance (IES and router interfaces)
system	1000	0	FlowTable rule is applicable to system filters
service	1100	service-id for existing VPLS or VPRN service	FlowTable rule is applicable to an existing VPRN or VPLS service specified by the sros-cookie value

To enable multi-service mode of operation, an operator must embed the OF switch in an ACL filter policy, and, since multi-service H-OFS supports a mix of VPRN/VPLS/GRT/System rules, an additional scope of embedding must be selected (embed open-flow service, embed open-flow system - grt scope used by default). Since after embedding H-OFS instance, an ACL policy contains rules specific to a particular VPRN or VPLS service instance or to a GRT or to a System Filter Policy, the ACL filter policy can only be used in the scope defined by H-OFS embedding.

Rules programmed by an OF controller with grt, system, and service cookies specified are accepted even if the H-OFS instance is not embedded by a filter activated in a given context. Rules programmed by an OF controller with a service cookie specified, when the service ID is not one of the supported service types, or when the service with the specified id does not exist, are rejected with an error returned back to the controller. If an H-OFS is embedded into a line card policy with a specific service context, the embedding must be removed before that service is deleted.

Table 12 summarizes the main differences between the two modes of operation.

Table 12: Differences Between GRT Mode and Multi-service Mode

Function	GRT Mode (no switch-defined-cookie)	Multi-service Mode (switch-defined-cookie)
Support OF on IES access interfaces	Yes	Yes
Support OF on router interfaces in GRT instance	Yes	Yes
Support OF on VPRN access and network interfaces	No (lack of native OF service virtualization)	Yes
Support OF on VPLS access and network interfaces	No (lack of native OF service virtualization)	Yes
Support port and VLAN in flowtable match (see the following section)	No	Yes
Support OF control of System ACL policies	No	Yes
Traffic steering actions	Forward, drop, redirect to LSP, Layer 3 PBR actions only	All
Scale	Up to ingress ACL filter policy entry scale	Up to OF system scale limit per H-OFS instance, and up to 64534 entries per unique sros-cookie value

Caveats:

- Please refer to the Release Notes for a full list of GRT/IES/VPRN/VPLS interfaces that support OF control for multi-service mode.
- The 7x50 H-OFS always requires sros-cookie to be provided for FlowTable operations and will fail any operation without the cookie when **switch-defined-cookie** is enabled.
- OF no-match-action is not programmed in hardware for system filters, since system filters are chained to other filter policies and no-match-action would break the chaining.
- An H-OFS instance does not support overlapping of priorities (flow_priority value) within a single sros-cookie (type+value). The supported values for priority differ based on a value for switch-define-cookie:
  → H-OFS with **switch-define-cookie** disabled
  — Valid flow_priority_range 1 to max-size – 1
  — flow_priority_value 0 is reserved (no match action)

→ H-OFS with **switch-define-cookie** enabled
— Valid flow_priority_range 1 to 65534
— flow-priority_value 0 is reserved (no match action)

- flow_priority must map to a valid filter ID. The following items show how flow_priority is mapped to a filter policy entry ID:

  → H-OFS with **switch-define-cookie** disabled
  — filter entry ID = max-size – flow_priority + embedding offset

  → H-OFS with **switch-define-cookie** enabled
  — filter entry ID = 65535 – flow_priority + embedding offset

- When multiple H-OFS instances are embedded into a single ACL filter, no two H-OFS instances can program the same filter entry ID.

Port and VLAN ID Match in Flow Table Entries

When operating in multi-service mode, SR OS H-OFS supports matching on port and VLAN IDs as part of Flow Table match criteria. When an OF controller specifies incoming port and VLAN values other than "ANY", the H-OFS instance translates them to an SR-OS VPLS SAP (sros-cookie must be set to a valid VPLS service ID). If the translation does not result in an existing VPLS SAP, the rule is rejected and an error is returned to the controller.

A flow table rule with a port/VLAN ID match is programmed only if the matching SAP has this H-OFS instance embedded in its ACL ingress filter policy using SAP scope of embedding (**embed open-flow sap**). Please see SR OS H-OFS Port and VLAN Encoding on page 676 for required encoding of port and VLAN IDs.

The SR OS H-OFS supports a mix of rules with service scope and with SAP scope. For VPLS SAPs, an H-OFS instance must be embedded twice: once for the VPLS service and once for the SAP if both service-level and SAP-level rules are to be activated.

An example of activating both service-level and SAP-level rules inside a single ACL policy 1 used on VPS SAP 1/1/1:100:

```
configure filter ip-filter 1

     scope exclusive

     embed open-flow "ofs1" service vpls100 offset 100

     embed open-flow "ofs1" sap 1/1/1:100  offset 200
```

Caveats:

- Since an H-OFS instance does not support overlapping priorities within a single sros-cookie (type+value), the priority for rules applicable to different SAPs within the same VPLS service must not overlap.

- Masking is not supported when adding a new flow table rule with a port and VLAN ID match.

Hybrid OpenFlow Switch Steering using Filter Policies

A router H-OFS instance is embedded into line card IPv4 and IPv6 filter policies to achieve OF-controlled Policy Based Routing (PBR). When H-OFS instance is created, embedded filters (IP and IPv6) required for that instance are automatically created. The filters are created with names, as follows:

`“_tmnx_ofs_<ofs_name>”`, with the same name for IPv4 and IPv6 filters used.

If embedded filters cannot be allocated due to the lack of filter policy instances, the creation of an H-OFS instance will fail. When the H-OFS instance is deleted, the corresponding embedded filters are freed.

The H-OFS can be embedded only in ingress filter policies on line cards/platforms supporting embedded filters (FP2-based or newer) and for services supporting H-OFS. Embedding of an H-OFS in filter policies on unsupported services is blocked, embedding of an H-OFS in filter policies in unsupported direction or on unsupported hardware follows the general filter policy misconfiguration behavior and is not recommended. Unsupported match fields are ignored. Other match criteria may cause a packet to match an entry.

As soon as an H-OFS instance is created, the controller can program OF rules for that instance. For instance, the rules can be created prior to the H-OFS instance embedding into a filter policy or prior to a filter policy with H-OFS instance embedded being assign to an interface. This allows operator to either pre-program H-OFS steering rules, or to disable the rules without removing them from a flow table by removing the embedding. An error is returned to controller if it attempts to program rules not supported by the system. The following lists examples of the errors returned:

- `unsupported instr: [OFPET_BAD_INSTRUCTION, OFPBIC_UNSUP_INST]`
- `unsupported action: [OFPET_BAD_ACTION, OFPBAC_BAD_TYPE]□`
- `unsupported output port   : [OFPET_BAD_ACTION, OFPBAC_BAD_OUT_PORT]□`
- `unsupported match field: [OFPET_BAD_MATCH, OFPBMC_BAD_FIELD]□`
- `unsupported match value: [OFPET_BAD_MATCH, OFPBMC_BAD_VALUE]□`
- `output port invalid/deleted after flow_mod is sent to filter: OFPET_BAD_ACTION, OFPBAC_BAD_OUT_PORT]□`

As the OF controller updates traffic steering rules, the Hybrid OpenFlow Switch updates the flow table rules. This automatically triggers programming of the embedded filter, which consequently causes instantiation of the rules for all services/interfaces that have a filter policy embedding this H-OFS instance. Embedding filter policy configuration/operational rules apply also to embedded filters auto-created for an H-OFS instance (see Embedded Filter Support for ACL Filter Policies section of this guide). Note that MPLS cannot be deleted if OFS rules are created that redirect to an LSP.

The auto-created embedded filters can be viewed through CLI but cannot be modified and/or deleted through filter policy CLI/SNMP. Operator can see the above embedded filters under show filter context, including the details on the filters themselves, entries programmed, interface association, statistics, etc.

The following picture depicts the H-OFS to service operator-configurable mapping example.

Note: For an H-OFS with **switch-defined-cookie** enabled, embedded filters are created for each unique context in the H-OFS instead.

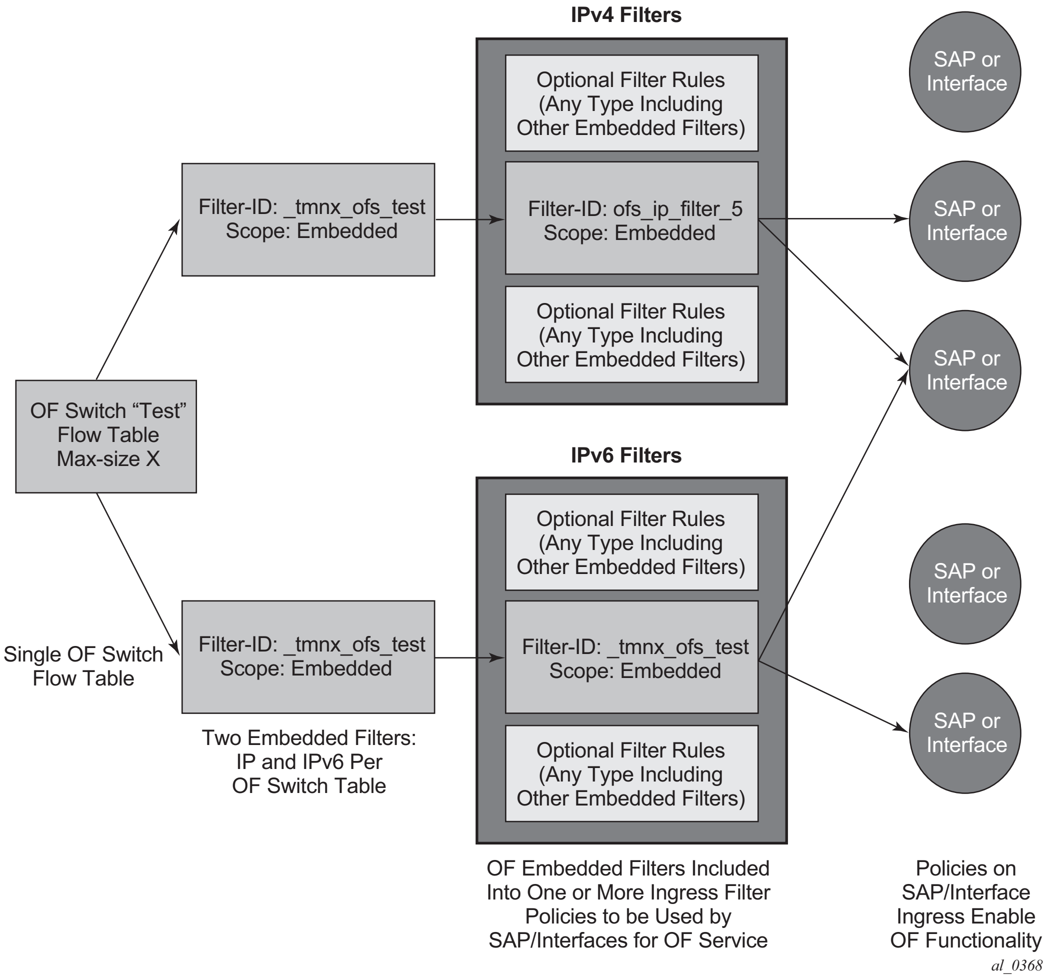
IPv4 Filters
Optional Filter Rules (Any Type Including Other Embedded Filters)
Filter-ID: _tmnx_ofs_test Scope: Embedded
Filter-ID: ofs_ip_filter_5 Scope: Embedded
Optional Filter Rules (Any Type Including Other Embedded Filters)
SAP or Interface
SAP or Interface
SAP or Interface
OF Switch “Test” Flow Table Max-size X
IPv6 Filters
Optional Filter Rules (Any Type Including Other Embedded Filters)
SAP or Interface
Single OF Switch Flow Table
Filter-ID: _tmnx_ofs_test Scope: Embedded
Filter-ID: _tmnx_ofs_test Scope: Embedded
SAP or Interface
Two Embedded Filters: IP and IPv6 Per OF Switch Table
Optional Filter Rules (Any Type Including Other Embedded Filters)
OF Embedded Filters Included Into One or More Ingress Filter Policies to be Used by SAP/Interfaces for OF Service
Policies on SAP/Interface Ingress Enable OF Functionality
al_0368

Figure 26: OF Flow Table Mapping to Router/Switch Service Infrastructure Example — switch-defined-cookie Disabled

The router allows mixing H-OFS rules from one or more H-OFS instances in a single filter policy. Co-existence of H-OFS rules in a single policy with CLI/SNMP programmed rules and/or BGP flowspec programmed rules in a single line card filter policy is also supported. When a management interface and an OF controller flow entry have the same filter policy entry, the management interface-created entry overrides the OF controller-created entry; see the embedded filter functional description. Note that for mixing of the rules from multiple management entities, the controller should not program an entry in its Flow Table that would match all traffic, as this would stop evaluation of the filter policy.

The router supports HA for the OF Flow Table content and statistics. On an activity switch the channel goes down and is re-established by the newly active CPM. “Fail secure mode” operation takes place during channel re-establishment (OpenFlow rules continue to be applied to the arriving traffic). OF controller is expected to re-synchronize the OF table when the channel is re-established.On a router reboot, H-OFS Flow Table rules and statistics are purged. The same takes place when H-OFS instance is shutdown. The H-OFS instance cannot be deleted unless the H-OFS instance is removed first from all embedding filter policies.

Hybrid OpenFlow Switch Statistics

SR-OS Hybrid OpenFlow switch supports statistics retrieval using the OpenFlow protocol. There are two types of statistics that can be collected:

1. **Statistics for SR-OS H-OFS logical ports**
   Logical port statistics are available for RSVP-TE and MPLS-TP LSP logical ports. The non-zero statistics will be returned as long as a given LSP has its statistics enabled through an MPLS configuration. The statistics can be retrieved irrespective of whether a given OF switch uses the specified LSP or not. The statistics account for an aggregate of all packets/bytes forwarded over a given LSP. High availability follows MPLS statistics support.

   Statistics are not available for any other logical ports encodings.
2. **Statistics for SR-OS H-OFS flow table**
   Flow table statistics can be retrieved for one or more flow table entries of a given H-OFS. The returned packet/bytes values are based on ACL statistics collected in hardware. An OpenFlow controller can retrieve statistics either directly from hardware or from the ACL CPM-based bulk request cache. The ACL cache is used when processing an OpenFlow statistics multi-part aggregate request message (OFPMP_AGGREGATE), or when an OpenFlow statistics multi-part flow message request (OFPM_FLOW) is translated to multiple flow table entries (a bulk request). When an OpenFlow multi-part flow statistics request message (OFPM_FLOW) is translated to a single flow table entries requests (a single entry request), the counters are read from hardware in real-time.

   A mix of the two methods can be used to retrieve some flow table statistics from hardware in real-time while retrieving other statistics from the cache. See the Filter Policy Statistics section of this guide for more details on ACL cache and ACL statistics.

Operational Notes:

- Flow table statistics displayed through the CLI debugging tools (tools>dump>open-flow>of-switch) are read in real-time from hardware. However, to protect the system, executing CLI debugging tool commands within 5 seconds will return the same statistics for any flow that had its statistics read from hardware within the last 5 seconds.
- When retrieving flow table statistics at scale, it is recommended to either use bulk requests, or to pace single entry requests in order to obtain the desired balance between stats real-time accuracy and CPM activity.

Hybrid OpenFlow Switch Traffic Steering Details

As described in the previous section, an update to an OpenFlow Switch's flow table, results in the embedded filter update(s), which triggers update to all filter policies embedding those filters. The router automatically downloads the new set of rules to the line cards as defined through service configuration. The rules became part of ingress line card pipeline as depicted by the below picture:

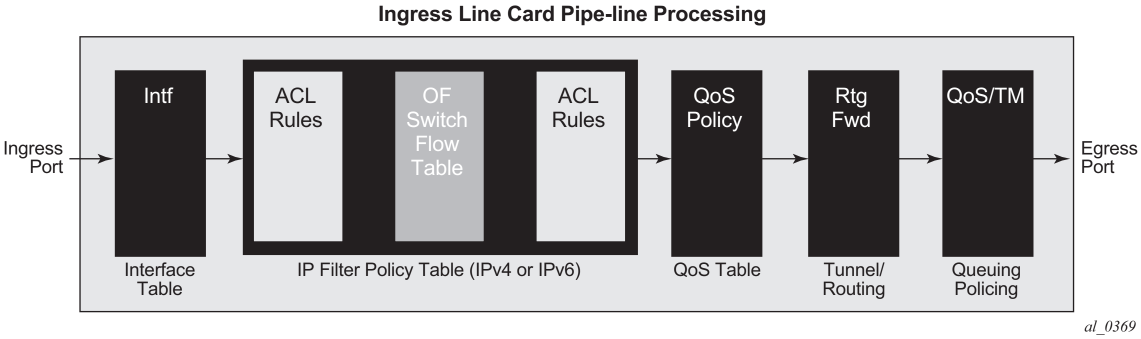

Figure 27: OpenFlow Switch Embedding in Ingress Pipeline Processing

SR OS H-OFS Logical Port

Logical ports are used in OpenFlow to encode switch-specific ports. SROS H-OFS uses logical ports in steering actions by encoding PBR targets. The following encoding shows logical port types supported by SR OS H-OFS:

Table 13: Encoding and Supported Logical Port Types

Bits 31..28	Bits 27..24	Bits 24..0
Logical port type (LPT)	Logical port type sub-type (LPT-S)	Logical port type value (LPT-V) — always padded with leading zeros

Encodings:

```
RSVP LSP: LPT: 0100, LPT-S: 0000 (tunnel), LPT-V: RSVP TE Tunnel ID
```

```
MPLS-TP LSP: LPT: 0100, LPT-S: 0000 (tunnel), LTP-V: MPLS-TP Tunnel Number
```

```
GRT instance: LPT: 0100, LPT-S: 0001 (L3 routing instance), LPT-V: 0
```

`VPRN Id: LPT: 0100, LPT-S: 0001 (L3 routing instance), LPT-V:` VPRN Service ID for a VPRN instance configured on the system. Note that the supported range in OF is limited to a 24-bit service ID value range (a subset of VPRN IDs supported by the SR OS system).

Logical port values other than RSVP-TE LSP and MPLS-TP LSP require H-OFS with **switch-defined-cookie** enabled. GRT instance and VPRN ID logical ports are not stored in the H-OFS logical port table, hence functionality such as retrieving statistics per port is not available for those values.

SR OS H-OFS Port and VLAN Encoding

The OF controller can use port and VLAN values other than ANY for VPLS SAP match and for VPLS steering to SAP for H-OFS instances with **switched-defined-cookie** enabled.

To specify a port in an OF message, SR-OS TmnxPortId encoding must be used. The allowed values are those for Ethernet physical ports and LAG.

To encode VLAN tags, OXM_OF_VLAN_ID and new experimenter OFL_OUT_VLAN_ID fields are used as per Table 14.

Table 14: VLAN Tag Encoding

NULL tag, dot1Q tag, inner QinQ tag VlanId	Outer QinQ tag VlanId
OXM_OF_VLAN_VID	OFL_OUT_VLAN_ID (Experimenter field uses same encoding as OXM_OF_VLAN_VID)

Table 15 shows how OF programmed values are translated to SR-OS SAPs.

Table 15: Translation of OF Programmed Values to SR OS SAPs

OXM_OF_IN_PORT	OXM_OF_VLAN_VID	OFL_OUT_VLAN_ID	Matching SAP SR OS Encoding	Supported in flow_add	Supported in flow_mod flow_del mp_req	Comment
TmnxPortId for port or LAG	Value: 0x0000 Mask: Absent	Must be absent	port-id lag-id	Yes	Yes	Mask must be absent
TmnxPortId for port or LAG	Value: 0x1yyy, yyy encodes qtag1 Mask: Absent	Must be absent	port-id:qtag1 lag-id:qtag1	Yes	Yes	Mask must be absent
TmnxPortId for port or LAG	Value: 0x1FFF Mask: Absent	Must be absent	port-id:* lag-id:*	Yes	Yes	Mask must be absent
TmnxPortId for port or LAG	Value: 0x1000 Mask: 0x1000	Must be absent	port-id: any lag-id: any where "any" is either * or a valid VLAN-ID (but not NULL)	No	Yes	Mask must be 0x1000
TmnxPortId for port or LAG	Value: 0x1yyy, yyy encodes qtag2 Mask: Absent	Value: 0x1zzz, zzz encodes qtag1 Mask: Absent	port-id:qtag1.qtag2 lag-id:qtag1.qtag2	Yes	Yes	Mask must be absent

Table 15: Translation of OF Programmed Values to SR OS SAPs

OXM_OF_IN_PORT	OXM_OF_VLAN_VID	OFL_OUT_VLAN_ID	Matching SAP SR OS Encoding	Supported in flow_add	Supported in flow_mod flow_del mp_req	Comment
TmnxPortId for port or LAG	Value: 0x1FFF Mask: Absent	Value: 0x1zzz, zzz encodes qtag1 Mask: Absent	port-id: qtag1.* lag-id: qtag1.*	Yes	Yes	Mask must be absent
TmnxPortId for port or LAG	Value: 0x1FFF Mask: Absent	Value: 0x1FFF Mask: Absent	port-id: *.* lag-id: *.*	Yes	Yes	Mask must be absent
TmnxPortId for port or LAG	Value: 0x1000 Mask: 0x1000	Value: 0x1zzz, zzz encodes qtag1 Mask: Absent	port-id: qtag1.any lag-id: qtag1.any where any is either * or a valid VLAN-ID (but not NULL)	No	Yes	Mask must be absent for OFL_OUT_VLAN_VID
TmnxPortId for port or LAG	Value: 0x1000 Mask: 0x1000	Value: 0x1FFF Mask: Absent	port-id: *.any lag-id: *.any where "any" is either * or a valid VLAN-ID (but not NULL)	No	Yes	Mask must be absent for OFL_OUT_VLAN_VID
TmnxPortId for port or LAG	Value: 0x1000 Mask: 0x1000	Value: 0x1000 Mask: 0x1000	port-id: any.any lag-id: any.any where "any" is either * or a valid VLAN-ID (but not NULL)	No	Yes	Masks must be 0x1000
TmnxPortId for port or LAG	Value: 0x0000 Mask: Absent	Value: 0x1FFF Mask: Absent	port-id: *.null	Yes	Yes	Mask must be absent

Redirect to IP next-hop

A router supports redirection of IPv4 or IPv6 next-hop for traffic arriving on a L3 interface. An OF controller can rely on this functionality and program PBR next-hop steering actions for H-OFS instances with **switched-defined-cookie** enabled using the following OF encoding:

```
ALU_IPD_EXPERIMENTER_ID:  0x000025BA

ALU_AXN_REDIRECT_TO_NEXTHOP: 2

flow_mod:

       instruction= OFPIT_WRITE_ACTION/OFPIT_APPLY_ACTION,
       action= OFPAT_EXPERIMENTER(ALU_AXN_REDIRECT_TO_NEXTHOP),
       encoding:

struct alu_axn_redirect_to_nhopv4{

       uint16_t  type;                   /* OFPAT_EXPERIMENTER. */
       uint16_t  len;                    /* Total  Length is a multiple of 8. */
       uint32_t  experimenter;           /* Experimenter ID vendor unique*/
       uint8_t    redirect_type ;        /*  Type = 1 for Nhop*/
       uint8_t    flags;                 /*  flags is 0-7 bits:
                                                       Bit  0  =  Ipv4,
                                                       Bit  1  = Ipv6,
                                                       Bit 2   = indirect
                                                       */
       uint8_t     pad[2];
       uint32_t   ipaddr;                /* ipv4 addr */
       unit8_t     pad[0];               /* Not needed */

}; ASSERT(sizeof(alu_axn_redirect_to_nhopv4) == 16)

struct alu_axn_redirect_to_nhopv6{

       uint16_t  type;                   /* OFPAT_EXPERIMENTER. */
       uint16_t  len;                    /* Total  Length is a multiple of 8. */
       uint32_t  experimenter;           /* Experimenter ID vendor unique*/
       uint8_t    redirect_type ;        /*  Type = 1 for Nhop*/
       uint8_t    flags;                 /*  flags is 0-7 bits:
                                                       Bit  0  =  Ipv4,
                                                       Bit  1  = Ipv6,
                                                       Bit 2   = indirect
                                                       */
       uint8_t       pad[2];
       uint128_t   ip6addr;              /* ipv6 addr */
       unit8_t     pad[4];               /* Make total len multiple of 8 */
}; ASSERT(sizeof(alu_axn_redirect_to_nhopv6) == 32)
```

In case of erroneous programming, the following experimenter-specific errors are returned to the controller:

```
enum alu_err_exp_class{

       ALU_ERR_CLASS_RD_TO_SDP        = 0,
       ALU_ERR_CLASS_RD_TO_NHOP       = 1,

}

enum alu_err_subtype_redirect_to_nhop

{

       ALU_ERR_RN_INVALID_FLAGS       = 0
       ALU_ERR_RN_INVALID_ARGS        = 1
       ALU_ERR_RN_INVALID_ADDR        = 2

}
```

Redirect to GRT Instance or VRF Instance

A router supports redirection of IPv4 or IPv6 traffic arriving on an L3 interface to a different routing instance (GRT or VRF). An OF controller can rely on this functionality and program PBR actions for GRT/VRF steering for H-OFS instances with **switched-defined-cookie** enabled using the following OF encoding:

```
flow_mod:

       instruction type: OFPIT_WRITE_ACTIONS/OFPIT_APPLY_ACTION,
       action type: OFPAT_OUTPUT,
```

`port=` SR-OS LOGICAL port encoding GRT or VPRN Service ID as outlined in the SR OS H-OFS Logical Port section.

Note that since a 24-bit value is used to encode the VPRN service ID in the logical port, redirection to a VPRN service with a service ID above that range is not supported.

Redirect to Next-hop and VRF/GRT Instance

A router supports redirection of IPv4 or IPv6 traffic arriving on an L3 interface to a different routing instance (GRT or VRF) and next-hop IP at the same time. An OF controller can rely on this functionality and program PBR steering action for H-OFS instances with **switched-defined-cookie** enabled using the following OF encoding:

```
ALU_IPD_EXPERIMENT_ID:0X000025BA

ALU_AXN_REDIRECT_TO_NEXTHOP:2

flow_mod:

        Instruction 1:
        instruction=OFPIT_WRITE_ACTION/OFPIT_APPLY_ACTION
        action=OFPAT_EXPERIMENTER(ALU_AXN_REDIRECT_TO_NEXTHOP),
```

Encoding as outlined in the Redirect to IP next-hop section (indirect flag must be set).

```
        Instruction 2:
        instruction type: OFPIT_WRITE_ACTIONS/OFPIT_APPLY_ACTION,
        action type: OFPAT_OUTPUT,
```

port= SR-OS LOGICAL port encoding GRT or VPRN Service ID as outlined in the SR OS H-OFS Logical Port section.

Redirect to LSP

The router supports traffic steering to an LSP. The following details the OF encoding to be used by an OF controller:

```
flow_mod:

        instruction type: OFPIT_WRITE_ACTIONS or OFPIT_APPLY_ACTION,
        action type: OFPAT_OUTPUT,
```

port= SR-OS LOGICAL port encoding RSVP-TE or MPLS-TP LSP as outlined in SR OS H-OFS Logical Port section

A received LSP in a flow rule is compared against those in the H-OFS logical port table, if the table does not contain the LSP the rule programming fails. Otherwise, the rule is installed in an ACL filter. As long as any path within the LSP is UP, the redirect rule will forward unicast IP(v6) traffic on the currently used best LSP path by adding LSP transport label and, in case of IPv6 traffic, additionally adding explicit NULL label.

When an LSP in the H-OFS logical port table goes down, the OF Switch removes the LSP from its logical port table and may notify the controller of that fact if the logical port status reporting is enabled. It is up to the OF controller to decide whether to remove rules using this LSP or not. If the rules are left in the flow table, the traffic that was to be redirected to this LSP will instead be subject to a forward action for this flow rule. Note that if the controller does not remove the entries and the system re-uses the LSP identified for another LSP, the rules left in the flow table will start redirecting traffic onto this new LSP.

In some deployments, an SDN controller may need to learn from the router H-OFS logical ports status. To support that function, the OF switch supports optional status reporting using asynchronous OF protocol messages for ports status change.

Redirect to SAP

For traffic arriving on a VPLS interface, a router supports PBF to steer traffic over another VPLS SAP in the same service. An OF controller can rely on this functionality and program PBF steering action for H-OFS instances with **switched-defined-cookie** enabled using the following OF encoding:

```
flow_mod:

       instruction type: OFPIT_WRITE_ACTIONS or OFPIT_APPLY_ACTION,
       Action 1:
       action type: OFPAT_OUTPUT,
       port: 
```

= Encoding as outlined in SR OS H-OFS Port and VLAN Encoding section

```
       Action 2:
       action type=OFPAT_SET_FIELD
```

OXM TLVs encode SAP VLANs as outlined in SR OS H-OFS Port and VLAN Encoding section:

- OXM_OF_VLAN_VID

- OFL_OUT_VLAN_ID (optional)

Redirect to SDP

For traffic arriving on a VPLS interface, a router supports PBF to steer traffic over a VPLS SDP in the same service. An OF controller can rely on this functionality and program PBF steering action for H-OFS instances with switched-defined-cookie enabled using the following OF encoding:

```
ALU_IPD_EXPERIMENTER_ID :  0x000025BA

ALU_AXN_REDIRECT_TO_SDP: 1

flow_mod:

       instruction= OFPIT_WRITE_ACTIONS/OFPIT_APPLY_ACTIONS,
       action= OFPAT_EXPERIMENTER(ALU_AXN_REDIRECT_TO_SDP),
       encoding:

struct alu_axn_redirect_to_sdp{

       uint16_t  type;                   /* OFPAT_EXPERIMENTER. */
       uint16_t  len;                    /* Total  Length is a multiple of 8. */
```

```
      uint32_t  experimenter;          /* Experimenter ID vendor unique*/
      uint8_t    redirect_type ;       /* Type = 0 for SDP*/
      uint8_t    flags;                /*  Flags that can be used to denote
      info(reserved)*/
      uint16_t  sdp-id;                /*  Sdp-id*/
      uint32_t  vcId;                  /*  Vc-id*/
      unit8_t    pad[0];               /* Not needed */
}; ASSERT(sizeof(alu_axn_redirect_to_sdp) == 16)
```

In case of erroneous programming, the following experimenter-specific errors are returned to the controller:

```
enum alu_err_exp_class
{
      ALU_ERR_CLASS_RD_TO_SDP      = 0,
      ALU_ERR_CLASS_RD_TO_NHOP     = 1,
}
enum alu_err_redirect_to_sdp
{
      ALU_ERR_RS_INVALID_FLAGS     = 0
      ALU_ERR_RS_INVALID_ARGS      = 1
      ALU_ERR_RS_INVALID_SDP_ID    = 2
      ALU_ERR_RS_INVALID_VC_ID     = 3
}
```

Forward action

An OF controller can program forward action, when a specific flow is to be forwarded using regular router forwarding. Note that this would be a default behavior if the filter-policy embedding this OF switch instance has a default-action forward and no filter policy rule matches the flow. To implement forward action, the following OF encoding is used:

```
flow_mod:
      instruction type: OFPIT_WRITE_ACTIONS or OFPIT_APPLY_ACTION,
      action type: OFPAT_OUTPUT,
      port= NORMAL
```

where NORMAL is a OF reserved value.

Drop action

An OF controller can program a drop action, when packets of a specific flow are to be dropped. To implement drop action, the following OF encoding is used:

- A wildcard rule with empty action-set

Configuration Notes

The following information describes OF implementation caveats:

- SROS Hybrid OpenFlow Switch requires S/W upgrade only and can be enabled on any SROS router/switch running IOM-2 (with restrictions) or newer line cards. For full functionality, performance and future scale IOM3-XP or newer line cards and CPM4 or newer control cards are recommended.
- Some platforms may not support all OF functionality based on underlying H/W. For example, if underlying H/W does not support IPv6, then OF IPv6 functionality will not be supported, if underlying H/W does not support redirect to LSP, redirect action will be ignored.
- Each flow in an OF flow table must have unique priority. Overlap is not supported
- Timed expiry of the flow entries is not supported
- The implementation is compliant by design with OpenFlow specification as applicable to supported router functionality only.