OVS Commands Reference

March, 2015

Version: 3

www.pica8.com

Pica8, Inc.
1032 Elwell Court, Suite 105
Palo Alto, CA. 94303
+1 (650) 614-5838

sales@pica8.com
support@pica8.com

Contents

Pica8's operating system, PicOS, leverages Open vSwitch (OVS) a production quality, multi-layer virtual switch licensed under the open source Apache 2.0 license. OVS runs as a process within PicOS. The OpenFlow (OF) protocol is driven by the Open Networking Foundation (ONF), a leader in software-defined networking (SDN). The OpenFlow protocol governs three essential components of SDN: an OpenFlow physical switch, an OpenFlow virtual switch to manage virtual machines, and an OpenFlow controller to organize all network pieces.

Organization

This document presents the OVS commands arranged in three major sections:

- **ovsappctl**--the utility for querying and controlling the OVS daemon
- **ovs-ofctl**--the OpenFlow switch management utility
- **ovs-vsctl**--the OpenFlow virtual switch management utility

Resources

For more detailed information on Open vSwitch, the OpenFlow protocol, and command usage, visit the following web sites.

Open vSwitch Advanced Features Tutorial
http://git.openvswitch.org/cgi-bin/gitweb.cgi?p=openvswitch;a=blob_plain;f=tutorial/Tutorial;hb=HEA

Open vSwitch http://openvswitch.org/
OpenFlow https://www.opennetworking.org/sdn-resources/onf-specifications/openflow

Command "ovsappctl"

- ovs-appctl common commands
- ovs-appctl target commands

Command "ovsofctl"

- ovs-ofctl common commands

Command "ovsvsctl"

- ovs-vsctl common commands

Command "ovsappctl"

NAME

ovsappctl: utility for querying and controlling Open vSwitch daemon

SYNTAX

ovs-appctl *[TARGET]* <COMMAND> [ARG...]
At Open vSwitch daemons running time, certain commands accepted to manage them including control their behavior and query their setting. Except ovs-vsctl, ovs-appctl is also available to manage the ovs-vswitchd itself, it sends some internal commands to ovs-vswitchd daemon to change some configurations and prints the daemon's response on standerd output

.

- ovs-appctl common commands
- ovs-appctl target commands

ovs-appctl common commands

This section includes the common comands of ovs-appctl.

- ovs-appctl help
- ovs-appctl hwlog/set-level <module> <level>
- ovs-appctl hwlog/set-type <mode> <type>
- ovs-appctl ofproto/set_l34_enable <true|false>
- ovs-appctl version
- ovs-appctl vlog/list
- ovs-appctl vlog/set [module] [level]

ovs-appctl help

Lists commands supported by the ovs-vswitchd.

ovs-appctl hwlog/set-level <module> <level>

Set hwlog level of <module> as detailed in <level>.
The <module> cantains sdk/api/all; the <level> contains debug/info/warn/error|/off.

ovs-appctl hwlog/set-level all debug

Set hwlog level of 'all' module as 'debug'.

ovs-appctl hwlog/set-level sdk debug

Set hwlog level of 'sdk' module as 'debug'.

ovs-appctl hwlog/set-level api debug

Set hwlog level of 'api' module as 'debug'.

ovs-appctl hwlog/set-type <mode> <type>

Enable/disable hwlog of <mode>.
The <mode> contains config/packet/all; the <type> contains true/false.

ovs-appctl hwlog/set-type config true

Enable hwlog of configurations.

ovs-appctl ofproto/set_l34_enable <true|false>

For the flows which unsupported by HW, such as modifying ip field flow. Pica8 support a switch to control the flows can be allowed adding or not. Default, these flows allowed adding.

'ovs-appctl ofproto/set_L34_enable true' means allow adding these flows, and 'ovs-appctl ofproto/set_L34_enable false' means not allow.

ovs-appctl version

Print the version of the ovs-vswitchd.

ovs-appctl vlog/list

List the known logging modules and their current levels.

ovs-appctl vlog/set [module] [level]

Set log level of [module] as detailed in [level].
The *[level]* contains 'off', 'emer', 'err', 'warn', 'info' and 'dbg'.
If special the [level] value but without [module], represent setting all modules as [level]. If special [module] but without [level], represent setting the [module] as dbg.

ovs-appctl vlog/set dpif_pica

Set vlog level of 'dpif_pica' module as 'dbg'.

ovs-appctl vlog/set odp_util

Set vlog level of 'odp_util' module as 'dbg'.

ovs-appctl vlog/set hwlog

Set vlog level of 'hwlog' module as 'dbg'.

ovs-appctl target commands

This section include one command of target. ovs-appctl -t <target> <command>,The default <tartget> is ovs-vswitchd.So this command can be write like this :

- ovs-appctl -t ovs-vswitchd <command>

ovs-appctl -t ovs-vswitchd <command>

ovs-appctl -t ovs-vswitchd lacp/show

Display the information of LACP.

ovs-appctl -t ovs-vswitchd dpif/dump-flows <bridge>

This command is useful in ovs2.0.

ovs-appctl -t ovs-vswitchd bridge/dump-flows <bridge>

ovs-appctl -t ovs-vswitchd pica/dump-flows

This command is useful in ovs2.0. Using this command can print hardward flows.

Command "ovsofctl"

NAME

ovs-ofctl: OpenFlow switch management utility

SYNTAX

ovs-ofctl [*OPTION*] <*COMMAND*> [*ARG...*]
The **ovsofctl** program is a command line tool for monitoring and administering OpenFlow switches. It can also show the current state of an OpenFlow switch, including features, configuration, and table entries. It should work with any OpenFlow switch, not just Open vSwitch.

ovs-ofctl common commands

- ovs-ofctl add-flow <bridge> <flow>
- ovs-ofctl add-flows <bridge> <file>
- ovs-ofctl add-group <bridge> group_id=<id>,type=<type>,bucket=<actions>
- ovs-ofctl add-meter <bridge> meter=<id>,<meter-parameter>
- ovs-ofctl bundle <bridge> <bundle>
- ovs-ofctl del-flows <bridge> <flow>
- ovs-ofctl del-group <bridge> [group_id=<id>]
- ovs-ofctl del-meter <bridge> meter=<id>
- ovs-ofctl del-meters <bridge>
- ovs-ofctl dump-desc <bridge>
- ovs-ofctl dump-flows <bridge> <flow>
- ovs-ofctl dump-ports <bridge> <port>
- ovs-ofctl dump-ports-desc <bridge>
- ovs-ofctl dump-tables <bridge>
- ovs-ofctl dump-tables-desc <bridge>
- ovs-ofctl mod-flows <bridge> <flow>
- ovs-ofctl mod-group <bridge> group_id=<id>,type=<type>,bucket=<actions>
- ovs-ofctl mod-meter <bridge> meter=<id>,<meter-parameter>
- ovs-ofctl modport <bridge> <iface> <action>
- ovs-ofctl mod-table <bridge> <iface> <mod>
- ovs-ofctl monitor <bridge> [MISSLEN] [invalid_ttl] [watch:[...]]
- ovs-ofctl show <bridge>
- ovs-ofctl snoop <bridge>

ovs-ofctl common commands

Most of these commands take an argument that specifies the method for connecting to an OpenFlow switch. The following connection methods are supported:

- ovs-ofctl add-flow <bridge> <flow>
- ovs-ofctl add-flows <bridge> <file>
- ovs-ofctl add-group <bridge> group_id=<id>,type=<type>,bucket=<actions>
- ovs-ofctl add-meter <bridge> meter=<id>,<meter-parameter>
- ovs-ofctl bundle <bridge> <bundle>
- ovs-ofctl del-flows <bridge> <flow>
- ovs-ofctl del-group <bridge> [group_id=<id>]
- ovs-ofctl del-meter <bridge> meter=<id>
- ovs-ofctl del-meters <bridge>
- ovs-ofctl dump-desc <bridge>
- ovs-ofctl dump-flows <bridge> <flow>
- ovs-ofctl dump-ports <bridge> <port>
- ovs-ofctl dump-ports-desc <bridge>
- ovs-ofctl dump-tables <bridge>
- ovs-ofctl dump-tables-desc <bridge>
- ovs-ofctl mod-flows <bridge> <flow>
- ovs-ofctl mod-group <bridge> group_id=<id>,type=<type>,bucket=<actions>
- ovs-ofctl mod-meter <bridge> meter=<id>,<meter-parameter>
- ovs-ofctl modport <bridge> <iface> <action>
- ovs-ofctl mod-table <bridge> <iface> <mod>
- ovs-ofctl monitor <bridge> [MISSLEN] [invalid_ttl] [watch:[...]]
- ovs-ofctl show <bridge>
- ovs-ofctl snoop <bridge>

ovs-ofctl add-flow <bridge> <flow>

Add flow described by flow.

ovs-ofctl add-flow <bridge> in_port=<port>,actions=<action>

Matches the in_port port in open flow,<port> can be a port number or keyword(eg:LOCAL).

ovs-ofctl add-flow <bridge> dl_vlan=<vlanid>,actions=<action>

Add a flow with the match field dl_vlan (IEEE 802.1q Virtual LAN tag). When the VLAN ID of the packets matches the flow' match field then it will be forward according to the actions.The vlan value ranges from 0 to 4095.

ovs-ofctl add-flow <bridge> dl_vlan_pcp=<value>,actions=<action>

dl_vlan_pcp is a identifier that matches IEEE 802.1q Priority Code Point (PCP) priority.It is a value between 0 and 7.When the value is higher, frame priority level is higher.

ovs-ofctl add-flow <bridge> dl_src=<mac>,actions=<action>

Add a flow with the match field dl_src that matches an Ethernet source address. This value uses 6 pairs of hexadecimal digits to specify,eg: 00:0B:C4:A8:22:B0.

ovs-ofctl add-flow <bridge> dl_dst=<mac>,actions=<action>

Add a flow with the match field dl_dst that matches an Ethernet destination address. This value uses 6 pairs of hexadecimal digits to specify,eg: 00:0B:C4:A8:22:B0.

ovs-ofctl add-flow <bridge> dl_src=<mac/mask>,actions=<action>

ovs-ofctl add-flow <bridge> dl_dst=<mac/mask>,actions=<action>

This type of source mac address provides a wider match field. When the 6 pairs of mask are full *ff* ,it indicates exact match ,as to say, it is the same with *dl_src=xx:xx:xx:xx:xx:xx.*Otherwise,1-bit in mask indicates that the corresponding bit in *mac* must match exactly,0-bit in mask indicates that wildcards that bit.

ovs-ofctl add-flow <bridge> dl_type=<ethernet type>,actions=<action>

Matches Ethernet protocol type *ethertype*, which is specified as an integer between 0 and 65535, inclusive, either in decimal or as a hexadecimal number prefixed by **0x.** (e.g. **0x0806** to match ARP packet).

ovs-ofctl add-flow <bridge> dl_type=<ethernet type>,nw_src=ip[/netmask],actions=<action>

ovs-ofctl add-flow <bridge> dl_type=<ethernet type>,nw_dst=ip[/netmask],actions=<action>

nw_src and nw_dst are the identifier of ip source/destination address.But before you set the ip source/destination address,you must set the Ethernet Type(dl_type).Type values include0x0800: matches IPv4 source/destination address ip(eg:ip , tcp).0x0806:thearp protocol type,matches the ar_sqa or ar_tpa field. respectively, in ARP packets for IPv4 and Ethernet. 0x8035:therarp protocol type, matches the ar_spa or ar_tpa field, respectively, in RARP packets for IPv4 and Ethernet. Other than 0x0800, 0x0806, or 0x8035, the values of nw_src and nw_dst are ignored

ovs-ofctl add-flow <bridge> dl_type=<ethernet type>, nw_proto=<proto>, actions=<action>

This parameter <proto> is specified as a decimal number with the value between 0-255.
When dl_type=0x86dd or IPv6 is specified, the packets must match IPv6 header type.Set proto to 58 to match ICMPv6 packets.
When dl_type=0x0806 or arp is specified,packets must match the lower 8 bits of the ARP opcodes. The opcodes greater than 255 are treated as 0.
When dl_type=0x8035 or rarp is specified,packets must match the lower 8 bits of the RARP opcodes The opcodes greater than 255 are treated as 0.

ovs-ofctl add-flow <bridge> dl_type=<ethernet type>, nw_tos=<tos>, actions=<action>

This parameter <tos> is specified as a decimal number with the value between 0-255.When dl_type=0x0800 or 0x86dd is specified,matches the IP TOS/DHCP or IPv6 traffic class field tos.Note that the two lower reserved bits of tos are ignored for matching purposes.When **dl_type** is wildcarded or set to a value other than 0x0800 or 0x86dd, the value of **nw_tos** is ignored (see **Flow Syntax** above).

ovs-ofctl add-flow <bridge> dl_type=<ethernet type>, nw_ecn=<ecn>,actions=<action>

This parameter <tos> is specified as a decimal number with the value between 0-3. When dl_type=0x0800 or 0x86dd is specified,matches the *ecn* bits in IP ToS or IPv6 traffic class fields .When **dl_type** is wildcarded or set to a value other than 0x0800 or 0x86dd, the value of **nw_ecn** is ignored

ovs-ofctl add-flow <bridge> dl_type=<ethernet type>,nw_proto=<proto>,tp_src=<port>,actions=<action>

ovs-ofctl add-flow <bridge> dl_type=<ethernet type>,nw_proto=<proto>,tp_dst=<port>,actions=<action>

When you want to match **tp_src** or **tp_dst**,you must specify**dl_type** and **nw_proto**as TCP or UDP ,<port> is specified as decimal number between 0-65535.When **dl_type** and **nw_proto** take other values, the values of these settings are ignored.

ovs-ofctl add-flow <bridge> dl_type=<ethernettype>,nw_proto=<proto>,tp_src=<port/mask>,ac

ovs-ofctl add-flow<bridge> dl_type=<ethernet type>,nw_proto=<proto>,tp_dst=<port/mask>,actions=<action>

When you set the **tp_src** and **tp_dst** like this,that mean bitwise match on TCP (or UDP) source or destination port. Both the port and mask are 16-bit decimal numbers or hexadecimal numbers prefixed by 0X.Each 1bit in mask requires the corresponding bit in port must match,while each 0-bit in mask requires the corresponding bit in port to be ignored.

ovs-ofctl add-flow <bridge> table=<number><flow>

By default,the table number is 0. When specified **table** number, that will limit flow manipulations and dump flows commands to apply to the specified table number which is between 0-254.

ovs-ofctl add-flow <bridge> metadata=value[/mask]<flow>

metadata is just useful across multiple flow tables. It makes matching flows easy sometimes, we will just check the **metadata** of flow because different metadata means different flows. But this field is only applied to software flow tables, not applied to the only hardware flow table in our switch.

ovs-ofctl add-flow<bridge> dl_type=<ethernettype>,nw_proto=1,icmp_type=<type>,actions=<a

ovs-ofctl add-flow<bridge> dl_type=<ethernettype>,nw_proto=1,icmp_code=<type>,actions=<

When you specify **dl_type** and**nw_proto** as ICMP or ICMPv6,then **icmp_type** must matches the ICMP type (eg:**icmp_type=1**),and **icmp_code** must matched ICMP code that the value of code is between 0-255.When **dl_type** and **nw_proto** take other values, the values of these settings are ignored.

ovs-ofctl add-flow<bridge> dl_type=0x8847,mpls_label=<label>,actions=<action>

The value of mpls label is between 0-1048575.That is to say you can add 1048575 mpls labels at most.

ovs-ofctl add-flow<bridge> dl_type=0x8847,mpls_tc=<tc>,actions=<action>

The value of **mpls_tc** is between 0-7.It is for experimental use.

ovs-ofctl add-flow<bridge> vlan_tci=<tci[/mask]>,actions=<action>

The tci matches VLAN TCI. The tci and mask are 16-bit values are decimal by default;you can use a 0x prefix to set the values in hexadecimal.when mask is specified,1-bit number in mask requires corresponding bit in tci must match exact,and 0-bit in mask makes corresponding bit in tci be ignored.If do not specify mask,thentci is the exact VLAN TCI to match.

ovs-ofctl add-flow<bridge> dl_type=0x86DD,ipv6_src=<ipv6[/netmask]>,actions=<action>

ovs-ofctl add-flow<bridge> dl_type=0x86DD,ipv6_dst=<ipv6[/netmask]>,actions=<action>

If you want to add a flow with the ipv6 source and destination,you must first set the dl_type as 0x86dd(or ipv6 or tcp6).
The actions of flow include the following values,all the values are applied into add-flow, add-flows, and mod-flows .These are just examples of add-flow below.

ovs-ofctl add-flow<bridge><match-field>actions=[target][,target...]

ovs-ofctl add-flow<bridge><match-field>actions=output:<port>

The port should be an OpenFlow port number or keyword as LOCAL and so on.That means outputting packets to port.

ovs-ofctl add-flow<bridge><match-field>actions=enqueue:<port>:<queue>

The *port* should be an OpenFlow port number or keyword(eg:LOCAL).This action means to enqueue the packets on specified *queue* within the port.The queues numbers depend on switch.

ovs-ofctl add-flow<bridge><match-field>actions=NORMAL

This action means making packets be processed as device' normal L2/L3.

ovs-ofctl add-flow<bridge><match-field>actions=flood

This action means outputting packets on all physical ports other than the port on which it was receivedand any ports on which flooding is disabled (typically, these would be ports disabled by the IEEE 802.1D spanning tree protocol).

ovs-ofctl add-flow <bridge> <match-field> actions=all

This action means outputting packets on all physical ports other than the port on which it was received.

ovs-ofctl add-flow <bridge> <match-field>actions=controller(key=value...)

When you set action as controller ,the packets will be sent in OpenFlow controller as packet-in message.

ovs-ofctl add-flow <bridge> <match-field>actions=drop

This action means discard the packets without further processing or forwarding.when this action **drop** is used,you should not specify other actions.

ovs-ofctl add-flow <bridge> <match-field> actions=push_vlan:<ethertype>,set_field:<value>-\>vlan_vid,outpu

Using **push_vlan**,you can push a new vlan to packets,*ethertype*is the ethertype for the tag**(**only support **0x8100** at the moment.**set_field** is used to set the value of priority ,the format is like this.

ovs-ofctl add-flow <bridge> <match-field> dl_type=<ethertype>,dl_src=<src>,dl_dst=<dst>,actions=push_mp

Using**push_mpls,**you can push a new mpls label for a packets without any mplslabel.At this time,the*ethertype*must be either MPLS unicast ethertype**0x8847,**or the MPLS multicast ethertype **0x8848.**if the packets has already contains a mplslabel,pushes a new outermost label as a copy of the existing outermost label.

ovs-ofctl add-flow<bridge><match-field>actions=pop_mpls:<ethertype>

You can pop outer mpls label by setting *ethertype*to MPLS Ethertype.and pop all MPLS label by setting *ethertype* to the types except mplsethertype.

ovs-ofctl add-flow<bridge><match-field>actions=push_pbb:<ethertype>,set

You can push outermost mac address for packets.PBB is a mac-in-mac technology.ethertype is **0x88e7.**

ovs-ofctl add-flow<bridge><match-field>actions=pop_pbb

You can pop the outer mac for packets by using pop_pbb.

ovs-ofctl add-flow<bridge><match-field>actions=mod_vlan_vid:<vlan_vid>

This action means modify the thevlan id of the packets.Thevlan tag is added or modified as necessary to match specifiedvalue.when the vlan tag is added ,then priority of the packets are 0 by default.You can set priority by **mod_vlan_pcp**.*vlan_id* is between 0-4095.

ovs-ofctl add-flow<bridge><match-field>actions=mod_vlan_pcp:<vlan_pcp>

You can modify the vlan priority of the packets.vlan_pcp value is between0-7, 0 presents the lowest priority and 7 means highest priority.

ovs-ofctl add-flow<bridge><match-field>actions=strip_vlan

Using this action ,you can pop vlan tag from packets.

ovs-ofctl add-flow<bridge><match-field>actions=mod_dl_src:<mac>

User can modify source mac of packets from the output port by **mod_dl_src.**

ovs-ofctl add-flow<bridge><match-field>actions=mod_dl_dst:<mac>

User can modify *destination* mac of packets from the output port by **mod_dl_dst**

ovs-ofctl add-flow<bridge><match-field>actions=mod_nw_src:<ip>

Modify the *source ipv4* addressof specified flow,by using mod_nw_src.

ovs-ofctl add-flow<bridge><match-field>actions=mod_nw_dst:<ip>

Modify the *destination ipv4* addressof specified flow,by using mod_nw_dst.

ovs-ofctl add-flow<bridge><match-field>actions=mod_tp_src:<port>

Modify the TCP or UDP source port.

ovs-ofctl add-flow<bridge><match-field>actions=mod_tp_dst:<port>

Modify the TCP or UDP destination port.

ovs-ofctl add-flow<bridge><match-field>actions=mod_nw_tos:<tos>

By using this action you can modify the IPv4 TOS or DSCP field to tos ,which must be a multiple of 4 between 0-255.This action can not modify the lower 2-bits of tos field which represents ECN bits.

ovs-ofctl add-flow<bridge><match-field>actions=resubmit([port],[table])

By this action,you can re-search OpenFlow flow table with the**in_port** field replaced by *port*. Or research the table whose number is specified by table.

ovs-ofctl add-flow<bridge><match-field>actions=set_queue:<queue>

By taking this action, the packets that is output the port will be output the specified *queue*. Different switch has different numbers of supported queues.

ovs-ofctl add-flow<bridge><match-field>actions=set_mpls_ttl:ttl

This action will set the TTL of outer MPLS label stack entry of a packet. The range of TTL number is between 0-255.

ovs-ofctl add-flow<bridge><match-field>actions=controller(key=value...)

Taking this action will send packets to OpenFlow controller as "packet-in" message. There are some key-value pairs as follow.
max_len=*nbytes:* limit the maximum length of packets sent to the controller. By default, the entire packet is sent.
reason=*reason*:Specify *reason* as the reason for sending the message in the ‘‘packet in’’ message. The supported reasons are **action** (the default), **no_match**, and **invalid_ttl**.

Summarize above flows as a table:

ovs-ofctl add-flow <bridge> <flow>

Add table:

feature	Match fields
L2	in_port=<port>
	dl_src=<mac>
	dl_dst=<mac>
	dl_src=<mac/mask>
	dl_dst=<mac/mask>

	dl_type=<ethernet type>
	<match-field>
	<match-field>
L3	**dl_type=<ethernet type>,nw_src=ip[/netmask]**
	dl_type=<ethernet type>,nw_dst=ip[/netmask]
	dl_type=<Ethernet type>, nw_proto=<proto>
	dl_type=<ethernet type>, nw_ecn=<ecn>
	dl_type=<ethernettype>,nw_proto=<proto>,tp_src=<port>
	dl_type=<ethernet type>,nw_proto=<proto>,tp_dst=<port>
	dl_type=<ethernettype>,nw_proto=<proto>,tp_src=<port/mask>
	dl_type=<ethernet type>,nw_proto=<proto>,tp_dst=<port/mask>
	<match-field>
	<match-field>
	<match-field>
	<match-field>
Table	**table=<number>**

Metadata	**metadata=value[/mask]**
Lcmp	**dl_type=<ethernettype>,nw_proto=1,icmp_type=<type>**
	dl_type=<ethernettype>,nw_proto=1,icmp_code=<type>
MPLS	**dl_type=0x8847,mpls_label=<label>**
	dl_type=0x8847,mpls_tc=<tc>
	<match-field> dl_type=<ethertype>,dl_src=<src>,dl_dst=<dst>
	<match-field>
	<match-field>
vlan	**vlan_tci=<tci[/mask]>**
	dl_vlan=<vlanid>
	dl_vlan_pcp=<value>
	<match-field>
	<match-field>
	<match-field>

	<match-field>
Ipv6	**dl_type=0x86DD,ipv6_src=<ipv6[/netmask]>**
	dl_type=0x86DD,ipv6_dst=<ipv6[/netmask]>
Openflow	**<match-field>**
	<match-field>
	<match-field>
	<match-field>
	<match-field>
	<match-field>
	<match-field>
	<match-field>
	<match-field>
PBB	**<match-field>**
	<match-field>
	<match-field>
Qos	**<match-field>**

	<match-field>
	dl_type=<ethernet type>, nw_tos=<tos>

ovs-ofctl add-flows <bridge> <file>

Add multiple flows from file.You can create a file with many flows in server then copy this file to your switch.Then use this command **ovs-ofctl add-flows <bridge> <file>** to add these flows in this file.

Command

ovs-ofctl add-flows <bridge> <file>

Example

There are two flows in file 111.txt.Add these flows in this file as below:

admin@PicOS-OVS$sudo scp sophia.sun@10.10.50.20:/home/sophia.sun/111.txt ./
The authenticity of host '10.10.50.20 (10.10.50.20)' can't be established.
ECDSA key fingerprint is 8f:f3:ca:a1:b9:a9:67:26:e1:54:dc:6a:62:74:d5:f6.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.50.20' (ECDSA) to the list of known hosts.
sophia.sun@10.10.50.20's password:
111.txt 100% 54 0.1KB/s 00:00
admin@PicOS-OVS$
admin@PicOS-OVS$
admin@PicOS-OVS$
admin@PicOS-OVS$ls
111.txt
admin@PicOS-OVS$
admin@PicOS-OVS$
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-ofctl add-flows br0 111.txt
admin@PicOS-OVS$
admin@PicOS-OVS$
admin@PicOS-OVS$ovs-ofctl dump-flows br0
OFPST_FLOW reply (OF1.4) (xid=0x2):
cookie=0x0, duration=10.804s, table=0, n_packets=n/a, n_bytes=0, in_port=7 actions=output:8
cookie=0x0, duration=10.800s, table=0, n_packets=n/a, n_bytes=0, in_port=8 actions=output:9

ovs-ofctl add-group <bridge> group_id=<id>,type=<type>,bucket=<actions>

PicOS OVS supports group table from Openflow 1.2.

ovs-ofctl add-group <bridge> group_id= <id> ,type=all,bucket= <actions>[,bucket=<actions>]

Add a group that group_id=<id>, type=all in <bridge>.The max buckets number can be created is 10.

Example:

Create a group type=all, include two buckets to modify the src-mac and dst-mac.

```
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=1,type=all,bucket=mod_dl_src=00:00:00:11:11:11,mod_dl_dst=00:00:00:22:
```

ovs-ofctl add-group <bridge> group_id= <id> ,type=indirect,bucket= <actions>

Add a group that group_id=<id>, type=indirect in <bridge>.

Example:

Create a group type=indirect, just include one bucket to modify the src-mac and dst-mac.

```
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=2,type=indirect,bucket=mod_dl_src=00:00:00:99:11:11,mod_dl_dst=00:00:0
```

ovs-ofctl add-group <bridge> group_id= <id> ,type=select,bucket= <actions>[,bucket=<actions>]

Add a group that group_id=<id>, type=select in <bridge>.

PisOS OVS is not support weight now, user can not to special the weight or special the weight=1.

Because OVS forward packet by TCAM, the traditional ECMP in routing table can not be used in OVS mode.

We implement a "dummy ECMP" by splitting the matching fields of a flow. By the group that type=select, system will choose one match field to split in nw_src, nw_dst, dl_src, dl_dst. The nw_src is the highest priority to choose, if has no special nw_src or the mask of nw_src is 32, will choose nw_dst. The last choose dl_src and dl_dst. The premise is the match fields must have mask except 32 nor ff:ff:ff:ff:ff:ff.

Except this using, other flows involved with select group will be packet-driven flows.

Example:

Create a group type=select, the actions of multi buckets only have output.

```

```

```
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=3,type=select,bucket=bucket=output:1,output:2,bucket=output:3,bucket=o
```

ovs-ofctl add-group <bridge> group_id= <id> ,type=ff,bucket= <actions>[,bucket=<actions>]

Add a group that group_id=<id>, type=ff, in <bridge>. The ff means fast_failover.
Example:

```
admin@PicOS-OVS$ovs-ofctl add-group br0
group_id=4,type=ff,bucket=watch_port:2,watch_group:2,output:3,watch_port:3,watc
```

Actions supported table

Chip	Firebolt3								Triumph2	Trident-II		
Model	3290	3295	3920	3922	3930	as5600_52x	3780	as4600_54t	3297	5101	5401	as6701_32x
output:<PORT_ID>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mod_dl_src	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mod_dl_dst	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mod_vlan_vid	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mod_vlan_pcp	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mod_nw_tos	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
strip_vlan	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
push_pbb	×	×	✓	✓	✓	✓	✓	×	✓	✓	✓	✓
pop_pbb	×	×	✓	✓	✓	✓	✓	×	✓	✓	✓	✓
set_queue	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
push_mpls	×	×	✓	✓	✓	✓	✓	×	✓	✓	✓	✓
pop_mpls	×	×	✓	✓	✓	✓	✓	×	✓	✓	✓	✓
mod_nw_src	×	×	×	×	×	×	×	×	×	✓	✓	✓
mod_nw_dst	×	×	×	×	×	×	×	×	×	✓	✓	✓
mod_tp_src	×	×	×	×	×	×	×	×	×	✓	✓	✓
mod_tp_dst	×	×	×	×	×	×	×	×	×	✓	✓	✓

Follow actions are not supported.

different modification for different bucket

eg.

bucket=mod_dl_src:22:11:11:11:11:11,output:2,bucket=mod_dl_src:22:22:22:22:22:22,output:3

ovs-ofctl add-meter <bridge> meter=<id>,<meter-parameter>

PicOS OVS supports meter from Openflow 1.3.

ovs-ofctl add-meter <bridge> meter= <id> ,kbps [,burst,stats] ,band=type=drop,rate= <rate>[,burst_size=<size>][,prec_level=<level>]

Add a meter, the type=drop.
Example:

Without burst_size. Limit the rate as 30000kbps.

```
root@PicOS-OVS$ovs-ofctl add-meter br0 meter=1,kbps,band=type=drop,rate=30000
```

With burst_size. Limit the rate as 30000kbps.

```
root@PicOS-OVS$ovs-ofctl add-meter br0
meter=2,kbps,burst,band=type=drop,rate=30000,burst_size=30000
```

ovs-ofctl add-meter <bridge> meter= <id> ,kbps [,burst,stats] ,band=type= dscp_remark,rate= <rate> ,prec_level= <level>[,burst_size=<size>]

Add a meter, the type=dscp_remark.
Example:

Without burst_size. The prec_level=14.

```
root@PicOS-OVS$ovs-ofctl add-meter br0
meter=2,kbps,band=type=dscp_remark,rate=30000,prec_level=14
```

With burst_size. The prec_level=14.

```
root@PicOS-OVS$ovs-ofctl add-meter br0
meter=2,kbps,burst,band=type=dscp_remark,rate=30000,prec_level=14,burst_size=30
```

ovs-ofctl bundle <bridge> <bundle>

PicOS OVS support bundle from version 2.3.

User can add a bundle with format as following:

ovs-ofctl bundle <bridge>
"[open:bundle_id=<id>,]add:bundle_id=<id>,message=<message>[,add:bundle_id=<id>,mes

Using bundles is to group related state changes on a switch so that all changes are applied together or that none of them is applied. The second goal is to better synchronise changes across a set of OpenFlow switches, bundles can be prepared and pre-validated on each switch and applied at the same time.
PisOS OVS support applaying 10 bundles max at the same time. And the max massages is 100 of each bundle.
And support flow/group/meter-mods at present.

messag=add-flow <match><,actions>

User can add a flow by bundle.
Example:

```
admin@PicOS-OVS$ovs-ofctl bundle br0
"open:bundle_id=1,add:bundle_id=1,message=add-flow
in_port=1,dl_src=22:11:11:11:11:11,dl_dst=22:22:22:22:22:22,dl_type=0x0800,nw_s
priority=40000,in_port=2,actions=mod_dl_src=00:00:00:11:11:11,close:bundle_id=1
```

messag=mod-flows <match><,actions>

Modify flows by bundle.

messag=del-flows <match>

Delete flows by bundle.

Example:

Delete the flows that special match field.

```
admin@PicOS-OVS$ovs-ofctl bundle br0
"open:bundle_id=1,add:bundle_id=1,message=del-flows
in_port=1,commit:bundle_id=1"
```

Delete all flows in br0.

```
admin@PicOS-OVS$ovs-ofctl bundle br0
"open:bundle_id=1,add:bundle_id=1,message=del-flows,commit:bundle_id=1"
```

message=add-group group_id=<id>,type=<type>,bucket=<actions>[,bucket=<actions>]

Add groups by bundle.

Example:

```
admin@PicOS-OVS$ovs-ofctl bundle br0
"open:bundle_id=1,add:bundle_id=1,message=add-group
group_id=1,type=all,bucket=output:2,bucket=output:3,add:bundle_id=1,message=add
group_id=2,type=indirect,bucket=output:1,commit:bundle_id=1"
```

message=mod-group group_id=<id>,type=<type>,bucket=<actions>[,bucket=<actions>]

Modify a group by bundle.

messag=del-groups group_id=<id>

Delete groups by bundle.

Example:

```
admin@PicOS-OVS$ovs-ofctl bundle br0
"open:bundle_id=1,add:bundle_id=1,message=del-groups ,commit:bundle_id=1"
```

messag=add-meter meter=<id> kbps[,burst,stats],band=type=<type>,rate=<rate>[,burst_size=<siz

Add meter by bundle.

messag=mod-meter meter=<id> kbps[,burst,stats],band=type=<type>,rate=<rate>[,burst_size=<siz

Modify meter by bundle.

messag=del-meter meter=<id>

Delete meter that meter=<id>.

Multiple bundles

User can apply multiple bundles at the same time. Each bundle has its own open/add/close/commit/discard.
Example:

```
root@PicOS-OVS$ovs-ofctl bundle br-iris
"open:bundle_id=1,open:bundle_id=2,add:bundle_id=1,message=add-flow
dl_src=00:00:00:11:11:11,actions=output:3,add:bundle_id=2,message=add-flow
dl_src=00:00:00:11:11:22,actions=output:4,close:bundle_id=1,close:bundle_id=2,c
```

ovs-ofctl del-flows <bridge> <flow>

Delete the flow entries from flow table of *<bridge>*. If the *[flow]* is omitted, delete all flows in *<bridge>*, otherwise, will delete the matched flows.

ovs-ofctl del-group <bridge> [group_id=<id>]

Delete group in <bridge>. If the group_id is specified, the group that <group_id> in <bridge> will delete, otherwise, all groups in <bridge> will be cleared.
Example:

```
admin@PicOS-OVS$ovs-ofctl del-groups br0 group_id=2
admin@PicOS-OVS$ovs-ofctl del-groups br0
```

ovs-ofctl del-meter <bridge> meter=<id>

Delete the meter that meter=<id> in <bridge>.

ovs-ofctl del-meters <bridge>

Delete all the meters in <bridge>.

ovs-ofctl dump-desc <bridge>

Prints *<bridge>* description, including manufacturer, hardware, software, serial num and dp description.

ovs-ofctl dump-flows <bridge> <flow>

Prints flow.entries of <bridge>. With the *[flow]* is specified will only print the matching flow. If the *[flow]* is omitted, all flow entries of the bridge will be printed.

ovs-ofctl dump-ports <bridge> <port>

Prints the port statistics of *<bridge>*. If the port is specified, only the port statistics of the bridge will be printed, otherwise, will print all the ports statistics.

ovs-ofctl dump-ports-desc <bridge>

Prints port statistics. Will show detail information about these interfaces in this bridge, include the state,peer and speed information ,etc.The information print by this command is a subset of command 'ovs-ofctl show ***<bridge>'.***

ovs-ofctl dump-tables <bridge>

Prints *<bridge>* all tables stats. Including all 254 tables.

ovs-ofctl dump-tables-desc <bridge>

Print the descriptions of tables of <bridge>.

ovs-ofctl mod-flows <bridge> <flow>

Modify the actions of matching flows in the *<bridge>.*

ovs-ofctl mod-group <bridge> group_id=<id>,type=<type>,bucket=<actions>

Modify a group if the group entry with the specified group identifier already residesin the group table. If the group identifier does not exist, the group will add in group table successfully.
Example:

```
admin@PicOS-OVS$ovs-ofctl mod-group br0
group_id=100,type=indirect,bucket=mod_dl_src=00:00:00:99:11:11,mod_dl_dst=00:00
```

ovs-ofctl mod-meter <bridge> meter=<id>,<meter-parameter>

Modify a meter if the meter entry with the specified meter identifier already exists.

The format is shown as following:

ovs-ofctl mod-meter <bridge> meter=<id>,kbps[,burst,stats],band=type=<type>,rate=<rate>,burst_size=<size>,prec_level=<

ovs-ofctl modport <bridge> <iface> <action>

Modify behaviors of port *<iface>* in *switch*. *<iface>* can be an OpenFlow port number or name or the **LOCAL** port name. The *<action>* can be as the following:

ovs-ofctl mod-port <bridge> <iface> up

ovs-ofctl mod-port <bridge> <iface> down

Enable or disable the port link status.

ovs-ofctl mod-port <bridge> <iface> receive

ovs-ofctl mod-port <bridge> <iface> noreceive

Allow or disallow this interface receive traffic. By default, receiving is allowed.

ovs-ofctl mod-port <bridge> <iface> forward

ovs-ofctl mod-port <bridge> <iface> noforward

Allow or disallow traffic forwarding on this interface. By default, forwarding is allowed.

ovs-ofctl mod-port <bridge> <iface> flood

ovs-ofctl mod-port <bridge> <iface> noflood

Controls wether the interface to flood the received traffic or not. By default, flooding is enabled. Disabling flooding is primarily useful to prevent loops when a spanning tree protocol is not in use.

ovs-ofctl mod-table <bridge> <iface> <mod>

ovs-ofctl mod-table <bridge> <table> evict

Enable eviction on <table> of <bridge>. Eviction adds a mechanism enabling the switch to automatically eliminate entries of lower importance to make space for newer entries. This enables to smoother degradation of behaviour when the table is full.
If want enable eviction on all tables, user can set the <table> as 'all'.

ovs-ofctl mod-table <bridge> <table> vacancy:<range>

Configure the vacancy <range> on <table> of <bridge>. Vacancy event adds a mechanism enabling the controller to get an early warning based on a capacity threshold chosen by the controller. This allows the controller to react in advance and avoid getting the table full. If want configure vacancy range on all tables, user can set the <table> as 'all'.
The syntax of <range> as <low..high>.

ovs-ofctl mod-table <bridge> <table> clear

Clear the eviction or vacancy on <table> of <bridge>.If want clear eviction on all tables, user can set the <table> as 'all'.

ovs-ofctl monitor <bridge> [MISSLEN] [invalid_ttl] [watch:[...]]

Print packets received from <bridge>.

ovs-ofctl show <bridge>

Show OpenFlow information on <*bridge*>, including OpenFlow features ans port descriptions.

ovs-ofctl snoop <bridge>

Snoop on the <bridge> and its controller. Prints to the console all OpenFlow messages of <bridge> received.

Command "ovsvsctl"

NAME

ovs-vsctl: OpenFlow ovs-vswitchd management utility

SYNTAX

ovs-vsctl *[~~-OPTION~~] <COMMAND> [ARG...] [-[OPTION] <COMMAND> [ARGs]]…*
The ovs-vsctl is used primarily to manage the ovs-vswitchd. The ovs-vswitchd is connected with ovs-server which manages it to save and change the configuration into a database.

OPTION

Connect to *<database>* with active database connection methods:

ovs-vsctl –db=tcp:<ip>:<port> *<COMMAND>*

Connect to database by tcp.

Example:
Connect to database by tcp, ip 10.10.50.154, port 6640.

```
root@PicOS-OVS$ovs-vsctl --db=tcp:10.10.50.154:6640 add-br br0 - set bridge
br0 datapath_type=pica8
```

ovs-vsctl –db=ssl:<ip>:<port> *<COMMAND>*

ovs-vsctl –db= punix:*<FILE>*

If not specialed, the default connection is unix:/ovs/var/run/openvswitch/db.sock**.**

ovs-vsctl common commands

The commands contain 'open vSwitch commands' 'Bridge commands' 'Port commands' 'Interface commands' 'Controller commands' 'SSL commands' 'Switch commands' and 'Database commands'. The details as follows:

- Bridge commands
- Controller commands
- Database commands
- Interface commands
- Mirror commands
- NetFlow commands
- Open vSwitch commands
- Pica commands
- Port commands
- QoS_queue commands

- sFlow commands

Bridge commands

ovs-vsctl [-- OPTION] add-br <bridge> [--[OPTION] <COMMAND> [args]]

Create a new bridge named *<bridge>.* If a bridge named *<bridge>* already exists, the command will fail.
Example:

```
ovs-vsctl add-br br0 - set bridge br0 datapath_type=pica8
```

ovs-vsctl set bridge br0 protocols=openflow10,openflow12,openflow13,openflow14

Modify the Openflow version for bridge
Example:

```
ovs-vsctl -- set bridge br0 protocols=openflow10
```

ovs-vsctl del-br <bridge>

Delete the bridge named *<bridge>* and all of its configuration includes ports.
Example:

```
ovs-vsctl del-br br0
```

ovs-vsctl list-br

Print the names of all the bridges.

Controller commands

ovs-vsctl [--OPTION] set-controller <bridge> <target> [target]

Set the controllers for *<bridge>.* Support set multiple controllers at same time.
The bridge is typically configured to connect to multiple controllers. The controllers may select a primary controller that takes change of the flow tables of the bridge to inplement a network policy.
The <target> can use any of the following forms:

ovs-vsctl [--OPTION] set-controller <bridge> tcp: <ip> : <port>

Connect to controller with tcp port, the default value is 6633.
Example:

```
root@PicOS-OVS$ovs-vsctl set-controller br0 tcp:10.10.50.47:6633
```

ovs-vsctl [--OPTION] set-controller <bridge> ssl: <ip> : <port>

Connect to controller with specified SSL port, the default value is 6633.
Example:

```
root@PicOS-OVS$ovs-vsctl set-controller br0 ssl:10.10.50.100:6633
```

ovs-vsctl [--OPTION] get-controller <bridge>

Print the controllers for *<bridge>.*
Example:

```
ovs-vsctl get-controller br0
```

ovs-vsctl [--OPTION] del-controller <bridge>

Delete the controllers for *<bridge>.*
Example:

```
ovs-vsctl del-controller br0
```

ovs-vsctl [--OPTION] set-fail-mode <bridge> <mode>

Set the fail-mode for *<bridge>* to *<mode>*.
The *<mode>* used in control flow table when controller failure setting. Support **standalone** and **secure** modes, the default mode is **secure** mode.
If set the mode is standalone, ovs-vswitchd will take over responsibility for setting up flows when connect to controller fail, and all flows in bridge will clear and there is a flow named "normal" to let the switch work as a L2 switch ; If fail-mode set to secure, ovs-vswitchd will not set up flows nor clear all flows. In secure mode, the packet is dropped default, in other word, there is no flow named "normal" to let switch work as a L2 switch
Example:

```
ovs-vsctl set-fail-mode br0 standalone
```

ovs-vsctl [--OPTION] get-fail-mode <bridge>

Print the fail-mode for *<bridge>*. If not set the fail-mode, will print nothing.
Example:

```
ovs-vsctl get-fail-mode br0
```

ovs-vsctl [--OPTION] del-fail-mode <bridge>

Delete the fail-mode for *<bridge>*. After delete current mode will return the default mode standalone.
Example:

```
ovs-vsctl del-fail-mode br0
```

Database commands

Database commands create, list and modify the contents of ovsdb tables.PicOS OVS support the following tables, as:

Bridge table

Configure a bridge within an Open vSwitch. Record bridge configurations by _uuid in bridge tables.

Controller table

Configure an OpenFlow controller, record controller infos by _uuid in controller table.

Interface table

Configure a network device attached to a port, record interface configurations by _uuid in interface table.

Mirror table

Configure a mirror port to a bridge, record mirror configurations by _uuid in mirror table.

Open_vSwitch table

The global configurations for ovs-vswitchd, record configurations by _uuid.

Pica8 table

Pica_match_mode table

Port table

Configure bridge ports, record port configurations by _uuid in port table.

QoS table

Configure quality-of-service for a port, record QoS configurations by _uuid in QoS table.

Queue table

Configure one queue within a QoS, record by _uuid in queue table.

SSL table

sFlow table

Configure an sFlow exporter attached to a bridge, record by _uuid in sFlow table.

NetFlow table

Configure a NetFlow attached to a bridge, records by _uuid in NetFlow table.

ovs-vsctl [--OPTION] list <table> [record]

List the *[record]* record if special *[record]*, otherwise, list all records in *<table>.*
The *[record]* mean _uuid or special name
Example:

```
admin@PicOS-OVS$ovs-vsctl list bridge
admin@PicOS-OVS$ovs-vsctl list port br0
admin@PicOS-OVS$ovs-vsctl list port ge-1/1/1
```

ovs-vsctl [--OPTION] find <table> <condition> [condition…]

List records satisfying *<condition>* in *<table>.*
The *<condition>* represent a column equels value or not. If the value is specified, the operations can be support: "=", "!=", "<", ">", "<=", ">=", "\{=\}", "\{!=}", "{<}", "{>}", "{<=}" and "{>=}".
Example:

```
ovs-vsctl find bridge datapath_id=5e3e089e01616580 name=br0
```

ovs-vsctl [--OPTION] get <table> <record > <column> [:key]

Print values of column in *<record>* in *<table>*. *<table>* mean table name, and *<record>* mean _uuid.
Example:(print the value of datapath_id in the _uuid=80984dfb-5f63-45c8-bd3f-6bda918ffc75 in bridge table.)

```
ovs-vsctl get bridge 80984dfb-5f63-45c8-bd3f-6bda918ffc75 datapath_id
```

ovs-vsctl [--OPTION] set <table> <record > <column> [:key] <=value>

Set or change the column values in *<record>* in *<table>.*
Example: (change the value of trunks in ge-1/1/2 in port table.)

```
root@PicOS-OVS$ovs-vsctl list port ge-1/1/2
_uuid : 6e7b862a-6503-4f1b-982d-b33b65e3dc01
```

```
bond_downdelay : 0
bond_fake_iface : false
bond_mode : []
bond_updelay : 0
external_ids : {}
fake_bridge : false
interfaces : [f1b5568f-bd6f-480d-a9ca-636486f387c2]
lacp : []
mac : []
name : "ge-1/1/2"
other_config : {}
qos : []
statistics : {}
status : {}
tag : 1
trunks : [2000, 4094]
root@PicOS-OVS$ovs-vsctl set port 6e7b862a-6503-4f1b-982d-b33b65e3dc01
trunks=100,200
root@PicOS-OVS$ovs-vsctl list port ge-1/1/2
_uuid : 6e7b862a-6503-4f1b-982d-b33b65e3dc01
bond_downdelay : 0
bond_fake_iface : false
bond_mode : []
bond_updelay : 0
external_ids : {}
fake_bridge : false
interfaces : [f1b5568f-bd6f-480d-a9ca-636486f387c2]
lacp : []
mac : []
name : "ge-1/1/2"
other_config : {}
qos : []
statistics : {}
status : {}
tag : 1
trunks : [100, 200]
vlan_mode : trunk
```

ovs-vsctl [--OPTION] destroy <table> <record>

Deletes<record> from <table>.

ovs-vsctl [--OPTION] add <table> <record > <column> [key=] <value>

Add the column values in *<record>* in *<table>.*

Example: (add the value of trunks in ge-1/1/2 in port table.)

```
root@PicOS-OVS$ovs-vsctl get port ge-1/1/2 trunks
[100, 200]
root@PicOS-OVS$
root@PicOS-OVS$ovs-vsctl add port ge-1/1/2 trunks 100,300
root@PicOS-OVS$ovs-vsctl get port ge-1/1/2 trunks
[100, 200, 300]
```

ovs-vsctl [--OPTION] remove <table> <record > <column> [key=] <value>

Remove the column value in *<record>* in *<table>.*

Example: (remove the value of trunks in ge-1/1/2 in port table.)

```
root@PicOS-OVS$ovs-vsctl get port ge-1/1/2 trunks
[100, 200, 300]
root@PicOS-OVS$ovs-vsctl remove port ge-1/1/2 trunks trunks 100
root@PicOS-OVS$
root@PicOS-OVS$ovs-vsctl get port ge-1/1/2 trunks
[200, 300]
```

ovs-vsctl [--OPTION] clear <table> <record > <column>

Clear all values from column in <record> in <table>.

Example:

Clear all values of trunks in ge-1/1/2 in port table.

```
root@PicOS-OVS$ovs-vsctl get port ge-1/1/2 trunks
[200, 300]
root@PicOS-OVS$ovs-vsctl clear port ge-1/1/2 trunks
[]
root@PicOS-OVS$
```

Interface commands

ovs-vsctl [--OPTION] list-ifaces <bridge>

Print the names of all interfaces on *<bridge>*.

Example:

```
ovs-vsctl list-ifaces br0
```

ovs-vsctl [--OPTION] iface-to-br <interface>

Print the name of bridge that contains *<port>*.

Example:

```
ovs-vsctl iface-to-br ge-1/1/1
```

Mirror commands

ovs-vsctl [--OPTION] -- set bridge <bridge> mirrors=@m -- --id=@<port1> get Port <port1> -- --id=@<port2> get Port <port2> [-- --id=@<port3> get Port <port3>]-- --id=@m create Mirror name=<mirror-name> select-src-port=@<port1>[,@<port3>] select-dst-port=@<port1>[,@<port3>] output-port=@<port2>

PicOS OVS supports mirroring, **select-src-port** and **select-dst-port** represent the source ports of mirroring, **select-dst-port** means some packets (in switch chip) will go-out from the specified port (egress); **select-src-port** means some packets will enter the specified port (ingress); **output_port** means the monitor port.

Example:

Add port ge-1/1/1, ge-1/1/2 and ge-1/1/3 to mirror, ge-1/1/1 and ge-1/1/2 as ingress and egress, the output port is ge-1/1/3.

```
root@PicOS-OVS$ovs-vsctl - set bridge br0 mirrors=@m - --id=@ge-1/1/1 get
Port ge-1/1/1 - --id=@ge-1/1/2 get Port ge-1/1/2 - --id=@ge-1/1/3 get Port
ge-1/1/3 - --id=@m create Mirror name=mymirror
select-src-port=@ge-1/1/1,@ge-1/1/2 select-dst-port=@ge-1/1/1,@ge-1/1/2
output-port=@ge-1/1/3
```

ovs-vsctl [--OPTION] destroy <table> <record> – clear Bridge <bridge> mirrors

Example:

Delete a mirror named mymirror from mirror table in bridge br0.

```
admin@PicOS-OVS$ ovs-vsctl destroy Mirror mymirror - clear Bridge br0 mirrors
```

NetFlow commands

ovs-vsctl – set Bridge <bridge> netflow=@nf – --id=@nf create NetFlow targets= <target> active-timeout= <timeout>

Example:

```
admin@PicOS-OVS$ovs-vsctl - set Bridge br0 netflow=@nf - --id=@nf create
NetFlow targets=\"10.10.50.207:5566\" active-timeout=30
In the above CLI, the parametersare shown as following:
COLLECTOR_IP=10.10.50.207
```

```
COLLECTOR_PORT=5566
ACTIVE_TIMEOUT=30
```

ovs-vsctl – clear Bridge <bridge> netflow

Delete NetFlow from NetFlow table.

Example:

Clear NetFlow from NetFlow table in bridge br0.

```
admin@PicOS-OVS$ovs-vsctl - clear Bridge br0 netflow
```

ovs-vsctl init (the command is not effect)

To initialize the Open vSwitch database if not yet initialized. If the database has already been initialized, the init command will have no effect.

ovs-vsctl show

Print a brief overview of database contents.

ovs-vsctl emer-reset

Reset the configurarion to clean state. Using this command can delete fail mode, OpenFLow controllers, port mirrors, NetFlow, sFlow, and IPFIX configuration.

Pica commands

ovs-vsctl [--OPTION] set-match-mode <mode:options=priority> [mode:options=priority]

By default, 2 TCAM entries are used to support all matching tuples for all flows even the flow does not use all matching tuples. To optimize the TCAM usage, PicOS-2.1 allows user to configure the switch in short flow TCAM match mode, in which, each flow will only consume 1 TCAM entry. To use this feature, the flow must use the exact fields described below and cannot mix fields from various modes:
mac mode: "in_port, dl_src, dl_dst, vlan_vid, dl_type"
ip mode: "in_port, nw_proto, nw_src, nw_dst, dl_type=0x0800"
arp_tpa mode: "in_port, arp_tpa, dl_type=0x0806"
For example, if mac mode is enabled, all the flows must only use one or more fields defined in the mac mode. If mac and ip modes are enabled, then you can configure either mac flows or ip flows based on the fields described above. However, you cannot mix the fields from mac and ip (that is, dl_src and nw_src).
Each mode is configured with a priority range that determines the flow priority. The flow priority

must be specified when user configure the flow through ovs commands or controller.
If user do not want match-mode, can configure match-mode as default. The default value is default.

Example:

Configure match-mode mac priority from 200 to 499; ip priority 500-999; arp_tpa priority 1000-2000.

```
root@PicOS-OVS$ovs-vsctl set-match-mode
mac=200-499,ip=500-999,arp_tpa=1000-2000
```

Remove match-mode configuration.

```
root@PicOS-OVS$ ovs-vsctl set-match-mode default
```

ovs-vsctl show-match-mode

Print current match-mode.

Example:

```
root@PicOS-OVS$ovs-vsctl show-match-mode
current match modes:
mac:
priorities={200-499}
fields={in_port,dl_src,dl_dst,dl_vlan,dl_type}
ip:
priorities={500-999}
fields={in_port,nw_proto,nw_src,nw_dst,tp_src,tp_dst,dl_type=0x0800}
arp_tpa(both):
priorities={1000-2000}
fields={in_port,arp_tpa,dl_type=0x0806}
root@PicOS-OVS$
```

ovs-vsctl set-cos-map <TRUE|FALSE>

From PicOS-2.1, user can enable/disable the cos-mapping, the default value is disabled. Once enable cos-mapping, the packet with the different cos value mapped to queue0-7. With DSCP(0-7) mapped to queue 0; with DSCP(8-15) mapped to queue 1; b y Such Analogy, with DSCP(56-63) mapped to queue 7. If configure a flow with actions: **set_queue**, the default cos-mapping will be replaced.

Example:

Enable cos-mapping.

```
admin@PicOS-OVS$ovs-vsctl set-cos-map true
```

ovs-vsctl show-cos-map

Display the configuration of cos-mapping.
Example:

```
admin@PicOS-OVS$ovs-vsctl show-cos-map
cos mapping:
{
0x00--0x07: 0
0x08--0x0f: 1
0x10--0x17: 2
0x18--0x1f: 3
0x20--0x27: 4
0x28--0x2f: 5
0x30--0x37: 6
0x38--0x3f: 7
}
```

ovs-vsctl enable-egress-mode <TRUE|FALSE>

Enable or disable egress-mode.

Use table 253 to support another 256 flow, the action of flow should be drop or mod_nw_tos.

eg.

ovs-ofctl add-flow br0 table=253,in_port=1,actions=drop
ovs-ofctl add-flow br0 table=253,dl_type=0x0800,nw_dst=192.168.1.3,actions=mod_nw_tos:16

ovs-vsctl show-egress-mode

Show the current configure egress-mode.

ovs-vsctl set-combinated-mode <TRUE|FALSE>

Enable combinated mode to deal with VCAP/ICAP flow conflict, only half of TCAM flow number can be used

ovs-vsctl show-combinated-mode

Show whether combinated-mode is enable or not

Port commands

ovs-vsctl [--OPTION] add-port <bridge> <port> [ARG...] [--[OPTION] <COMMAND> [ARGs]]

Add a new port named *<port>* to *<bridge>*. Including physic port, LAG port and GRE tunnel.

Physic port

Example:

Add a physical port.

```
ovs-vsctl add-port br0 ge-1/1/1 vlan_mode=trunk tag=1 trunks=2000,4094 - set
interface ge-1/1/1 type=pica8
```

Add a physical port and configure special link_speed.

```
ovs-vsctl add-port br0 te-1/1/49 vlan_mode=trunk tag=1 trunks=2000,4094 - set
interface te-1/1/49  type=pica8 options:link_speed=1G
```

LAG port

Example:

Add a lag interface:

```
root@PicOS-OVS#ovs-vsctl add-port br0 ae1 vlan_mode=trunk tag=1
trunks=2000,4094 - set Interface ae1 type=pica8_lag options:members=ge-1/1/1
root@PicOS-OVS#ovs-vsctl - set Interface ae1 options:lag_type=static
```

Modify the mumbers of the lag

```
root@PicOS-OVS#ovs-vsctl set Interface ae1 options:members=ge-1/1/2,ge-1/1/3
```

LACP port

Example:

Add a lacp port and configure the parameter

```
root@PicOS-OVS#ovs-vsctl add-port br0 ae1 vlan_mode=trunk tag=1
trunks=2000,4094 - set Interface ae1 type=pica8_lag
root@PicOS-OVS#ovs-vsctl - set Interface ae1 options:lag_type=lacp
root@PicOS-OVS#ovs-vsctl set Interface ae1 options:members=ge-1/1/2,ge-1/1/3
root@PicOS-OVS#ovs-vsctl - set Interface ae1
options:lacp-system-id=00:11:11:11:11:11
root@PicOS-OVS#ovs-vsctl - set Interface ae1
options:lacp-system-priority=32768
root@PicOS-OVS#ovs-vsctl - set Interface ae1 options:lacp-time=fast
root@PicOS-OVS#ovs-vsctl - set Interface ae1 options:lacp-time=slow
root@PicOS-OVS#ovs-vsctl - set Interface ae1 options:lacp-mode=active
root@PicOS-OVS#ovs-vsctl - set Interface ae1 options:lacp-mode=passive
root@PicOS-OVS#ovs-vsctl - set Interface ge-1/1/2 options:lacp-port-id=2
root@PicOS-OVS#ovs-vsctl - set Interface ge-1/1/2
options:lacp-port-priority=32768
root@PicOS-OVS#ovs-vsctl - set Interface ge-1/1/2
options:lacp-aggregation-key=0
```

GRE port

Example:

Add a GRE port.

```
ovs-vsctl add-port br0 gre1 - set Interface gre1 type=pica8_gre
options:remote_ip=10.10.61.10 options:local_ip=10.10.60.10 options:vlan=1012
options:src_mac=C8:0A:A9:9E:14:A5 options:dst_mac=00:E0:0C:00:00:FD
options:egress_port=ge-1/1/2
```

ovs-vsctl [--OPTION] list-ports <bridge>

Print the names of all the ports on *<bridge>.*

Example:

```
ovs-vsctl list-ports br0
```

ovs-vsctl [--OPTION] del-port <bridge> <port>

Delete the port named *<port>* from *<bridge>.*

Example:

```
ovs-vsctl del-port br1 ge-1/1/1
```

ovs-vsctl [--OPTION] port-to-br <port>

Print name of the bridge which contrains the special *<port>*.

Example:

```
ovs-vsctl port-to-br ge-1/1/1
```

VXLAN port

Example:

Add a VXLAN port.

```
admin@PicOS-OVS$ovs-vsctl add-port br0 vxlan1 – set interface vxlan1
type=pica8_vxlan options:remote_ip=10.10.10.2 options:local_ip=10.10.10.1
options:vlan=1 options:vnid=1122867 options:udp_dst_port=4789
options:src_mac=C8:0A:A9:04:49:1A  options:dst_mac=C8:0A:A9:9E:14:A5
options:egress_port=qe-1/1/2
```

L2GRE port

Example:

Add a L2GRE port.

```
admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre1 – set interface l2gre1
type=pica8_l2gre options:remote_ip=10.10.61.10 options:local_ip=10.10.60.10
options:vlan=1 options:l2gre_key=1234 options:src_mac=C8:0A:A9:49:1A
options:dst_mac=C8:0A:A9:9E:14:A5 options:egress_port=qe-1/1/2
```

admin@PicOS-OVS$ovs-vsctl add-port br0 l2gre1 – set interface l2gre1 type=pica8_l2gre options:remote_ip=10.10.61.10 options:local_ip=10.10.60.10 options:vlan=1 options:l2gre_key=1234 options:src_mac=C8:0A:A9:49:1A options:dst_mac=C8:0A:A9:9E:14:A5 options:egress_port=qe-1/1/2

QoS_queue commands

ovs-vsctl [--OPTION] -- set port <port> qos=@newqos -- --id=@newqos create qos type=PRONTO_STRICT queues:<queueid>=@newqueue [queues:<queueid>=@newqueue1] -- --id=@newqueue create queue other-config:min-rate= <minrate> other-config:max-rate= <maxrate> [-- --id=@newqueue1 create queue other-config:min-rate= <minrate> other-config:max-rate= <maxrate>]

Queue 0~7 represent priority 0~7 resprectively.

Example:

Configure qos contain two queues: queue 0 and queue 7. And min and max rate of queue 0 and queue 7 is set as 10M.

```
admin@PicOS-OVS$ovs-vsctl - set port ge-1/1/3 qos=@newqos - --id=@newqos
create qos type=PRONTO_STRICT queues:0=@newqueue queues:7=@newqueue1 -
--id=@newqueue create queue other-config:min-rate=100000000
other-config:max-rate=100000000 - --id=@newqueue1 create queue
other-config:min-rate=100000000 other-config:max-rate=100000000
```

ovs-vsctl [--OPTION] clear port <port> qos

Delete QoS applied *<port>*.

Example:

Clear QoS and queues applied port ge-1/1/3 from Qos table and queue table.

```
admin@PicOS-OVS$ovs-vsctl clear port ge-1/1/3 qos
admin@PicOS-OVS$ovs-vsctl - --all destroy qos
admin@PicOS-OVS$ovs-vsctl - --all destroy queue
```

sFlow commands

ovs-vsctl -- --id=@s create sFlow agent= <agent> target= <target> header= <header> samping= <samping> polling= <polling> -- set Bridge <bridge> sflow=@s

PicOS OVS supports sFlow v5.

Example:

```
admin@PicOS-OVS$ovs-vsctl -- --id=@s create sFlow agent=eth0
target=\"10.10.50.207:9901\" header=128 sampling=2000 polling=30 -- set
Bridge br0 sflow=@s
In the above CLI, the parameters are shown as following:
COLLECTOR_IP=10.10.50.207
COLLECTOR_PORT=9901
AGENT_IP=eth0
HEADER_BYTES=128
SAMPLING_N=64
POLLING_SECS=10
```

ovs-vsctl – clear Bridge <bridge> sflow

Delete sFlow from sFlow table.

Example:

Clear sFlow from sFlow table in bridge br0.

```
admin@PicOS-OVS$ovs-vsctl -- clear Bridge br0 sflow
```

Troubleshoot the configuration

```
admin@5712$ovs-vsctl list sflow
_uuid               : 3362a543-4a1a-47db-898a-0ec0eddb7aa0
agent               : "eth0"
external_ids        : {}
header              : 128
polling             : 30
sampling            : 2000
targets             : ["172.16.0.173:8008"]
```