Dell Networking Operating System for OpenFlow on N-Series (DNOS-OF)

User Guide v1.0f

Dell Networking Engineering (Campus)
August 2015

NOTE: A NOTE indicates important information that helps you make better use of your computer.
CAUTION: A CAUTION indicates either potential damage to hardware or loss of data and tells you how to avoid the problem.
WARNING: A WARNING indicates a potential for property damage, personal injury, or death.

2015–03, Rev. A02

Table of contents

1 Executive summary – DNOS-OF

SDN and OpenFlow is fast becoming a requirement in campus networking products due to the perceived strategic value of SDN and the high perceived cost of proprietary legacy network gear. DNOS-OF is a campus networking product based on existing N Series hardware and a custom firmware image.

1.1 DNOS-OF Overview

DNOS-OF is a web downloadable firmware image available for the Dell Networkinging N-Series hardware that enables OpenFlow 1.3.4 support as a pure OpenFlow switch. It is intended to:

- Provide basic easy to use pure OpenFlow mode support on N-Series switches to enable SDN for Campus networks, no hybrid mode is supported.
- Co-exist with the existing N-Series firmware images and image management, while not affecting any existing functionality. This requires the ability to load DNOS-OF code from within the users existing firmware, run as a pure OF switch, and revert back with no impact to the users existing firmware, including their running configuration and switch settings.
- Leverage Broadcom's OFDPA (OpenFlow Data Path Abstraction) SDK to provide basic OpenFlow agent integration, along with abstracting the SOC hardware tables when presenting them as OpenFlow flow tables.
- Network administrators are able to select the OS for the switch in the same manner they select which firmware image they want to run today.
- Provide limited and simpler features and functionality in order to allow for delivery of a quick and low cost solution that is easy to test out in customer lab environments.

1.2 About This Document

This guide describes the product and its purpose, how to configure, monitor, and maintain DNOS-OF on the Dell Networking N-Series switches, a reference for the DNOS-OF command-line interface (CLI) and some basic examples showing how to set up a single end to end Layer 2 traffic flow in DNOS-OF.

1.3 Additional Documentation

Documents for the Dell Networking series switches are available at **dell.com/support**.

2 User Guide

2.1 Embedded Management (CLI/GUI)

There is a CLI provided by the DNOS-OF platform that is accessible from the serial port, telnet, and SSH. There is currently no other management access. The CLI reference is included in an appendix at the end of this document.

2.2 Supported Hardware

The DNOS-OF firmware is supported on the following N-Series platforms as of release 1.0:

- 3024
- 3024P
- 3048
- 3048P

The current plan is for support of all N-Series hardware in following releases.

2.3 Limitations and Product Constraints

- Dell Networking N-Series switches with the DNOS-OF firmware installed are pure OpenFlow only switches. No legacy functions are available
- Only 1 SDN agent is supported per physical switch, no hybrid mode, and a CLI for a user interface.
- Only a single OpenFlow instance is supported, which includes all physical ports.
- The OpenFlow 1.3.4 spec is the only initial mode of compatibility supported, there is no backwards compatibility with prior versions of the OpenFlow spec.
- DNOS-OF is quallified with the Ryu controller. This is in alignment with DNOS-9.x and their OpenFlow / SDN mode release compatibility and testing.
- All OpenFlow 1.3.4 commands that are listed as “mandatory” in the spec are supported except for those having to do with hybrid mode. Some OpenFlow 1.3.4 commands that are “optional” but that enhance the product serviceability and value are supported as well.
- Stacking of DNOS-OF OpenFlow switches using standard Higig protocol is not supported.
- Basic SSH is initially provided in 1.0, with enhanced encryption and TLS in following releases.
- Packet buffering is not supported.
- Except for minimal system internal logging and output to the serial console, only remote logging being supported so as not to impact the existing code.

3 Product Features – SDN/OpenFlow and DNOS-OF

3.1 Overview – What is SDN?

The physical separation of the network control plane from the forwarding plane where a control plane controls several devices. Software-Defined Networking (SDN) is a networking architecture that is dynamic, manageable, and adaptable, making it useful for high-bandwidth dynamic applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. The OpenFlow™ protocol is a foundational element for building SDN solutions.

Some attributes of SDN architecture are:

- Directly programmable: Network control is directly programmable because it is decoupled from forwarding functions.
- Agile: Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
- Centrally managed: Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
- Programmatically configured: SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.
- Open standards-based and vendor-neutral: When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multple, vendor-specific devices and protocols.

3.2 Overview – What is OpenFlow?

The OpenFlow protocol is one instance of an SDN architecture, based on a set of specifications maintained by the Open Networking Forum (ONF). At the core of the specifications is a definition of an abstract packet processing machine, called a switch. The switch processes packets using a combination of packet contents and switch configuration state. A protocol is defined for manipulating the switch's configuration state as well as receiving certain switch events. Finally, a controller is an element that speaks the protocol to manage the configuration state of many switches and respond to events.

More Information on the verview and genesis, current state of protocol: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

OpenFlow from Flowgrammable website: http://flowgrammable.org/

OpenFlow from Open Networking Spec website: https://www.opennetworking.org/sdn-resources/openflow

3.3 Overview – What is DNOS-OF?

As a product line, the Dell Networking Operating System – for OpenFlow, DNOS-OF is a firmware bundle that allows a traditional N series switch to be used as a pure OpenFlow switch. DNOS-OF is designed to

1. Deliver a pure OpenFlow switch for the N series
2. Enable SDN in campus networks
3. Interoperate with any controller supporting OpenFlow 1.3.4 and multiple table support.

DNOS-OF leverages the BigSwitch Networks open source Indigo agent (Indigo) and Broadcom's OpenFlow Data Plane Abstraction (OF-DPA) packages to provide OpenFlow support.

The DNOS-OF based abstract switch is a specialization of the OpenFlow 1.3.4 OFLS (OpenFlow logical switch). Section 4.3.1 below describes the DNOS-OF Abstract Switch in terms of abstract objects that are visible to the OpenFlow controller.

The DNOS-OF abstract switch objects can be thought of as programming points for the Ethernet switching hardware. These include flow tables with action sets, group table entries, physical ports, and queues. The DNOS-OF adaptation layer provides support for OpenFlow specific state, for example, statistics counters. It also maps OpenFlow objects to hardware and manages hardware resources. Supporting OpenFlow in switch hardware involves some tradeoffs. As has been noted elsewhere, the generality promised by OpenFlow can come at a cost of latency, as well as cost and power inefficiencies. In addition, to effectively use this generality a specific multi-table pipeline first needs to be designed and configured. The DNOS-OF Abstract Switch may be viewed as coming preconfigured and optimized to support single pass, full bandwidth packet processing performance that makes efficient use of the hardware and available table memory resources, trading off unrestricted generality in favor of latency, performance, and cost, while enabling a logically centralized control plane with programming flexibility.

The DNOS-OF Abstract Switch includes functionality to support bridging and routing functionality on the switch chip, among others. These flows expose the proper functionality in the switching hardware. Future versions of DNOS-OF are expected to support additional features and packet flow use cases.

DNOS-OF implements a basic SDK switch OS with primary functionality being an SDN agent, a CLI and platform support for the various N-Series hardware components.

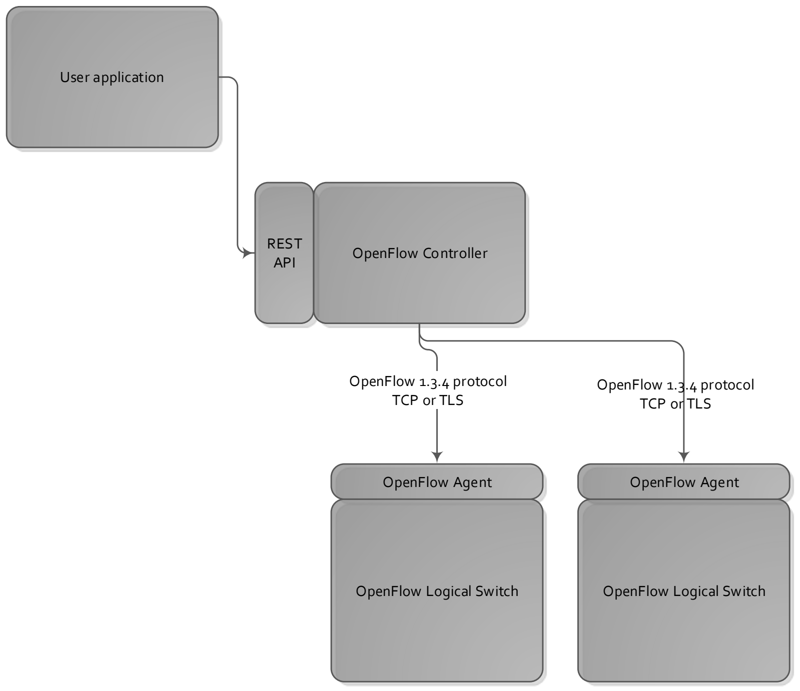
4 Product Details

4.1 Details SDN and the OpenFlow Architecture

4.1.1 SDN/OpenFlow High Level Architectural Components

At the core of the OpenFlow specifications is the definition of an abstract packet processing machine, called a switch. The switch processes packets using a combination of packet contents and switch configuration state. A protocol is defined for manipulating the switch's configuration state as well as receiving certain switch events. Finally, a controller is an element that speaks the OpenFlow protocol down to the switch based agent in order to manage the configuration state of many switches and respond to switch events.

Controller Topology



The controllers provide flow programming instructions to agents running in the switches for setting up switch functions and tables that are normallu programmed to run on the legacy firmware on the switch itself, such as VLAN’s, ACL entries, routing and bridging.

4.2 DNOS-OF Product Details

4.2.1 Design

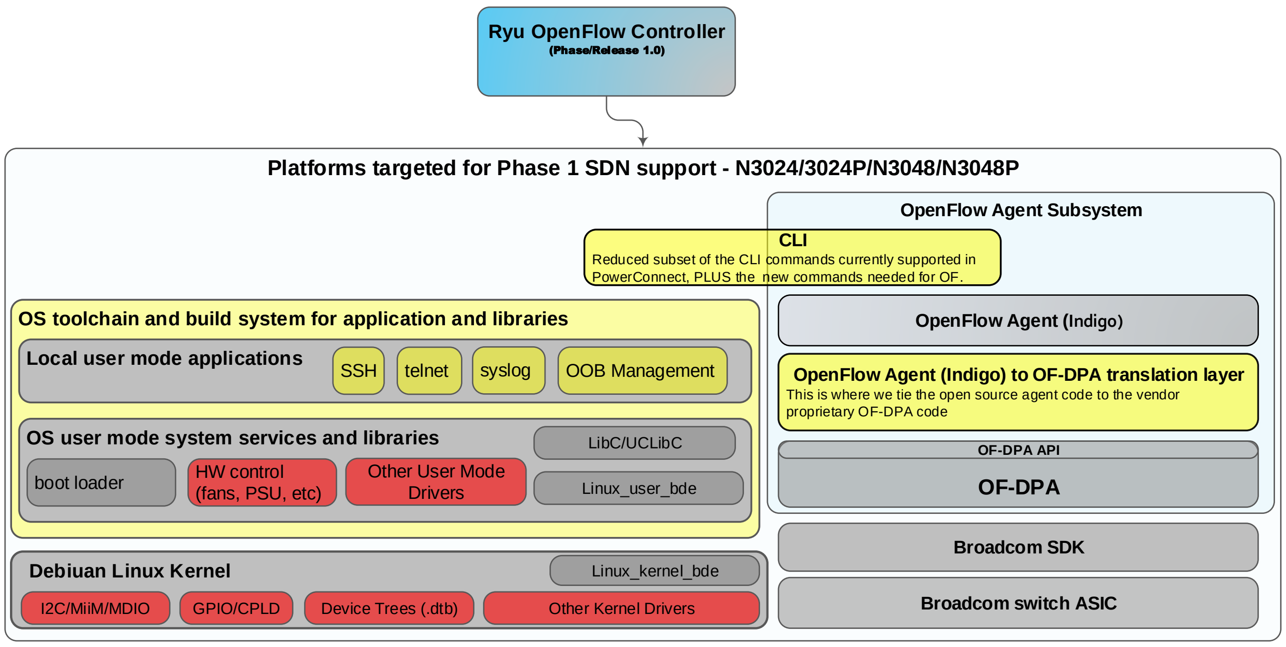
The network OS portion of DNOS-OF consists of modified Debian Linux kernel for the main underlying core of the platform operating environment, along with an additional kernel framework consisting of the Debian Linux user and kernel mode interface BDE drivers provided by the switch SDK.

There are also user mode drivers for all of the target hardware, which together make up the Board Support Package (BSP). There is SDN agent code to interface north bound to an SDN controller. There is code to implement and encapsulate SDN OpenFlow 1.3.4 capability specifically on the switch chips (the OF-DPA layer), as well as interface code between the open source agent and our agent. Additionally there is a CLI provided by the underlying platform code and developed inhouse by Dell Networking Engineering.

The DNOS-OF 1.0 firmware was initially targeted for the N3024 and N3048 N-series family of campus Ethernet switches from Dell. Later adding the PoE versions, N3024P and N3048P. With the use of Debian Linux, and built from a modular architecture, all of the DNOS-OF components and code have been designed and built to be 100% compatible with industry standard architectures and approaches.

Below are the primary components that make up the DNOS-OF firmware architecture in release 1.0.

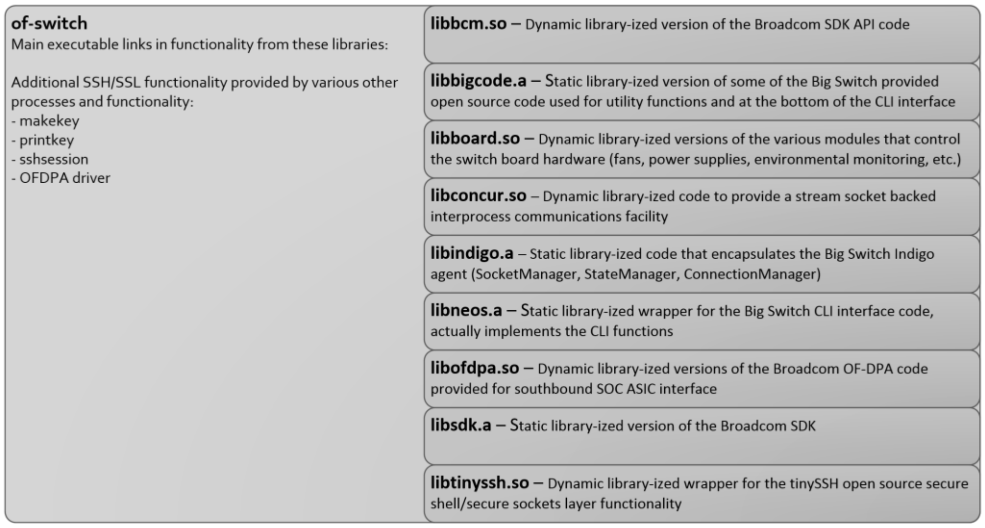
Campus N Series OpenFlow Switch – Phase 1 High Level Firmware Overview


Ryu OpenFlow Controller
(Phase/Release 1.0)
Platforms targeted for Phase 1 SDN support - N3024/3024P/N3048/N3048P
OpenFlow Agent Subsystem
CLI
Reduced subset of the CLI commands currently supported in PowerConnect, PLUS the new commands needed for OF.
OS toolchain and build system for application and libraries
Local user mode applications
SSH
telnet
syslog
OOB Management
OS user mode system services and libraries
LibC/UCLibC
boot loader
HW control (fans, PSU, etc)
Other User Mode Drivers
Linux_user_bde
Debiuan Linux Kernel
Linux_kernel_bde
I2C/MiiM/MDIO
GPIO/CPLD
Device Trees (.dtb)
Other Kernel Drivers
OpenFlow Agent (Indigo)
OpenFlow Agent (Indigo) to OF-DPA translation layer
This is where we tie the open source agent code to the vendor proprietary OF-DPA code
OF-DPA API
OF-DPA
Broadcom SDK
Broadcom switch ASIC

Note that the OpenFlow controller shown here were the initial ones targeted, but any controller that is OpenFlow 1.3.4 multi-table compliant should work with DNOS-OF. This is mainly dependent on the ability of the specific controller software to work with 1) OpenFlow 1.3.4, 2) multiple tables presented to the controller, and 3) some knowledge and support for the limitations of the hardware SOC based flow tables.

4.2.2 Module Architecture

While the primary application architecture, build system and OS infrastructure were designed and built in house at Dell, the product uses several open source and vendor packages, all tied together with Dell proprietary glue code, as shown below:



of-switch
Main executable links in functionality from these libraries:

Additional SSH/SSL functionality provided by various other processes and functionality:
- makekey
- printkey
- sshsession
- OFDPA driver

libbcm.so – Dynamic library-ized version of the Broadcom SDK API code

libbigcode.a – Static library-ized version of some of the Big Switch provided open source code used for utility functions and at the bottom of the CLI interface

libboard.so – Dynamic library-ized versions of the various modules that control the switch board hardware (fans, power supplies, environmental monitoring, etc.)

libconcur.so – Dynamic library-ized code to provide a stream socket backed interprocess communications facility

libindigo.a – Static library-ized code that encapsulates the Big Switch Indigo agent (SocketManager, StateManager, ConnectionManager)

libneos.a – Static library-ized wrapper for the Big Switch CLI interface code, actually implements the CLI functions

libofdpa.so – Dynamic library-ized versions of the Broadcom OF-DPA code provided for southbound SOC ASIC interface

libsdk.a – Static library-ized version of the Broadcom SDK

libtinyssh.so – Dynamic library-ized wrapper for the tinySSH open source secure shell/secure sockets layer functionality

4.2.3 Kernel

The DNOS-OF software architecture consists of a standard kernel.org Debian Linux kernel for the main underlying core of the platform operating environment, along with a kernel framework consisting of the Linux user and kernel mode interface BDE drivers provided by the switch SDK. This overall wrapper is layered around a Debian Linux distribution.

4.2.4 Platform

There are several user mode drivers that provide communications and control of the target hardware, which together make up the Board Support Package (BSP).   All user manageable or visible HW on the product are accessible via these user mode platform drivers.

4.2.5 Indigo

Indigo is an open source Big Switch Networks provided north bound OpenFlow API layer which has been tied into the OF-DPA libraries and consequently into the of-switch application. It handles communications from the OpenFlow controller to the OpenFlow agent in the DNOS-OF switch.

4.2.6 of-switch Main Application

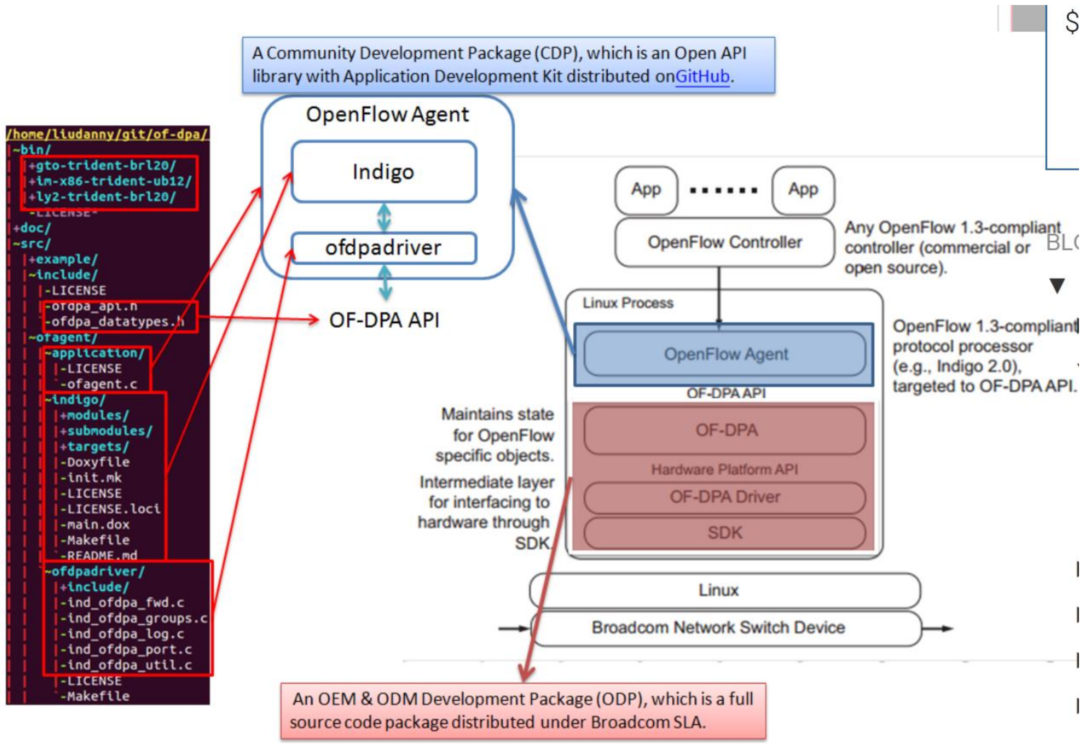
The SDN agent code to interface north bound to an SDN controller, the CLI, the platform monitoring, the overall initiation and control of the process and various tasks takes place in the of-switch application.
Embedded Management (CLI/GUI)

The CLI is provided by the underlying DNOS-OF platform layer, and is ciurrently accessible through the serial console as well as telnet and SSH. Support for a REST API based GUI, SNMP MIB’s, and various other management approaches are planned for later releases.

4.2.7 OF-DPA SDK

The OpenFlow Data Plane Abstraction SDK, translates general OpenFlow protocol commands received and processed by the OpenFlow agent on an abstract or virtual switch into specific rules and tables as implemented on a Broadcom SOC product in order to establish flows on the hardware.

Here is a high level diagram of what is provided by OF-DPA.


```
/home/liudanny/git/of-dpa/
|~bin/
| |+gto-trident-brl20/
| |+im-x86-trident-ub12/
| |+ly2-trident-brl20/
|  -LICENSE-
|+doc/
|~src/
| |+example/
| |~include/
| | |-LICENSE
| | |-ofdpa_api.h
| | `-ofdpa_datatypes.h
| |~ofagent/
| | |~application/
| | | |-LICENSE
| | | `-ofagent.c
| | |~indigo/
| | | |+modules/
| | | |+submodules/
| | | |+targets/
| | | |-Doxyfile
| | | |-init.mk
| | | |-LICENSE
| | | |-LICENSE.loci
| | | |-main.dox
| | | |-Makefile
| | | `-README.md
| | `~ofdpadriver/
| |   |+include/
| |   |-ind_ofdpa_fwd.c
| |   |-ind_ofdpa_groups.c
| |   |-ind_ofdpa_log.c
| |   |-ind_ofdpa_port.c
| |   |-ind_ofdpa_util.c
| |   |-LICENSE
| |   `-Makefile
```
A Community Development Package (CDP), which is an Open API library with Application Development Kit distributed on GitHub.
OpenFlow Agent
Indigo
ofdpadriver
OF-DPA API
App
App
OpenFlow Controller
Any OpenFlow 1.3-compliant controller (commercial or open source).
Linux Process
OpenFlow Agent
OpenFlow 1.3-compliant protocol processor (e.g., Indigo 2.0), targeted to OF-DPA API.
OF-DPA API
Maintains state for OpenFlow specific objects.
OF-DPA
Hardware Platform API
Intermediate layer for interfacing to hardware through SDK.
OF-DPA Driver
SDK
Linux
Broadcom Network Switch Device
An OEM & ODM Development Package (ODP), which is a full source code package distributed under Broadcom SLA.

4.3 OpenFlow Table Programming supported by DNOS-OF

4.3.1 Bridging and Routing Functions

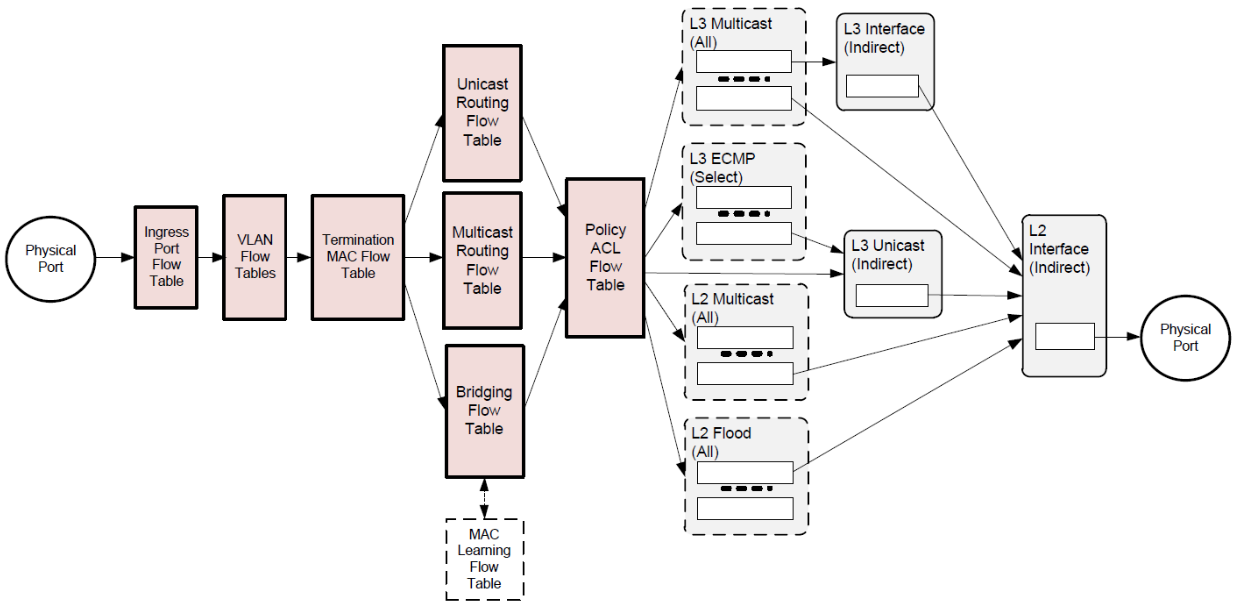


Figure 2. Abstract switch Objects Used for Bridging and Routing

The DNOS-OF Abstract Switch objects that can be programmed for bridging and routing are shown in Figure 2.

Packets enter and exit the pipeline on physical ports local to the switch. The Ingress Port Flow Table (table 0) is always the first table to process a packet. Flow entries in this table can distinguish traffic from different types of input ports by matching associated Tunnel Id metadata. Normal bridging and routing packets from physical ports have a Tunnel Id value of 0. To simplify programming, this table provides a default rule that passes through packets with Tunnel Id 0 that do not match any higher priority rules. Logical ports are not supported in DNOS-OF, so the Tunnel Id will always be 0.

All packets in the Bridging and Routing flow must have a VLAN. The VLAN Flow Table can do VLAN filtering for tagged packets and VLAN assignment for untagged packets. If the packet has more than one VLAN tag, the outermost VLAN Id is the one used for forwarding.

The Termination MAC Flow Table matches destination MAC addresses to determine whether to bridge or route the packet and, if routing, whether it is unicast or multicast. MAC learning is supported using a “virtual” flow table that is logically synchronized with the Bridging Flow Table.

When MAC learning is enabled, DNOS-OF does a lookup in the Bridging Flow Table using the source MAC, outermost VLAN Id, and IN_PORT. A miss is reported to the controller using a Packet In message. Logically this occurs before the Termination MAC Flow Table lookup. The MAC Learning Flow Table cannot be directly read or written by the controller. The MAC Learning Flow Table has a “virtual” table number which is reported to the Controller in a table miss Packet-In message. It does not appear as part of the pipeline since its table number assignment would violate the OpenFlow requirement for packets to traverse tables in monotonically increasing order.

The ACL Policy Flow Table can perform multi-field wildcard matches, analogous to the function of an ACL in a conventional switch.

DNOS-OF makes extensive use of OpenFlow Group entries, and most forwarding and packet edit actions are applied based on OpenFlow group entry buckets. Groups support capabilities that are awkward or inefficient to program in OpenFlow 1.0, such as multi-path and multicast forwarding, while taking advantage of functionality built into the hardware.

4.3.2 DNOS-OF Object Descriptions – Flow Tables and Group Tables

DNOS-OF presents the application writer with a set of objects that can be programmed using OpenFlow 1.3.4. The programmable objects include flow tables and group table entries.

This section provides programming descriptions for these objects. For details consult the DNOS-OF TTP (Table Type Patterns) supplied with the firmware.

Flow tables have specific attributes, including entry types (rules) that have specific match fields, actions, and instructions. Flow entries can have "Goto-Table" instructions that determine the next table to process the packet. In other words, the flow entry programming determines the order in which packets traverse tables and accumulate actions in an action set. Actions in the action set are applied prior to the packet being forwarded when there is no next table specified. Specific forwarding actions, including egress packet edits, are for the most part included within the action sets of the group entries. DNOS-OF uses specific types of group entries to support different packet flow scenarios. Apply-actions instructions and action lists are also used for some VLAN tag packet editing, and to send packets to the controller.

In the general OpenFlow case packets pass from flow table to flow table and can be arbitrarily modified between tables. To take advantage of this generality each table stage would need to include a packet parser. In DNOS-OF this kind of packet flow is conceptual - packets are parsed early in the pipeline and header fields are extracted. After that it is only these fields that are passed between tables and used for matching or modification by "apply actions" instructions. It is not expected that this distinction will matter to applications.

The next section describes the DNOS-OF flow tables in terms of their supported match fields, flow entry rule types, instructions, actions, expiration provisions, and statistics counters. Default miss actions are also specified for each table as applicable. Group table entry types and action set constraints are then described. Ingress packets always have an associated Tunnel Id metadata value. For packets from physical ports this value is always zero. Only Physical ports are supported in DNOS-OF, so no Tunnel Id values other than 0 are allowed.

NOTE: The software may have other undocumented tables and groups implemented or partially implemented.   Only the features described here to support bridging and routing are supported.

4.3.2.1 Ingress Port Flow Table

The Ingress Port Flow Table is the first table in the pipeline and, by convention, is numbered zero. OpenFlow uses a 32 bit value for ifNums. In this version of DNOS-OF, the high order 16 bits are zero for physical ports since no other port types are supported in 1.0.

The Ingress Port Flow Table presents what is essentially a de-multiplexing logic function as an OpenFlow table that can be programmed from the controller. By default, packets from physical ports with null (zero) Tunnel Id metadata go to the VLAN Flow Table. Entries in this table must admit ingress packets by matching the ingress ifNum exactly, by matching Tunnel Id, or by some combination.

Note: DNOS-OF may prevent certain types of rules from being added to other tables unless there is appropriate flow entry in the Ingress Port Flow Table.
The default on miss is for packets from physical ports to go to the VLAN Flow Table. There is no default rule for data center overlay tunnel packets from logical ports, which are dropped on miss.

4.3.2.1.1 Match Criteria, Instructions, Actions/Action List/Action Set, Counters, Flow Expiry

The Ingress Port Flow Table supports the flow entry types listed in Table 1. This table would typically have one rule enabling ingress packets from each port type. However, since in this release, only the physical port type is supported, only one rule is enabled.

Table 1: Ingress Port Flow Table Entry Types

Type	Description
Normal Ethernet Frames	Matches packets from local physical ports, identified by zero Tunnel Id. Normal Ethernet rules have Goto-Table instructions that specify the VLAN Flow Table.

Table 2: Ingress Port Flow table Match Fields

Field	Bits	Maskable	Optional	Description
IN_PORT	32	No	Yes	Ingress port. Depending on rule may be omitted to match any IN_PORT.

Table 3: Ingress Port Flow Table Instructions

The Ingress Port Flow Table supports the single Goto-Table instruction listed in Table 3.

Name	Argument	Description
Goto-Table	Table	Next table. For this release, must be the VLAN Flow Table.
Apply-Actions	Action list	Can contain at most one instance of each of the actions listed in Table 3.1

The Ingress Port Flow Table actions can optionally set the packet VRF using an action list.

Table 4: Ingress Port Flow Table Action List

Name	Argument	Description
Set-Field	VRF	VRF for L3 lookups. Only applicable to Normal Ethernet Frame rules. Optional.

Table 5: Ingress Port Flow Table Counters

Name	Type	Description
Active Entries,	Table	Reference count of number of active entries in the table
Duration	Per-entry	Seconds since this flow entry was installed

4.3.2.2 VLAN Flow Table

The VLAN Flow Table is used for IEEE 801.Q VLAN assignment and filtering to specify how VLANs are to be handled on a particular port. All packets must have an associated VLAN id in order to be processed by subsequent tables. Packets that do not match any entry in the VLAN table are filtered, that is, dropped by default. Note that IEEE defined BPDUs are always received untagged.
The VLAN Flow Table can optionally assign a nonzero VRF value to the packet based on the VLAN. OF-DPA defines VRF as a new pipeline metadata field. The VRF defaults to zero if not set.

4.3.2.2.1 Match Criteria, Instructions, Actions/Action List/Action Set, Counters, Flow Expiry

The VLAN Flow Table supports the Flow Entry Types listed in Table 10. Flow entries are differentiated based on IN_PORT, whether or not the packet was tagged, and the VLAN id in the tag.

OpenFlow has traditionally used a 16-bit field for VLAN id. Since only the low order 12 bits are needed to express a VLAN id, OpenFlow has defined special values to indicate tagged and untagged packets. In particular, the VLAN id 0x0000 (OFPVID_NONE, defined in the OpenFlow specification) is used to represent an untagged packet, and 0x1000 (OFPVID_PRESENT) for a priority tagged packet. All tagged packets are represented by VLAN id values between 0x1001 and 0x1FFE[24] (OFPVID_PRESENT | VLAN id value). This convention must be followed in programming rules from the controller. For further explanation consult the OpenFlow 1.3.4 specification.

Note: DNOS-OF does not support matching packets just on whether or not they have a VLAN tag as described in Table 13 of OpenFlow 1.3.4.

Note: At most two tags are supported. Entries in the OF-DPA VLAN Flow table are mutually exclusive. Any explicit rule priority assignments are ignored.
Table 6: VLAN Flow Table Flow Entry Types

Type	Description
VLAN Filtering	Exact match on IN_PORT and VLAN_VID parsed from the packet. For tagged packets with a VLAN tag containing a VLAN_VID greater than zero. Cannot be masked. VLAN_VID cannot be used in a Port VLAN Assignment rule for untagged packets. The only instruction is Goto-Table and must specify the Termination MAC Flow Table. Tagged packets that do not match any rule are treated as VLAN_VIDs that are not allowed on the port and are dropped. Can optionally assign a VRF for routed packets.
Untagged Packet Port VLAN Assignment	Exact match on IN_PORT and VLAN id == 0 (lower 12 bits of match field) value using a mask value of 0x0fff (masks off OFPVID_PRESENT). Action set must assign a VLAN_VID. The VLAN_VID value cannot be used in a VLAN Filtering rule. If the packet does not have a VLAN tag, one will be pushed if necessary at packet egress. Rule must have a Goto-Table instruction specifying the Termination MAC Flow Table. Untagged packets are dropped if there is no port VLAN assignment rule. Can optionally assign a VRF for routed packets.
Allow All VLANs	Wildcard VLAN match for a specific IN_PORT. Essentially turns off VLAN filtering and/or assignment for a physical port. Must be lower priority than any overlapping translation, filtering, MPLS, or VLAN assignment rule. Untagged packets that match this rule will be assigned an illegal VLAN and may be subsequently dropped. Should also define an L2 Unfiltered Interface group entry for the port.
VLAN Translate, Single Tag, or Single Tag to Double Tag	Used to either modify the VLAN id on a single tagged packet, or to optionally modify the VLAN id and then push another tag onto a single tagged packet. Can also optionally assign a VRF for routed packets. By OpenFlow convention, the outermost VLAN tag is matched independent of TPID.

Note: The untagged packet rule applies to both untagged packets, which match VLAN_VID = 0x1000, and IEEE 802.1P priority tagged packets, which match VLAN_VID = 0x0000. However the VLAN-PCP match field will be set from the value in a priority VLAN tag rather than default to zero in the case of a packet without a VLAN tag.

Note: A VLAN Flow Table rule cannot specify an IN_PORT and VLAN_VID combination that is used in a VXLAN Access Logical Port configuration. Conversely, it must include a rule to permit an IN_PORT and VLAN_VID combination used in a VXLAN Tunnel Next H

Table 8. VLAN Flow Table Instructions

Name	Argument	Description
Apply-Actions	Action List	The VLAN Flow Table supports the actions specified in Table 13.
Goto-Table	Table	For VLAN filtering or Port VLAN assignment the next table should be the Termination MAC Flow Table.

The VLAN table uses Apply Actions for port VLAN tagging and assignment. The action list can have at most one of each action type.

Table 9: VLAN Flow Table Action List

Name	Argument	Description
Set Field	VLAN_VID, must be between 1 and 4094.	Sets the VLAN id on the outermost tag. If the packet is untagged then one is pushed with the specified VLAN id and priority zero.
Set Field	VRF	Optionally sets the VRF pipeline field. VRF must be the same in all rules for the same VLAN.
Push VLAN	TPID	Used in translating single to double tag. TPID must be 0x8100 (inner VLAN tag) or 0x88a8 (outer VLAN tag).

Note: The untagged packet action is the same as in OpenFlow 1.0. The implicit addition of a tag to an untagged packet is tolerated but not condoned in OpenFlow 1.3.4.

Only hard interval time-out ageing per entry is supported, as indicated in Table 9.

Table 11: VLAN Flow Table Expiry

Name	Bits	Description
Hard Timeout	32	Number of seconds after which flow entry is removed. Optional, entry does not age out if zero or not specified.

4.3.2.3 Termination MAC Flow Table

The Termination MAC Flow Table determines whether to do bridging or routing on a packet. It identifies destination MAC, VLAN, and Ethertype for routed packets. Routed packet rule types use a goto instruction to indicate that the next table is one of the routing tables. The default on a miss is to go to the Bridging Flow Table.

4.3.2.3.1 Match Criteria, Instructions, Actions/Action List/Action Set, Counters, Flow Expiry

The Termination MAC Flow Table implements the flow entry types listed in Table 12.

Table 12: Termination MAC Flow Table Entry Types

Name	Description
Unicast MAC	Used to identify an IPv4 or IPv6 router MAC. Priority must be assigned so as to be lower than a multicast MAC rule, if one exists.
IPv4 Multicast MAC	Wildcard rule that recognizes all IPv4 multicast MAC addresses specified in RFC 1112. If specified, this must be ETH_DST = 01-00-5e-00-00-00 with mask ff-ff-ff-80-00-00. There can only be one flow entry of this type.
IPv6 Multicast MAC	Wildcard rule that recognizes all IPv6 MAC addresses specified in RFC 2464. If specified, this must be ETH_DST = 33-33-00-00-00-00 with mask ff-ff-00-00-00-00. There can only be one flow entry of this type.

The Termination MAC Flow Table match fields are listed in Table 13. Strict rule priority must be assigned by the controller so that every flow entry has a unique priority.

Table 13: Termination MAC Flow Table Match Fields

\Field	Bits	Maskable	Optional	Description
IN_PORT	32	No	Yes	Physical (local) input port.
ETH_TYPE	16	No	No	Prerequisite for IPv4 (0x0800) or IPv6 (0x86dd).
ETH_DST	48	No	No	Ethernet destination MAC. Prefix maskable for only the specific multicast IP flow entries in Table 28. Can only be field masked for unicast destination MACs.
VLAN_VID	16	Yes	Yes	Matches against the Outer VLAN id. Must be either omitted or exact.
IPV4_DST	32	\Yes	Yes	Can only be used with 224/8 address and 224.0.0.0 mask values, otherwise must be omitted. Prerequisite ETH_TYPE must be 0x0800.
IPv6_DST	128	Yes	Yes	Can only be used with FF00::/8 address and FF00:0:0:0:0:0:0:0 mask values, otherwise must be omitted. Prerequisite ETH_TYPE must be 0x86dd.

The Termination MAC Flow Table can have the instructions shown in Table 14.

Name	Argument	Description
Goto-Table		Unicast MAC rules with multicast IPV4_DST or IPV6-DST should specify the Multicast Routing Flow Table, otherwise they can only specify the Unicast Routing Flow Table. Multicast MAC rules can only specify the Multicast Routing Flow Table. The packet is dropped if the rule matches and there is no Goto-Table instruction.
Apply Actions		Optional. If supplied can only contain one action, output a copy to CONTROLLER.

Table 15: Termination MAC Flow Table Counters

Name	Bits	Type	Description
Active Entries	32	Table	Number of active flow entries in the table
Duration (sec.)	32	Per entry	Seconds since this flow entry was installed

Table 16: Termination MAC Flow Table Expiry

Name	Bits	Description
Hard Timeout	32	Number of seconds after which flow entry is removed. Optional, entry does not age out if zero or not specified.

4.3.2.4 Bridging Flow Table

The Bridging Flow Table supports Ethernet packet switching for potentially large numbers of flow entries using the hardware L2 tables. The default on a miss is to go to the Policy ACL Flow Table.
Note: The Policy ACL Flow Table is preferred for matching BPDUs.
The Bridging Flow Table forwards based on VLAN (normal switched packets) using the flow entry types in Table 17.

Table 17: Bridging Flow Table Flow Entry Types

Type	Description
Unicast VLAN Bridging	Matches switched unicast Ethernet frames by VLAN id and MAC_DST. MAC_DST must be unicast and cannot be masked. VLAN id must be present and nonzero. Tunnel id must be masked or omitted.
Multicast VLAN Bridging	Matches switched multicast Ethernet frames by VLAN id and MAC_DST. MAC_DST must be multicast and cannot be masked. VLAN id must be present and nonzero. Tunnel id must be masked or omitted.
DLF VLAN Bridging	Matches switched Ethernet frames by VLAN id only. MAC_DST must be field masked and match any destination. Must have lower relative priority than any unicast or multicast flow entries that specify this VLAN. VLAN id must be present and nonzero. Tunnel id must be masked or omitted.

Note: Exact match rules must be given higher priority assignments than any wildcard rules. In any event, exact match rules are evaluated before any wildcard rules.

4.3.2.4.1 Match Criteria, Instructions, Actions/Action List/Action Set, Counters, Flow Expiry

Match fields for flow entry types are described in the following tables.

Table 18. Bridging Flow Table Match Fields

Field	Bits	Maskable	Optional	Description
ETH_DST	48	Yes	Yes	Ethernet destination MAC, allowed values depend on flow entry type. Exact match only (mask must be all 1's if supplied).
VLAN_VID	16	Yes	Yes	VLAN id, allowed values depend on flow entry type. Exact match only (mask must be all 1's if supplied).

Default next table if no match is the ACL Policy Flow Table.

Table 19: Bridging Flow Table Instructions

Name	Argument	Description
Write-Actions	Action set	Only the actions in Section 3.2.4.3 can be specified.
Apply-Actions	Action list	Optional. If specified, can contain only a single output action to send a copy to CONTROLLER
Goto-Table	Table	Must be the ACL Policy Flow Table if specified. If packet matches and no next table is specified then the packet is dropped.

The Bridging Flow Table supports the actions in Table 20 by flow entry type. The DNOS-OF API validates consistency of flow entry type and DNOS-OF group entry type references.

Table 20: Bridging Flow Table Actions by Flow Entry Type

Type	Argument	Description
Unicast VLAN Bridging	Group ID	Must be a DNOS_OF L2 Interface group entry for the forwarding VLAN.
Multicast VLAN Bridging	Group ID	Must be a DNOS_OF L2 Multicast group entry for the forwarding VLAN.
DLF VLAN Bridging	Group ID	Must be a DNOS_OF L2 Flood group entry for the forwarding VLAN.

The Bridging Flow Table counters are listed in Table 21.

Table 21: Bridging Flow Table Counters

Name	Bits	Type	Description
Active Entries	32	Table	Number of active entries in the table.
Duration (sec.)	32	Per entry	Seconds since this flow entry was installed.

Bridging Flow Table expiry provisions are shown in Table 22.

Table 22: Bridging Flow Table Expiry

Name	Bits	Description
Hard Timeout	32	Number of seconds after which flow entry is removed. Optional, entry does not age out if zero or not specified.
Idle Timeout	32	Number of seconds of inactivity - after which a flow entry is removed. Optional, flow entry does not age out if unspecified or zero.

4.3.2.5 Unicast Routing Flow Table

The Unicast Routing Flow Table supports routing for potentially large numbers of IPv4 and IPv6 flow entries using the hardware L3 tables.

The Unicast Routing Flow Table is a single table organized as two mutually exclusive logical subtables by IP protocol, and supports flow entry types listed in Table 23. One table number is used for both logical tables.

Table 23. Unicast Routing Flow Table Entry Types

Type	Table	Prerequisite(s)	Description
IPv4 Unicast	Table 40	Ethertype=0x0800	Matches routed unicast IPv4 packets. The Goto-Table instruction specifies the Policy ACL Table.
IPv6 Unicast	Table 41	Ethertype=0x86dd	Matches routed unicast IPv6 packets. The Goto-Table instruction specifies the Policy ACL Table.

4.3.2.5.1 Match Criteria, Instructions, Actions/Action List/Action Set, Counters, Flow Expiry

Table 24. Unicast Routing Flow Table IPv4 Header Match Fields

Field	Bits	Maskable	Optional	Description
ETH_TYPE	16	No	No	Must be 0x0800
VRF	16	No	Yes	If omitted or zero indicates the default routing table.
IPv4 DST	12	Yes	No	Must be a unicast IPv4 address. Prefix maskable only, mask used for LPM forwarding.

Table 25. Unicast Routing Flow Table IPv6 Header Match Fields

Field	Bits	Maskable	Optional	Description
ETH_TYPE	16	No	No	Must be 0x86dd
VRF	16	No	Yes	If omitted or zero indicates the default routing table.
IPV6_DST	128	Yes	No	Must be a unicast IPv6 address. Prefix maskable only, used for LPM forwarding.

Note: Exact match rules must be given higher priority assignments than any LPM prefix match rules. In any event, the hardware evaluates exact match rules before any wildcard rules.

Note: Rules that specify a nonzero VRF must have higher relative priority than other overlapping rules. The wildcard rules are effectively "global" or "default" in that they are matched last, that is, if no specific VRF rule matches the packet. If the packet VRF is zero it can only match one of the wildcard rules.
Default next table on a miss is the ACL Policy Flow Table.

Table 26: Unicast Routing Flow Table Instructions

Name	Argument	Description
Write-Actions	Action set	Only the actions in Table 27 can be specified.
Clear-Actions	-	Used to delete any forwarding decision so that the packet will be dropped.

Other instruction types, specifically Apply Actions, are not supported.

Table 27: Unicast Routing Flow Table Actions

Name	**Argument**	**Description**
Group	Group ID	Must be a DNOS-OF L3 Unicast Group Entry.
Decrement TTL and do MTU check	-	MTU check is a vendor extension. An invalid TTL (zero before or after decrement) is always dropped and a copy sent to the CPU for forwarding to the CONTROLLER. Similarly, a packet that exceeds the MTU is dropped and a copy sent to the CONTROLLER. Required.

The group entry includes the decrement TTL and MTU check actions, so these need not be explicitly specified in the action set. The Routing Flow Table counters are listed in Table 28.

Table 28: Unicast Routing Flow Table Counters

Name	Bits	Type	Description
Active Entries	32	Table	Reference count of number of active entries in the table.
Duration (sec.)	32	Per entry	Seconds since this flow entry was installed.

Unicast Routing Flow Table expiry provisions are shown in Table 29.

Table 29: Unicast Routing Flow Table Flow Expiry

Name	Bits	Description
Hard Timeout	32	Number of seconds after which flow entry is removed. Optional, entry does not age out if zero or not specified.
Idle Timeout	32	Number of seconds of inactivity, after which a flow entry is removed. Optional, entry does not age out if zero or not specified.

4.3.2.6 Multicast Routing Flow Table

The Multicast Routing Flow Table supports routing for IPv4 and IPv6 multicast packets.
The Multicast Routing Flow Table is also organized as two mutually exclusive logical sub tables by IP protocol, and supports the flow entry types listed in Table 30.

Table 30: Multicast Routing Flow Table Entry Types

Type	Table	Prerequisite(s)	Description
IPv4 Multicast	Table 31	Ethertype=0x0800	Matches routed multicast IPv4 packets.
IPv6 Multicast	Table 32	Ethertype=0x86dd	Matches routed multicast IPv6 packets.

4.3.2.6.1 Match Criteria, Instructions, Actions/Action List/Action Set, Counters, Flow Expiry

Match fields for flow entry types are described in the following tables.

Table 31. Multicast Routing Flow Table IPv4 Match Fields

Field	Bits	Maskable	Optional	Description
ETH_TYPE	16		No	Must be 0x0800. Required prerequisite.
VLAN_VID	16		No	VLAN id
VRF	16		Yes	VRF.
IPV4_SRC	32		Yes	Cannot be bit masked, but can be omitted.
IPV4_DST	32		No	Must be an IPv4 multicast group address.

Table 32. Multicast Routing Flow Table IPv6 Match Fields

Field	Bits	Maskable	Optional	Description
ETH_TYPE	16	No	No	Must be 0x86dd. Required prerequisite.
VLAN_VID	16	No	No	VLAN id
VRF	16	No	Yes	VRF.
IPV6_SRC	128	Yes	Yes	Cannot be bit masked, but can be omitted.
IPV6_DST	128	Yes	No	Must be an IPv6 multicast group address.

Default next table on miss is the ACL Policy Flow Table.

Table 33: Multicast Routing Flow Table Instructions

Name	Argument	Description
Write Actions	Action set	Only the actions in Table 34 can be specified.

Goto-Table	Table	Must be the Policy ACL Flow Table. In the event that there is no group entry referenced and no next table specified, the packet will be dropped.

Other instruction types, specifically Apply Actions, are not supported.

Table 34: Mutlicast Routing Flow Table Actions

Name	**Argument**	**Description**
Group	Group ID	Must be a DNOS-OF L3 Multicast group entry with the forwarding VLAN ID as a name component.
Decrement TTL and do MTU check	-	MTU check is a vendor extension. An invalid TTL (zero before or after decrement) is always dropped and a copy sent to the CPU for forwarding to the CONTROLLER. Similarly, a packet that exceeds the MTU is dropped and a copy sent to the CONTROLLER. Required.

Note: The group entry includes the decrement TTL and MTU check actions.

Table 35: Multicast Routing Flow Table Counters

Name	Bits	Type	Description
Active Entries	32	Table	Reference count of number of active entries in the table.
Duration (sec.)	32	Per entry	Seconds since this flow entry was installed.

Table 36: Multicast Routing Flow Table Flow Expiry

Name	Bits	Description
Hard Timeout	32	Number of seconds after which flow entry is removed. Optional, entry does not age out if zero or not specified.
Idle Timeout	32	Number of seconds of inactivity after which a flow entry is removed. Optional: entry does not age out if zero or not specified.

4.3.2.7 Policy ACL Flow Table

The Policy ACL Flow Table supports wide, multi-field matching. Most fields can be wildcard matched, and explicit priority must be included in all flow entry modification. This is the preferred table for matching BPDU and ARP packets. It is also the only table where QoS actions are available.

The Policy ACL Flow Table is organized as mutually exclusive logical sub tables. Flow entries in the IPv6 logical tables match only IPv6 packets by VLAN ID. The non-IPv6 logical table matches any packet except for IPv6 packets by VLAN ID. By OpenFlow single-entry match semantics, since the Policy ACL Flow Table is considered a single table, a packet can match, at most, one rule in the entire table.

Note: The Ethertype prerequisite must be explicitly provided and cannot be masked.
The default on table miss is to do nothing. The packet will be forwarded using the output or group in the action set, if any. If the action set does not have a group or output action the packet is dropped. The Policy ACL Flow Table supports the flow entry types listed in Table 37.

Table 37: Policy ACL Flow Table Entry Types

Type	Table	Prerequisite	Description
IPv4 VLAN	Table 38	Ethertype != 0x86dd IN_PORT is a physical port.	Matches packers by VLAN ID except for IPv6. VLAN ID is optional but must be nonzero if supplied.
IPv6 VLAN	Table 39	Ethertype = 0x86dd	Matches only IPv6 packets by VLAN ID. VLAN ID is optional but must be nonzero if supplied.

4.3.2.7.1 Match Criteria, Instructions, Actions/Action List/Action Set, Counters, Flow Expiry

The available match fields for Policy ACL Flow Table flow entry types are as described in the following tables.

Table 38: Policy ACL Flow Table IPv4 Match Fields

Field	Bits	Maskable	Optional	Description or Prerequisite
IN_PORT	32	No	Yes	Physical or logical ingress port.
ETH_SRC	48	Yes	Yes	Ethernet source MAC
ETH_DST	48	Yes	Yes	Ethernet destination MAC
ETH_TYPE	16	No	Yes	Any value except 0x86dd. Explicit prerequisite must be 0x800 if IP fields are to be matched.
VLAN_VID	16	Yes	Yes	VLAN id. Cannot be masked for a VLAN bridging rule that redirects to a different L2 output group. Only applicable to VLAN flow entry types.
VLAN_PCP	3	No	Yes	802.1p priority field from VLAN tag. Always has a value, will be zero if packet did not have a VLAN tag.
VLAN_DEI	1	No	Yes	802.1p drop eligibility indicator field from VLAN tag. Always has a value, will be zero if packet did not have a VLAN tag.
VRF	16	No	Yes	VRF.
IPV4_SRC	32	Yes	Yes	Matches SIP if Ethertype = 0x0800
ARP_SPA	32	Yes	Yes	Matches ARP source protocol address if Ethertype = 0x0806
IPV4_DST	32	Yes	Yes	Matches DIP if Ethertype = 0x0800
IP_PROTO	8	No	Yes	IP protocol field from IP header if Ethertype = 0x0800
IP_DSCP	6	No	Yes	Bits 0 through 5 of the IP ToS Field as defined in RFC 2474 if Ethertype = 0x0800

IP_ECN	2	No	Yes	Bits 6 through 7 of the IP ToS Field as defined in RFC 3168 if Ethertype = 0x0800
TCP_SRC	16	No	Yes	If Ethertype = 0x0800 and IP_PROTO = 6
UDP_SRC	16	No	Yes	f Ethertype = 0x0800 and IP_PROTO = 17
SCTP_SRC	16	No	Yes	If Ethertype = 0x0800 and IP_PROTO = 132
ICMPV4_TYPE	8	No	Yes	If Ethertype = 0x0800 and IP_PROTO = 1
TCP_DST	16	No	Yes	If Ethertype = 0x0800 and IP_PROTO = 6
UDP_DST	16	No	Yes	if Ethertype = 0x0800 and IP_PROTO = 17
SCTP_DST	16	No	Yes	If Ethertype = 0x0800 and IP_PROTO = 132
ICMPv4_CODE	8	No	Yes	If Ethertype = 0x0800 and IP_PROTO = 1

Table 39: Policy ACL Flow Table IPv6 Match Fields.

Field	Bits	Maskable	Optional	Description
IN_PORT	32	No	Yes	Physical or logical ingress port.
ETH_SRC	48	Yes	Yes	Ethernet source MAC
ETH_DST	48	Yes	Yes	Ethernet destination MAC
ETH_TYPE	16	No	Yes	Must be 0x86dd
VLAN_VID	16	Yes	Yes	VLAN id. Cannot be masked for a VLAN bridging rule that redirects to a different L2 output group. Only applicable to VLAN flow entry types.
\VLAN_PCP	3	No	Yes	802.1p priority field from VLAN tag. Always has a value, will be zero if packet did not have a VLAN tag.
VLAN_DEI	1	No	Yes	802.1p drop eligibility indicator field from VLAN tag. Always has a value, will be zero if packet did not have a VLAN tag.
VRF	16	No	Yes	VRF
IPV6_SRC	128	Yes	Yes	Matches IPv6 SIP
IPV6_DST	128	Yes	Yes	Matches IPv6 DIP
IP_PROTO	8	No	Yes	Matches IPv6 Next header
IPV6_FLABEL	20	No	Yes	Matches IPv6 flow label
IP_DSCP	6	No	Yes	Bits 0 through 5 of the IP ToS Field as defined in RFC 2474 if Ethertype = 0x86dd
IP_ECN	2	No	Yes	Bits 6 through 7 of the IP ToS Field as defined in RFC 3168 if Ethertype = 0x86dd
TCP_SRC	16	No	Yes	If Ethertype = 0x86dd and IP_PROTO = 6
UDP_SRC	16	No	Yes	If Ethertype = 0x86dd and IP_PROTO = 17
SCTP_SRC	16	No	Yes	If Ethertype = 0x86dd and IP_PROTO = 132
ICMPV6_TYPE	8	No	Yes	If Ethertype = 0x86dd and IP_PROTO = 58
TCP_DST	16	No	es	If Ethertype = 0x86dd 00 and IP_PROTO = 6
UDP_DST	16	No	Yes	If Ethertype = 0x86dd and IP_PROTO = 17

SCTP_DST	16	No	es	If Ethertype = 0x86dd and IP_PROTO = 132
ICMPv6_CODE	8	No	es	If Ethertype = 0x86dd and IP_PROTO = 58

Notes:

- IPv6 Neighbor Discovery field matching is not supported in this version of DNOS-OF.
- Not all IPv6 match fields are supported on all platforms.
- DNOS-OF permits bit masking L4 source and destination ports, as well as ICMP code. The OpenFlow does not require these to be maskable.

The only instruction is write actions. Since there is no next table, there can be no Goto-Table or Write Metadata instructions.

Table 40: Policy ACL Flow Table Instruction Set

Name	Argument	Description
Apply Actions	Action list	Optional. Only the actions in Table 41 can be specified.
Clear Actions		Used to clear the action set for dropping the packet. Cannot be combined with write actions.
Write Actions	Action set	Only the actions in Table 42 or Table 43 can be specified, depending on rule type.

The packet is dropped if there is no group action that specifies output ports, since there is no next table.
Note: Apply-actions to CONTROLLER would be used in order to output the packet to the CONTROLLER reserved port, rather than an output action in the write-actions action set.

The Policy ACL Flow Table supports the actions listed in Table 41.

Table 41: Policy ACL Flow Table Action List Actions

Name	Argument	Description
Set-Field	Traffic Class	

The Policy ACL Flow Table action set supports the actions listed in Table 70 for VLAN match rule types, and the actions in Table 71 for tunnel match rule types.

Table 42: Policy ACL Flow Table VLAN Flow Entry Action Set

Name	Argument	Description
Group	Group	Sets output group entry for processing the packet after this table. Group must exist, be consistent with the type of rule and packet;, and can be any of: L2 Interface, L2 Rewrite, L2 Multicast, L3 Unicast, L3 Multicast, or L3 ECMP; must respect VLAN id naming conventions. In particular, if the output is an L2 Rewrite group that does not set the VLAN id, the L2 Interface group it references must be consistent with the VLAN id in the matched flow entry.
Set-Queue	Queue-id	Determines queue to be used when packet is finally forwarded. Zero indicates the default queue. Cannot be used together with Set Traffic Class in the action list.

As with Unicast and Multicast Routing Flow Table actions, the decrement TTL and MTU checks are encoded by referencing an L3 Unicast or Multicast group entry. Note that if the group entry type is L2 Interface. L2 Rewrite, or L2 Multicast then these checks will not be done.

The Policy ACL Flow Table counters are listed in Table 43. These are applicable to VLAN flow entries.

Table 43: Policy ACL Flow Table Counters

Name	Bits	Type	Description
Active Entries	32	Table	Reference count of number of active entries in the table.
Duration (sec.)	32	Per entry	Seconds since this flow entry was installed.
Received Packets	64	Per entry	Number of packets to which this rule applies.
Received Bytes	64	Per entry	Number of bytes to which this rule applies.

Policy ACL Flow Table expiry provisions are shown in Table 44. Each flow entry can have its own time-out values.

Table 44: Policy ACL Flow Table Expiry

Name	Bits	Description
Hard Timeout	32	Number of seconds after which flow entry is removed. Optional, entry does not age out if zero or not specified.
Idle Timeout	32	Number of seconds of inactivity, after which a flow entry is removed. Optional: entry does not age out if zero or not specified.

4.4 Group Table

Most forwarding actions are embodied in group table entries. DNOS-OF supports a defined set of group table entry types, effectively partitioning the group table into logical sub tables.

Each group entry has an identifier, type, counters, and one or more action buckets. OpenFlow has a single monolithic group table, but DNOS-OF differentiates among types of group entries. For this purpose, DNOS-OF encodes the group entry type in a group entry identifier field. The basic naming convention followed is illustrated in Table 45.

Table 45: DNOS-OF Group Table Entry Identifier Naming Convention

Field	Bits	Description
Index	[27:0]	28-bit field, used to uniquely identify a group entry of the indicated type. May be used to further encode properties of the group entry, such as VLAN ID.
Type	[32:38]	-bit field that encodes the entry type, one of: 0: DNOS-OF L2 Interface 1: DNOS-OF L2 Rewrite 2: DNOS-OF L3 Unicast 3: DNOS-OF L2 Multicast 4: DNOS-OF L2 Flood 5: DNOS-OF L3 Interface 6: DNOS-OF L3 Multicast 7: DNOS-OF L3 ECMP

DNOS-OF performs consistency checks on the group entry type when a group action is used in a flow entry. The index scheme varies by DNOS-OF group entry type and is described in the following sections.

4.4.1 DNOS-OF flow tables

DNOS-OF flow tables accommodate specific types of flow entries with associated semantic rules, including constraints such as which match fields are available, which instructions and actions are supported, how priorities can be assigned to flow entries, which next table(s) flow entries can go to, and so forth. The flow tables conform to the OpenFlow 1.3.4 specification. In addition to normal flows, two types of special flow entries are supported as follows:

- **Built-in**: Built-in flow entries come preinstalled in specific tables. They are visible to the controller but cannot be modified or deleted. Built-in entries have preassigned match fields, priority, and cookie values. They are typically used for default entries.
- **Automatic:** Automatic flow entries are added by the switch as a side effect of the controller adding a flow entry. They are visible to the controller but cannot be directly modified or deleted except by modifying or deleting the rule that caused the automatic entry to be added. Match fields and priority are predetermined, and the switch assigns the same cookie value as the initiating rule.

In addition to flow tables, DNOS-OF defines a set of group table entry types. The OpenFlow 1.3.4 specification defines four types of groups: indirect, all, select, and fast failover. DNOS-OF further types group entries according to how they can be used in packet flows. This is done using specific naming conventions, properties, and supported action buckets. All DNOS-OF group table entry types can be programmed using OpenFlow 1.3.4 as long as group mods respect the typing conventions.
One motivation for group typing is supporting fundamental differences in use-case requirements. For example, in order to support "one-arm" routing using group table entries, there needed to be a way to override OpenFlow's default source removal and allow routing back to the IN_PORT. This was accomplished by defining L3 group entry types with different properties from L2 groups. Group entry typing is also useful to enforce constraints on group entry chains and for Virtual Local Area Network (VLAN) configuration on physical ports.
Remember that DNOS-OF tables are programming abstractions and do not necessary directly correspond one-to-one with hardware tables. However, they are designed to faithfully capture both use-case requirements and the hardware packet flow semantics, while being straightforward to program from standard controllers.
Users must program flow tables and group entries according to the allowed entry types. The DNOS-OF validates calls and returns errors if constraints and/or conventions are violated. This includes the requirement that objects must exist before they can be referenced from other objects. The OpenFlow agent that interfaces to OF-DPA may also do some argument validation and execute local iterative procedures.

Many forwarding and editing actions for encapsulation/push and field modify are programmed using one or more action buckets in group table entries. This not only proves to be a very efficient and modular programming approach, in that the controller can optimize hardware resources better than the switch, but the controller intrinsically has more CPU power and memory than the control processor on a typical switch for this task. The controller also understands what the application is trying to do, especially when programming requires updating multiple tables. However, when compared with OpenFlow 1.0 programming, it may require more messages between the controller and switches, since more objects need to be programmed. It also potentially requires the controller to keep track of more switch state, although this state can be interrogated as needed.

4.4.2 DNOS-OF L2 Interface Group Entries

L2 Interface Group entries are of OpenFlow indirect type, with a single action bucket.

DNOS-OF L2 Interface group entries are used for egress VLAN filtering and tagging. The identifier convention is shown in Table 94. If a specific set of VLANs is allowed on a port, appropriate group entries must be defined for the VLAN and port combinations.

4.4.2.1 Naming Convention

Table 46 details the DNOS-OF L2 Interface group entry identifier subfields that encode combinations of egress port and VLAN ID.

Table 46: OF-DPA L2 Interface Group Entry Type Naming Convention

Field	Bits	Description
Port identifier	[15:0]	Identifies a physical port (ifNum).
VLAN Id	[27:16]	VLAN ID.
Type	[31:28]	0 (L2 Unicast).

4.4.2.2 Action Buckets

The single action bucket specifies the output port, and whether or not the packet is egressed tagged. Although the pop action is a NOP if the packet has no VLAN tag, packets should always have a VLAN tag when the actions in the output group table are applied.

Note: If the packet came in untagged and a port VLAN was assigned, a VLAN tag was pushed as a VLAN Flow Table action.

Table 47: DNOS-OF L2 Interface Group Entry Bucket Actions

Field	**Argument**	**Description**
Output	Port	Physical output port.
Pop VLAN	None	Pop the VLAN tag before sending the packet.
Set Field	DSCP	Static DSCP value for IP packets
Set Field	VLAN PCP	Static 802.1p value
Set-Field	VLAN DEI	Static 802.1p value

Clearly DNOS-OF L2 Interface group entries must be defined before being used. DNOS-OF maintains reference counts for used entries, and an entry cannot be deleted if it is referenced by a flow entry or another group.

4.4.2.3 Counters

DNOS-OF L2 Interface group entry counters are as shown in Table 48.

Table 48: OF-DPA L2 Interface Group Entry Counters

Name	Bits	Type	Description
Reference Count	32	Per entry	Number of flow entries or group entities currently referencing this group entry.
Duration (secs.)	32	Per entry	Seconds since this group entry was installed.

4.4.3 DNOS-OF L2 Rewrite Group Entries

DNOS-OF L2 Rewrite group entries are of indirect type and have a single action bucket. They are used when it is desired to modify Ethernet header fields for bridged packets. Use of an DNOS-OF L2 Rewrite group entry is optional, and can only be a Policy ACL Flow Table action.

DNOS-OF L2 Rewrite actions are optional with the exception of group. This permits an DNOS-OF L2 Rewrite group entry to selectively modify the source MAC, destination MAC, and/or VLAN ID.

If a Set Field action sets the VLAN id, the VLAN id must be the same as in a chained L2 Interface group entry. Note that if the VLAN id is not rewritten, the VLAN id in the L2 Interface group entry must be the same as the VLAN id matched in the Policy ACL Flow Table flow entry that forwarded to the rewrite group.

4.4.3.1 Naming Convention

Table 49 details the DNOS-OF L2 Rewrite group entry identifier subfields that encode the type and VLAN ID.

Table 49: OF-DPA L2 Rewrite Group Entry Type Naming Convention

Field	Bits	Description
Id	[27:0]	Index to differentiate group entries of this type.
Type	[31:28]	1 (OF-DPA L2 Rewrite).

4.4.3.2 Action Buckets

The single action bucket specifies the output group for forwarding the packet and optional Ethernet header modifications.

Field	Argument	Description
Group	Group entry	Must chain to a L2 Interface group entry. Required.
Set Field	MAC_SRC	Re-write the source MAC. Optional.
Set Field	MAC_DST	Re-write the destination MAC. Optional.
Set Field	VLAN-id	Re-write the VLAN id. Optional.

Chained group entries must be defined before being used. DNOS-OF maintains reference counts for used entries, and a group entry cannot be deleted if it is referenced by a flow entry or another group.

4.4.3.3 Counters

DNOS-OF L2 Rewrite group entry counters are as shown in Table 51 for completeness.

Table 51: OF-DPA L2 Rewrite Group Entry Counters

Name	Bits	Type	Description
Reference Count	32	Per entry	Number of flow or group entities currently referencing this group entry.
Duration (sec.)	32	Per entry	Seconds since this group entry was installed.

4.4.4 DNOS-OF L3 Unicast Group Entries

DNOS-OF L3 Unicast group entries are used to supply the routing next hop and output interface for packet forwarding. To properly route a packet from either the Routing Flow Table or the Policy ACL Flow Table, the forwarding flow entry must reference a DNOS-OF L3 Unicast Group entry.

DNOS-OF L3 Unicast automatically includes the ALLOW-IN_PORT vendor extension property to allow packets to be sent out IN_PORT. This property overrides the OpenFlow default behavior, which is to not forward a packet to IN_PORT, and is inherited by chained group entries. It is not visible to the controller and hence cannot be modified or read.

All packets must have a VLAN tag. A chained L2 Interface group entry must be in the same VLAN as assigned by the DNOS-OF L3 Unicast Group entry.

4.4.4.1 Naming Convention

The naming convention for DNOS-OF L3 Unicast Group entries is shown in Table 52.

Table 52: OF-DPA L3 Unicast Group Entry Naming Conventioin

Field	Bits	Description
Id	[27:0]	Index to differentiate group entries of this type.
Type	[31:28]	2 (OF-DPA L3 Unicast).

4.4.4.2 Action Buckets

The single action bucket is as shown in Table 53. All actions are required.

Table 53: DNOS-OF L3 Unicast Bucket Actions

Field	**Argument**	**Description**
Group	Group-id	Must chain to a L2 Interface group entry. ALLOW-IN_PORT permits the chained group entry output action to include the packet IN_PORT. Required.
Set Field	MAC_DST	Write the next hop destination MAC. Required.
Set Field	MAC_SRC	Write the source MAC corresponding to the L3 output interface. Required.
Set Field	VLAN-id	Write the VLAN id corresponding to the L3 output interface. Required.

4.4.4.3 Counters

The DNOS-OF L3 Unicast group entry counters are as shown in Table 54.

Table 54: OF-DPA L3 Unicast Group Entry Counters

Name	Bits	Type	Description
Reference Count	32	Per entry	Number of flow or group entities currently referencing this group entry.
Duration (sec.)	32	Per entry	Seconds since this group entry was installed.

4.4.5 DNOS-OF L2 Multicast Group Entries

DNOS-OF L2 multicast group entries are of OpenFlow ALL type. There can be multiple action buckets, each referencing an output port by chaining to a DNOS-OF L2 Interface Group entry.

Note: By OpenFlow default, a packet cannot be forwarded back to the IN_PORT from which it came in. An action bucket that specifies the particular packet's ingress port is not evaluated.

All of the DNOS-OF L2 Interface Group entries referenced by the DNOS-OF Multicast Group entry, and the DNOS-OF Multicast Group entry itself, must be in the same VLAN.

4.4.5.1 Naming Convention

DNOS-OF L2 Multicast group entries use the naming convention in Table 55.

Table 55: OF-DPA L2 Multicast Group Entry Type Naming Convention

Field	Bits	Description
Id	[15:0]	Index to differentiate group entries of this type.
VLAN Id	[27:16]	VLAN ID.
Type	[31:28]	3 (L2 Multicast).

4.4.5.2 Action Buckets

The contents of DNOS-OF L2 Multicast Group entry buckets can contain only the value shown in Table 56.

Table 56: OF-DPA L2 Multicast Bucket Actions

Field	Argument	Description
Group	Group-id	Must reference an OF-DPA L2 Multicast group entry whose VLAN ID name component matches the referencing group entry.

4.4.5.3 Counters

The L2 Multicast group entry counters are as shown in Table 57.

Table 57: OF-DPA L2 Multicast Group Entry Counters

Name	Bits	Type	Description
Reference Count	32	Per entry	Number of flow or group entities currently referencing this group entry.
Duration (sec.)	32	Per entry	Seconds since this group entry was installed.

4.4.6 DNOS-OF L2 Flood Group Entries

The OF-DPA L2 Flood Group entries are used by VLAN Flow Table wildcard (destination location forwarding, or DLF) rules. Like OF-DPA L2 Multicast group entry types they are of OpenFlow ALL type. The action buckets each encode an output port. Each OF-DPA L2 Flood Group entry bucket forwards a replica to an output port, except for packet IN_PORT.

The main difference from OF-DPA L2 Multicast Group entries is how they are processed in the hardware.

All of the DNOS-OF L2 Interface Group entries referenced by the OF-DPA Flood Group entry, and the OF-DPA Flood Group entry itself, must be in the same VLAN.

Note: There can only be one DNOS-OF L2 Flood Group entry defined per VLAN.

4.4.6.1 Naming Convention

DNOS-OF L2 Flood group entries follow the naming convention shown in Table 58.

Table 58: OF-DPA L2 Flood Group Entry Naming Convention

Field	Bits	Description
Id	[15:0]	Index to differentiate group entries of this type.
VLAN Id	[27:16]	VLAN ID.
Type	[31:28]	4 (OF-DPA L2 Flood).

4.4.6.2 Action Buckets

The contents of the DNOS-OF L2 Flood Group Entry action buckets can contain only the values shown in Table 59.

Table 59: OF-DPA L2 Flood Bucket Actions

Field	Argument	Description
Group	Group-id	Must reference an OF-DPA L2 Unicast group entry whose VLAN ID name component matches the VLAN ID in the referencing group entry name.

4.4.6.3 Counters

The DNOS-OF L2 Multicast group entry counters are as shown in Table 60.

Table 60: OF-DPA L2 Flood Group Entry Counters

Name	Bits	Type	Description
Reference Count	32	Per entry	Number of flow or group entities currently referencing this group entry.
Duration (sec.)	32	Per entry	Seconds since this group entry was installed.

4.4.7 DNOS-OF L3 Interface Group Entries

DNOS-OF L3 interface group entries are of indirect type and have a single action bucket. They are used to supply outgoing routing interface properties for multicast forwarding. For unicast forwarding, use of DNOS-OF L3 Unicast group entries is recommended.

DNOS-OF L3 Interface uses the ALLOW-IN-PORT vendor extension that permits packets to be sent out IN_PORT.

The VLAN id in the name must be the same as the VLAN_VID assigned in the Set Field action and the VLAN id in the name of the chained OF-DPA L2 Interface group.

4.4.7.1 Naming Convention

Table 61 details the DNOS-OF L3 Interface group entry identifier subfields.

Table 61: OF-DPA L3 Interface Group Entry Type Naming Convention

Field	Bits	Description
Id	[27:0]	Index to differentiate group entries of this type.
Type	[31:28]	5 (OF-DPA L3 Interface).

4.4.7.2 Action Buckets

The single action bucket specifies the MAC_SRC, VLAN_VID, TTL decrement action, and an output group for forwarding the packet. All actions are required.

Table 62: DNOS-OF L3 Interface Group Entry Bucket Actions

Field	Argument	Description
Group	Group entry	Must chain to a L2 Interface group entry. This group entry can output the packet to IN_PORT. The VLAN id component of the chained group entry's name must match the Set Field value for VLAN id.
Set Field	MAC_SRC	Write the source MAC corresponding to the L3 output interface.
Set Field	VLAN-id	Write the VLAN id corresponding to the L3 output interface.

Referenced group entries must be defined before being used. DNOS-OF maintains reference counts for used entries, and an entry cannot be deleted if it is referenced by a flow entry or another group.

4.4.7.3 Counters

DNOS-OF L3 Interface group entry counters are as shown in Table 63 for completeness.

Table 63: OF-DPA L3 Interface Group Entry Counters

Name	Bits	Type	Description
Reference Count	32	Per entry	Number of flow or group entities currently referencing this group entry.
Duration (sec.)	32	Per entry	Seconds since this group entry was installed.

4.4.8 DNOS-OF L3 Multicast Group Entries

DNOS-OF L3 Multicast group entries are of OpenFlow *all type*. The action buckets describe the interfaces to which multicast packet replicas are forwarded.

IP multicast packets are forwarded differently, depending on whether they are switched or routed. Packets must be switched in the VLAN in which they came in and cannot be output to IN_PORT. Packets that are multicast in other VLANs must be routed and must be allowed to egress via IN_PORT. This difference is reflected in the actions that are programmed in the action buckets.

Note that any chained DNOS-OF L2 Interface Group entries must be in the same VLAN as the DNOS-OF L3 Multicast group entry. However chained DNOS-OF L3 Interface Group entries must be in different VLANs from the DNOS-OF L3 Multicast Group entry, and from each other.

4.4.8.1 Naming Convention

The naming convention for DNOS-OF L3 Multicast Group entries is shown in Table 64.

Table 64: OF-DPA L3 Multicast Group Entry Naming Convention

Field	Bits	Description
Index	[15:0]	Used to differentiate between OF-DPA L3 multicast group entries.
VLAN Id	[27:16]	VLAN ID.
Type	[31:28]	6 (OF-DPA L3 Multicast).

4.4.8.2 Naming Convention

The action buckets contain the values shown in Table 65.

Table 65: DNOS-OF L3 Multicast Bucket Actions

Field	Argument	Description
Group	Group-id	Can chain to one of: L3 Interface; L2 Interface. Chained group entry names must conform to the VLAN id requirements above.

4.4.8.1 Counters

The DNOS-OF L3 Multicast group entry counters are as shown in Table 66.

Table 66: OF-DPA L3 Multicast Group Entry Counters

Name	Bits	Type	Description
Reference Count	32	Per entry	Number of flow or group entities currently referencing this group entry.
Duration (sec.)	32	Per entry	Seconds since this group entry was installed.

4.4.9 DNOS-OF L3 ECMP Group Entries

DNOS-OF L3 ECMP group entries are OpenFlow type SELECT. The action buckets reference the DNOS-OF L3 Unicast group entries that are members of the multipath group for ECMP forwarding.

A DNOS-OF L3 ECMP Group entry can be specified as a routing target instead of an DNOS-OF L3 Unicast Group entry. Selection of an action bucket for forwarding a particular packet is hardware specific.

4.4.9.1 Naming Convention

The naming convention for DNOS-OF L3 ECMP Group entries is as shown in Table 67.

Table 67: DNOS-OF L3 ECMP Group Entry Naming Convention

Field	Bits	Description
Id	[27:0]	Used to differentiate OF-DPA L3 ECMP group entries.
Type	[31:28]	7 (OF-DPA L3 ECMP)

4.4.9.2 Action Buckets

The action buckets contain the single value listed in Table 68.

Table 68. DNOS-OF L3 ECMP Group Entry Bucket Actions

Field	Argument	Description
Group	Group-id	May chain to an DNOS-OF L3 Unicast.

4.4.9.3 Counters

The DNOS-OF L3 ECMP group entry counters are as shown in Table 69.

Table 69: OF-DPA L3 ECMP Group Entry Counters

Name	Bits	Type	Description
Reference Count	32	Per entry	Number of flow or group entities currently referencing this group entry.
Duration (secs.)	32	Per entry	Seconds since this group entry was installed.

4.4.10 Fast Failover Group Entries

DNOS-OF does not support meter bands in Release 1.0.

4.4.11 Meters

DNOS-OF does not support meter bands in Release 1.0.

4.4.12 Ports

This section lists the DNOS-OF supported properties for physical and reserved ports.

4.4.12.1 Physical Ports

DNOS-OF supports physical ports that are available on specific target platforms. Ports are identified using a 32-bit ifNum value. The most significant two bytes indicate the type of port. Only physical ports are supported in DNOS-OF. Physical ports are front panel ports on the abstract switch.

DNOS-OF supports the physical port features listed in Table 73.

Table 70: Port Type Numbering Conventions

Type	Numbering	Description
Physical	0x0000 xxxx	Physical (front panel) port
Reserved	0xFFFF xxxx	Reserved ports as defined in the OpenFlow specification.

Table 73. DNOS-OF Port Features

Name	Bits	Configurable?	Description
Number	32	No	ifNum (should be the same as in interface MIB)
Hardware Address	48	No	MAC address assigned to port.
Name	128	Yes	16-byte string name (should be the same as in interface MIB)
Configured State	32	Yes	Port is administratively up (0) or down (1)
Current State	32	No	Port link (operational) state is up (0), live (4), or down (1). Generally a port is live if operationally up.
Current Features	32	No	DNOS-OF supports the feature bitmap in Table 74. A one indicates the feature is currently active.
Advertised Features	32	No	DNOS-OF supports the feature bitmap in Table 74. A zero bit indicates the feature is not available.
Supported Features	32	No	DNOS-OF supports the features in Table 74. A zero bit indicates the feature is not supported.
Peer Features	32	No	Bitmap indicating capabilities advertised by the peer from Table 74.
Current Speed	32	No	Current port bitrate in kbps
Max Speed	32	No	Maximum port bitrate in kbps

Note: Not all of the above may be applicable to the LOCAL or CONTROLLER reserved port.

Table 74 shows the port features bitmap referenced from the table above and the OpenFlow Port Features subclasses in Figure 16.

Table 74: Port Features Bitmap

Feature	Bit	Description
10 Mbps HD	0	10 Mbps half-duplex
10 Mbps FD	1	10 Mbps full-duplex
100 Mbps HD	2	100 Mbps half-duplex
100 Mbps FD	3	100 Mbps full-duplex
1GB HD	4	1 Gbps half-duplex
1GB FD	5	1 Gbps full-duplex
10GB FD	6	10 Gbps full-duplex
40GB FD	7	40 Gbps full-duplex
100GB FD	8	100 Gbps full-duplex
1TB FD	9	1 Tbps full-duplex
Other	10	Other rate, not in the above list
Copper	11	Copper medium
Fiber	12	Fiber medium
Autoneg	13	Auto-negotiation
Pause	14	Pause enabled
Pause_Asym	15	Asymmetric pause supported

4.4.12.2 Counters

DNOS-OF supports the port counters listed in Table 75.

Table 75: OF-DPA Physical Port Counters

Name	Bits	Description
Received Packets	64	Total packets received.
Transmitted Packets	64	Total packets transmitted.
Received Bytes	64	Total bytes received.
Transmitted Bytes	64	Total bytes transmitted.
Receive Drops	64	Received packets dropped for any reason.
Transmit Drops	64	Transmitted packets dropped for any reason.
Receive Errors	64	Received packet errors.
Transmit Errors	64	Transmit packets errors.
Receive Frame Alignment Errors	64	Received packets with frame alignment errors.
Receive Overrun Errors	64	Received packet overruns.
Receive CRC Errors	64	Received packet CRC errors.
Collisions	64	Collisions.
Duration (sec.)	32	Time in seconds since configured.

4.4.12.3 Reserved Ports

DNOS-OF supports the reserved ports listed in Table 76.

Table 76. DNOS-OF Reserved Ports

Name	Required	Description	Use	Supported?
ALL	Yes	Required but not supported in DNOS-OF.	Output	No
IN_PORT	Yes	Used to send packets to the ingress port to override OpenFlow default behavior. DNOS-OF uses group ALLOW-IN_PORT property instead. Not to be confused with the IN_PORT match field.	Output	No
CONTROLLER	Yes	The OpenFlow controller. Output destination for sending packets to the Agent which, in turn, sends to the OpenFlow Controller in a Packet_In message. Also can optionally be used to indicate the source of packets received by the Agent in a Packet_Out message.	Input or output	Yes
TABLE	Yes	Used in Packet_Out messages to indicate that the packet must be recirculated through the pipeline. Must always be the first table in the pipeline if specified.	Output	Yes
ANY	Yes	Special value used in some requests.	Neither	Yes
LOCAL	No	Used to send and receive packets with the local Network Protection App. Analogous to Controller but the destination is a local OAM engine rather than the Agent. The exact mechanism is implementation-dependent.	Input or output	Yes, for OAM
NORMAL	No	Not supported in OF-DPA	Output	No
FLOOD	No	Not supported in OF-DPA	Output	No

4.4.13 Vendor Extension Features

In many cases the vendor extension features only affect the OpenFlow abstract switch and can be accommodated by the existing OpenFlow 1.3.4 protocol. In others, an OpenFlow 1.3.4 agent and compatible controller can be extended using the OpenFlow Experimental facility to add new protocol elements as needed. DNOS-OF provides vendor extensions for source MAC learning, and L3 forwarding IN_PORT control.

4.4.13.1 Source MAC Learning

OF-DPA provides vendor extensions for source MAC learning, L3 forwarding IN_PORT control, MPLS and OAM actions and pipeline match fields, and new ancillary object types.
In many cases the vendor extension features only affect the OpenFlow abstract switch and can be accommodated by the existing OpenFlow 1.3.4 protocol. In others, an OpenFlow 1.3.4 agent and compatible controller can be extended using the OpenFlow Experimental facility to add new protocol elements as needed.

4.4.13.2 Group Properties

DNOS-OF adds the vendor extension property "ALLOW-IN_PORT" to DNOS-OF L3 Interface and L2 Loopback group entries. This property applies to the group entry and to any referenced group entries. L3 Interface and L2 Loopback group entries automatically come with the property set, and it cannot be overridden. This obviates the need for special protocol support in OpenFlow 1.3.4.

5 Installation, Configuration, Deployment

Since the target delivery mechanism is the web download, the only thing needed other than the actual N series switch is the DNOS .STK firmware image file. This section shows how it is downloaded, configured and started.

This firmware image file is identical to the standard N Series firmware image file format and is loaded and enabled the same way. The user can switch back and forth between the standard N-Series image .STK file and the DNOS-OF .STK file the same way they do now between versions of the N-Series firmware with “boot system <primary/backup>” command and then reloading the switch firmware.

In release 1.0, DNOS-OF can currently only be installed on the N3xxx/N3xxxP switche. It will be installable on any N-Series switch in following releases. Dell Networking N-series switches can currently store up to two software images in the flash partitions. The dual image feature allows you to upgrade the switch without deleting the older software image. You designate one image as the active image and the other image as the backup image.

The DNOS-OF switch firmware is obtained at **www.dell.com/support**

5.1 View current installed OS

From a terminal or the serial console use the show version command as below:

```
> sh ver
Machine Description............... Dell Networking Switch
System Model ID................... N3024
Machine Type...................... Dell Networking N3024
Serial Number..................... CN0PDJ93282984CT0290A02
Manufacturer...................... 0xbc00
Burned In MAC Address............. f8:b1:56:69:9d:9b
SOC Version....................... BCM56342_A0
HW Version........................ 5
CPLD Version...................... 13

unit active      backup      current-active next-active
---- ----------- ----------- -------------- --------------
1    6.2.0.5     6.2.0.5     7.28.16.0      6.2.0.5
```

You can currently download system images to the switch by using TFTP, or by copying files to and from a USB Flash drive that is plugged into the USB port on the front panel of the switch as shown in the following section.

5.2 Install DNOS-OF

To install from a TFTP server, copy the DNOS-OF image to the download directory of your TFTP server, then from the terminal or serial console of the switch use the "copy" command as shown below. This example puts the firmware into the backup partition on the switch:

ZBA123_console> copy tftp://<IP address of TFTP server>/DNOS-OF-xx.yy.zz.bb.stk <backup>

To install from USB, plug the USB stick containing the image into the switch and use the following command:

ZBA123_console> copy usb://<path to image>/DNOS-OF-xx.yy.zz.bb.stk <backup>

Once the file is copied, point the boot loader to the partition where the DNOS-OF firmware image was copied by using the "boot" command as shown below. This example points the boot loader to the backup partition where the new DNOS-OF image was just copied to and then issues a reload to boot into the new firmware:

ZBA123_console> boot system <backup>
ZBA123_console> reload

5.3 Configure Management Access

You can use any of the following methods to manage the switch and access the CLI:

- Telnet client
- SSH client
- direct serial console connection

The CLI syntax and semantics for all of the CLI commands are shown in Appendix A.

DNOS-OF comes initially configured with DHCP enabled, so if it is connected to a DHCP enabled network it will pull the management port values from the DHCP server. You can see the initial management configuration by using the show ip command as shown below:

```
FJ6K0Z1_console> show ip
    Incomplete command!
possible subcommands:
    address                 display management ip address
    gateway                 display management ip gateway
    ssh                     display ssh server status
    syslog                  show system syslog information
    telnet                  show telnet service status

FJ6K0Z1_console> show ip address
address:  198.18.3.119/24
FJ6K0Z1_console> show ip gateway
gateway:  198.18.3.254
```

The management access can be configured using the following commands:

```
FJ6K0Z1_console> set ip
    set ip <subcommand>
possible subcommands:
    address                 set ip address for the management port
    dhcp                    enable the dhcp client service on the management port
    gateway                 set the default gateway for the management port

FJ6K0Z1_console> set ip address 198.18.3.155/24
FJ6K0Z1_console> set ip gateway 198.18.3.254
```

You can also see the management configuration information that is stored in the switch configuration and shown in the running configuration:

```
"Management Interface":
{
   "dhcp": false,
   "ip address": "172.25.11.94/27",
   "ip gateway": "172.25.11.254"
},
"SSH Service":
{
   "enabled": false
},
"Telnet Service":
{
   "enabled": true
}
```

5.4 Configure Controller Communications Channel

Since it is a pure OpenFlow switch, before the DNOS-OF switch can be used to control any switch traffic a communications channel to the OpenFlow controller must be set up. This is done using the following command:

set openflow controller <IP address of controller> <IP port of controller>

Example use and expected output is shown below:

```
FJ6K0Z1_console> set openflow controller 198.18.3.114
663308-12 02:36:06.760361 of-switch: MSG: src/of-
switch/OF_SDNControllerInterface.cpp:HandleSetOpenFlowController:Adding controller 198.18.3.114:6633,
(TCP connection), to configuration
08-12 02:36:06.760586 ofconnectionmanager: INFO: Added remote connection: 198.18.3.114:6633
08-12 02:36:06.762918 ofconnectionmanager: INFO: cxn 198.18.3.114:6633: DISCONNECTED->CONNECTING
FJ6K0Z1_console> 08-12 02:36:06.800228 ofconnectionmanager: INFO: cxn 198.18.3.114:6633: CONNECTING-
>HANDSHAKE_COMPLETE
```

You can also see the controller configuration in the switch using **show running-config**:

```
"OpenFlow Controller":
{
   "connection max retries": 0,
   "connection retry interval": 2000,
   "enabled": false,
   "ip address": "",
   "ip port": 6653,
   "periodic echo ms": 10000,
   "priority": 0,
   "protocol version": "OpenFlow 1.3.4",
   "reset echo count": 3,
   "tls": false
},
```

In DNOS-OF 1.0 only the IP address, port, connection max retries and connection retry interval are able to be explicitly configured by the user, and the enabled state reflects whether there is a valid IP address and port specified, so it is implicitly configured. The remainder of the controller configuration currently takes default values.

The connection max retries and connection retry interval for the controller communication channel are configured using the following commands:

5.5 Verifying the Switch to Controller Communications

Once the firmware is installed, the management access is configured, and the controller access is configured, you can verify the status of the controller connection. This is done using the following command:

show openflow config

Example use and expected output is shown below:

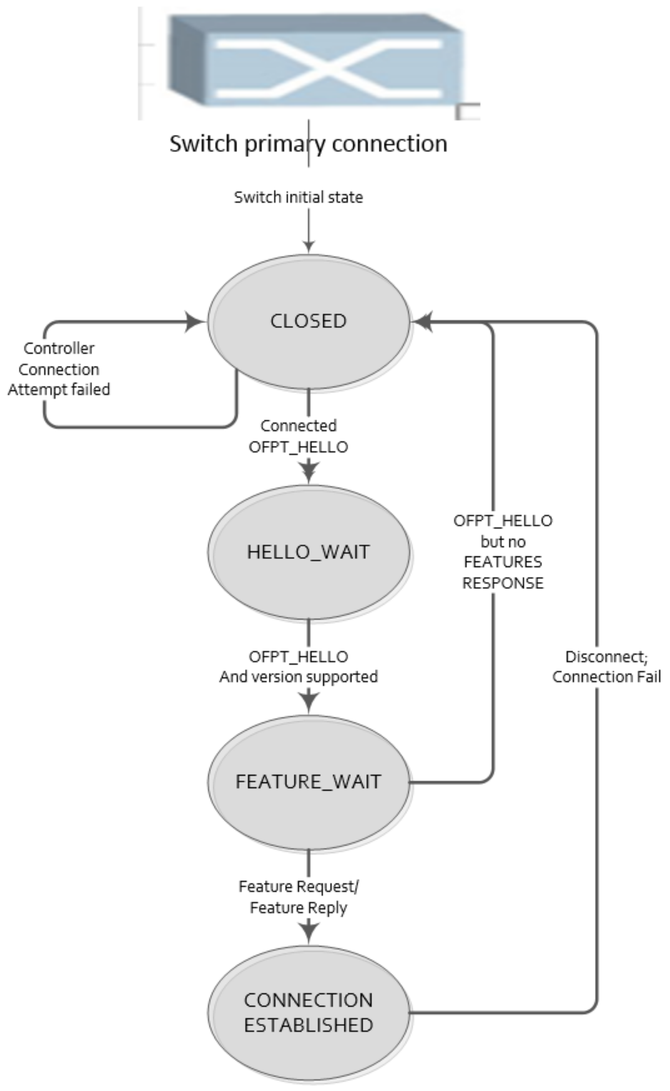
```
FJ6K0Z1_console> show openflow config

OpenFlow Configuration
----------------------
        System Model ID                                   : N3024P
        System Serial Number                              : CN0C3M5M282984CE0129A02
        System MAC Address                                : f8:b1:56:67:99:91
        OpenFlow Protocol Version                         : 1.3.4
        OpenFlow Protocol Connection Type                 : TCP
        OpenFlow Datapath ID                              : 160088049758712
        OpenFlow Datapath Description                     : DNOS-OF: N3024P(FJ6K0Z1)
        OpenFlow SDN Controller Connection Retry Interval : 2000
        OpenFlow SDN Controller Connection Max Retries    : 0
        PID                                               : 971
        Failure Mode                                      : SECURE
        Flow Misses                                       : CONTROLLER

        Flow Tables Synopsis
        --------------------
        Ingress Flow Table (0)
               Current Number of Flows: 0
               Max Number of Flows: 2000
        VLAN Flow Table (10)
               Current Number of Flows: 0
        Termination MAC Flow Table (20)
               Current Number of Flows: 0
               Max Number of Flows: 512
        Unicast Routing Flow Table (30)
               Current Number of Flows: 0
               Max Number of Flows: 40960
        Multicast Routing Flow Table (40)
               Current Number of Flows: 0
               Max Number of Flows: 8191
        Bridging Flow Table (50)
               Current Number of Flows: 0
               Max Number of Flows: 32767
        ACL/Policy Flow Table (60)
               Current Number of Flows: 0
               Max Number of Flows: 7680

        SDN Controllers
        ---------------
        Controller      Role    IP address:     IP Port Status
        ---------------------------------------------------------
        0               Master  198.18.3.114:   6633    HANDSHAKE_COMPLETE, CONNECTED
```

The controller communications channel goes through the OpenFlow connection handshake protocol (shown below) and should end up at HANDSHAKE_COMPLETE/CONNECTED with communications established.


Switch primary connection
Switch initial state
CLOSED
Controller Connection Attempt failed
Connected OFPT_HELLO
HELLO_WAIT
OFPT_HELLO but no FEATURES RESPONSE
OFPT_HELLO And version supported
Disconnect; Connection Fail
FEATURE_WAIT
Feature Request/ Feature Reply
CONNECTION ESTABLISHED

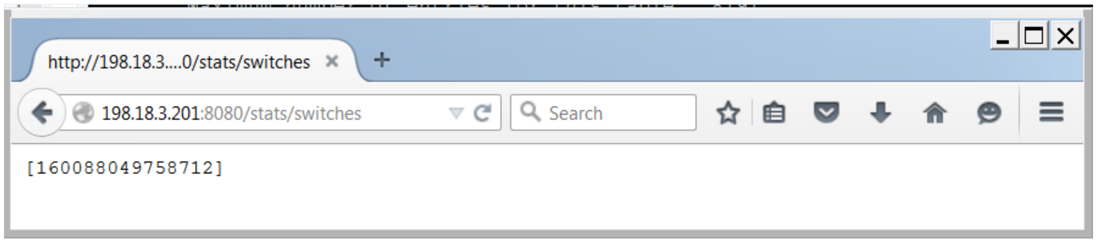
Next you need to verify that the switch shows up on the controller as an OpenFlow node. For the Ryu controller, in order to verify that the controller sees the switch and to obtain the unique datapath ID, you can use the controller's REST API. To start Ryu with the REST API enabled, use the Ryu ofctl_rest.py script included with the controller package. From the Ryu server command line:

> ryu-manager –verbose ofctl_rest.py

You can then use the Ryu REST API's to communicate with the switch via OpenFlow. The first API shown here wil allow you to test the controller connection and the controllers visibility into the switch node:

/stats/switches

Accessing this URL on the controllers IP address through a web server will allow you to see the Datapath ID (DPID) of the switches known to that controller:


http://198.18.3....0/stats/switches
198.18.3.201:8080/stats/switches
Search

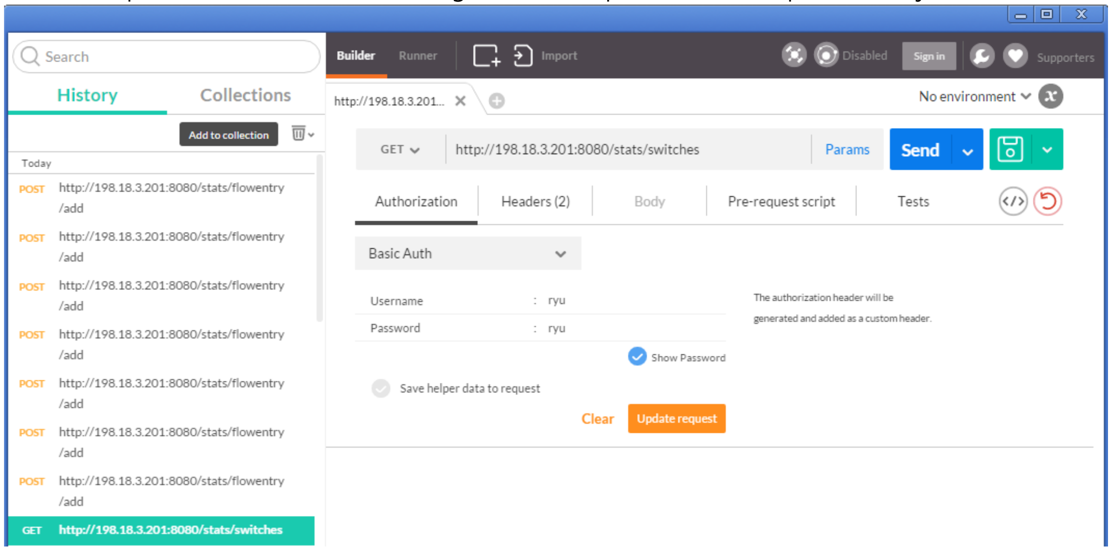
```
[160088049758712]
```

The datapath ID of the switch is shown above as **160088049758712**. This datapath ID uniquely identifies this switch node within the OpenFlow network and is used for all other OpenFlow ccommunications with the switch. Any REST capable interface will be able to access this, but for the following examples we use the Postman REST application.

An example of accessing the switch stats and datapath ID using the Postman REST API application is also show below. First set the REST command type to GET, and use the URL for accessing the stats/switches command call, using your controller IP address for the address, i.e.:

```
http://<my controller IP address>:8080/stats/switches
```

Next set up basic authentication using username/password set up for the Ryu controller when it was installed:



POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
GET http://198.18.3.201:8080/stats/switches

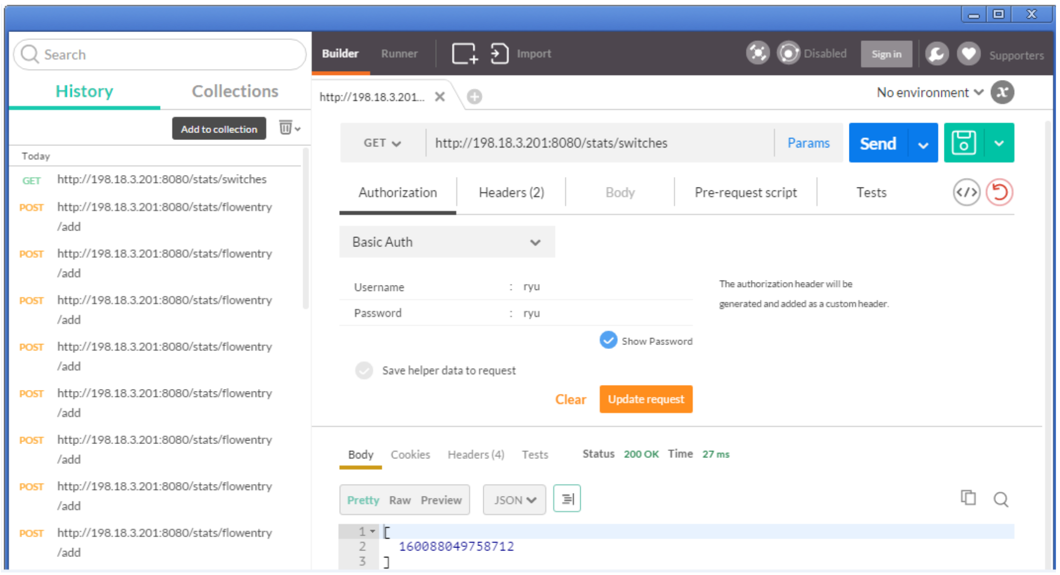
GET http://198.18.3.201:8080/stats/switches

Basic Auth

Username : ryu
Password : ryu

The authorization header will be generated and added as a custom header.

There is no body required for this API command so you can then hit the Send button to send the command to the REST API. The output from the controller is captured and a status shown as seen below where you see "200 OK" as the status of the call. If this call fails there will be an error code where the "200 OK" is, and error status information shown below that, otherwise it returns the list of datapath ID's that the controller knows about. In this case since there is only one controller connected, it shows us our switch datapath ID. This datapath ID will be used for all of the other communications with the switch via the Ryu controller for actions such as adding, modifying and deleting flows, retrieving error and counter status and establishing default switch behaviors.



Today

GET http://198.18.3.201:8080/stats/switches
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add

GET http://198.18.3.201:8080/stats/switches

Basic Auth

Username : ryu
Password : ryu

Show Password ✓

The authorization header will be generated and added as a custom header.

Save helper data to request

Status 200 OK Time 27 ms

```
[
    160088049758712
]
```

If you can read the switch node ID from the controller then you have established communications with the controller and can begin programming it for flows. For a continued example of setting up a simple end to end traffic L2 bridging flow, see Appendix B.

5.6 Logging

Due to the requirement not to impact any of the existing N series firmware, the DNOS-OF switch maintains only a minimal set of in-memory trace logs that are accessible by engineering and support for internal program debugging. All other user configurable logging is intended to use a remote syslog server or, if necessary/desired, the serial console. Later releases of the firmware may add local logging.

The logging system allows for use of the component and verbosity as well as whether it is enabled or disabled, to be controller. It also allows specification of the IP address and port for the syslog service to use for external logging, the parameters needed to set the logging to that service and whether it is enabled or disabled.

The logging levels and components are configurable for either the runtime or stored configured values used by the system. The **set logging level/component** commands are runtime only and do not affect the stored logging settings in the running-config. The **set default logging level/component** commands are used to store a given logging level or component to the running-config.

The default persistent logging values from the switch configuration are shown below:

```
"Default Logging":
{
   "components":
   {
      "API": true,
      "Mapping": true,
      "OFDB": true,
      "datapath": true
   },
   "level": 1
},
"Remote Syslog":
{
   "enabled": false,
   "ip address": "",
   "ip port": 514
},
```

5.6.1 View Logging Configuration

5.6.1.1 show logging

Shows the current state of system logging. There are 8 levels and 4 different components that can be each be enabled or disabled independently for logging. The levels are 0-7 and the components are API, Mapping, OpenFlow Database, and Datapath. These can be seen in the output of the show logging command.

ZYA123_console> show logging

```
Current debug log settings
--------------------------
        Debug logging components available: API, Mapping, OpenFlow Database, Datapath
        Debug logging components enabled: API Mapping OpenFlow Database Datapath
        Debug logging verbosity levels available:
        0 = OFF (FATAL only)
        1 = BASIC (FATAL,ERROR,WARNING)
        2 = INFO (FATAL,ERROR,WARNING,INFO)
        3 = MESSAGE (FATAL,ERROR,WARNING,INFO,MESSAGES)
        4 = VERBOSE (and then some)
        5 = TRACING
        6 = ALMOST ALL (except for in progress debugging)
        7 = ALL
        Current debug logging verbosity level: 1 (BASIC: FATAL, ERROR, WARNING)
```

5.6.1.2 show ip syslog service

To see the state of the syslog service in the logging, use the following command. This shows the default value when the switch is first set up:

```
FJ6K0Z1_console> show ip syslog servers
Remote Syslog server(s): Not enabled
```

5.6.2 Enable or Change Runtime Logging Levels/Components

The runtime logging options can be seen by using the following command:

```
FJ6K0Z1_console> set logging
possible subcommands:
    component                set component for logging
    level                     set level for logging
```

This shows that there are 2 logging subcommands available: level and component.

5.6.2.1 set logging component

Enables or disables logging components for specific internal code paths during runtime. This only changes the values during this run of the system. To make this persistent in the running configuration use the **set default logging component** command. See the examples below:

```
FJ6K0Z1_console> set logging component
set logging component command: valid components are 0-5
logging components currently set to 1,2

FJ6K0Z1_console> set logging component 1
Logging component 1 (API) enabled

FJ6K0Z1_console> set logging component 2
Logging component 2 (OFDB) enabled
```

5.6.2.2 set logging level

Sets the level that log messages are generated for during runtime. This only changes the values during this run of the system. To make this persistent in the running configuration use the **set default logging level** command. See the example below:

```
FJ6K0Z1_console> set logging level
set logging level command: valid levels are 0-7
```

5.6.3 Enable or Change Default Logging Levels/Components

Default logging values allow the users to set persistent values into the switch's running-configuration for the initial state of logging when the switch initially loads. This allows the initial configuration of the logging components to be set to a user configurable value, while the runtime log levels can be changed as needed to debug the system.

5.6.3.1set default logging component

set default logging component
Sets the logging components in the running configuration to be used as the initial configuration when the switch loads. It has the same range of values that the set logging component level has.

5.6.3.2 set default logging level

set default logging level
Sets the logging level in the running configuration to be used as the initial configuration when the switch loads. It has the same range of values that the set logging level has.

5.6.4 Syslog Configuration

As mentioned previously, there are currently no user accessible local logs kept in DNOS-OF, so setting up the external syslog is very important if you want to actually see the log entries you have configured.

ZBA123_console> set ip syslog service <IP address> <IP port> – enables logging to the remote syslog server

Without a valid argument, logging level gives this message:
ZBA123_console> set logging level
set logging level command: valid levels are 0-7
NOT setting to level -1

With a valid argument, logging level gives this message:
ZBA123_console> set logging level 4
logging level set to 4

Sets the address and port for remote syslog from the management port.

5.7 Switch Configuration Storage

There are 3 configuration files used by DNOS-OF to control switch configuration: **startup-config, running-config,** and **backup-config**. These store the configuration data in standard JSON object format. Note that password is stored encrypted.

5.7.1 startup-config

The initial switch configuration is stored in flash, in a file called **startup-config**. When the switch is first booted into DNOS-OF at the start of day, this will contain a set of default values as shown in the example below:

```
JFTP0Z1_console# show startup-config
{
   "Default Logging":
   {
      "components":
      {
         "API": true,
         "Mapping": true,
         "OFDB": true,
         "datapath": true
      },
      "level": 1
   },
   "Management Interface":
   {
      "dhcp": true,
      "ip address": "",
      "ip gateway": ""
   },
   "OpenFlow Controller":
   {
      "connection max retries": 0,
      "connection retry interval": 2000,
      "enabled": false,
      "ip address": "",
      "ip port": 6653,
      "periodic echo ms": 10000,
      "priority": 0,
      "protocol version": "OpenFlow 1.3.4",
      "reset echo count": 3,
      "tls": false
   },
   "Password Hash": "dWAj3KHZgdo",
   "Remote Syslog":
   {
      "enabled": false,
      "ip address": "",
      "ip port": 514
   },
   "SSH Service":
   {
      "enabled": false
   },
   "Telnet Service":
   {
      "enabled": true
   }
}
```

5.7.2 running-config

The switch configuration that it is currently running with is also stored in flash, in a file called **running-config**. If any changes are made to the switch configuration from the time it starts up it will be contained in this file. This running information can be saved as the default starup configuration by using the **write** command. The configuration parameters in the running config and are accessed through the various CLI commands.

Below is an example of what the switch running configuration looks like showing changes from the default values:

```
JFTP0Z1_console# show running-config
{
   "Default Logging":
   {
      "components":
      {
         "API": false,
         "Mapping": true,
         "OFDB": true,
         "datapath": true
      },
      "level": 4
   },
   "Management Interface":
   {
      "dhcp": false,
      "ip address": "172.25.11.94/27",
      "ip gateway": "172.25.11.254"
   },
   "OpenFlow Controller":
   {
      "connection max retries": 0,
      "connection retry interval": 5000,
      "enabled": true,
      "ip address": "172.25.11.93",
      "ip port": 6633,
      "periodic echo ms": 10000,
      "priority": 0,
      "protocol version": "OpenFlow 1.3.4",
      "reset echo count": 3,
      "tls": false
   },
   "Remote Syslog":
   {
      "enabled": true,
      "ip address": "172.25.11.93",
      "ip port": 514
   },
   "SSH Service":
   {
      "enabled": false
   },
   "Telnet Service":
   {
      "enabled": true
   },
   "password": "9CAE2CCD86E7E9EFE4DF7AA04A6F5609BF0956A23425311954FC88C5E413B635"
}
```

5.7.3 backup-config

The switch configuration can be saved off and also stored in flash, in a file called **backup-config**. This can be done using the "copy" command as shown below, and displayed using the **show backup-config** command.

```
JFTP0Z1_console# copy running-config backup-config
JFTP0Z1_console# show running-config
FJ6K0Z1_console> show backup-config

{
   "Default Logging":
   {
      "components":
      {
         "API": true,
         "Mapping": true,
         "OFDB": true,
         "datapath": true
      },
      "level": 1
   },
   "Management Interface":
   {
      "dhcp": true,
      "ip address": "",
      "ip gateway": ""
   },
   "OpenFlow Controller":
   {
      "connection max retries": 0,
      "connection retry interval": 2000,
      "enabled": true,
      "ip address": "198.18.3.201",
      "ip port": 6633,
      "periodic echo ms": 10000,
      "priority": 0,
      "protocol version": "OpenFlow 1.3.4",
      "reset echo count": 3,
      "tls": false
   },
   "Password Hash": "dWAj3KHZgdo",
   "Remote Syslog":
   {
      "enabled": false,
      "ip address": "",
      "ip port": 514
   },
   "SSH Service":
   {
      "enabled": false
   },
   "Telnet Service":
   {
      "enabled": true
   }
}
```

A Appendix - DNOS-OF CLI Command Reference

The following list of CLI commands below represent the those available in the latest DNOS-OF firmware. At any time, to get help on the DNOS-OF firmware from the command line interface, you can use the **help** command, as shown below:

```
JFTP0Z1_console# help

Special keys:
    DEL, BS    .... delete previous character
    Ctrl-A     .... go to beginning of line
    Ctrl-E     .... go to end of line
    Ctrl-F     .... go forward one character
    Ctrl-B     .... go backward one character
    Ctrl-D     .... delete current character
    Ctrl-U, X  .... delete to beginning of line
    Ctrl-K     .... delete to end of line
    Ctrl-W     .... delete previous word
    Ctrl-P     .... go to previous line in history buffer
    Ctrl-R     .... rewrites or pastes the line
    Ctrl-N     .... go to next line in history buffer
    Ctrl-Z     .... return to root command prompt
    Tab        .... command-line completion
    ?          .... list choices

possible subcommands:
    boot                    change boot settings
    clear                   clear system data structures
    copy                    copy a file
    crypto                  system crypto key management
    debug                   debug system components
    delete                  delete a file from local or usb storage
    dir                     show files on local or usb storage
    exit                    exit the current console session
    help                    display help information
    mount                   mount usb device
    no                      clear system settings
    reload                  shutdown and restart the system
    set                     configure system settings
    show                    show system information
    system                  change system settings
    unmount                 unmount usb device
    write                   save configuration to local storage
```

The possible subcommands above represent all first level CLI commands supported by DNOS-OF. There are significantly fewer of these available since much of the CLI that works with legacy or hybrid mode switches is not there in a pure OpenFlow switch.

A.1 Commands

boot top level command

Examples

```
JFTP0Z1_console# boot
    boot <subcommand>
possible subcommands:
    system                select system partition to boot
```

boot system

Use the **boot system** command to specify the system image that the switch loads at bootup.

Syntax

```
boot system <image to boot>
boot system <backup>
```

User Guidelines

Use the **show bootvar** or **show version** commands to find out which images are in the active and backup partitions.

Examples

```
boot system <active>
    Tells switch to load from image in active partition
boot system <backup>
    Tells switch to load from image in backup partition
```

clear top level command

The clear top level command shows what subcommands are available from it.

Examples

```
JFTP0Z1_console# clear
    clear <subcommand>
possible subcommands:
    counters              clear all interface statistics counters
    openflow              clear openflow data structures
```

clear counters

Use the **clear counters** command to clear all interface statistics counters.

Examples

```
JFTP0Z1_console# clear counters
This operation may take a few minutes. The console prompt will return when the operation is complete
Port Stats cleared
```

clear openflow statistics

Use the **clear openflow statistics** command to clear all openflow data structures

Examples

```
JFTP0Z1_console# clear openflow
    clear openflow <subcommand>
possible subcommands:
    statistics             clear openflow statistics counters

JFTP0Z1_console# clear openflow statistics
Clearing all OpenFlow statistics
Clearing OpenFlow flow statistics for all flow tables
Clearing OpenFlow port statistics
Clearing all openflow statistics
Clearing queue statistics for all ports
                Clearing OpenFlow statistics for port 1 for 8 queues
                Clearing OpenFlow statistics for port 2 for 8 queues
                Clearing OpenFlow statistics for port 3 for 8 queues
                Clearing OpenFlow statistics for port 4 for 8 queues
                Clearing OpenFlow statistics for port 5 for 8 queues
                Clearing OpenFlow statistics for port 6 for 8 queues
                Clearing OpenFlow statistics for port 7 for 8 queues
                Clearing OpenFlow statistics for port 8 for 8 queues
                Clearing OpenFlow statistics for port 9 for 8 queues
                Clearing OpenFlow statistics for port 10 for 8 queues
                Clearing OpenFlow statistics for port 11 for 8 queues
                Clearing OpenFlow statistics for port 12 for 8 queues
                Clearing OpenFlow statistics for port 13 for 8 queues
                Clearing OpenFlow statistics for port 14 for 8 queues
                Clearing OpenFlow statistics for port 15 for 8 queues
                Clearing OpenFlow statistics for port 16 for 8 queues
                Clearing OpenFlow statistics for port 17 for 8 queues
                Clearing OpenFlow statistics for port 18 for 8 queues
                Clearing OpenFlow statistics for port 19 for 8 queues
                Clearing OpenFlow statistics for port 20 for 8 queues
                Clearing OpenFlow statistics for port 21 for 8 queues
                Clearing OpenFlow statistics for port 22 for 8 queues
                Clearing OpenFlow statistics for port 23 for 8 queues
                Clearing OpenFlow statistics for port 24 for 8 queues
                Clearing OpenFlow statistics for port 50 for 8 queues
                Clearing OpenFlow statistics for port 51 for 8 queues
                Clearing OpenFlow statistics for port 52 for 8 queues
                Clearing OpenFlow statistics for port 53 for 8 queues
All OpenFlow statistics cleared
```

copy top level command

Use the **copy** command to copy files within the switch and to upload and download files to and from the switch.

User Guidelines

copy <source URL> <destination URL>

Copies a file from source to destination URL.
Valid URLs:
active, backup, backup-config, startup-config
tftp://<server ip>/<file name>
usb://<file name>Valid source and destination URLs:
active, backup, backup-config, startup-config
tftp://<server ip>/<file name>
usb://<file name>

Examples

```
copy tftp://aaa.bbb.cccc.ddd/<file to upload> <backup> - copies the file at the tftp location
specified into the backup partition.
copy <startup-config> <backup-config> - copies the config
```

crypto top level command

```
Example:
JFTP0Z1_console# crypto
    crypto <subcommand>
possible subcommands:
    key                     system crypto key management

JFTP0Z1_console# crypto key
    crypto key <subcommand>
possible subcommands:
    generate                generate ssh server encrytion keys
    zeroize                 remove the ssh server encryption
```

crypto key generate

Use the **crypto key generate** command to generate server encryption keys

Examples

```
JFTP0Z1_console> crypto key generate
```

crypto key zeroize

Use the **crypto key zeroize** command to remove the keys.

```
JFTP0Z1_console> crypto key
    crypto key zeroize
    Remove the SSH server encryption keys.
JFTP0Z1_console> crypto key zeroize
```

debug top level command

Examples

```
FTP0Z1_console# debug
    debug <subcommand>
possible subcommands:
    interface         system debug interface commands...
    ip                      system debug ip commands...
JFTP0Z1_console# debug interface
    debug interface <subcommand>
possible subcommands:
    no                      system debug interface no commands
    shutdown         shutdown a front-panel port
JFTP0Z1_console# debug interface no
    debug interface no <subcommand>
possible subcommands:
    shutdown                re-enable a front-panel port
```

debug interface shutdown

Use the **debug interface shutdown** command to shut down a front panel port.

Examples

```
JFTP0Z1_console# debug interface shutdown
```

debug interface no shutdown

Use the **debug interface no shutdown** command to reenable a front panel port.

Examples

```
JFTP0Z1_console# debug interface no shutdown
```

debug ip

Use the **debug ip** command to show the subcommands available under debug ip.

Examples

```
JFTP0Z1_console# debug ip
    debug ip <subcommand>
possible subcommands:
    ping                   check accessibility of a network node
    traceroute             check routed path of a network node
```

debug ip ping

Use the **debug ip ping** command to check the accessibility of a network node via the management port.

Examples

```
JFTP0Z1_console# debug ip ping <IP address>
```

debug ip traceroute

Check the routed path to another network node via the management port.

Examples

```
JFTP0Z1_console# debug ip traceroute <IP address>
JFTP0Z1_console# debug ip traceroute
BusyBox v1.22.1 (Debian 1:1.22.0-9+deb8u1) multi-call binary.
Usage: traceroute [-46FIldnrv] [-f 1ST_TTL] [-m MAXTTL] [-p PORT] [-q PROBES]
        [-s SRC_IP] [-t TOS] [-w WAIT_SEC] [-g GATEWAY] [-i IFACE]
        [-z PAUSE_MSEC] HOST [BYTES]
Trace the route to HOST
        -4,-6   Force IP or IPv6 name resolution
        -F      Set the don't fragment bit
        -I      Use ICMP ECHO instead of UDP datagrams
        -l      Display the TTL value of the returned packet
        -d      Set SO_DEBUG options to socket
        -n      Print numeric addresses
        -r      Bypass routing tables, send directly to HOST
        -v      Verbose
        -m      Max time-to-live (max number of hops)
        -p      Base UDP port number used in probes (default 33434)
        -q      Number of probes per TTL (default 3)
        -s      IP address to use as the source address
        -t      Type-of-service in probe packets (default 0)
        -w      Time in seconds to wait for a response (default 3)
        -g      Loose source route gateway (8 max)
```

delete top level command

Use the **delete** command to delete files from the local or USB file system. Only certain files can be deleted on the local system, while file on the USB can be deleted.

Examples

```
JFTP0Z1_console# delete
    delete <target URL>
    Deletes a file from local or usb storage.
    Valid URLs:
        backup-config, startup-config
        usb://<file name>
```

dir top level command

Use the **dir** command to list files in the file system. Valid file system areas are :

flash – list contents of the flash file system
usb – list contents of memory plugged into USB slot

Examples
```
JFTP0Z1_console# dir
    dir <subcommand>
possible subcommands:
    flash                   list files on the local flash storage
    usb                     list files on a mounted usb device
```

dir flash

Lists files in the flash file system

Examples
```
JFTP0Z1_console# dir flash
-rw-r--r--    1 root     root         922 Aug  7 10:25 startup-config
```

dir usb

Lists files in the usb file system

Examples
```
JFTP0Z1_console# dir usb
total 0
```

exit top level command

Use the **exit** command to quit a telnet or SSH login session. **NOTE: This command has no effect when connected via the serial console.**

Examples
```
JFTP0Z1_console# exit
```

help top level command

Displays the CLI help screen shown below:

Examples
```
JFTP0Z1_console> help
Special keys:
    DEL, BS    .... delete previous character
    Ctrl-A     .... go to beginning of line
    Ctrl-E     .... go to end of line
    Ctrl-F     .... go forward one character
    Ctrl-B     .... go backward one character
    Ctrl-D     .... delete current character
    Ctrl-U, X  .... delete to beginning of line
    Ctrl-K     .... delete to end of line
    Ctrl-W     .... delete previous word
    Ctrl-P     .... go to previous line in history buffer
    Ctrl-R     .... rewrites or pastes the line
    Ctrl-N     .... go to next line in history buffer
    Ctrl-Z     .... return to root command prompt
    Tab        .... command-line completion
    ?          .... list choices
possible subcommands:
    boot                    change boot settings
    clear                   clear system data structures
    copy                    copy a file
    crypto                  system crypto key management
    debug                   debug system components
    delete                  delete a file from local or usb storage
    dir                     show files on local or usb storage
    exit                    exit the current console session
    help                    display help information
    mount                   mount usb device
    no                      clear system settings
    reload                  shutdown and restart the system
    set                     configure system settings
    show                    show system information
    system                  change system settings
    unmount                 unmount usb device
    write                   save configuration to local storage
```

mount top level command

Displays the available subcommands for the mount command.

Examples

```
JFTP0Z1_console> mount
    mount <subcommand>
possible subcommands:
    usb                     mount usb drive
```

mount usb

Use the **mount usb** command to mount the usb file system.

Examples

mount usb

no top level command

Displays the available subcommands for the no command.

Examples

```
JFTP0Z1_console# no
    no <subcommand>
possible subcommands:
    ip                      clear system ip settings
    openflow                clear system openflow settings
    password                remove the password for remote ssessions
```

no ip subcommand

no ip shows what available subcommands are below that.

Examples

```
JFTP0Z1_console# no ip
    no ip <subcommand>
possible subcommands:
    address
    dhcp                    disable the management port dhcp client
    gateway                 remove the default gateway from the management port
    ssh                     clear system ip ssh settings
    syslog                  clear system ip syslog settings
    telnet                  clear system ip telnet settings
```

no ip address

no ip address removes the static IP address from the management interface.

Examples

```
JFTP0Z1_console# no ip address
    no ip address
    Delete ip address from management interface
```

no ip dhcp

This command shuts down the DHCP client on the switch and disables it in the running configuration. Use the **set ip dhcp** command to enable it.

Examples

```
FJ6K0Z1_console> no ip dhcp
    no ip dhcp
    Disable the management port dhcp client.
```

no ip gateway

This command removes the default gateway from the management port.

Examples

```
FJ6K0Z1_console> no ip gateway
    no ip gateway
    Remove the default gateway from the management port.
```

no ip ssh server

This command disables the SSH server on the management port.

Examples

```
FJ6K0Z1_console> no ip ssh
no ip ssh <subcommand>
possible subcommands:
server                  disable the ssh server on the management port
```

no ip syslog service

This command removes remote syslog service settings.

Examples

```
FJ6K0Z1_console> no ip syslog
    no ip syslog <subcommand>
possible subcommands:
    service                 remove remote syslog service settings
```

no ip telnet server

This command disables the telnet server on the management interface.

Examples

```
FJ6K0Z1_console> no ip telnet
    no ip telnet <subcommand>
possible subcommands:
    server                  disable the telnet server on the management interface
```

no openflow controller

This command removes an openflow controller channel.

Examples

```
FJ6K0Z1_console> no openflow controller
    no openflow controller <ip address> <port>
    Remove an OpenFlow controller channel.
```

no password

Removes the password for remote sessions.

Examples

```
FJ6K0Z1_console> no password
    no password
    Remove the password for remote ssessions.
```

reload

Use the **reload** command to reboot the switch.

Examples

```
FJ6K0Z1_console> reload
    reload
    Shuts down and restarts the system. The reload command takes no arguments.
```

set top level command

Displays the available subcommands for the set command.

Examples

```
FJ6K0Z1_console> set
    set <subcommand>
possible subcommands:
    default                 set system default logging settings
    ip                      set system ip settings
    logging                 set system logging settings
    openflow                set system openflow settings
    password                set password for remote ssessions
```

set clock

Sets the system date and time.Use the **set default logging components** command to change the default logging components that are enabled when the system starts.

Examples

```
FJ6K0Z1_console> set clock
Enter date:
Enter time:
```

set default logging subcommands

Allows the user to change the default logging level and component that are loaded when the system initializes. These are written to the runtime configuration as seen here:

```
"Default Logging":
  {
     "components":
     {
        "API": true,
        "Mapping": true,
        "OFDB": true,
        "datapath": true
     },
     "level": 1
  },
```

Examples

```
FJ6K0Z1_console> set default
    set default <subcommand>
    Set system default logging settings.
possible subcommands:
    logging                 set system default logging settings
FJ6K0Z1_console> set default logging
    set default logging <subcommand>
possible subcommands:
    component      enable or disable a specific component as a default for logging in the running
configuration
    level          set the logging level stored as a default in the running configuration
```

set default logging components

Use the **set default logging components** command to change the default logging components that are enabled when the system starts.

Examples

```
FJ6K0Z1_console> set default logging component
    set default logging component <component>
    Set default component for logging where component is:
        0 = NONE
        1 = API
        2 = MAPPING
        3 = OFDB
        4 = DATAPATH
        5 = ALL
```

set default logging level

Use the **set default logging level** command to change the default logging level that the switch uses when the system starts.

Examples

```
FJ6K0Z1_console> set default logging level

    set default logging level <default level>

    Set the default logging level. This is the persistent log level that the switch initializes to
    by default when it starts up.
    The level can be:
        0 = OFF (FATAL only)
        1 = BASIC (FATAL,ERROR,WARNING)
        2 = INFO (FATAL,ERROR,WARNING,INFO)
        3 = MESSAGE (FATAL,ERROR,WARNING,INFO,MESSAGEs)
        4 = VERBOSE (and then some)
        5 = TRACING
        6 = ALMOST ALL (except for in progress debugging)
        7 = ALL
```

set ip subcommands

Displays the subcommands supported under set ip.

Examples

```
FJ6K0Z1_console> set ip
    set ip <subcommand>
possible subcommands:
    address                 set ip address for the management port
    dhcp                    enable the dhcp client service on the management port
    gateway                 set the default gateway for the management port
    ssh                     set system ip ssh settings
    syslog                  set system ip syslog settings
    telnet                  set system ip telnet settings
```

set ip address

Use to set the static IP address of the switch.

Examples

```
FJ6K0Z1_console> set ip address
    set ip address <ip address>/<masklength>
    Set ip address for the management port. This command uses CIDR notation.
```

set ip dhcp

Use to enable DHCP on the switch and store this information into the switch configuration. Use the **no ip dhcp** command to disable it.

Examples

```
FJ6K0Z1_console> set ip dhcp
    set ip dhcp
    Enable the DHCP client service.
```

set ip gateway

Use to set the default IP gateway on the management port and store this information to the switch configuration.

Examples

```
FJ6K0Z1_console> set ip gateway
    set ip gateway <gateway address>
    Set the default_gateway for the management port.
```

set ip ssh server

Use to enable the ssh server on the management port and store that state to the switch configuration.

Examples

```
FJ6K0Z1_console> set ip ssh ?
    set ip ssh <subcommand>
possible subcommands:
    server                enable the ssh server on the management port
```

set ip syslog service

Use to enable the syslog service on the management port and store that state to the switch configuration.

Examples

```
FJ6K0Z1_console> set ip syslog service ?
    set ip syslog service <ip address> <port>
    Set the address and port for remote syslog from the management port. The port
    is optional and will default to 514.
```

set ip telnet server

Used to enable the telnet service on the management port and store that state to the switch configuration.

Examples

```
FJ6K0Z1_console> set ip telnet server
    set ip telnet server
    Enable the IP telnet server on the management port.
```

set logging subcommand

Use to enable the telnet service on the management port and store that state to the switch configuration.

Examples

```
FJ6K0Z1_console> set logging
    set logging <subcommand>
possible subcommands:
    component             set component for logging
    level
```

set logging component

Use this API to set the runtime logging components on the switch until rebooted. Use the "**set default logging …"** command to store the setting into the system flash running-donfiguration

Examples

```
FJ6K0Z1_console> set logging
    set logging <subcommand>
possible subcommands:
    component             set component for logging
    level
FJ6K0Z1_console> set logging component
set logging component command: valid components are 0-5
FJ6K0Z1_console> help set logging component
    set logging component <component>
    Set component for logging where component is:
        0 = NONE
        1 = API
        2 = MAPPING
        3 = OFDB
        4 = DATAPATH
        5 = ALL
```

set logging level

Use this API to set the runtime logging level for the switch until rebooted. Use the "**set default logging …"** command to store the setting into the system flash running-donfiguration

Examples
```
FJ6K0Z1_console> set logging level
    set logging level <level>
    Set the logging level. Values above 3 should probably not be turned on for
    serial port backed IO like screen consoles/maintenance ports.
    The level can be:
        0 = OFF (FATAL only)
        1 = BASIC (FATAL,ERROR,WARNING)
        2 = INFO (FATAL,ERROR,WARNING,INFO)
        3 = MESSAGE (FATAL,ERROR,WARNING,INFO,MESSAGEs)
        4 = VERBOSE (and then some)
        5 = TRACING
        6 = ALMOST ALL (except for in progress debugging)
        7 = ALL
```

set openflow subcommand

Displays available subcommands.

Examples
```
FJ6K0Z1_console> set openflow
    set openflow <subcommand>
possible subcommands:
    connection              set system openflow connection settings
    controller              create an openflow controller channel
```

```
In the switch configuration:
   "OpenFlow Controller":
   {
      "connection max retries": 0,
      "connection retry interval": 5000,
      "enabled": true,
      "ip address": "172.25.11.93",
      "ip port": 6633,
      "periodic echo ms": 10000,
      "priority": 0,
      "protocol version": "OpenFlow 1.3.4",
      "reset echo count": 3,
      "tls": false
   },
```

set openflow connection maxretries

Use this API to set the OpenFlow max retries, the number of times that the switch will resend control frames to try and reestablish connection with the controller. This is a persistent value that is stored in the switch running-configuration and can be saved to the backup-configuration.

Examples
```
FJ6K0Z1_console> set openflow connection maxretries
```

set openflow connection retryinterval

Use this API to set the OpenFlow retry interval used with the retries above. This is the interval at which frequency the switch will look for the controller if it loses communications. The switch will resend control frames to try and reestablish connection with the controller. This is a persistent value that is stored in the switch running-configuration and can be saved to the backup-configuration.

Examples
```
FJ6K0Z1_console> set openflow connection maxretries
```

set openflow controller <ip address> <ip port>

Use the **set openflow controller** command to point the DNOS-OF switch to where the controller it is supposed to connect to resides.

Examples

```
FJ6K0Z1_console> set openflow controller 172.25.11.93 6633
08-07 22:00:22.036504 ofconnectionmanager: INFO: cxn 172.25.11.93:6633: DISCONNECTED
08-07 22:00:22.078219 ofconnectionmanager: INFO: cxn 172.25.11.93:6633: CONNECTING->HANDSHAKE_COMPLETE
```

set password command

Use the **set password** command to change the password stored in the running configuration and used to authenticate SSH sessions.

Examples

```
FJ6K0Z1_console> set password
Enter password: 123456
```

show subcommand

Lists the subcommands that are available for the show command.

Examples

```
FJ6K0Z1_console> show
    show <subcommand>
possible subcommands:
    backup-config          display the backup configuration file
    bootvar                display system software boot settings
    clock                  show current time of day
    crypto                 show system crypto information
    interfaces             show system interfaces information
    ip                     show system ip information
    logging                show logging information
    memory                 show system memory information
    openflow               show system openflow information
    power                  show poe power information
    process                show system process information
    running-config         show the running configuration
    startup-config         display the startup configuration file
    statistics             show system statistics information
    switch                 show switch information
    system                 show system information
    tech-support           print the output of multiple show commands
    version                display system hardware and software versions
```

show backup-config command

Shows the data stored in the backup configuration

Examples

```
FJ6K0Z1_console> show backup-config
{
   "Default Logging":
   {
      "components":
      {
         "API": true,
         "Mapping": true,
         "OFDB": true,
         "datapath": true
      },
      "level": 1
   },
   "Management Interface":
   {
      "dhcp": true,
      "ip address": "",
      "ip gateway": ""
   },
   "OpenFlow Controller":
   {
      "connection max retries": 0,
      "connection retry interval": 2000,
      "enabled": true,
      "ip address": "198.18.3.201",
      "ip port": 6633,
      "periodic echo ms": 10000,
      "priority": 0,
      "protocol version": "OpenFlow 1.3.4",
      "reset echo count": 3,
      "tls": false
   },
   "Password Hash": "dWAj3KHZgdo",
   "Remote Syslog":
   {
      "enabled": true,
      "ip address": "172.25.2.51",
      "ip port": 514
   },
   "SSH Service":
   {
      "enabled": true
   },
   "Telnet Service":
   {
      "enabled": true
   }
}
```

show bootvar command

Displays the environment variables that the system is set up to use in the boot loader when it starts up. Note that similar information can be retrieved from the "show version" platform CLI command.

Examples

```
FJ6K0Z1_console> show bootvar

Images currently available in Flash

unit active      backup      current-active next-active
---- ----------- ----------- -------------- --------------
1    8.10.2.0     6.1.0.6     8.10.2.0       8.10.2.0
```

show clock command

Shows the current time and date on the system.

Examples

```
FJ6K0Z1_console> show clock

Sat Aug 15 01:53:33 2015
```

show crypto key command

Prints the SSH server encryption key. Note that the "crypto key generate" command allows this key to be created, while the "crypto key zeroize" command clears it.

Examples

```
FJ6K0Z1_console> show crypto ?
    show crypto <subcommand>
possible subcommands:
    key                     print the ssh server encryption keys
FJ6K0Z1_console> show crypto key

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINSybSubK7fgoCXmFKh7PbhxpsP7R1hiBuEdt4zsLyH/
ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBOtADcORH+KoLRSaULZ25AZ2B2lVpPLckJAJEqCWPtqplE+r
XjAoPuGV9f/W5bXARY6sqkb6jht1KiA2H8Hb/Eo=
```

show interfaces top level command

Lists the subcommands available under show interfaces.

Examples

```
FJ6K0Z1_console> show interfaces ?
    show interfaces <subcommand>
possible subcommands:
    configuration           show interfaces configuration information
    status                  show interfaces status information
    switchport              show operational status of an interface
```

show interfaces configuration command

Shows the interfaces configuration information.

Examples

```
FJ6K0Z1_console> show interfaces
    show interfaces <subcommand>
possible subcommands:
    configuration           show interfaces configuration information
    status                  show interfaces status information
    switchport              show operational status of an interface
FJ6K0Z1_console> show interfaces configuration ?
    show interfaces configuration
    Show configuration information for all available interfaces.

FJ6K0Z1_console> show interfaces configuration
Port  Duplex  Speed     Neg    MTU  Admin State
----  ------  -----  ------  -----  -----------
   1    half      0    auto  16356      enabled
   2    half      0    auto  16356      enabled
   3    half      0    auto  16356      enabled
   4    half      0    auto  16356      enabled
   5    half      0    auto  16356      enabled
   6    half      0    auto  16356      enabled
   7    half      0    auto  16356      enabled
   8    half      0    auto  16356      enabled
   9    half      0    auto  16356      enabled
  10    half      0    auto  16356      enabled
  11    half      0    auto  16356      enabled
  12    half      0    auto  16356      enabled
  13    half      0    auto  16356      enabled
  14    half      0    auto  16356      enabled
  15    half      0    auto  16356      enabled
  16    half      0    auto  16356      enabled
  17    half      0    auto  16356      enabled
  18    half      0    auto  16356      enabled
  19    half      0    auto  16356      enabled
  20    half      0    auto  16356      enabled
  21    half      0    auto  16356      enabled
  22    half      0    auto  16356      enabled
  23    full   1000    auto  16356      enabled
  24    full   1000    auto  16356      enabled
  50    full  10000  noauto  16356      enabled
  51    full  10000  noauto  16356      enabled
```

NOTE: the port numbering shown with a gap in port numbers is consistent with the way this is done in traditional PowerConnect images, however this diverges from the way ports are handled in both OpenFlow and in many other switch vendors. The port naming reflected back in the GetFeatures OpenFlow response has the ports named as seen in traditional PowerConnect (i.e. Te 2/0/1, Ge 1/0/1, etc.)

show interfaces status command

Shows the interfaces status information.

Examples

```
FJ6K0Z1_console> show interfaces status
    Show interfaces status  Show status information for all available interfaces.

FJ6K0Z1_console> show interfaces status
Port  Duplex  Speed     Neg  Link  Flow Ctrl
----  ------  -----  ------  ----  ---------
  1    half       0   auto   down
  2    half       0   auto   down
  3    half       0   auto   down
  4    half       0   auto   down
  5    half       0   auto   down
  6    half       0   auto   down
  7    half       0   auto   down
  8    half       0   auto   down
  9    half       0   auto   down
 10    half       0   auto   down
 11    half       0   auto   down
 12    half       0   auto   down
 13    half       0   auto   down
 14    half       0   auto   down
 15    half       0   auto   down
 16    half       0   auto   down
 17    half       0   auto   down
 18    half       0   auto   down
 19    half       0   auto   down
 20    half       0   auto   down
 21    half       0   auto   down
 22    half       0   auto   down
 23    full    1000   auto   down
 24    full    1000   auto   down
 50    full   10000  noauto  down
 51    full   10000  noauto  down
```

show interfaces switchport command

Shows the switchport information.

Examples

```
FJ6K0Z1_console> show interfaces switchport
    show interfaces switchport <port number>
    Shows the operational status of an interface by port number.
FJ6K0Z1_console> show interfaces switchport 1
Port 1 is down, MTU 16356 bytes, Half-duplex, 0 Mb/s
L2 Switched: ucast: 0 pkt, 0 bytes - mcast: 0 pkt
Received: 0 broadcasts, 0 multicasts, 0 unicasts
    0 packets input, 0 bytes, 0 drops
    0 jabbers, 0 runts, 0 giants, 0 alignments
    0 MAC Errors, 0 FCS, 0 overrun, 0 not forwarded
Transmitted: 0 broadcasts, 0 multicasts, 0 unicasts
    0 packets output, 0 bytes, 0 discards
    0 output_errors, 0 collisions
    0 multiple collisions, 0 excessive collisions
```

show memory cpu command

Shows the statistics for the DNOS-OF process system memory use in Linux.

Examples

```
FJ6K0Z1_console> show memory cpu
    show memory cpu
    Show system memory use.
FJ6K0Z1_console> show memory cpu
Total Memory................ 1032188  bytes
Available Memory Space...... 752868  bytes
```

show openflow top level command

Displays the show commands available for OpenFlow

Examples

```
FJ6K0Z1_console> show openflow
    Incomplete command!
possible subcommands:
    config                  show openlow configuration
    flows                   show openflow flows
    statistics              show openflow statistics
    tables                  show openflow table information
```

show openflow config command

Shows information about the openflow configuration and processes, as well as information about the OpenFlow controller interfaces active in DNOS-OF.

Examples

```
FJ6K0Z1_console> show openflow config
OpenFlow Configuration
----------------------
  System Model ID                                : N3024P
  System Serial Number                           : CN0C3M5M282984CE0129A02
  System MAC Address                             : f8:b1:56:67:99:91
  OpenFlow Protocol Version                      : 1.3.4
  OpenFlow Protocol Connection Type              : TCP
  OpenFlow Datapath ID                           : 160088049758712
  OpenFlow Datapath Description                  : DNOS-OF:N3024P(FJ6K0Z1)
  OpenFlow SDN Controller Connection Retry Interval : 2000
  OpenFlow SDN Controller Connection Max Retries    : 0
  PID                                            : 644
  Failure Mode                                   : SECURE
  Flow Misses                                    : CONTROLLER
  Flow Tables Synopsis
  --------------------
            Ingress Flow Table (0)
                    Current Number of Flows: 0
                    Max Number of Flows: 2000
            VLAN Flow Table (10)
                    Current Number of Flows: 0
                    Max Number of Flows: 12288
            Termination MAC Flow Table (20)
                    Current Number of Flows: 0
                    Max Number of Flows: 512
            Unicast Routing Flow Table (30)
                    Current Number of Flows: 0
                    Max Number of Flows: 40960
            Multicast Routing Flow Table (40)
                    Current Number of Flows: 0
                    Max Number of Flows: 8191
            Bridging Flow Table (50)
                    Current Number of Flows: 0
                    Max Number of Flows: 32767
            ACL/Policy Flow Table (60)
                    Current Number of Flows: 0
                    Max Number of Flows: 7680

    `SDN Controllers
   Controller      Role    IP address:     IP Port Status
    ------------------------------------------------------
    0               Master  198.18.3.201:   6633    DISCONNECTED
```

show openflow tables command

Shows information about the openflow tables available, supported in DNOS-OF.

Examples

```
FJ6K0Z1_console> show openflow tables
Flow tables and table ID Summary
--------------------------------
Ingress Port Table (0) [NumberOfEntries=0, MaxEntries=2000] [Available to OpenFlow controller]
Port DSCP Trust Table (5) [NumberOfEntries=0, MaxEntries=8127]
Port PCP Trust Table (6) [NumberOfEntries=0, MaxEntries=1023]
Tunnel DSCP Trust Table (7) [NumberOfEntries=0, MaxEntries=0]
Tunnel PCP Trust Table (8) [NumberOfEntries=0, MaxEntries=0]
Injected OAM Table (9) [NumberOfEntries=0, MaxEntries=2048]
VLAN Table (10) [NumberOfEntries=0, MaxEntries=12288][Available to OpenFlow controller]
VLAN 1 Table (11) [NumberOfEntries=0, MaxEntries=12288]
MAINTENANCE POINT Table (12) [NumberOfEntries=0, MaxEntries=0]
MPLS L2 Port Table (13) [NumberOfEntries=0, MaxEntries=0]
MPLS QoS DSCP Trust Table (15) [NumberOfEntries=0, MaxEntries=0]
MPLS QoS PCP Trust Table (16) [NumberOfEntries=0, MaxEntries=0]
Termination MAC Table (20) [NumberOfEntries=0, MaxEntries=512]  [Available to OpenFlow controller]
MPLS 0 Table (23) [NumberOfEntries=Unknown, MaxEntries=Unknown]
MPLS 1 Table (24) [NumberOfEntries=0, MaxEntries=0]
MPLS 2 Table (25) [NumberOfEntries=Unknown, MaxEntries=Unknown]
MPLS-TP MAINTENANCE POINT Table (26) [NumberOfEntries=0, MaxEntries=0]
BFD Table (27) [NumberOfEntries=0, MaxEntries=8127]
Unicast Routing Table (30) [NumberOfEntries=0, MaxEntries=40960][Available to OpenFlow controller]
Multicast Routing Table (40) [NumberOfEntries=0, MaxEntries=8191][Available to OpenFlow controller]
Bridging Table (50) [NumberOfEntries=0, MaxEntries=32767][Available to OpenFlow controller]
L2 Policer Table (55) [NumberOfEntries=0, MaxEntries=0]
L2 Policer Actions Table (56) [NumberOfEntries=0, MaxEntries=0]
ACL Policy Table (60) [NumberOfEntries=0, MaxEntries=7680]  [Available to OpenFlow controller]
Color Based Actions Table (65) [NumberOfEntries=0, MaxEntries=100]
Egress VLAN Table (210) [NumberOfEntries=0, MaxEntries=4096]
Egress VLAN 1 Table (211) [NumberOfEntries=0, MaxEntries=4096]
Egress MAINTENANCE POINT Table (226) [NumberOfEntries=0, MaxEntries=0]

No group entries found.
```

show openflow flows command

Shows information about the entire contents of the openflow flow database.

Examples

To show information about the entire contents of the openflow flow database omit the “table” objecto:

```
FJ6K0Z1_console> show openflow flows

Showing openflow flow entries for all tables
Ingress Port Table (0) Flow Entries
----------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 2000
        Number of entries actually found = 0
VLAN Table (10) Flow Entries for
--------------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 12288
        Number of entries actually found = 0
Termination MAC Table (20) Flow Entries
----------------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 512
        Number of entries actually found = 0
Unicast Routing Table (30) Flow Entries
----------------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 40960
        Number of entries actually found = 0
Multicast Routing Table (40) Flow Entries
-----------------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 8191
        Number of entries actually found = 0
Bridging Table (50) Flow Entries
--------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 32767
        Number of entries actually found = 0
ACL Policy Table (60) Flow Entries
----------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 7680
        Number of entries actually found = 0
```

To show information about a single flow table in the openflow flow database use the following syntax. These particular commands shows a request of the VLAN fow table and the Termination MAC flow table.

```
FJ6K0Z1_console> show openflow flows 10

Showing openflow flow entries for table 10, VLAN Table
VLAN Table (10) Flow Entries
--------------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 12288
        Number of entries actually found = 0

FJ6K0Z1_console> show openflow flows 20

Showing openflow flow entries for table 20, Termination MAC Table
Termination MAC Table (20) Flow Entries
----------------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 512
        Number of entries actually found = 0
```

show openflow statistics command

Shows what openflow statistics are available from DNOS-OF given certain qualifier arguments. See below.

Examples

```
FJ6K0Z1_console> show openflow statistics
    show openflow statistics <statistic type> <optional port number>
    Current supported OpenFlow statistics
        all   = ALL Statistics
        flow  = Flow Statistics
        port  = Port Statistics
        queue = Queue Statistics
        group = Group Statistics
```

This shows that there are 4 categories you can enable/disable completely and mutually exclusive of each other to filter what statistics are displayed. You can also filter by port number.

With either no argument por ALL shown below, there is a large amount of statistics data that comes back:

```
=======
FJ6K0Z1_console> show openflow statistics
Showing statistics for all ports
Showing statistics for port 255
OpenFlow Flow Statistics by Table
---------------------------------
Ingress Port Table (0) Flow Statistics
Ingress Port Table (10) Flow Statistics
Ingress Port Table (20) Flow Statistics
Ingress Port Table (30) Flow Statistics
Ingress Port Table (40) Flow Statistics
Ingress Port Table (50) Flow Statistics
Ingress Port Table (60) Flow Statistics
OpenFlow Port Statistics
------------------------
Port 1: RxPackets=0, RxByte=0, TxPackets=0, TxBytes=0
        RxErrors=0, RxDrops=0, TxDrops=0
        RxFrameErrors=0, RxOverErrors=0, RxCRCErrors=0, Collisions=0, Duration (seconds)=0
Port 2: RxPackets=0, RxByte=0, TxPackets=0, TxBytes=0
        RxErrors=0, RxDrops=0, TxDrops=0
        RxFrameErrors=0, RxOverErrors=0, RxCRCErrors=0, Collisions=0, Duration (seconds)=0
.
. <Note that display of some port data was removed for illustration purposes>
.
Port 24: RxPackets=0, RxByte=0, TxPackets=0, TxBytes=0
        RxErrors=0, RxDrops=0, TxDrops=0
        RxFrameErrors=0, RxOverErrors=0, RxCRCErrors=0, Collisions=0, Duration (seconds)=0
Port 50: RxPackets=0, RxByte=0, TxPackets=0, TxBytes=0
        RxErrors=0, RxDrops=0, TxDrops=0
        RxFrameErrors=0, RxOverErrors=0, RxCRCErrors=0, Collisions=0, Duration (seconds)=0
Port 51: RxPackets=0, RxByte=0, TxPackets=0, TxBytes=0
        RxErrors=0, RxDrops=0, TxDrops=0
        RxFrameErrors=0, RxOverErrors=0, RxCRCErrors=0, Collisions=0, Duration (seconds)=0

OpenFlow Queue Statistics by Port
---------------------------------
        OpenFlow statistics for port 1, 8 queues
Port:1, queue:0 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:1, queue:1 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:1, queue:2 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:1, queue:3 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:1, queue:4 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:1, queue:5 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:1, queue:6 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:1, queue:7 - txBytes=0, txPackets=0, durationInSeconds=97876
        OpenFlow statistics for port 2, 8 queues
Port:2, queue:0 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:2, queue:1 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:2, queue:2 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:2, queue:3 - txBytes=0, txPackets=0, durationInSeconds=97876
```

```
Port:2, queue:4 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:2, queue:5 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:2, queue:6 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:2, queue:7 - txBytes=0, txPackets=0, durationInSeconds=97876
.
. <Note that display of some port data was removed for illustration purposes>
.
        OpenFlow statistics for port 24, 8 queues
Port:24, queue:0 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:24, queue:1 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:24, queue:2 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:24, queue:3 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:24, queue:4 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:24, queue:5 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:24, queue:6 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:24, queue:7 - txBytes=0, txPackets=0, durationInSeconds=97876
        OpenFlow statistics for port 50, 8 queues
Port:50, queue:0 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:50, queue:1 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:50, queue:2 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:50, queue:3 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:50, queue:4 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:50, queue:5 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:50, queue:6 - txBytes=0, txPackets=0, durationInSeconds=97876
Port:50, queue:7 - txBytes=0, txPackets=0, durationInSeconds=97876
        OpenFlow statistics for port 51, 8 queues
Port:51, queue:0 - txBytes=0, txPackets=0, durationInSeconds=97875
Port:51, queue:1 - txBytes=0, txPackets=0, durationInSeconds=97875
Port:51, queue:2 - txBytes=0, txPackets=0, durationInSeconds=97875
Port:51, queue:3 - txBytes=0, txPackets=0, durationInSeconds=97875
Port:51, queue:4 - txBytes=0, txPackets=0, durationInSeconds=97875
Port:51, queue:5 - txBytes=0, txPackets=0, durationInSeconds=97875
Port:51, queue:6 - txBytes=0, txPackets=0, durationInSeconds=97875
Port:51, queue:7 - txBytes=0, txPackets=0, durationInSeconds=97875
OpenFlow Group Statistics
-------------------------
No group entries found.
```

You can also filter output by specific object and ports, as in the following:

```
FJ6KOZ1_console> show openflow
    Incomplete command!
possible subcommands:
    config                  show openlow configuration
    flows                   show openflow flows
    statistics              show openflow statistics
    tables                  show openflow table information

FJ6KOZ1_console> show openflow statistics ?
    show openflow statistics <statistic type> <optional port number>

    Current supported OpenFlow statistics
        all   = ALL Statistics
        flow  = Flow Statistics
        port  = Port Statistics
        queue = Queue Statistics
        group = Group Statistics

FJ6KOZ1_console> show openflow statistics ALL port 2

Showing statistics for all ports
Showing statistics for port 255
OpenFlow Flow Statistics by Table
---------------------------------
Ingress Port Table (0) Flow Statistics (port 2)
Ingress Port Table (10) Flow Statistics (port 2)
Ingress Port Table (20) Flow Statistics (port 2)
Ingress Port Table (30) Flow Statistics (port 2)
Ingress Port Table (40) Flow Statistics (port 2)
Ingress Port Table (50) Flow Statistics (port 2)
Ingress Port Table (60) Flow Statistics (port 2)
```

show power inline command

Show power inline settings and usage from the PoE controller.

Examples

```
FJ6K0Z1_console> show power inline
Port  Watts  PoE Status
----  -----  ----------
  1    0.0   off (detecting)
  2    0.0   off (detecting)
  3    0.0   off (detecting)
  4    0.0   off (detecting)
  5    0.0   off (detecting)
  6    0.0   off (detecting)
  7    0.0   off (detecting)
  8    0.0   off (detecting)
  9    0.0   off (detecting)
 10    0.0   off (detecting)
 11    0.0   off (detecting)
 12    0.0   off (detecting)
 13    0.0   off (detecting)
 14    0.0   off (detecting)
 15    0.0   off (detecting)
 16    0.0   off (detecting)
 17    0.0   off (detecting)
 18    0.0   off (detecting)
 19    0.0   off (detecting)
 20    0.0   off (detecting)
 21    0.0   off (detecting)
 22    0.0   off (detecting)
 23    0.0   off (detecting)
 24    0.0   off (detecting)
```

show process cpu command

Displays the Linux process statistics for the CPU

Examples

```
FJ6K0Z1_console> show process cpu
Mem: 279164K used, 753024K free, 0K shrd, 0K buff, 56464K cached
CPU:   0% usr   0% sys   0% nic 100% idle   0% io   0% irq   0% sirq
Load average: 0.09 0.11 0.13 2/61 1094
  PID  PPID USER     STAT   VSZ %VSZ %CPU COMMAND
  644   951 root     S     297m  30%   0% of-switch ?
  951     1 root     S     2452   0%   0% /bin/sh
  961     1 root     S     2340   0%   0% /sbin/udhcpc -b -R
    1     0 root     S     2340   0%   0% init
 1093   644 root     S     2340   0%   0% /bin/sh -c { /bin/top -b -n 1; } 2> /dev/null
 2231   644 root     S     2340   0%   0% /bin/sh -c ps aux | grep '^[ ]*[0-9]* root[ ]*/sbin/micro-
inetd' || nohup /sbin/micro-inetd 22 /sbin/sshsession -q -l -o /mnt/flash/orion/ssh 2> /dev/null >
/dev/null < /dev/null
 1094  1093 root     R     2340   0%   0% /bin/top -b -n 1
 2234  2231 root     S     1528   0%   0% /sbin/micro-inetd 22 /sbin/sshsession -q -l -o
/mnt/flash/orion/ssh
    3     2 root     SW       0   0%   0% [ksoftirqd/0]
 1344     2 root     SW       0   0%   0% [kworker/0:2]
   12     2 root     SW       0   0%   0% [ksoftirqd/1]
   18     2 root     SW       0   0%   0% [kworker/1:1]
    8     2 root     SW       0   0%   0% [migration/0]
    9     2 root     SW       0   0%   0% [migration/1]
   15     2 root     SW       0   0%   0% [kdevtmpfs]
    6     2 root     SW       0   0%   0% [kworker/u:0]
    2     0 root     SW       0   0%   0% [kthreadd]
    4     2 root     SW       0   0%   0% [kworker/0:0]
    5     2 root     SW<      0   0%   0% [kworker/0:0H]
    7     2 root     SW<      0   0%   0% [kworker/u:0H]
   10     2 root     SW        0   0%   0% [kworker/1:0]
   11     2 root     SW<       0   0%   0% [kworker/1:0H]
   13     2 root     SW<       0   0%   0% [cpuset]
   14     2 root     SW<       0   0%   0% [khelper]
   16     2 root     SW<       0   0%   0% [netns]
   19     2 root     SW        0   0%   0% [kworker/u:1]
  138     2 root     SW        0   0%   0% [bdi-default]
  140     2 root     SW<       0   0%   0% [kblockd]
  148     2 root     SW        0   0%   0% [khubd]
  169     2 root     SW<       0   0%   0% [ip_addr_conflic]
  170     2 root     SW<       0   0%   0% [rpciod]
  196     2 root     SW        0   0%   0% [kswapd0]
  246     2 root     SW        0   0%   0% [fsnotify_mark]
  250     2 root     SW<       0   0%   0% [nfsiod]
  257     2 root     SW<       0   0%   0% [crypto]
  645     2 root     SW<       0   0%   0% [kworker/0:1H]
  883     2 root     SW        0   0%   0% [mtdblock0]
  888     2 root     SW        0   0%   0% [mtdblock1]
  893     2 root     SW        0   0%   0% [mtdblock2]
  898     2 root     SW        0   0%   0% [mtdblock3]
  903     2 root     SW        0   0%   0% [mtdblock4]
  908     2 root     SW        0   0%   0% [mtdblock5]
  918     2 root     SW<       0   0%   0% [deferwq]
  923     2 root     SW        0   0%   0% [ubi_bgt0d]
  948     2 root     SW        0   0%   0% [ubifs_bgt0_0]
  972     2 root     SW<       0   0%   0% [kworker/1:1H]
```

show running-config command

Shows the data stored currenty in the running configuration

Examples

```
FJ6K0Z1_console> show running-config
{
   "Default Logging":
   {
      "components":
      {
         "API": true,
         "Mapping": true,
         "OFDB": true,
         "datapath": true
      },
      "level": 1
   },
   "Management Interface":
   {
      "dhcp": true,
      "ip address": "",
      "ip gateway": ""
   },
   "OpenFlow Controller":
   {
      "connection max retries": 0,
      "connection retry interval": 2000,
      "enabled": true,
      "ip address": "198.18.3.201",
      "ip port": 6633,
      "periodic echo ms": 10000,
      "priority": 0,
      "protocol version": "OpenFlow 1.3.4",
      "reset echo count": 3,
      "tls": false
   },
   "Password Hash": "dWAj3KHZgdo",
   "Remote Syslog":
   {
      "enabled": true,
      "ip address": "172.25.2.51",
      "ip port": 514
   },
   "SSH Service":
   {
      "enabled": true
   },
   "Telnet Service":
   {
      "enabled": true
   }
}
```

show startup-config command

Shows the data stored for the configuration that the switch will use when it boots up next time.

Examples

```
FJ6K0Z1_console> show startup-config
{
   "Default Logging":
   {
      "components":
      {
         "API": true,
         "Mapping": true,
         "OFDB": true,
         "datapath": true
      },
      "level": 1
   },
   "Management Interface":
   {
      "dhcp": true,
      "ip address": "",
      "ip gateway": ""
   },
   "OpenFlow Controller":
   {
      "connection max retries": 0,
      "connection retry interval": 2000,
      "enabled": true,
      "ip address": "198.18.3.201",
      "ip port": 6633,
      "periodic echo ms": 10000,
      "priority": 0,
      "protocol version": "OpenFlow 1.3.4",
      "reset echo count": 3,
      "tls": false
   },
   "Password Hash": "dWAj3KHZgdo",
   "Remote Syslog":
   {
      "enabled": true,
      "ip address": "172.25.2.51",
      "ip port": 514
   },
   "SSH Service":
   {
      "enabled": true
   },
   "Telnet Service":
   {
      "enabled": true
   }
}
```

show statistics command

Lists out the statistics that are kept for the switch..

Examples

```
FJ6K0Z1_console> show statistics
    Incomplete command!
possible subcommands:
    switchport              show statistics for the switch
FJ6K0Z1_console> show statistics
    show statistics <subcommand>
possible subcommands:
    switchport              show statistics for the switch
FJ6K0Z1_console> show statistics 1
    Unknown subcommand!
    show statistics <subcommand>
possible subcommands:
    switchport              show statistics for the switch

FJ6K0Z1_console> show statistics switchport 1
Total Packets Received (Octets).........0
Packets Received Without Error..........0
Unicast Packets Received................0
Multicast Packets Received..............0
Broadcast Packets Received..............0
Receive Packets Discarded...............0
Octets Trasmitted.......................0
Packets Trasmitted Without Error........0
Unicast Packets Trasmitted..............0
Multicast Packets Transmitted...........0
Broadcast Packets Transmitted...........0
Transmit Packets Discarded..............0
Time Since Counters Last Cleared .......0
```

show system command

Lists out information showing the state of the system

Examples

```
FJ6K0Z1_console> show system

System Description:    Dell Networking N3024P
Burned in MAC Address: f8:b1:56:67:99:91
System Model ID:       N3024P
Machine Type:          Dell Networking N3024P
Serial Number:         CN0C3M5M282984CE0129A02
Service Tag:           FJ6K0Z1
Asset Tag:             12345

System Thermal Conditions:
Unit Temperature State
      (Celsius)
---- ----------- -----
0    32          OK

Temperature Sensors:
Unit Description Temperature
                  (Celsius)
---- ----------- -----------
0    MAC         32
0    PHY         32

Fans:
Unit Description Status
---- ----------- ------
0    Fan-1       OK
0    Fan-2       OK

Power supplies:
Unit Description Status Average Current  Since
                         Power   Power  (seconds ago)
                        (Watts) (Watts)
---- ----------- ------ ------- ------- ---------
0    System      OK     28      28      248837
0    System      Failed 0       0       0

USB Port Power Status:
----------------------
USB is Enabled  and status is OK
```

show tech-support command

Lists out information showing the state of the system. This is a very comprehensive dump of everything listed here under the "show' commands along with internal debugging information and other information about the system.

Examples

```
FJ6K0Z1_console> show tech-support
<dump of system begins>
.
.
.
<dump of system ends>
```

show version command

Displays the versions of firmware on the switch.

Examples

```
FJ6K0Z1_console> show version
Machine Description............... Dell Networking Switch
System Model ID................... N3024P
Machine Type...................... Dell Networking N3024P
Serial Number..................... CN0C3M5M282984CE0129A02
Manufacturer...................... 0xbc00
Burned In MAC Address............. f8:b1:56:67:99:91
SOC Version....................... BCM56342_A0
HW Version........................ 5
CPLD Version...................... 13
unit active      backup      current-active next-active
---- ----------- ----------- -------------- --------------
1    8.10.2.0     6.1.0.6     8.10.2.0       8.10.2.0
```

system top level command

Displays information available under the system top level command.

Examples

```
FJ6K0Z1_console> system
    system <subcommand>
possible subcommands:
    asset-tag               set the switch asset tag
    clock                   set the system clock
    locate                  locate a switch by LED blinking
```

system asset-tag command

Allows the user to configure an asset tag for the switch in the switch configuration.

Examples

```
FJ6K0Z1_console> system asset-tag
    system asset-tag <asset tag>
    Set the switch asset tag. The asset tag is a customer configurable string
    that is up to 16 characters long.

FJ6K0Z1_console> system asset-tag 12345
FJ6K0Z1_console> show system

System Description:    Dell Networking N3024P
Burned in MAC Address: f8:b1:56:67:99:91
System Model ID:       N3024P
Machine Type:          Dell Networking N3024P
Serial Number:         CN0C3M5M282984CE0129A02
Service Tag:           FJ6K0Z1
Asset Tag:             12345
```

system clock command

Allows the user to configure current system time and date.

Examples

```
FJ6K0Z1_console> system clock

    Incomplete command!

possible subcommands:
    date                    set the system date
    time                    set the system time
```

system locate command

Starts a beacon timer running for the specified number of seconds to allow the ystem to be located in the racks.

Examples

```
FJ6K0Z1_console> system locate
    system locate <time in seconds (1-300)>
    Activate's the switch's blinking locate LED function for the alloted amount
    of time. This can assist a user in locating the switch among other hardware.
FJ6K0Z1_console> system locate 30
```

unmount command

Allows the user to unmount the device from the system. It shows an informational message if the device is not mounted but causes no other problems.

unmount	unmount usb device

Examples

Shows no message if usb device was mounted:

```
FJ6K0Z1_console> unmount
    unmount <subcommand>
possible subcommands:
    usb                     prepare the usb drive for safe removal
FJ6K0Z1_console> unmount usb
```

Shows following message if usb device was not mounted:

```
FJ6K0Z1_console> unmount usb
umount: can't umount /media: Invalid argument
```

write

Use the **write** command to write the contents of the running-config to the backup-config.

B Appendix – Setting up flows with the DNOS-OF SDN Agent, Ryu SDN Controller, and Ryu REST API

The following shows annotated console extracts of installing the Ryu 3.23 controller, setting up the Ryu REST API JSON parser, connecting the DNOS-OF switch agent to the Ryu controller, setting up sone test flows, and shows some other useful utility commands available through the Ryu controller. This example is shown running on a Debian Linux machine but also runs fine on Ubuntu and CentOS.

1. Ryu 3.23.2 Installation

Install the RYU controller using the below set of commands on a Debian Linux machine.
Prerequisites: (for Debian-based distributives)

```
sudo apt-get install git
sudo apt-get install python-setuptools
sudo apt-get install python-pip
sudo apt-get install python-dev
```

To install but not build Ryu controller:

```
sudo pip install ryu
```

To install and build sources : Ryu BUILD:

```
git clone git://github.com/osrg/ryu.git
cd ryu
sudo python ./setup.py install
```

Make sure you get one of the latest RYU controller releases, 3.23 or later. The prior versions had a bug that would not allow the JSON scripts to execute properly in a multi table environment. Also note that these later releases of Ryu require a newer Python release, V3.4, to run properly. However many things rely on their old Python 2.7 installations for many applications, so you will want to have both with one listed as the primary alternative and the other as the backup alternative.

2. Starting the Ryu controller with the REST API enabled

Once you have the controller downloaded, installed and validated, you are ready to start it up and establish a link from the DNOS-OF SDN agent.

Navigate to the ryu/ryu/app subdirectory created when Ryu is installed and find the ofctl_rest.py file.

Start the Ryu controller with the REST API as shown below and you should see something like the following messages:

```
khughes@ODL42:~/Ryu/ryu/ryu/app$ ryu-manager --verbose ofctl_rest.py

          loading app ofctl_rest.py
          loading app ryu.controller.ofp_handler
          instantiating app None of DPSet
          creating context dpset
          creating context wsgi
          instantiating app ryu.controller.ofp_handler of OFPHandler
          instantiating app ofctl_rest.py of RestStatsApi
          BRICK dpset
            CONSUMES EventOFPStateChange
            CONSUMES EventOFPPortStatus
            CONSUMES EventOFPSwitchFeatures
          BRICK ofp_event
```

```
PROVIDES EventOFPMeterConfigStatsReply TO {'RestStatsApi': set(['main'])}
PROVIDES EventOFPPortStatsReply TO {'RestStatsApi': set(['main'])}
PROVIDES EventOFPAggregateStatsReply TO {'RestStatsApi': set(['main'])}
PROVIDES EventOFPQueueStatsReply TO {'RestStatsApi': set(['main'])}
PROVIDES EventOFPStatsReply TO {'RestStatsApi': set(['main'])}
PROVIDES EventOFPDescStatsReply TO {'RestStatsApi': set(['main'])}
PROVIDES EventOFPMeterStatsReply TO {'RestStatsApi': set(['main'])}
PROVIDES EventOFPGroupStatsReply TO {'RestStatsApi': set(['main'])}
PROVIDES EventOFPPortStatus TO {'dpset': set(['main'])}
PROVIDES EventOFPMeterFeaturesStatsReply TO {'RestStatsApi': set(['main'])}
PROVIDES EventOFPSwitchFeatures TO {'dpset': set(['config']), 'RestStatsApi': set(['main'])}
PROVIDES EventOFPGroupFeaturesStatsReply TO {'RestStatsApi': set(['main'])}
PROVIDES EventOFPStateChange TO {'dpset': set(['main', 'dead'])}
PROVIDES EventOFPFlowStatsReply TO {'RestStatsApi': set(['main'])}
PROVIDES EventOFPPortDescStatsReply TO {'RestStatsApi': set(['main'])}
PROVIDES EventOFPGroupDescStatsReply TO {'RestStatsApi': set(['main'])}
CONSUMES EventOFPErrorMsg
CONSUMES EventOFPHello
CONSUMES EventOFPEchoRequest
CONSUMES EventOFPSwitchFeatures
CONSUMES EventOFPPortDescStatsReply
      BRICK RestStatsApi
CONSUMES EventOFPMeterConfigStatsReply
CONSUMES EventOFPPortStatsReply
CONSUMES EventOFPAggregateStatsReply
CONSUMES EventOFPQueueStatsReply
CONSUMES EventOFPStatsReply
CONSUMES EventOFPDescStatsReply
CONSUMES EventOFPMeterStatsReply
CONSUMES EventOFPGroupStatsReply
CONSUMES EventOFPSwitchFeatures
CONSUMES EventOFPMeterFeaturesStatsReply
CONSUMES EventOFPGroupFeaturesStatsReply
CONSUMES EventOFPFlowStatsReply
CONSUMES EventOFPPortDescStatsReply
CONSUMES EventOFPGroupDescStatsReply
```

In my example, the IP address of the Ryu controller is 172.25.11.93 and the IP address of the DNOS-OF switch we were hooking to it is 172.25.161.222

You can log into the instance of the Ryu controller if desired to see it is up and running by SSH to IP using username: ryu and password: ryu.

Point the DNOS-OF switch to connect to the Ryu controller, you should see something like the following on the switch console:

```
ZYX123_console> set openflow controller 172.25.11.93 6633

07-13 21:38:59.311044 of-switch: MSG: src/of-switch/OF_SDNAgent.cpp:SetOpenFlowController:Adding
controller 172.25.11.93:6633, (TCP connection), to configuration
```

```
07-13 21:38:59.312255 of-switch: MSG: src/of-switch/OF_SDNController.cpp:Connect:Initializing
controller 172.25.11.93:6633
07-13 21:38:59.312482 ofconnectionmanager: INFO: Added remote connection: 172.25.11.93:6633
07-13 21:39:00.336444 ofconnectionmanager: INFO: cxn 172.25.11.93:6633: DISCONNECTED->CONNECTING
07-13 21:39:00.453541 ofconnectionmanager: INFO: cxn 172.25.11.93:6633: CONNECTING-
>HANDSHAKE_COMPLETE
```

And something like the following on the Ryu console:

```
(20961) wsgi starting up on http://0.0.0.0:8080/
connected socket:<eventlet.greenio.base.GreenSocket object at 0x33add90> address:('172.25.161.222',
48165)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x33aa650>
move onto config mode
EVENT ofp_event->dpset EventOFPSwitchFeatures
switch features ev version: 0x4 msg_type 0x6 xid 0x9fee4cd4
OFPSwitchFeatures(auxiliary_id=0,capabilities=71,datapath_id=133571005559288,n_buffers=0,n_tables=7)
move onto main mode
EVENT ofp_event->dpset EventOFPStateChange
DPSET: register datapath <ryu.controller.controller.Datapath object at 0x33aa050>
EVENT ofp_event->dpset EventOFPStateChange
DPSET: unregister datapath <ryu.controller.controller.Datapath object at 0x33aa050>
connected socket:<eventlet.greenio.base.GreenSocket object at 0x33aa610> address:('172.25.161.222',
48166)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x33aa7d0>
move onto config mode
EVENT ofp_event->dpset EventOFPSwitchFeatures
switch features ev version: 0x4 msg_type 0x6 xid 0xae4a1b6c
OFPSwitchFeatures(auxiliary_id=0,capabilities=71,datapath_id=133571005559288,n_buffers=0,n_tables=7)
move onto main mode
EVENT ofp_event->dpset EventOFPStateChange
DPSET: register datapath <ryu.controller.controller.Datapath object at 0x33aa350>
```

The log above shows the sequence of handshake when the switch agent connects to the controller. Hello messages are exchanged and then switch features are requested. One of the features exchanged is the OpenFlow protocol version (0x4=1.3.4), along with the datapath ID of the switch that just connected, which you will need to install flows.

You can also see the datapath ID of all switches connected to the Ryu by issuing a GET with Postman or similar REST client, or just pointing at it with a web browser or with some thing like CURL:

http://172.25.11.93:8080/stats/switches

[133571005559288]

COMMAND: `curl -X GET http:// 172.25.11.93:8080/stats/switches`

```
Output: [133571005559288]
```

```
ZYX123_console>
07-13 21:48:20.949575 ofconnectionmanager: INFO: cxn 172.25.11.93:6633: Exceeded outstanding echo requests.  Resetting cxn
07-13 21:48:20.949727 ofconnectionmanager: INFO: cxn 172.25.11.93:6633: HANDSHAKE_COMPLETE->CLOSING
07-13 21:48:20.949927 ofconnectionmanager: INFO: cxn 172.25.11.93:6633: CLOSING->DISCONNECTED
07-13 21:48:20.952296 ofconnectionmanager: INFO: cxn 172.25.11.93:6633: DISCONNECTED->CONNECTING
07-13 21:48:21.076249 ofconnectionmanager: INFO: cxn 172.25.11.93:6633: CONNECTING->HANDSHAKE_COMPLETE
```

To push a flow, you can use Postman and a URL like this:

http://172.25.11.93:8080/stats/flowentry/add

Or you can once again use the CURL command line based REST API tool. Here is one example of a flow description that successfully installs via the Ryu REST API:

```
{
    "dpid": 133571005559288,
    "cookie": 1,
    "cookie_mask": 1,
    "table_id": 10,
    "idle_timeout": 30,
    "hard_timeout": 30,
    "priority": 1,
    "flags": 1,
    "_name" : "vlan10",
    "cmd" : "add",
    "mask" : "0",
    "port" : "any",
    "group" : "any",
    "match":
    {
        "in_port" : "10",
        "vlan_vid" : "2"
    },
    "instructions":
    [
        {
            "goto":
            {
                "table_id":"20"
            }
        }
    ]
}
```

1) To fetch the switch description we can use the below cURL command on controller console.
   COMMAND: `curl -X GET http://localhost:8080/stats/flow/{dpId}`

   `Where dpId –` **133571005559288** `(We will get this from above command's response)`

2) Use the below commands on controller as per their usage:

   a) **FLOW ENTRY ADD COMMAND:**
   ```
   curl -i -v -X POST -d '@flow_vlan_add.json'
   http://localhost:8080/stats/flowentry/add
   ```
   Here flow_vlan_add.json is a json file to create a vlan flow entry

   b) **FETCHING FLOWS USING MATCH FILTER COMMAND:**
   ```
   curl -i -v -X POST -d '@flow_vlan_stats.json'
   http://localhost:8080/stats/aggregateflow/{dpid}
   ```
   Here flow_vlan_stats.json is a JSON file contains the filter parameters and values

   c) **DELETE ALL FLOW ENTRIES COMMAND:**
   ```
   curl -i -v -X DELETE
   http://localhost:8080/stats/flowentry/clear/{dpid}
   ```

   d) **GROUP ENTRY ADD COMMAND:**
   ```
   curl -i -v -X POST -d '@group_l2_add.json'
   http://localhost:8080/stats/groupentry/add
   ```
   Here group_l2_add.json is a JSON file contains the parameters to create a group

   e) **DELETE GROUP COMMAND :**
   ```
   curl -i -v -X POST -d '@group_delete.json'
   http://localhost:8080/stats/groupentry/delete
   ```

Here group_delete.json is a JSON file contains data of group id and dpid

3) In the same way, we can change the HTTP method and URL based on our requirement, please check the list of services available from RYU controller, but below are some Examples

```
# To get meters stats of the switch
# GET /stats/meter/<dpid>

# To get group features stats of the switch
# GET /stats/groupfeatures/<dpid>

# To get groups desc stats of the switch
# GET /stats/groupdesc/<dpid>

# To  get groups stats of the switch
# GET /stats/group/<dpid>

# To get ports description of the switch
# GET /stats/portdesc/<dpid>

# To modify all matching flow entries
# POST /stats/flowentry/modify

# To modify flow entry strictly matching wildcards and priority
# POST /stats/flowentry/modify_strict

# To delete flow entry strictly matching wildcards and priority
# POST /stats/flowentry/delete_strict

# To delete all flow entries of the switch
# DELETE /stats/flowentry/clear/<dpid>

# To add a meter entry
# POST /stats/meterentry/add

# To modify a meter entry
# POST /stats/meterentry/modify

# To delete a meter entry
# POST /stats/meterentry/delete

# To modify a group entry
# POST /stats/groupentry/modify

# To modify behavior of the physical port
# POST /stats/portdesc/modify

# To send a experimeter message
# POST /stats/experimenter/<dpid>
```

3. Where within the Ryu directory structure to find the REST API script

Navigate to ryu/app folder and find "ofctl_rest.py" which has the logic to parse the input and translate it to an openflow packet via the controller and send it to the agent.

C Appendix - Example of setting up a basic Ethernet L2 Bridging topology for end to end traffic

The following section illustrates how to establish an end to end traffic flow using basic Ethernet Layer 2 bridging. The Ryu controller, a web browser and the REST API web utility Postman are used in these examples, but any OpenFlow 1.3.4 compliant controller and REST API tool can be used.

Note: that Ryu 3.23 or later controller must be used for the example flows shown since it supports JSON REST API calls properly.

Once DNOS-OF is set up on the switch, it is configured for management access, and a communication channel is established to the controller and verified as described in the section on deployment, flows can be set up and traffic tested.

The basic minimal steps required for creating a single end to end flow using DNOS-OF are shown below:

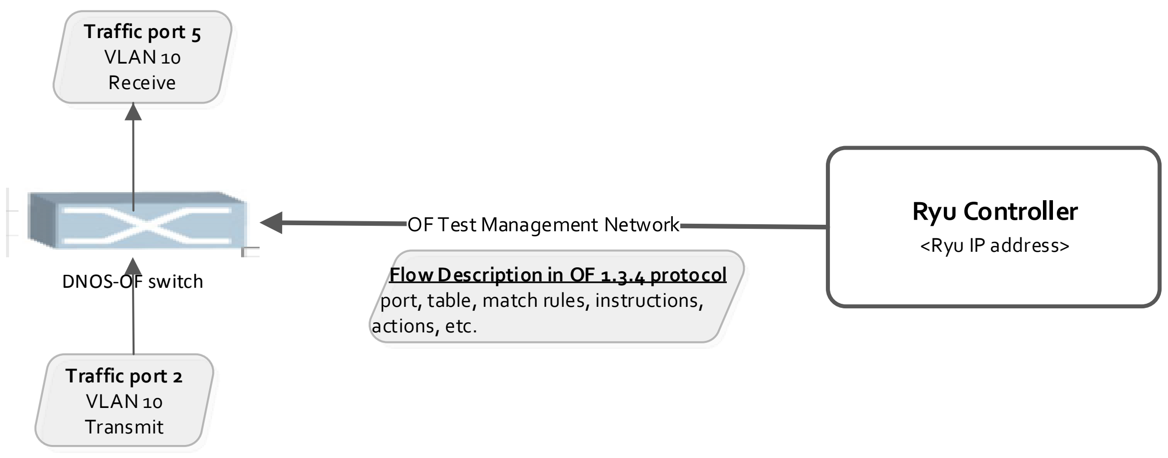
2) Set up a VLAN flow in the VLAN flow table.
3) Create a group that includes the proper VLAN flow and its associated port, and attaches it to an output port that will be specified in the bridging flow.
4) Set up a bridging flow in the Bridging flow table to specify the output port.

The details of each of these flows required for end to end data traffic are outlined in the following diagrams and the JSON scripts shown below. You can use XML, JSON, Java, C/C++, Perl, Python, or any other mechanism to specify the metadata of the flow, as long as at the end of it the OpenFlow protocol commands sent on the wire down to the switch describe the flow properly per the supported OpenFlow protocol specification standard (currently V1.3.4 for Release 1.0 of DNOS-OF).

For the Ryu controller, you can use any REST API URL tool to retrieve switch information and to send and retrieve flow information instructions which are then sent down to the switch via the controller using the OpenFlow 1.3.4 protocol. The examples below show the use of the Ryu controller with a web browser and with the Postman REST API application.

Once the switch datapath ID is retrieved, it can then be used to set up flows on the switch as shown below

C.1 System Diagram for example flow

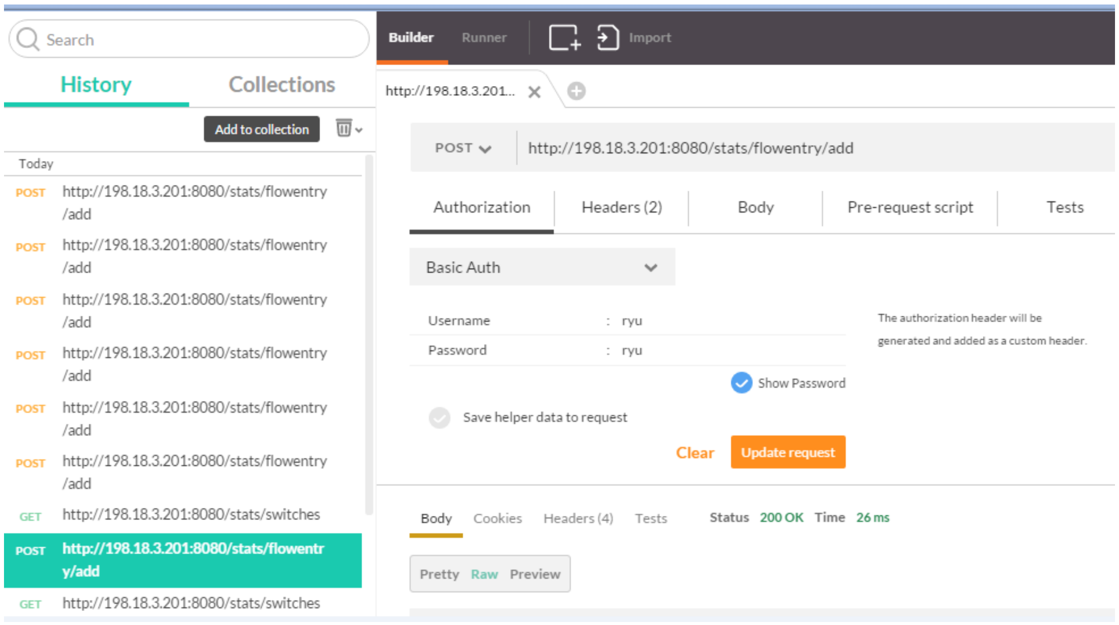

Traffic port 5
VLAN 10
Receive
DNOS-OF switch
OF Test Management Network
Flow Description in OF 1.3.4 protocol
port, table, match rules, instructions, actions, etc.
Ryu Controller
<Ryu IP address>
Traffic port 2
VLAN 10
Transmit

C.2 Step 1 - Set up a VLAN flow

The script below shows the JSON code that is used to create the first flow, in the VLAN table (table 10 in the SOC).

```
{
    "dpid"          : <your switch datapath ID goes here>,
    "cookie"        : 1,
    "cookie_mask"   : 1,
    "table_id"      : 10,
    "hard_timeout"  : 30,
    "priority"      : 1,
    "flags"         : 1,
    "_name"         : "vlan10",
    "cmd"           : "add",
    "mask"          : "0",
    "port"          : "any",
    "group"         : "any",
    "match"         : {"in_port" : 2, "vlan_vid" : 10},
    "instructions"  : [{"goto": {"table_id":20}} ]
}
```

This script is fed into the REST API in the Ryu controller via the Postman application as shown below. First the authorization parameters are set using the controller username and password, as with the Get /stats/switches, and the API URL is set to **http://<ryu controller IP>:8080/stats/flowentry/add**.



Today

POST http://198.18.3.201:8080/stats/flowentry/add

POST http://198.18.3.201:8080/stats/flowentry/add

POST http://198.18.3.201:8080/stats/flowentry/add

POST http://198.18.3.201:8080/stats/flowentry/add

POST http://198.18.3.201:8080/stats/flowentry/add

POST http://198.18.3.201:8080/stats/flowentry/add

GET http://198.18.3.201:8080/stats/switches

POST http://198.18.3.201:8080/stats/flowentry/add

GET http://198.18.3.201:8080/stats/switches

http://198.18.3.201...

POST http://198.18.3.201:8080/stats/flowentry/add

Basic Auth

Username : ryu

Password : ryu

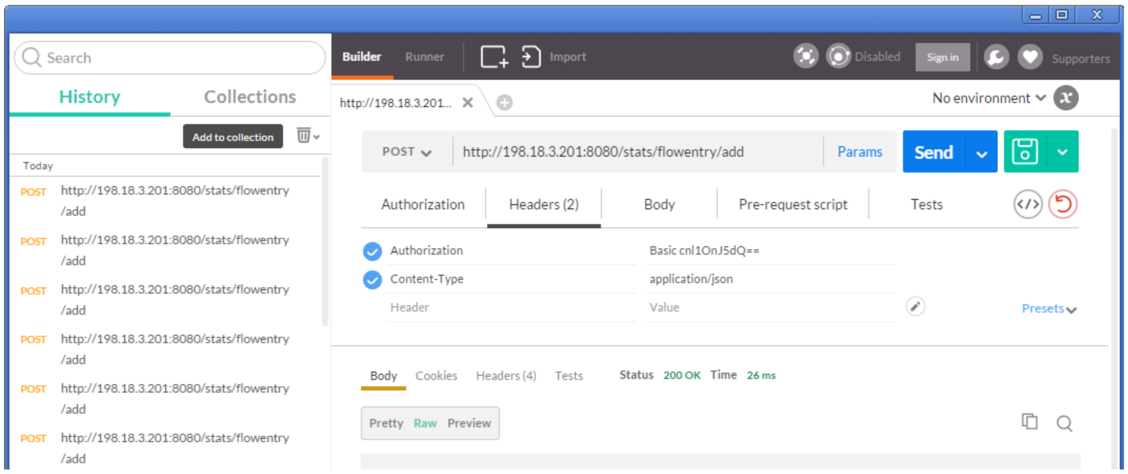
The authorization header will be generated and added as a custom header.

Show Password

Save helper data to request

Body Status 200 OK Time 26 ms

Next the headers are verified set as shown below, with the content type set to JSON and the basic authorization header in place, and the JSON transaction type set to POST:



```
POST  http://198.18.3.201:8080/stats/flowentry/add
```

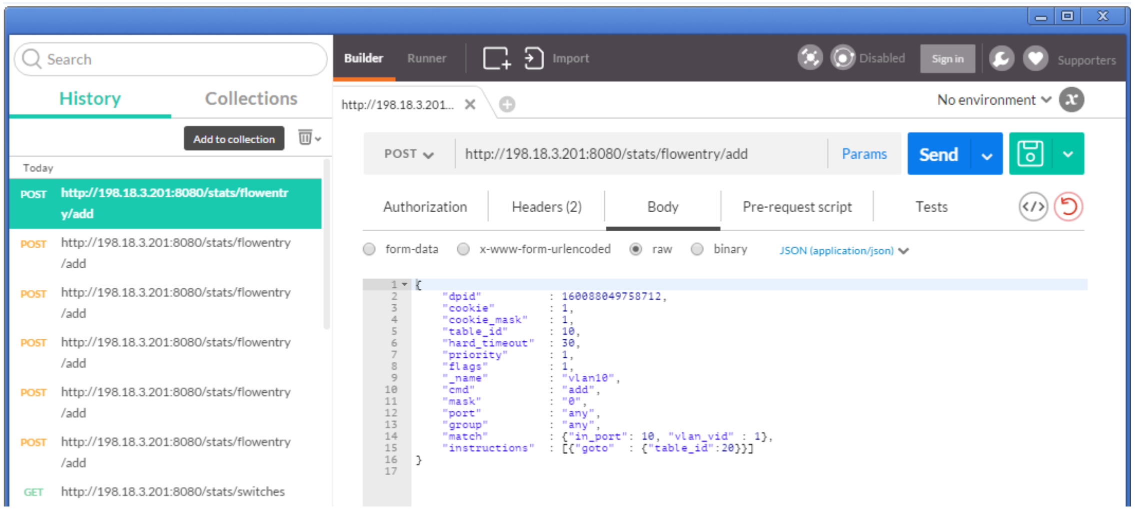
Header	Value
Authorization	Basic cnl1OnJ5dQ==
Content-Type	application/json

Status 200 OK Time 26 ms

History (Today):

```
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
```

Finally the body of the flow request is filled in with our REST flow add request, as shown below:



```
POST  http://198.18.3.201:8080/stats/flowentry/add
Body: raw  JSON (application/json)
```

```
{
    "dpid"          : 160088049758712,
    "cookie"        : 1,
    "cookie_mask"   : 1,
    "table_id"      : 10,
    "hard_timeout"  : 30,
    "priority"      : 1,
    "flags"         : 1,
    "_name"         : "vlan10",
    "cmd"           : "add",
    "mask"          : "0",
    "port"          : "any",
    "group"         : "any",
    "match"         : {"in_port": 10, "vlan_vid" : 1},
    "instructions"  : [{"goto"  : {"table_id":20}}]
}
```

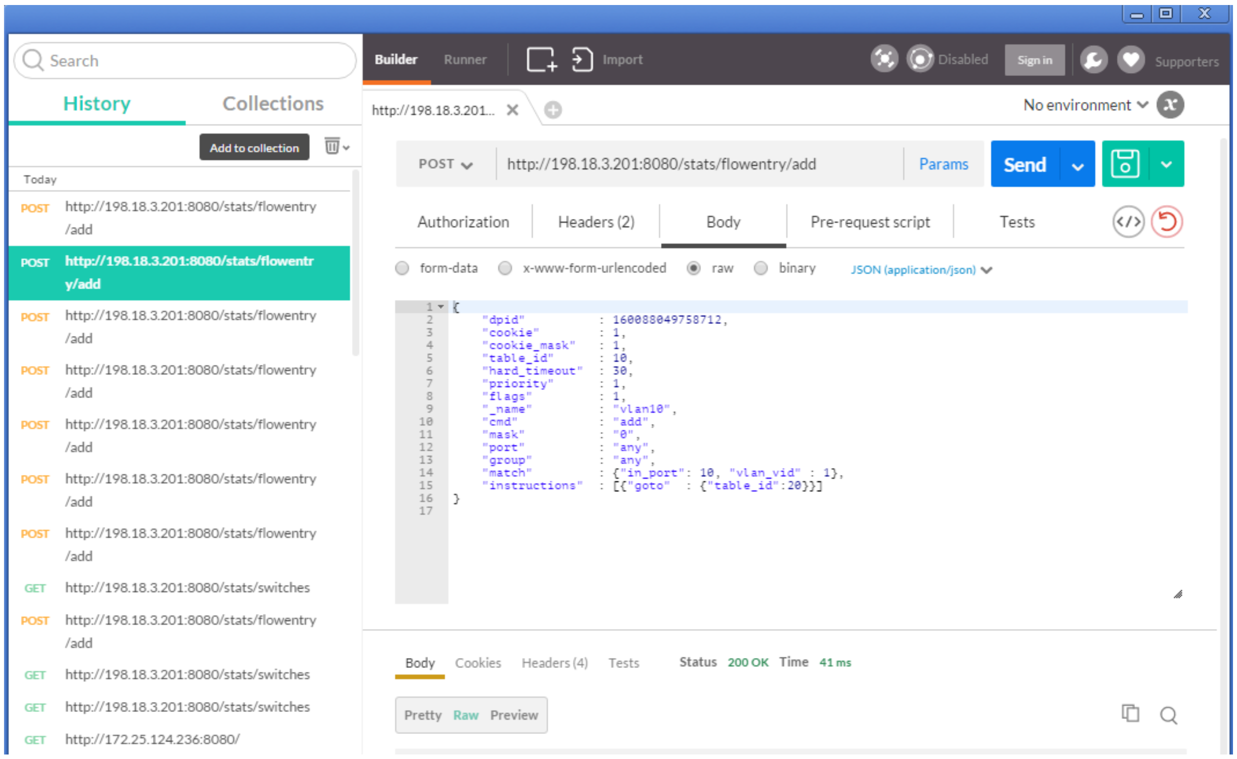
History (Today):

```
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
GET  http://198.18.3.201:8080/stats/switches
```

To send the flow add request to the switch via the controller, hit the Send button. You should see a couple of things at this point. On the switch console, if the flow is successfully added, you will see something like the following:

```
FJ6K0Z1_console> 08-10 19:39:56.799895 indigo_ofdpa_driver: MSG:
src/indigo/ofdpadriver/ind_ofdpa_fwd.c:indigo_fwd_flow_create:flow create called, incoming table
ID=10
08-10 19:39:56.800101 indigo_ofdpa_driver: MSG:
src/indigo/ofdpadriver/ind_ofdpa_fwd.c:indigo_fwd_flow_create:Flow match criteria for table 10
retrieved successfully)
08-10 19:39:56.800223 indigo_ofdpa_driver: MSG:
src/indigo/ofdpadriver/ind_ofdpa_fwd.c:indigo_fwd_flow_create:Flow match fields and masks criteria
for table 10 retrieved successfully)
08-10 19:39:56.800363 indigo_ofdpa_driver: MSG:
src/indigo/ofdpadriver/ind_ofdpa_fwd.c:indigo_fwd_flow_create:Flow match instructions for table 10
retrieved successfully, adding flow)
```

On the Postman application you should see the same "200 OK" output status back from the Ryu controller REST API command, as shown below:



POST http://198.18.3.201:8080/stats/flowentry/add

Body: raw, JSON (application/json)

```
{
    "dpid"          : 160088049758712,
    "cookie"        : 1,
    "cookie_mask"   : 1,
    "table_id"      : 10,
    "hard_timeout"  : 30,
    "priority"      : 1,
    "flags"         : 1,
    "_name"         : "vlan10",
    "cmd"           : "add",
    "mask"          : "0",
    "port"          : "any",
    "group"         : "any",
    "match"         : {"in_port": 10, "vlan_vid" : 1},
    "instructions"  : [{"goto"  : {"table_id":20}}]
}
```

History (Today):

```
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
POST http://198.18.3.201:8080/stats/flowentry/add
GET  http://198.18.3.201:8080/stats/switches
POST http://198.18.3.201:8080/stats/flowentry/add
GET  http://198.18.3.201:8080/stats/switches
GET  http://198.18.3.201:8080/stats/switches
GET  http://172.25.124.236:8080/
```

Status 200 OK Time 41 ms

To verify the flow that was indeed added to the switch, the **show openflow flows** command can be used as shown below:

```
FJ6K0Z1_console> show openflow flows

Showing openflow flow entries for all tables
Ingress Port Table (0) Flow Entries
----------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 2000
        Number of entries actually found = 0

VLAN Table (10) Flow Entries
--------------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 12288
| Flow ID:0x9c157cc8 Priority:0 Hard_time:0 Idle_time:0 Cookie:3 inPort:mask = 0:0xffff0000
srcMac:mask = 0000.0000.0000:0000.0000.0000 destMac:mask = 0000.0000.0000:0000.0000.0000 etherType =
0000 vlanId:mask = 100:0xfff srcIp4 = 0.0.0.0/0.0.0.0 dstIp4 = 0.0.0.0/0.0.0.0 srcIp6 = ::/:: dstIp6
= ::/:: DSCP = 0 VRF = 0 DEI = 0 ECN = 0 IP Protocol = 0x00 Source L4 Port = 0 Destination L4 Port =
0 ICMP Type = 0 ICMP Code = 0 | Set CoS queue = 6 outPort = 0
        Number of entries actually found = 0

Termination MAC Table (20) Flow Entries
----------------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 512
        Number of entries actually found = 0

Unicast Routing Table (30) Flow Entries
----------------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 40960
        Number of entries actually found = 0

Multicast Routing Table (40) Flow Entries
-----------------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 8191
        Number of entries actually found = 0

Bridging Table (50) Flow Entries
--------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 32767
        Number of entries actually found = 0

ACL Policy Table (60) Flow Entries
----------------------------------
        Number of entries reported = 0
        Maximum number of entries for this table = 7680
        Number of entries actually found = 0
```

What this flow does in he switch, is to say that anything coming into port 2 on VLAN 10 should be forwarded on via the GOTO TABLE statement to the Termination MAC table, table ID 20,

C.3 Step 2 - Set up a Group Entry

The script below shows the JSON code that is used to create the second flow, in the Group table:

```
{
    "group_mod":
    {
            "_name"                 : "l2_0xa0005",
            "_description"          : "Description",
            "cmd"                   : "add",
            "type"                  : "indirect",
            "group_id"              : "0xa0005",
            "buckets":
            [{
                    "weight"        : "0",
                    "watch_port"    :"any",
                    "watch_group"   :"any",
                    "actions"       : [{ "output": { "port":"5" }}]
            }]
    }
}
```

Use the same steps described in installing and verifying the VLAN flow to install this group flow which forwards any frame coming into any port on the switch on VLAN 10, coming through the termination MAC data path, should be sent out port 5.

C.4 Step 3 - Set up a Bridging Flow

The script below shows the JSON code that is used to create the third flow, in the bridging table

```
{
    "flow_mod":
    {
            "_name"        : "Bridging",
            "_description" : "Description",
            "table"        : "bridging",
            "cmd"          : "add",
            "mask"         : "0",
            "port"         : "any",
            "group"        : "any",
            "match":
            {
                    "vlan_vid"      : "10, 0x1fff",
                    "eth_dst"       :"00:01:02:03:04:05",
                    "eth_dst_mask"  :"ff:ff:ff:ff:ff:ff"
            },
            "instructions":
            [{
                    "write": [{"actions": [{ "group": { "group_id":"0xa0005" }}]}]
             },
             {
                 "goto": {"table":"acl"}
             }]
    }
}
```

Once again, use the same steps described in installing and verifying the previous 2 flows with Postman to install this 3rd flow that takes traffic on VLAN 10 coming into the bridging table and forwards it to the last table, ACL/Policy, where it egresses the switch on port 5. This gets an end to end flow from an ingress of port 2 to an egress of port 5.

D Appendix - additional resources

Dell.com/spport Dell is focused on meeting your needs with proven services and support.

DellTechCenter.com is an IT Community where you can connect with Dell Customers and Dell employees for the purpose of sharing knowledge, best practices, and information about Dell products and installations.

Referenced and/or recommended DNOS-OF, SDN, OpenFlow and OF-DPA publications:

- ONF site: **https://www.opennetworking.org/**
- Broadcom OFDPA site: **https://github.com/Broadcom-Switch/of-dpa**
- TechTarget OpenFlow/SDN site: **http://searchsdn.techtarget.com/guides/OpenFlow-protocol-tutorial-SDN-controllers-and-applications-emerge**
- Flowgrammable site: **http://flowgrammable.org/**
- Ryu site: **http://osrg.github.io/ryu/**
- Linux kernel - **https://www.kernel.org/**
- Debian Linux kernel - **https://www.debian.org/**
- ONL **–** **http://opennetlinux.org/**
- ONIE **–** **https://github.com/onie/onie**
- OCP and open-source projects **–** **http://www.opencompute.org/wiki/Networking/SpecsAndDesigns**