Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 10, 2013

E. Haleplidis
University of Patras
O. Cherkaoui
University of Quebec in Montreal
S. Hares
Huawei
W. Wang
Zhejiang Gongshang University

July 9, 2012

Forwarding and Control Element Separation (ForCES) OpenFlow Model Library
draft-haleplidis-forces-openflow-lib-01

Abstract

This document describes the OpenFlow switch in Logical Function Blocks (LFBs) used in the Forwarding and Control Element Separation (ForCES). The LFB classes are defined according to the ForCES Forwading Element (FE) model and ForCES protocol specifications. The library includes the descriptions of the OpenFlow LFBs and the XML definitions.

Status of this Memo

Copyright Notice

Table of Contents

```
   1.  Introduction . . . . . . . . . . . . . . . . . . . . . . . . .  5
     1.1.  ForCES . . . . . . . . . . . . . . . . . . . . . . . . . .  5
     1.2.  OpenFlow . . . . . . . . . . . . . . . . . . . . . . . . .  5
   2.  Terminology and Conventions  . . . . . . . . . . . . . . . . .  7
     2.1.  Requirements Language  . . . . . . . . . . . . . . . . . .  7
     2.2.  Definitions  . . . . . . . . . . . . . . . . . . . . . . .  7
   3.  OpenFlow ForCES library  . . . . . . . . . . . . . . . . . . . 10
     3.1.  OpenFlow Specification . . . . . . . . . . . . . . . . . . 10
     3.2.  ForCES-based OpenFlow Specification  . . . . . . . . . . . 12
   4.  OpenFlow Base Types  . . . . . . . . . . . . . . . . . . . . . 21
     4.1.  Data Types . . . . . . . . . . . . . . . . . . . . . . . . 21
       4.1.1.  Atomic . . . . . . . . . . . . . . . . . . . . . . . . 21
       4.1.2.  Compound Struct  . . . . . . . . . . . . . . . . . . . 22
       4.1.3.  Compound Array . . . . . . . . . . . . . . . . . . . . 23
     4.2.  Frame Types  . . . . . . . . . . . . . . . . . . . . . . . 23
     4.3.  MetaData Types . . . . . . . . . . . . . . . . . . . . . . 23
   5.  OpenFlow LFBs  . . . . . . . . . . . . . . . . . . . . . . . . 25
     5.1.  OpenFlowSwitch . . . . . . . . . . . . . . . . . . . . . . 25
       5.1.1.  Data Handling  . . . . . . . . . . . . . . . . . . . . 25
       5.1.2.  Components . . . . . . . . . . . . . . . . . . . . . . 25
       5.1.3.  Capabilities . . . . . . . . . . . . . . . . . . . . . 26
       5.1.4.  Events . . . . . . . . . . . . . . . . . . . . . . . . 26
     5.2.  OFFlowTables . . . . . . . . . . . . . . . . . . . . . . . 26
       5.2.1.  Data Handling  . . . . . . . . . . . . . . . . . . . . 27
       5.2.2.  Components . . . . . . . . . . . . . . . . . . . . . . 28
       5.2.3.  Capabilities . . . . . . . . . . . . . . . . . . . . . 29
       5.2.4.  Events . . . . . . . . . . . . . . . . . . . . . . . . 29
     5.3.  OFGroupTable . . . . . . . . . . . . . . . . . . . . . . . 29
       5.3.1.  Data Handling  . . . . . . . . . . . . . . . . . . . . 29
       5.3.2.  Components . . . . . . . . . . . . . . . . . . . . . . 29
       5.3.3.  Capabilities . . . . . . . . . . . . . . . . . . . . . 30
       5.3.4.  Events . . . . . . . . . . . . . . . . . . . . . . . . 30
     5.4.  OFPort . . . . . . . . . . . . . . . . . . . . . . . . . . 30
       5.4.1.  Data Handling  . . . . . . . . . . . . . . . . . . . . 30
       5.4.2.  Components . . . . . . . . . . . . . . . . . . . . . . 30
       5.4.3.  Capabilities . . . . . . . . . . . . . . . . . . . . . 31
       5.4.4.  Events . . . . . . . . . . . . . . . . . . . . . . . . 31
     5.5.  OFQueue . . . . . . . . . . . . . . . . . . . . . . . . . 31
       5.5.1.  Data Handling  . . . . . . . . . . . . . . . . . . . . 31
       5.5.2.  Components . . . . . . . . . . . . . . . . . . . . . . 31
       5.5.3.  Capabilities . . . . . . . . . . . . . . . . . . . . . 32
       5.5.4.  Events . . . . . . . . . . . . . . . . . . . . . . . . 32
     5.6.  OFRedirectIn . . . . . . . . . . . . . . . . . . . . . . . 32
       5.6.1.  Data Handling  . . . . . . . . . . . . . . . . . . . . 32
       5.6.2.  Components . . . . . . . . . . . . . . . . . . . . . . 32
       5.6.3.  Capabilities . . . . . . . . . . . . . . . . . . . . . 32
       5.6.4.  Events . . . . . . . . . . . . . . . . . . . . . . . . 32
     5.7.  OFRedirectOut  . . . . . . . . . . . . . . . . . . . . . . 33
       5.7.1.  Data Handling  . . . . . . . . . . . . . . . . . . . . 33
       5.7.2.  Components . . . . . . . . . . . . . . . . . . . . . . 33
       5.7.3.  Capabilities . . . . . . . . . . . . . . . . . . . . . 33
       5.7.4.  Events . . . . . . . . . . . . . . . . . . . . . . . . 33
     5.8.  OFAction . . . . . . . . . . . . . . . . . . . . . . . . . 33
       5.8.1.  Data Handling  . . . . . . . . . . . . . . . . . . . . 33
       5.8.2.  Components . . . . . . . . . . . . . . . . . . . . . . 34
       5.8.3.  Capabilities . . . . . . . . . . . . . . . . . . . . . 34
       5.8.4.  Events . . . . . . . . . . . . . . . . . . . . . . . . 34
     5.9.  OFActionLFBs . . . . . . . . . . . . . . . . . . . . . . . 34
       5.9.1.  OFActionOutput . . . . . . . . . . . . . . . . . . . . 34
       5.9.2.  OFActionSetVLANVID . . . . . . . . . . . . . . . . . . 35
       5.9.3.  OFActionSetVLANPriority  . . . . . . . . . . . . . . . 35
       5.9.4.  OFActionSetMACSource . . . . . . . . . . . . . . . . . 35
       5.9.5.  OFActionSetMACDestination  . . . . . . . . . . . . . . 35
       5.9.6.  OFActionSetIPSource  . . . . . . . . . . . . . . . . . 36
       5.9.7.  OFActionSetIPDestination . . . . . . . . . . . . . . . 36
       5.9.8.  OFActionSetIPTOS . . . . . . . . . . . . . . . . . . . 36
       5.9.9.  OFActionSetIPECN . . . . . . . . . . . . . . . . . . . 36
       5.9.10. OFActionSetTCPSource . . . . . . . . . . . . . . . . . 37
       5.9.11. OFActionSetTCPDestination  . . . . . . . . . . . . . . 37
       5.9.12. OFActionCopyTTLOut . . . . . . . . . . . . . . . . . . 37
       5.9.13. OFActionCopyTTLIn  . . . . . . . . . . . . . . . . . . 38
       5.9.14. OFActionSetMPLSLabel . . . . . . . . . . . . . . . . . 38
       5.9.15. OFActionSetMPLSTC  . . . . . . . . . . . . . . . . . . 38
       5.9.16. OFActionSetMPLSTTL . . . . . . . . . . . . . . . . . . 38
       5.9.17. OFActionDecrementMPLSTTL . . . . . . . . . . . . . . . 39
       5.9.18. OFActionPushVLan . . . . . . . . . . . . . . . . . . . 39
       5.9.19. OFActionPopVLAN  . . . . . . . . . . . . . . . . . . . 39
       5.9.20. OFPushMPLSOFAction . . . . . . . . . . . . . . . . . . 39
       5.9.21. OFPopMPLSOFAction  . . . . . . . . . . . . . . . . . . 40
       5.9.22. OFSetQueueOFAction . . . . . . . . . . . . . . . . . . 40
       5.9.23. OFSetIPTTLOFAction . . . . . . . . . . . . . . . . . . 40
       5.9.24. OFDecrementIPTTLOFAction . . . . . . . . . . . . . . . 40
       5.9.25. OFExperimenterOFAction . . . . . . . . . . . . . . . . 41
   6.  XML for OpenFlow library . . . . . . . . . . . . . . . . . . . 42
   7.  Acknowledgements . . . . . . . . . . . . . . . . . . . . . . . 92
```

8. IANA Considerations 93
9. Security Considerations 94
10. References 95
  10.1. Normative References 95
  10.2. Informative References 96
Authors' Addresses 97

1. Introduction

The purpose of this document is to create a library of Logical Functional Blocks that are necessary to describe an OpenFlow switch using the ForCES model. This includes DataTypes, MetaData and of course the LFBs.

Readers of this document can get a better understanding of what are the internal parts of an OpenFlow switch in a more formal approach. Additionally having a ForCES-defined OpenFlow switch allows developers to build a purely ForCES based solution that understands the OF model or even a middleware so that ForCES-implemented OpenFlow switches may be controlled by an OpenFlow controller, or a ForCES Control Element (CE) may control OpenFlow switches

1.1. ForCES

ForCES [RFC3746], focuses on the communication and model necessary to separate control-plane functionality such as routing protocols, signaling protocols, and admission control, from data-forwarding-plane per-packet activities, such as packet forwarding, queuing, and header editing.

The modeling of FEs is based on an abstraction using distinct Logical Functional Blocks (LFBs), which are interconnected in a directed graph, and receive, process, modify, and transmit packets along with metadata. An LFB is a block of encapsulated fine-grained operation of the forwarding plane. The ForCES model [RFC5812] additionally includes both a capability and a state model. One of the advantages of the ForCES Model is that it is independent of the actual implementation of the FE; it only provides a view of its capabilities and state that can be acted upon using the ForCES protocol. It is left to the forwarding plane developers to define how the FE functionality is represented using the model.

The ForCES protocol [RFC5810] was developed to allow the CEs to determine the capabilities of each FE expressed by the FE model, to add and remove entries, parameters, query for statistics, and register for and receive events in a scalable fashion over secure and reliable means. The strength of the ForCES protocol stems from the fact that it is agnostic of the model, as a CE can control any Forwarding Element described with the ForCES model.

1.2. OpenFlow

OpenFlow [OpenFlowSpec1.1] is conceptually similar to ForCES on separating the control and forwarding plane. It provides a protocol that mediates between the controller and the switch. Unlike ForCES, the OpenFlow switch is statically defined to deal with flows and the protocol is aware of the flow components. An OpenFlow Switch consists of one or more flow tables, a group table that performs packet lookups and forwarding, and an OpenFlow channel to an external controller. A flow table is consisted of flow entries, each containing a set of match fields to match against packets, counters and instructions. The controller manages the switch via the OpenFlow protocol. Using this protocol, the controller can add, update, and delete flow and group entries.

2. Terminology and Conventions

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.2. Definitions

This document follows the terminology defined by ForCES related documents of RFC3654, RFC3746, RFC5810,RFC5811,RFC5812,RFC5812. The definitions are repeated below for clarity. Also additional definitions from the OpenFlow specification 1.1 [OpenFlowSpec1.1] are also included.

Control Element (CE) - A logical entity that implements the ForCES protocol and uses it to instruct one or more FEs on how to process packets. CEs handle functionality such as the execution of control and signaling protocols.

Forwarding Element (FE) - A logical entity that implements the ForCES protocol. FEs use the underlying hardware to provide per-packet processing and handling as directed/controlled by one or more CEs via the ForCES protocol.

LFB (Logical Functional Block) - The basic building block that is operated on by the ForCES protocol. The LFB is a well defined, logically separable functional block that resides in an FE and is controlled by the CE via the ForCES protocol. The LFB may reside at the FE’s datapath and process packets or may be purely an FE control or configuration entity that is operated on by the CE. Note that the LFB is a functionally accurate abstraction of the FE’s processing capabilities, but not a hardware-accurate representation of the FE implementation.

LFB Class and LFB Instance - LFBs are categorized by LFB Classes. An LFB Instance represents an LFB Class (or Type) existence. There may be multiple instances of the same LFB Class (or Type) in an FE. An LFB Class is represented by an LFB Class ID, and an LFB Instance is represented by an LFB Instance ID. As a result, an LFB Class ID associated with an LFB Instance ID uniquely specifies an LFB existence.

LFB Metadata - Metadata is used to communicate per-packet state from one LFB to another, but is not sent across the network. The FE model defines how such metadata is identified, produced, and consumed by the LFBs. It defines the functionality but not how metadata is encoded within an implementation.

LFB Components - Operational parameters of the LFBs that must be visible to the CEs are conceptualized in the FE model as the LFB components. The LFB components include, for example, flags, single-parameter arguments, complex arguments, and tables that the CE can read and/or write via the ForCES protocol (see below).

ForCES Protocol - While there may be multiple protocols used within the overall ForCES architecture, the term "ForCES protocol" and "protocol" refer to the "Fp" reference points in the ForCES framework in [RFC3746]. This protocol does not apply to CE-to-CE communication, FE-to-FE communication, or to communication between FE and CE managers. Basically, the ForCES protocol works in a master-slave mode in which FEs are slaves and CEs are masters.

ForCES Protocol Transport Mapping Layer (ForCES TML) - A layer in ForCES protocol architecture that uses the capabilities of existing transport protocols to specifically address protocol message transportation issues, such as how the protocol messages are mapped to different transport media (like TCP, IP, ATM, Ethernet, etc.), and how to achieve and implement reliability, multicast, ordering, etc. The ForCES TML specifications are detailed in separate ForCES documents, one for each TML.

Match Field - a field against which a packet is matched, including packet headers, the ingress port, and the metadata value.

Action - an operation that forwards the packet to a port or modifies the packet, such as decrementing the TTL field. Actions may be specified as part of the instruction set associated with a flow entry or in an action bucket associated with a group entry.

Flow entry - an element in a flow table used to match and process packets. It contains a set of match fields for matching packets, a set of counters to track packets, and a set of instructions to apply.

Instruction - an operation that either contains a set of actions to add to the action set, contains a list of actions to apply immediately to the packet, or modifies pipeline processing.

Flow Table - A stage of the pipeline, contains flow entries.

OpenFlow pipeline - the set of linked flow tables that provide matching, forwarding, and packet modifications in an OpenFlow switch.

Groups - a list of action buckets and some means of choosing one or more of those buckets to apply on a per-packet basis.

Group Entry - an element in a group table. It contains a group identifier to distinguish groups, a group type to define the type of the group, a set of counters to track packets, and a set of action buckets.

Action Bucket - a set of actions and associated parameters, defined for groups.

Action Set - a set of actions associated with the packet that are accumulated while the packet is processed by each table and that are executed when the instruction set instructs the packet to exit the processing pipeline.

Ports - where packets enter and exit the OpenFlow pipeline. May be a physical port, a logical port defined by the switch, or a reserved port defined by the specification.

3. OpenFlow ForCES library

3.1. OpenFlow Specification

An OpenFlow switch as described in the OpenFlow Specification document [OpenFlowSpec1.1] appears in Figure 1

```
    +---------+                +---------+                   +---------+
    |         |                |         |                   |         |
    |  Port   |                |  Group  |                   |  Port   |
    |         |                |  Table  |                   |         |
    +---------+                +---------+                   +---------+
         | M1                                                     /\
         | M2                                                     |
         \/                                                       |
    +---------+      +---------+            +---------+      +---------+
    |         | M1   |         | M1         |         | M2   | Execute |
    |  Flow   | ---> |  Flow   | -->...---> |  Flow   | ---> |  Action |
    | Table 0 | M2   | Table 1 | M2         | Table N |      |   Set   |
    +---------+ M3   +---------+ M3         +---------+      +---------+

    Legend
    M1: Ingress Port
    M2: Action Set{}
    M3: Metadata
```

Figure 1: OpenFlow switch datapath

A packet enters the switch through a Port and is passed on the first Flow Table along with the Ingress Port as a Metadata (M1). Additionally each frame carries around a list of actions, called Action Set (M2), which have initially no actions in it. The Action Set will be executed at the end of the DataPath in the Execute Action Set block. After the first Flow Table another metadata called Metadata (M3) also accompanies the packet. This data inside the metadata maybe written by the Flow Tables when the Write Metadata instruction is applied.

Each Flow Table performs a match based on certain fields (e.g IP Source Address or Source MAC Address) and then perform a specific instruction if there is a match. If no match occurs, the frame is processed based on the Flow Table’s configuration. The choices are either:

a. Forward to the OpenFlow controller

b. Send to the next flow table

c. Drop the frame

The list of instructions a Flow Table may perform upon a match are:

o Apply a List of actions

o Clear the Action Set

o Write actions on the action set

o Write Metadata

o Go to Flow Table (allows a FlowTable X to send the packet and metadata to any FlowTable Y, provided that X>Y)

In OpenFlow there are two types of action executions which are independent of each other. The first one refered to as action list and is programmed into the flow table to be executed immediately within the packet pipeline upon a match on a flow table. The second one gets executed at the end of the pipeline in the execute action set. The second type of actions is collected in a metadata refered to as Action Set during the datapath processing with hte Write Actions instruction.

The type of actions the Flow Table can perform or write in the Action Set is:

o Setting of a field (e.g. IP address, MAC address)

o Push or Pop tags (VLAN, MPLS)

o Copy TTL inwards or outwards

o Decrease TTLs

o Output the packet to ports (a copy of the original packet will be sent to the port)

o Apply QoS to a packet

o Apply the packet to a group (a copy of the original packet will be sent to the group)

Additionally a Flow Table may drop the packet. The drop is implicit based on the Flow Table’s configuration (e.g. when there are no more instructions).

An Action Set MUST contains a maximum of one action of each of the following class of types which MUST be executed in the order specified below regardless of the order they were added to the Action Set. The output action in the action set is executed last. If both an output action and a group action are specified in an action set, the output action is ignored and the group action takes precedence. If no output action and no group action were specified in an action set, the packet is dropped.

1. Copy TTL inwards

2. Pop a tag (maximum one of VLAN tag, MPLS tag)

3. Push a tags (maximum one of VLAN tag, MPLS tag)

4. Copy TTL outwards

5. Decrease TTL

6. Setting of a field (maximum one of set IP address, set VLAN ID, etc...)

7. Apply QoS to a packet

8. Apply the packet to a group

9. Output the packet

The Group Table contains a set of Group Entries, each of which contains a set of Action Buckets, each of which contains a set of actions which can be applied to a group of packets that don’t have the same set of matching fields. This alleviates the problem of having to set up the same set of actions in flow tables for different set of matching fields by having these set of actions in one place only.

3.2. ForCES-based OpenFlow Specification

ForCES models FEs using LFBs, fine-grained operations of the forwarding plane. It is logical to have at least the following LFB classes:

1. OFPort

2. OFFlowTables

3. OFGroupTable

4. OFActions

5. OFQueue

Additionally packets may be sent to the controller or the controller may send packets to the switch to be put on the datapath. RedirectIn and RedirectOut are two LFBs defined in the Base LFB Library [I-D.ietf-forces-lfb-lib]. However as some more metadata are required for the OpenFlow switch, the OFRedirectIn and OFRedirectOut will be defined by extending the initial LFBs.

While it may seem that having multiple OFFlowTables instances to represent each Flow Table in the OpenFlow, the authors decided to model the OFFlowTables to contain all the Flow Tables in one instance of the OFFlowTables. On of the OFFlowTables’s components is an array of Flow Tables entries and each entry contains its own Flow Entries, Flow Table Counter and Miss Behaviour. The index of the Flow Tables entry represents the Flow Table ID. The rationale behind such a decision is the simplification of the model. With multiple ActionLFBs and multiple FlowTables, the resulting connection graph between Flow Tables and FlowTables and ActionLFBs would be very complex. Addionally this simplifies also the hanlding of two metadatas, the ActionSet Metadata and the Metadata which are now invisible to the model as they are passed only between Flow Tables. Figure 2 shows an example of how the OFFlowTables is internally.

```
        | From          /\To OFOutput        /\To Group    /\To Redirect
        | OFPorts       | Action             | Table        | Out
   +----|---------------|--------------------|--------------|----------+
   |    |               |                    |              |          |
   |    |   +-----------+--------------------+--------------+          |
   |    \/  |           |                    |              |          |
   | +--------+     +--------+           +--------+       +--------+   |
   | |        | --> |        |           |        |       |        |   |
   | |  Flow  |     |  Flow  |--+---+--> |  Flow  |-+--+->|  Flow  |   |
   | | Table 0|--+  | Table 1|  |   /\   | Table 2| /\ /\ | Table N|   |
   | +--------+  |  +--------+  |   |    +--------+ |  |  +--------+   |
   |   /\  /\    |    /\  |     |   |      /\  |    |  |       /\      |
   |   |   |     |    |   |     +----------|---|----+  |       |       |
   |   |   |     |    |   |         |      |   |       |       |       |
   |   |   |     +----|---+---------+------|---+-------+       |       |
   |   |   |          |                    |                   |       |
   |   |   +----------+-------------+------+-------------------+       |
   |   |                            |                                  |
   +---|----------------------------|----------------------------------+
       |From                        | To/From
       |Redirect In                 \/OFActions
```

Figure 2: FlowTable Internal

Figure 3 depicts what an OFFlowTables with its own set of ActionLFBs would look like. Figure 4 depicts how the OFFlowTables work with a shared set of ActionLFBs with the OFGroupTable LFB look like.

```
                +------+ M1,M2,M3        M1,M2,M3   +---------+
                | Push | P(2)            P(1)       |   Set   |
                | Vlan |<----------+   +----------->|    IP   |
                |Header|           |   |            | Address |
                +------+           |   |            +---------+
                   |               |   |                |
                   | P(3),M1     +--------+  P(2),M1    |
                   +------------>|        |<------------+
                                 |        |
          ---------------------->| OFFlow |---------------------->
              P                  | Tables |    P(3)
                                 |        |
            +---------+  P,M1,M2 |        |
            |         | <------- |        |       Legend:
            |Decrement|          |        |       P:  Packet
            |  IP TTL |          +--------+       M1: PacketID
            |         |  P(1),M1     /\           M2: LFBClassID
            +---------+ -------------+            M3: ActionIndex
```

Figure 3: ForCES FlowTable with each own set of ActionLFBs

A packet P enters the OFFlowTables from an OFPortLFB. If a match occurs for the packet within the OFFlowTables and if the instruction for that match is an "Apply Action List" then the actions must be performed immediately and in the order specified in the action list. Alongside the packet, some metadata are passed as well. M1, the PacketID required by the OFFlowTables to identify the packet and continue exectution from where it stopped when it returns, M2 the LFBClassID so that the ActionLFB knows the ClassID of the LFB to return the packet and M3 the ActionIndex required in some ActionLFBs to determine parameters for the action. For example in Figure 3 let’s assume that the action list includes the following four actions:

1. Decrement IP TTL

2. Set IP Address

3. Push VLAN header

4. Final Action

The packet P will be first sent to the Decrement IP TTL Action LFB. Upon completion it will be returned as P1 to the OFFlowTables and then will be sent to the Set IP Address LFB. Upon return as P2 it will be sent to the Push VLAN Header and returned finally as P3 to the Flow Table LFB. Then depending upon the final action the packet may:

o A copy of P3 will be sent to a Port LFB if it is an output action and the action set will be executed.

o A copy of P3 will be sent to the Group LFB if it is an group action and the action set will be executed.

o Remain in the OFFlowTables and checked for a match in the Flow Table specified by a Goto action if it is a goto action.

```
              +---------+  P(1),M1,M2,M3
              | Set IP  |<-----------------+
              | Address |                  |
              |         |--------------+   |
              +---------+  P(2),M1     |   |
                                       |   |
                                       |   |
                         P(2),M1       \/  |
                +------+ M2,M3      +---------+
                | Push |<-----------|  Group  |   P3
                | Vlan |            |  Table  |------->
                |Header|----------->|         |
                +------+   P(3),M1  +---------+
              P  /\  |              P(1) /\
              M1 |   |                   |
              M2 |   |P(1),M1            |
              M3 |   \/                  |
               +--------+                |          Legend:
           P   |        |----------------+          P: Packet
         ----->| OFFlow |                           M1: PacketID
               | Tables |----------------->         M2: LFBClassID
               +--------+   P(1)                    M3: ActionIndex
```

Figure 4: ForCES FlowTables and GroupTable with a common ActionLFB

A packet enters the OFFlowTables from an OFPortLFB. If a match occurs for the packet within the OFFlowTables and if the instruction for that match is an "Apply Action List" then the actions must be performed immediately and in the order specified in the action list. For example in Figure 4 let’s assume that the action list includes the following three actions:

1. Push VLAN Header

2. Group

3. Output

The packet P will be first sent to the Push VLAN header Action LFB and upon completion will be returned as P1 to the OFFlowTables. Then a copy of P1 will be sent to the Group Table LFB and then finally a copy of P1 will be sent to the Port LFB for output and the action set will be executed.

Let’s assume that the following actions will be executed for P1 in the action bucket in the Group Table.

1. Set IP Address

2. Push Vlan Header

3. Output

The packet P1 will be sent to the Set IP Address Action LFB and upon completion will be returned as P2 to the Group Table. Then the packet P2 will be sent to the Push Vlan Header Action LFB and be returned as P3 to the Group Table and be sent to the Port LFB for output.

Regarding the OFActions, when a match occurs in the OFFlowTables or an action bucket must be executed in the OFGroupTable it may contain multiple actions, it seems reasonable to separate each action as an individual LFB that performs that specific action. For every action needed to be executed, the OFFlowTables or the OFGroupTable will send the frame to the appropriate action LFB(s) in the order defined either by the Action Bucket or the instructions in the OFFlowTables’s match entry. Once the packet has been processed from an Action LFB, it MUST be returned to the LFB instance that made that call. OFFlowTables and OFGroupTable LFB may either have shared ActionLFBs or separate.

Current specified Action LFBs are:

Output Actions

- OFActionOutput

Set Queue Actions

- OFActionSetQueue

Push/Pop Tag Actions

- OFActionPushVLan
- OFActionPopVLAN
- OFActionPushMPLS
- OFActionPopMPLS

Set Actions

- OFActionSetMACSource
- OFActionSetMACDestination
- OFActionSetVLANVID
- OFActionSetVLANPriority
- OFActionSetMPLSLabel
- OFActionSetMPLSTC
- OFActionSetMPLSTTL
- OFActionDecrementMPLSTTL
- OFActionSetIPSource
- OFActionSetIPDestination
- OFActionSetIPTOS
- OFActionSetIPECN
- OFActionSetIPTTL
- OFActionDecrementIPTTL
- OFActionSetTCPSource
- OFActionSetTCPDestination
- OFActionCopyTTLOut
- OFActionCopyTTLIn

Experimenter Actions

- OFActionExperimenter

Most Action LFBs have data associated with the action, e.g. an IP Address for the SetIPSource or SetIPDestination actions, stored in an array in the LFB. The FlowTable sending the packet needs to send additionally as a metadata an index pointing to the action parameter needed for the execution. Each Action LFB has one group input port that accepts a packet, the LFBClassID of the LFB that sent the packet, so that it can be returned after the action has been performed, and optionally the Action Index. Furthermore, one more metadata is required, the PacketID with which the Flow Table LFB or the Group Table LFB can keep track of a packet's progress. Additionally each Action LFB has one group output port that returns the altered packet to the sender. Since the action LFBs have these ports in common and the ForCES model can support augmentation of LFB classes, similar to inheritance in object oriented programming, an OFActionLFB has been specified from which all Action LFBs are derived from.

The Action LFBs can be used also by the OFGroupTable using the same input and output port.

Additionally each OFFlowTables can output a packet to a specific port through the OFOutputAction LFB. Figure 5 shows an example of a topology and how the various LFBs are interconnected. The controller can obtain the topology information by querying the FEObject's LFBTopology.

```
        /\          /\            /\           /\             /\
        | Out       | Out         | Out        | Out          | Out
        |           |             |            |              |
    +-------+    +-------+    +-------+     +-------+      +-------+
    |       |    |       |    |       |     |       |      |       |
    |OFQueue|    |OFQueue|    |OFQueue| ... |OFQueue|      |OFQueue|
    |       |    |       |    |       |     |       |      |       |
    +-------+    +-------+    +-------+     +-------+      +-------+
        /\           /\           /\           /\             /\
        |            |            |            |              |
        |            +--+---------+            |              |
        |               |                      |              |
      +--------+     +--------+            +--------+     +--------+
   In |        |     |        |<--  In  -->|        |     |        | In
   -->| OFPort |     | OFPort |            | OFPort |     | OFPort |<--
      |        |     |        |     ...    |        |     |        |
      +--------+     +--------+            +--------+     +--------+
        |   /\         |   /\                |   /\           |  /\
        |   |          |   |                 |   |            |  |
        +---|----------+---|-----------------+---|------------+  |
        |   |              |                     |            |  |
        |   +--------------+------+--------------+---------------+
        |                         |
        |                    +----------+       +------------+
        |                    |          |       |            |To
        |   +---------------->| OFOutput |------->| OFRedirect |-->
        |   |                |  Action  |       |    Out     |Controller
        |   |                +----------+       +------------+
        |   |                    /\                    /\
        \/  |                    |                     |
     +--------+              +---------+               |
     |        |------------->|         |               |
     | OFFlow |              | OFGroup |               |
     | Tables |-----+        |  Table  |               |
     +--------+     |        +---------+               |
       /\   /\      |           /\                     |
       |    |       +-----------|----------------------+
       |    |                   |
       |    |    +--------------+-------------+--------------+
       |    |    |              |             |              |
       |    +----|----------+---|----------+--|-----------+  |
       |    |    |          |   |          |  |           |  |
       |    \/   \/         \/  \/         \/ \/          \/ \/
       |   +--------+     +--------+     +--------+     +--------+
       |   |        |     |        |     |        |     |        |
       |   |OFAction|     |OFAction|     |OFAction|     |OFAction|
       |   |        |     |        |     |        |     |        |
       |   +--------+     +--------+     +--------+     +--------+
       |
   +------------+
   |            |From
   | OFRedirect |<--
   |     In     |Controller
   +------------+
```

Figure 5: ForCES OpenFlow Switch example LFB connectivity

Regarding the execution of the Action Set, it is considered implementation specific and should be performed internally in the OFFlowTables using the ActionLFBs. Additionally the execution of OpenFlow’s PacketOut message which contains a list of actions to be performed on a buffered or a redirected packet in the switch is also implementation specific and should also be performed internally in the OFFlowTables.

OpenFlow 1.1 provides, with the use of the experimenter concept new instruction or action types. In this model, new instruction types can be modeled by expanding the InstructionTypes datatype definition and new action types can be modeled by creating new ActionLFBs.

4. OpenFlow Base Types

Some datatypes in this LFB library are imported from Base LFB Library [I-D.ietf-forces-lfb-lib] as they have already been defined there.

4.1. Data Types

Data types defined in the OpenFlow library are categorized by types of atomic, compound struct, and compound array.

4.1.1. Atomic

The following data types are defined as atomic data types in the OpenFlow library:

Data Type Name	Brief Description
MPLSLabelValue	An MPLS label
MPLSTrafficClassValues	The MPLS Traffic Class
IPv4ToSbits	TOSBits
ActionType	The possible actions
InstructionTypes	Instructions supported
FlowTableMissConfigType	Types to configure the default behavior of unmatched packets in a Flow Table
PacketInTypes	Packet In Types
GroupBucketExecuteType	To determine which Action Bucket(s) should be executed (all, select, indirect, fast failover)
PortNumberType	Port Number values
QueuePropertyType	Property type for a queue

OpenFlow Atomic Types

4.1.2. Compound Struct

The following data types are defined as struct data types in the OpenFlow library:

Data Type Name	Brief Description
SwitchDescriptionType	Fields of the switch description
WildcardsType	Wildcards for fields
MatchFieldType	A Match Field Type (contains all possible match fields)
FlowEntry	A Flow entry type
ActionRowType	An Action Row for the action table
TableCounterType	Counter per table
FlowCounterType	Counter per flow
WriteMetadataTableType	Metadata and mask for the write metadata instruction per row
GroupCounterType	Counters per group
BucketCounterType	Counters per bucket
GroupTableEntry	A Row of the Group Table
ActionBucket	An Action Bucket
PortConfigurationType	Types of configuration for the OpenFlow port
PortStateType	Current State of the port
PortFeaturesType	Port Features
PortCounterType	Counter per port
QueueArrayPropertiesType	Type Definition for property
QueueCounterType	Counters per queue

OpenFlow Struct Types

4.1.3. Compound Array

The following data types are defined as an array data type in the OpenFlow library

Data Type Name	Brief Description
Actions	Actions to perform. An Array of ActionRowTypes

OpenFlow Array Types

4.2. Frame Types

No additional frame types are defined in this library

4.3. MetaData Types

The following metadata are defined in the OpenFlow type library:

MetaData Name	Brief Description	MetaData ID	Metadata Type
IngressPort	The Ingress port the packet has arrived from.	1024	uint32
InPhyPort	The Port Index of the Physical interface the frame entered the switch	1025	uint32
PacketID	The PacketID metadata is used to uniquelly identify a packet within the Flow Table or the Group Table to continue processing it after it has been returned from an OFActionLFB.	1026	uint32

ActionIndex	The Action Index metadata is used to point the row in the array in an Action LFB	1027	uint32
GroupIndex	The Group index metadata is used to point to the row of the array in an Group LFB	1028	uint32
LFBClassIDMetadata	The LFBClassID	1029	uint32
QueueIDMetadata	The Queue ID the packet should be sent to	1030	uint32
BufferID	The Buffer ID of a stored packet in the switch requried for the PacketOut message	1031	uint32
RedirectReason	The reason the packet was redirected to the controller	1032	uchar
FlowTableID	The FlowTable ID the packet was sent to the controller from	1033	uchar
ActionListMetadata	The Action List that may come along with the packet in a PacketOut message	1032	octetstring

OpenFlow Metadata Types

5. OpenFlow LFBs

5.1. OpenFlowSwitch

Similar to the concept of the FEProtocol LFB and the FEObject LFB, the OpenFlowSwitchLFB contains information and configuration parameters regarding the functionality of the switch but play no role in the datapath processing. Therefore there are no input or output ports.

5.1.1. Data Handling

This LFB does not handle any data.

5.1.2. Components

The DatapathID component, a unsigned integer of 64 bits, uniquely identifies a datapath. The lower 48 bits are intended for the switch MAC address, while the top 16 bits are up to the implementer.

The MissSendLen component, an unsigned integer of 16 bits, defines the maximum number of bytes of each packet sent to the controller as a result of both flow table misses and flow table hits with the controller as the destination.

The HandleFragments component, a Boolean, defines what the switch does with fragments. If true the switch will drop fragments. If false there is no special handling.

The ReassembleFragments component, a Boolean, defines if the switch will reassemble fragments.

The InvalidTTLtoController component, a Boolean, defines whether the switch will send packets with invalid TTL to the controller.

The SwitchDescription component, a structure, contains the following information about the switch:

- o Manufacturer description
- o Hardware description
- o Software description
- o Serial Number
- o Human readable description of datapath

Lastly the Ports component is an array which contains in its rows, all the port numbers.

5.1.3. Capabilities

The following capabilities have been defined for the OpenFlowSwitch LFB

An assortment of Boolean type capabilities to define:

- o FlowStatistics. If the switch keeps flow statistics
- o TableStatistics. If the switch keep table statistics
- o PortStatistics. If the switch keep port statistics
- o GroupStatistics. If the switch keep group statistics
- o IPReassembly. If the switch can reassemble IP fragments
- o QueueStats. If the switch keeps queue statistics
- o ARPMatchIP. If the switch matches IP addresses in ARP packets

The MaxBufferedPackets capability, an unsigned integer of 32 bits, defines the maximum packets the switch can buffer when sending packets to the controller.

The TablesSupported capability, an unsigned integer of 8 bits, defines the number of tables supported by the switch, each of which can have a different set of supported wildcard bits and number of entries.

Additionally the another capability, the ActionSupported, defines the supported actions for the switch.

5.1.4. Events

Three events have been specified regarding the ports. The first event will be triggered when a new port is added to the switch, the second when a port has been removed from the switch and the third when a port has been modified

5.2. OFFlowTables

An LFB that houses all OpenFlow Flow Tables residing in the switch.

5.2.1. Data Handling

The OFFlowTables describes the process of selecting packets and classify them into specific flows based on specific match fields assigned by the controller.

The LFB is expected to receive all types of Ethernet packets through a group input named InputPort from an OFPort along with the IngressPort and the InPhyPort as metadata.

All Flow Tables reside in an array within the LFB, the index of the array being the Flow Table ID.

Each Flow Table compares the packet with the MatchFields inside the FlowEntries Table. If there is no match, depending upon the MissBehaviour component, one of the following actions will occur:

1. The packet or part of it will be sent to the controller via the singleton output port RedirectPacketOut along with the IngressPort, InPhyPort, RedirectReason and FlowTableID as metadata and optionally the BufferID if the packet is buffered in the switch. How the packet is buffered in the switch is implementation specific.

2. The packet will be sent it to the next table in the pipeline (the next flow table row entry in the array).

3. The packet will be dropped.

If there is a match the LFB will decide based on the InstructionType of the component Instructions inside the matched FlowEntry.

If the instruction is Apply Actions, the LFB will use the InstructionIndex to find the Actions inside the ApplyActionList. Each row of the ApplyActionList is an array containing rows of ActionRowTypes. For every ActionRowType, the LFB will send the packet to the corresponding Action LFB through the group output ActionPort port alongside with the LFBClassIDMetadata of the LFB, the PacketID and the ActionIndex, if the specific action has any parameters, like the Set MAC Address action. The ActionIndex is used as an index for the table inside the Action LFB. The packet is then returned from the Action LFB through the group input port PacketReturn to continue further processing if exists. The PacketID is an identifier issued by the OFFlowTables LFB that will uniquely identify the packet so that when it returns it will continue processing from whence it stopped.

One exception to the rule when applying the action list is in regards to the Group action. The OFGroup LFB handles groups. The OFFlowTables using the ActionIndex locates the Group Identifier in the OFFlowTables’s GroupTable component. Then it sends a copy of the packet to the GroupTableLFB using the group output ActionPort and the original packet will continue in the Flow Table.

As the ActionSet metadata is an internal component of the OFFlowTables LFB, unreadable and unsettable by the Controller, both instructions Clear Actions and Write Actions are implementation specific. The same applies with the Write Metadata instruction.

If the instruction is Goto Table, the LFB will use the InstructionIndex to find the value OFFlowTables InstanceID in the GoToFlowTable table and send the packet to the FLowTables row entry with the index of the FlowTable ID.

Additionally the OFFlowTables handle packets incoming from the controller expecting either a whole packet through the RedirectPacketIn singleton input port or the buffer ID through the RedirectBufferIn singleton input port from the OFRedirectIn LFB, along with the ActionList and the IngressPort as metadata.

5.2.2. Components

The FlowTables, an array holding all Flow Tables of the switch. Each row of the array is a struct of the FlowEntries, the FlowTableCounter and the MissBehaviour for each flow. The FlowEntries component of the struct is an array and each row of the array is a struct containing the cookie, the MatchFields, the Counters, the Instructions, the Timeouts, the Timers and the priority of the specific flow entry. The FlowTableCounter are the counters of the specific FlowTable and the MissBehaviour component of the struct specifies what the specific Flow Table shall do with the packet if there is no match.

The ApplyActionList is a component to maintain the actions required per flow. It is an array of Actions, which is an array of a struct of ActionType and ActionTableIndex.

The WriteActions is a component to maintain the actions to be writen for a write actions instruction. It is an array of Actions, which is an array of a struct of ActionType and ActionTableIndex.

The WriteMetadataTable is a component to hold the metadata values required for the write metadata value. It is an array of WriteMetadataTableType, which is a struct of the Metadata value and the MetadataMask.

The GotoFlowTable component contains the FlowTable IDs flows should go to for the goto table action. It is an array of uint32. The value is selected using the InstructionIndex.

The GroupTable component contains group identifiers. It is an array of group identifiers indexed by the ActionTableIndex.

5.2.3. Capabilities

This LFB has no capabilities specified.

5.2.4. Events

One event has been defined regarding the Flow Table. The event will be triggered when a flow is deleted from the Flow Table whether due to the idle timeout, or to the hard timeout or a flow was deleted by the controller.

5.3. OFGroupTable

The Group LFB contains Action Buckets that can be applied to a packet during its path in the Flow Tables pipeline.

5.3.1. Data Handling

The OFGroupTable does not take part in the actual handling of the data. Rather, it contains the action per group which are required by all Flow Tables in the pipeline. Packets initially enter this LFB from an OFActionSet LFB or a OFFlowTables via the group input port PacketIn and using the GroupIndex metadata the LFB finds the group requested for this packet. Then the LFB depending on the requested actions sends the packet to the required OFActionLFBs via the group output ActionPort and expects results via the group input PacketReturn.

5.3.2. Components

The LFB has only one component which is the Group Table. This is an array of GroupTableEntry types. Each GroupTableEntry contains a Group Identifier, the type of Group, the required counters and an array of action buckets.

An action bucket is a struct which contains the Group weight required for select groups, the watch port and watch group required for fast failover groups, the bucket counters and the actions for this bucket.

The structure of actions in a bucket are identical to the actions in the flow table LFB containing the type of action and an action table index. With the action type and action index the Group LFB can identify the component and index of the action details.

5.3.3. Capabilities

This LFB has no capabilities specified.

5.3.4. Events

This LFB has no events specified.

5.4. OFPort

5.4.1. Data Handling

This LFB abstract the point where packets enter and exit the OpenFlow switch pipeline. May be a physical port, a virtual port defined by the switch. The LFB handles Ethernet frames coming in or going out to/of the OpenFlow switch. Ethernet frames are received and passed to an OFFlowTables through the singleton output port PacketIn, along with the IngressPortID and InPhyPort metadata.

When a packet is ready to be send on the wire, it is sent to an OFPort instance through the group input port PacketOut and then depending of whether the packet has been assigned a queue with the QueueID as metadata and then is sent to the OFQueue LFB via the group output port QueueOut.

5.4.2. Components

The PortNumber component uniquely identifies the port within a switch.

The IEEEMAC component contains the MAC Address of the port.

The Name component is a human readable name of the port.

The Configuration component specifies port behaviour. It’s a struct component with the following boolean fields. PortDown, NoReceiving, NoForwarding and NoPacket_In.

The State component defines the current state of the OpenFlow Switch. It is a struct component that defines whether the link is down, the port is blocked or the port can be used for live fast failover.

The Current Features component describes the current features of the port. It is a struct component and specifies the Speed Types, the Connected Medium, the Auto Negotiation and the Pause Types

The Advertised Features component describes the advertised features of the port. The component is of the same structure as the current features.

The CurrentSpeed component defines the current port bitrate in kbps.

The MaximumSpeed component defines the maximum port bitrate in kbps.

The PortCounter component contains the necessary counters for the port. It is a struct component comprised of counters for Packets Received, Packets Transmitted, Bytes Received, Bytes Transmitted, Drops Received, Transmit Drops, Errors in reception, Errors in transmittion, Frame Alignment Errors received, Received Overrun Errors, CRC Errors in received packets, Collisions.

5.4.3. Capabilities

Two capabilities has been defined for the Port LFB. Supported Features and Peer Features. These include:

- o Types of Speed supported
- o Medium Connected to the port
- o Auto-negotiation
- o Pause Types supported of the port

5.4.4. Events

This LFB has no events specified.

5.5. OFQueue

5.5.1. Data Handling

This LFB manages the queuing algorithm for handling packets prior to be output from the switch. Multiple OFQueue LFBs can be attached to an OFPort LFB to handle queues. If a packet has been set a QueueID with a Set Queue action, it is sent to the OFQueue LFB after the OFPort LFB and it enters this LFB via the group input port PacketIn where it will be handled according to the LFBs configuration and then be output from the switch.

5.5.2. Components

The QueueID component, a uint32, defines the ID for the specific queue.

The Properties component, is an array of Property Types an the length of the property, defines the current queue mode. Current specified modes are none and minimum rate.

The QueueCounter component, a struct of TransmitPackets, TransmitBytes, TransimtOverrunErrors holds the necessary counter for the LFB.

5.5.3. Capabilities

This LFB has no capabilities specified.

5.5.4. Events

This LFB has no events specified.

5.6. OFRedirectIn

The OFRedirectIn LFB abstracts the process for the controller to inject data packets into the switch to input data packets into the data path. The LFB is derived from the RedirectIn LFB defined in the Base LFB Library [I-D.ietf-forces-lfb-lib].

5.6.1. Data Handling

A packet or a bufferID arrives from the controller depending on whether the packet was buffered or not in the switch. If the packet was not buffered or the controller wishes to inject a packet into the switch, the packet will be output from the singleton output port PacketIn along with an ActionList and an IngressPort metadata to be sent to the OFFlowTables for processing. If the packet was buffered in the switch, the no packet is sent from the singleton output port BufferIn but only the BufferID, the ActionList and the IngressPort metadatas.

5.6.2. Components

This LFB has no components specified.

5.6.3. Capabilities

This LFB has no capabilities specified.

5.6.4. Events

This LFB has no events specified.

5.7. OFRedirectOut

The OFRedirectOut LFB abstracts the process for the switch to deliver data packets to the controller. The LFB is derived from the RedirectIn LFB defined in the Base LFB Library [I-D.ietf-forces-lfb-lib].

5.7.1. Data Handling

A packet or part of a packet along with the BufferID metadata arrives from the group input port Outgoing from either the OFFlowTables or the OFActionOutput LFBs. Besides the optional BufferID metadata, the IngressPort, the InPhyPort and the RedirectReason and the optional FlowTableID are sent to the Outgoing input port required to be sent to the Controller as metadata.

5.7.2. Components

This LFB has no components specified.

5.7.3. Capabilities

This LFB has no capabilities specified.

5.7.4. Events

This LFB has no events specified.

5.8. OFAction

This LFB is a template used for create OFActionLFBs. All OFActionLFBs have the input and output port in common but have different components. This LFB defines how input and output port of all OFActionLFBs. Inside OFActionLFBs there is a table with the required attributes where applicable as some OFActionLFBs don’t require attributes.

5.8.1. Data Handling

A packet arrives in an OFActionLFB via the group input PacketIn from an OFFlowTables or an OFGroupTable, along with the LFBClassID metadata, required to uniquely identify the sender, the PacketID and optionally the ActionIndex metadata if the action requires a specific attributes, the IngressPort, the InPhyPort and the QueueID required by the OFActionOutput. Once the packet has been processed it is return to the sender LFB via the group output PacketOut along with the PacketID.

5.8.2. Components

This LFB has no components specified.

5.8.3. Capabilities

This LFB has no capabilities specified.

5.8.4. Events

This LFB has no events specified.

5.9. OFActionLFBs

As none of the OFActionLFBs have any capabilities or events, these sections are ommited from the draft.

5.9.1. OFActionOutput

5.9.1.1. Data Handling

The OFActionOutputLFB does not modify the packet in any way, rather forwards a packet to a specified OFPort. Additionally there are several virtual ports that the OFActionOutputLFB may send the packet to:

All - Group output, sends the packet out all standard ports, but not to the ingress port or ports configured not to forward

Controller - Sends the packet to the controller

Table - Submit the packet to the first flow table so that the packet can be processed through the regular OpenFlow pipeline. Only valid in the action list of a packet-out message

InPort - Sends the packet out the ingress port.

Local - Sends the packet to the switch’s local networking stack

Normal - Processes the packet using the traditional non-OpenFlow pipeline of the switch.

Flood - Floods the packet using the normal pipeline of the switch.

5.9.1.2. Components

This LFB has only one component, the OutputActionTable, which is an array of a struct of the port number and optionally the maximum length in bytes, if the receiving end is the controller.

5.9.2. OFActionSetVLANVID

5.9.2.1. Data Handling

The OFActionSetVLANVIDLFB replaces the existing VLAN ID. Only applies to packets with an existing VLAN tag.

5.9.2.2. Components

This LFB has only one component, the SetVLANVIDActionTable, which is an array of uint16 VLAN tag values.

5.9.3. OFActionSetVLANPriority

5.9.3.1. Data Handling

The OFActionSetVLANPriorityLFB replaces the existing VLAN priority. Only applies to packets with an existing VLAN tag.

5.9.3.2. Components

This LFB has only one component, the SetVLANPriorityActionTable, which is an array of uchar VLAN priority values.

5.9.4. OFActionSetMACSource

5.9.4.1. Data Handling

The OFActionSetMACSourceLFB replaces the existing Ethernet source MAC address.

5.9.4.2. Components

This LFB has only one component, the SetMACSourceActionTable, which is an array of IEEEMAC addresses.

5.9.5. OFActionSetMACDestination

5.9.5.1. Data Handling

The OFActionSetMACDestinationLFB replaces the existing Ethernet source MAC address.

5.9.5.2. Components

This LFB has only one component, the SetMACSourceActionTable, which is an array of IEEEMAC addresses.

5.9.6. OFActionSetIPSource

5.9.6.1. Data Handling

The OFActionSetIPSourceLFB replaces the existing IP source address with new value and update the IP checksum (and TCP/UDP/SCTP checksum if applicable). This action is only applicable to IPv4 packets.

5.9.6.2. Components

This LFB has only one component, the SetIPSourceActionTable, which is an array of IPv4 addresses.

5.9.7. OFActionSetIPDestination

5.9.7.1. Data Handling

The OFActionSetIPDestinationLFB replaces the existing IP destination address with new value and update the IP checksum (and TCP/UDP/SCTP checksum if applicable). This action is only applicable to IPv4 packets.

5.9.7.2. Components

This LFB has only one component, the SetIPDestinationActionTable, which is an array of IPv4 addresses.

5.9.8. OFActionSetIPTOS

5.9.8.1. Data Handling

The OFActionSetIPTOSLFB replaces the existing IP TOS value and update the IP checksum. Only applies to IPv4 packets.

5.9.8.2. Components

This LFB has only one component, the SetIPTOSActionTable, which is an array of IPv4 uchar TOS values.

5.9.9. OFActionSetIPECN

5.9.9.1. Data Handling

The OFActionSetIPECNLFB replaces the existing IP ECN value and update the IP checksum. Only applies to IPv4 packets.

5.9.9.2. Components

This LFB has only one component, the SetIPECNActionTable, which is an array of IPv4 uchar ECN values.

5.9.10. OFActionSetTCPSource

5.9.10.1. Data Handling

The OFActionSetTCPSourceLFB replaces the existing TCP/UDP/SCTP source port with new value and update the TCP/UDP/SCTP checksum. This action is only applicable to TCP, UDP and SCTP packets.

5.9.10.2. Components

This LFB has only one component, the SetTCPSourceActionTable, which is an array of uint16 values.

5.9.11. OFActionSetTCPDestination

5.9.11.1. Data Handling

The OFActionSetTCPDestinationLFB replaces the existing TCP/UDP/SCTP destination port with new value and update the TCP/UDP/SCTP checksum. This action is only applicable to TCP, UDP and SCTP packets.

5.9.11.2. Components

This LFB has only one component, the SetTCPDestinationActionTable, which is an array of uint16 values.

5.9.12. OFActionCopyTTLOut

5.9.12.1. Data Handling

The OFActionCopyTTLOutLFB copies the TTL from next-to-outermost to outermost header with TTL. Copy can be IP-to-IP, MPLS-to-MPLS, or IP-to-MPLS.

5.9.12.2. Components

This LFB has no components specified.

5.9.13. OFActionCopyTTLIn

5.9.13.1. Data Handling

The OFActionCopyTTLOutLFB copies the TTL from outermost to next-to-outermost header with TTL. Copy can be IP-to-IP, MPLS-to-MPLS, or IP-to-MPLS.

5.9.13.2. Components

This LFB has no components specified.

5.9.14. OFActionSetMPLSLabel

5.9.14.1. Data Handling

The OFActionSetMPLSLabelLFB replaces the existing MPLS label. Only applies to packets with an existing MPLS shim header.

5.9.14.2. Components

This LFB has only one component, the SetMPLSLabelActionTable, which is an array of uint32 MPLS label values.

5.9.15. OFActionSetMPLSTC

5.9.15.1. Data Handling

The OFActionSetMPLSTCLFB replaces the existing MPLS traffic class. Only applies to packets with an existing MPLS shim header.

5.9.15.2. Components

This LFB has only one component, the SetMPLSTCActionTable, which is an array of uchar MPLS label values.

5.9.16. OFActionSetMPLSTTL

5.9.16.1. Data Handling

The OFActionSetMPLSTTLLFB replaces the existing MPLS TTL. Only applies to packets with an existing MPLS shim header.

5.9.16.2. Components

This LFB has only one component, the SetMPLSTTLTable, which is an array of uchar MPLS TTL values.

5.9.17. OFActionDecrementMPLSTTL

5.9.17.1. Data Handling

The OFActionDecrementMPLSTTLLFB decrements the MPLS TTL. Only applies to packets with an existing MPLS shim header.

5.9.17.2. Components

This LFB has no components specified.

5.9.18. OFActionPushVLan

5.9.18.1. Data Handling

The OFActionPushVLanLFB pushes a new VLAN header onto the packet. The Ethertype is used as the Ethertype for the tag. Only Ethertype 0x8100 and 0x88a8 should be used.

5.9.18.2. Components

This LFB has only one component, the PushVLANTable, which is an array of uint16 ethertypes.

5.9.19. OFActionPopVLAN

5.9.19.1. Data Handling

The OFActionPopVLANLFB pops the outer-most VLAN header from the packet.

5.9.19.2. Components

This LFB has no components specified.

5.9.20. OFPushMPLSOFAction

5.9.20.1. Data Handling

The OFPushMPLSOFActionLFB pushes a new MPLS shim header onto the packet. The Ethertype is used as the Ethertype for the tag. Only Ethertype 0x8847 and 0x8848 should be used.

5.9.20.2. Components

This LFB has only one component, the PushMPLSTable, which is an array of uint16 MPLS header values.

5.9.21. OFPopMPLSOFAction

5.9.21.1. Data Handling

The OFPopMPLSOFActionLFB pops the outer-most MPLS tag or shim header from the packet. The Ethertype is used as the Ethertype for the resulting packet (Ethertype for the MPLS payload).

5.9.21.2. Components

This LFB has only one component, the PopMPLSTable, which is an array of uint16 ethertype values.

5.9.22. OFSetQueueOFAction

5.9.22.1. Data Handling

The OFSetQueueOFActionLFB sets the queue ID for the packet. This LFB will return the packet along with the QueueID as a metadata.

5.9.22.2. Components

This LFB has only one component, the SetQueueTable, which is an array of uint32 queue identifiers.

5.9.23. OFSetIPTTLOFAction

5.9.23.1. Data Handling

The OFSetIPTTLOFActionLFB replaces the existing IP TTL and update the IP checksum. Only applies to IPv4 packets.

5.9.23.2. Components

This LFB has only one component, the SetIPTTLTable, which is an array of uchar TTL values.

5.9.24. OFDecrementIPTTLOFAction

5.9.24.1. Data Handling

The OFDecrementIPTTLOFActionLFB decrements the existing IP TTL and update the IP checksum. Only applies to IPv4 packets.

5.9.24.2. Components

This LFB has no components specified.

5.9.25. OFExperimenterOFAction

5.9.25.1. Data Handling

The OFExperimenterOFActionLFB handles experimenter actions.

5.9.25.2. Components

This LFB has only one component, the SetIPTTLTable, which is an array of uint32 Experimenter ID values.

6. XML for OpenFlow library

```xml
 <?xml version="1.0" encoding="UTF-8"?>
 <LFBLibrary xmlns="urn:ietf:params:xml:ns:forces:lfbmodel:1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  provides="OpenFlowLibrary">
  <load library="BaseTypeLibrary"
  location="urn:ietf:params:xml:ns:forces:lfbmodel:1.0"/>
   <dataTypeDefs>
     <!-- Data Type Definition for the OpenFlow Switch -->
     <dataTypeDef>
       <name>SwitchDescriptionType</name>
       <synopsis>The type of the switch description</synopsis>
       <struct>
         <component componentID="1">
           <name>MFR</name>
           <synopsis>Manufacturer description</synopsis>
           <typeRef>string[256]</typeRef>
         </component>
         <component componentID="2">
           <name>HW</name>
           <synopsis>Hardware description</synopsis>
           <typeRef>string[256]</typeRef>
         </component>
         <component componentID="3">
           <name>SF</name>
           <synopsis>Software description</synopsis>
           <typeRef>string[256]</typeRef>
         </component>
         <component componentID="4">
           <name>SerialNum</name>
           <synopsis>Serial Number</synopsis>
           <typeRef>string[32]</typeRef>
         </component>
         <component componentID="5">
           <name>DP</name>
           <synopsis>Human readable description of datapath</synopsis>
           <typeRef>string[256]</typeRef>
         </component>
       </struct>
     </dataTypeDef>
     <!-- Data Type Definition for the Flow Match -->
     <dataTypeDef>
       <name>MPLSLabelValue</name>
       <synopsis>An MPLS label.</synopsis>
       <atomic>
         <baseType>uint32</baseType>
         <rangeRestriction>
           <allowedRange min="0" max="1048575"/>
         </rangeRestriction>
       </atomic>
     </dataTypeDef>
     <dataTypeDef>
       <name>MPLSTrafficClassValues</name>
       <synopsis>The MPLS Traffic Class</synopsis>
       <atomic>
         <baseType>uchar</baseType>
         <rangeRestriction>
           <allowedRange min="0" max="7"/>
         </rangeRestriction>
       </atomic>
     </dataTypeDef>
     <dataTypeDef>
       <name>IPv4ToSbits</name>
       <synopsis>TOSBits</synopsis>
       <atomic>
         <baseType>uchar</baseType>
         <rangeRestriction>
           <allowedRange min="0" max="63"/>
         </rangeRestriction>
       </atomic>
     </dataTypeDef>
     <dataTypeDef>
       <name>WildcardsType</name>
       <synopsis>Wildcards for fields</synopsis>
       <struct>
         <component componentID="1">
           <name>InPort</name>
           <synopsis>Input Port Wildcard</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="2">
           <name>VLANID</name>
           <synopsis>Vlan ID Wildcard</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="3">
           <name>VLANPCP</name>
           <synopsis>Vlan priority Wildcard</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="4">
           <name>DLType</name>
           <synopsis>Ethernet frame typ Wildcard</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="5">
           <name>IPToS</name>
           <synopsis>IP ToS Wildcard</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="6">
           <name>IPProtocol</name>
           <synopsis>IP Protocol Wildcard</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="7">
           <name>TPSource</name>
           <synopsis>TCP/UDP/SCTP source port Wildcard</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="8">
           <name>TPDestination</name>
           <synopsis>TCP/UDP/SCTP destination port Wildcard</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="9">
           <name>MPLSLabel</name>
           <synopsis>MPLS label Wildcard</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="10">
           <name>MPLSTC</name>
           <synopsis>MPLS TC Wildcard</synopsis>
           <typeRef>boolean</typeRef>
         </component>
       </struct>
     </dataTypeDef>
     <dataTypeDef>
       <name>MatchFieldType</name>
       <synopsis>A Match Field Type</synopsis>
       <struct>
         <component componentID="1">
           <name>IngressPort</name>
           <synopsis>Numerical representation of incoming port, starting
           at 1. This may be a physical or switch-defined virtual port.
           </synopsis>
           <typeRef>uint32</typeRef>
         </component>
         <component componentID="2">
           <name>Wildcards</name>
           <synopsis>Wildcards for match fields</synopsis>
           <typeRef>WildcardsType</typeRef>
         </component>
         <component componentID="3">
           <name>EthernetSourceAddress</name>
           <synopsis>Ethernet source address</synopsis>
           <typeRef>IEEEMAC</typeRef>
         </component>
         <component componentID="4">
           <name>EthernetSourceAddressMask</name>
           <synopsis>Ethernet source address mask</synopsis>
           <typeRef>IEEEMAC</typeRef>
         </component>
         <component componentID="5">
           <name>EthernetDestinationAddress</name>
           <synopsis>Ethernet destination address</synopsis>
           <typeRef>IEEEMAC</typeRef>
         </component>
         <component componentID="6">
           <name>EthernetDestinationAddressMask</name>
           <synopsis>Ethernet destination address mask</synopsis>
           <typeRef>IEEEMAC</typeRef>
         </component>
         <component componentID="7">
           <name>VlanID</name>
           <synopsis>VLAN identifier of outermost VLAN tag.</synopsis>
           <typeRef>VlanIDType</typeRef>
         </component>
         <component componentID="8">
           <name>VlanPriority</name>
           <synopsis>VLAN PCP Field of outermost VLAN tag.</synopsis>
           <typeRef>VlanPriorityType</typeRef>
         </component>
         <component componentID="9">
           <name>EtherType</name>
           <synopsis>Ethernet type of the OpenFlow packet payload, after
           VLAN tags. 802.3 frames have special handling.</synopsis>
           <typeRef>uint16</typeRef>
         </component>
         <component componentID="10">
           <name>IPv4TOS</name>
           <synopsis>Specify as 8-bit value and place ToS in upper 6
           bits for match</synopsis>
           <typeRef>IPv4ToSbits</typeRef>
         </component>
         <component componentID="11">
           <name>IPProto</name>
           <synopsis>IP protocol or lower 8 bits of
           ARP opcode. Only the lower 8 bits of the ARP opcode are used
           for the match</synopsis>
```

```
            <typeRef>uchar8</typeRef>
          </component>
          <component componentID="12">
            <name>IPv4SourceAddress</name>
            <synopsis>IPv4 Source Address to match</synopsis>
            <typeRef>IPv4Addr</typeRef>
          </component>
          <component componentID="13">
            <name>IPv4SourceAddressMask</name>
            <synopsis>IPv4 Source Address mask</synopsis>
            <typeRef>IPv4Addr</typeRef>
          </component>
          <component componentID="14">
            <name>IPv4DestinationAddress</name>
            <synopsis>IPv4 Destination Address to match</synopsis>
            <typeRef>IPv4Addr</typeRef>
          </component>
          <component componentID="15">
            <name>IPv4DestinationAddressMask</name>
            <synopsis>IPv4 Destination Address mask</synopsis>
            <typeRef>IPv4Addr</typeRef>
          </component>
          <component componentID="16">
            <name>TCPSourcePort</name>
            <synopsis>Source Port for TCP and ICMP to match</synopsis>
            <typeRef>uint16</typeRef>
          </component>
          <component componentID="17">
            <name>TCPDestinationPort</name>
            <synopsis>Destination Port for TCP and ICMP to match
            </synopsis>
            <typeRef>uint16</typeRef>
          </component>
          <component componentID="18">
            <name>MPLSlabel</name>
            <synopsis>Match on outermost MPLS tag.</synopsis>
            <typeRef>MPLSLabelValue</typeRef>
          </component>
          <component componentID="19">
            <name>MPLSTrafficClass</name>
            <synopsis>Match on outermost MPLS tag for traffic class.
            </synopsis>
            <typeRef>MPLSTrafficClassValues</typeRef>
          </component>
          <component componentID="20">
            <name>Metadata</name>
            <synopsis>MetaData</synopsis>
            <typeRef>uint64</typeRef>
```

```
         </component>
         <component componentID="21">
           <name>MetadataMask</name>
           <synopsis>MetaData Mask</synopsis>
           <typeRef>uint64</typeRef>
         </component>
       </struct>
     </dataTypeDef>
     <!-- Datatype Definition for Flow Table -->
     <dataTypeDef>
       <name>FlowEntry</name>
       <synopsis>A Flow entry</synopsis>
       <struct>
         <component componentID="1">
           <name>Cookie</name>
           <synopsis>Opaque data chosen by controller</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="2">
           <name>MatchFields</name>
           <synopsis>Match Fields: to match against packets. These
           consist of the ingress port and packet headers, and
           optionally metadata specified by a previous table</synopsis>
           <typeRef>MatchFieldType</typeRef>
         </component>
         <component componentID="3">
           <name>Counters</name>
           <synopsis>Counters: to update for matching packets</synopsis>
           <typeRef>FlowCounterType</typeRef>
         </component>
         <component componentID="4">
           <name>Instructions</name>
           <synopsis>Instruction: what to do with the packet of the flow
           </synopsis>
           <array>
             <struct>
               <component componentID="1">
                 <name>InstructionType</name>
                 <synopsis>The instruction type</synopsis>
                 <typeRef>InstructionTypes</typeRef>
               </component>
               <component componentID="2">
                 <name>InstructionIndex</name>
                 <synopsis>The instruction index.</synopsis>
                 <typeRef>uint32</typeRef>
               </component>
             </struct>
             <contentKey contentKeyID="1">
```

```
               <contentKeyField>InstructionType</contentKeyField>
             </contentKey>
           </array>
         </component>
         <component componentID="5">
           <name>Timeouts</name>
           <synopsis>Timeouts for the flow entry</synopsis>
           <struct>
             <component componentID="1">
               <name>IdleTimeout</name>
               <synopsis>Timeout to expire if no flows are matched for
               this flow entry</synopsis>
               <typeRef>uint16</typeRef>
             </component>
             <component componentID="2">
               <name>HardTimeout</name>
               <synopsis>Timeout to expire for this flow entry
               regardless of idle timeout</synopsis>
               <typeRef>uint16</typeRef>
             </component>
           </struct>
         </component>
         <component componentID="6">
           <name>Timers</name>
           <synopsis>Timers per flow</synopsis>
           <struct>
             <component componentID="1">
               <name>Duration_Sec</name>
               <synopsis>Time flow has been alive in seconds</synopsis>
               <typeRef>uint32</typeRef>
             </component>
             <component componentID="2">
               <name>Duration_nSec</name>
               <synopsis>Time flow has been alive in nanoseconds beyond
               Duration_Sec</synopsis>
               <typeRef>uint32</typeRef>
             </component>
           </struct>
         </component>
         <component componentID="7">
           <name>Priority</name>
           <synopsis>Priority within the specified flow table</synopsis>
           <typeRef>uint16</typeRef>
         </component>
       </struct>
     </dataTypeDef>
     <dataTypeDef>
       <name>ActionRowType</name>
       <synopsis>An Action Row for the action table</synopsis>
       <struct>
         <component componentID="1">
           <name>Action</name>
           <synopsis>The type of action</synopsis>
           <typeRef>ActionType</typeRef>
         </component>
         <component componentID="2">
           <name>ActionTableIndex</name>
           <synopsis>Index of the Table this action applies to
           </synopsis>
           <typeRef>uint32</typeRef>
         </component>
       </struct>
     </dataTypeDef>
     <dataTypeDef>
       <name>ActionType</name>
       <synopsis>The type of action</synopsis>
       <atomic>
         <baseType>uint16</baseType>
         <specialValues>
           <specialValue value="1">
             <name>OUTPUT</name>
             <synopsis>Output to switch port</synopsis>
           </specialValue>
           <specialValue value="2">
             <name>SetVLANVID</name>
             <synopsis>Set the 802.1q VLAN id</synopsis>
           </specialValue>
           <specialValue value="3">
             <name>SetVLANPCP</name>
             <synopsis>Set the 802.1q priority</synopsis>
           </specialValue>
           <specialValue value="4">
             <name>SetDLSrc</name>
             <synopsis>Set Ethernet source address</synopsis>
           </specialValue>
           <specialValue value="5">
             <name>SetDLDst</name>
             <synopsis>Set Ethernet destination address</synopsis>
           </specialValue>
           <specialValue value="6">
             <name>SetIPSrc</name>
             <synopsis>Set IP source address</synopsis>
           </specialValue>
           <specialValue value="7">
             <name>SetIPDst</name>
             <synopsis>Set IP Destination address</synopsis>
           </specialValue>
           <specialValue value="8">
             <name>SetIPTOS</name>
             <synopsis>Set ToS field</synopsis>
           </specialValue>
           <specialValue value="9">
             <name>SetIPECN</name>
             <synopsis>Set ECN field</synopsis>
           </specialValue>
           <specialValue value="10">
             <name>SetTPSource</name>
             <synopsis>TCP/UDP/SCTP source port</synopsis>
           </specialValue>
           <specialValue value="11">
             <name>SetTPDestination</name>
             <synopsis>TCP/UDP/SCTP destination port</synopsis>
           </specialValue>
           <specialValue value="12">
             <name>CopyTTLOut</name>
             <synopsis>Copy TTL "outwards" -- from next-to-outermost to
             outermost</synopsis>
           </specialValue>
           <specialValue value="13">
             <name>CopyTTLIn</name>
             <synopsis>Copy TTL "inwards" -- from outermost to
             next-to-outermost</synopsis>
           </specialValue>
           <specialValue value="14">
             <name>SetMPLSLabel</name>
             <synopsis>Set MPLS label</synopsis>
           </specialValue>
           <specialValue value="15">
             <name>SetMPLSTC</name>
             <synopsis>Set MPLS TC</synopsis>
           </specialValue>
           <specialValue value="16">
             <name>SetMPLSTTL</name>
             <synopsis>Set MPLS TTL</synopsis>
           </specialValue>
           <specialValue value="17">
             <name>PushVLANTag</name>
             <synopsis>Push a new VLAN tag</synopsis>
           </specialValue>
           <specialValue value="18">
             <name>PopVLANTag</name>
             <synopsis>Pop the outer VLAN tag</synopsis>
           </specialValue>
           <specialValue value="19">
```

```
             <name>PushMPLSTag</name>
             <synopsis>Push a new MPLS tag</synopsis>
           </specialValue>
           <specialValue value="20">
             <name>PopMPLSTag</name>
             <synopsis>Pop the outer MPLS tag</synopsis>
           </specialValue>
           <specialValue value="21">
             <name>SetQueue</name>
             <synopsis>Set queue ID when outputting to a port</synopsis>
           </specialValue>
           <specialValue value="22">
             <name>Group</name>
             <synopsis>Apply group</synopsis>
           </specialValue>
           <specialValue value="23">
             <name>SetIPTTL</name>
             <synopsis>Set IP TTL</synopsis>
           </specialValue>
           <specialValue value="24">
             <name>DecIPTTL</name>
             <synopsis>Decrement IP TTL</synopsis>
           </specialValue>
         </specialValues>
       </atomic>
     </dataTypeDef>
     <dataTypeDef>
       <name>TableCounterType</name>
       <synopsis>Counter per table</synopsis>
       <struct>
         <component componentID="1">
           <name>ReferenceCount</name>
           <synopsis>Active Entries</synopsis>
           <typeRef>uint32</typeRef>
         </component>
         <component componentID="2">
           <name>PacketLookups</name>
           <synopsis>Packet Lookups</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="3">
           <name>PacketMatches</name>
           <synopsis>Packet Matches</synopsis>
           <typeRef>uint64</typeRef>
         </component>
       </struct>
     </dataTypeDef>
     <dataTypeDef>
```

```
       <name>Actions</name>
       <synopsis>Actions to perform. An Array of ActionRowTypes
       </synopsis>
       <array>
         <typeRef>ActionRowType</typeRef>
       </array>
     </dataTypeDef>
     <dataTypeDef>
       <name>FlowCounterType</name>
       <synopsis>Counter per flow</synopsis>
       <struct>
         <component componentID="1">
           <name>ReceivedPackets</name>
           <synopsis>Packets Received</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="2">
           <name>ReceivedBytes</name>
           <synopsis>Bytes Received</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="3">
           <name>DurationS</name>
           <synopsis>Duration in seconds</synopsis>
           <typeRef>uint32</typeRef>
         </component>
         <component componentID="4">
           <name>DurationNS</name>
           <synopsis>Duration in nanoseconds</synopsis>
           <typeRef>uint32</typeRef>
         </component>
       </struct>
     </dataTypeDef>
     <dataTypeDef>
       <name>InstructionTypes</name>
       <synopsis>Instructions supported</synopsis>
       <atomic>
         <baseType>short</baseType>
         <specialValues>
           <specialValue value="1">
             <name>GotoTable</name>
             <synopsis>Indicates the next table in the processing
             pipeline. The table-id must be greater than the current
             table-id. The flows of last table of the pipeline can not
             include this instruction</synopsis>
           </specialValue>
           <specialValue value="2">
             <name>WriteMetadata</name>
             <synopsis>Writes the masked metadata value into the
             metadata field. The mask specifies which bits of the
             metadata register should be modified (i.e. new metadata =
             old metadata and ~mask | value and mask)</synopsis>
           </specialValue>
           <specialValue value="3">
             <name>WriteAction</name>
             <synopsis>Merges the specifieed action(s) into the current
              action set. If an action of the given type exists in the
              current set, overwrite it, otherwise add it.</synopsis>
           </specialValue>
           <specialValue value="4">
             <name>ApplyActions</name>
             <synopsis>Applies the specific action(s) immediately,
             without any change to the Action Set. This instruction may
             be used to modify the packet between two tables or to
             execute multiple actions of the same type. The actions are
             specified as an action list</synopsis>
           </specialValue>
           <specialValue value="5">
             <name>ClearActions</name>
             <synopsis>Clears all the actions in the action set
             immediately.</synopsis>
           </specialValue>
         </specialValues>
       </atomic>
     </dataTypeDef>
     <dataTypeDef>
       <name>WriteMetadataTableType</name>
       <synopsis>Metadata and mask for the write metadata instruction
       per row</synopsis>
       <struct>
         <component componentID="1">
           <name>Metadata</name>
           <synopsis>The metadata</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="2">
           <name>MetadataMask</name>
           <synopsis>The metadata mask</synopsis>
           <typeRef>uint64</typeRef>
         </component>
       </struct>
     </dataTypeDef>
     <dataTypeDef>
       <name>FlowTableMissConfigType</name>
       <synopsis>Types to configure the default behavior of unmatched
       packets in a Flow Table</synopsis>
       <atomic>
         <baseType>uint32</baseType>
         <specialValues>
           <specialValue value="0">
             <name>Controller</name>
             <synopsis>Send to the controller</synopsis>
           </specialValue>
           <specialValue value="1">
             <name>Continue</name>
             <synopsis>Continue to the next table in the pipeline or
             send to the controller if the FlowTable is the last.
             </synopsis>
           </specialValue>
           <specialValue value="2">
             <name>Drop</name>
             <synopsis>Drop the packet</synopsis>
           </specialValue>
         </specialValues>
       </atomic>
     </dataTypeDef>
     <dataTypeDef>
       <name>PacketInTypes</name>
       <synopsis>Packet-in Types</synopsis>
       <atomic>
         <baseType>uchar</baseType>
         <specialValues>
           <specialValue value="0">
             <name>NoMatch</name>
             <synopsis>No Matching flow</synopsis>
           </specialValue>
           <specialValue value="1">
             <name>Action</name>
             <synopsis>Explicit action to send to controller</synopsis>
           </specialValue>
         </specialValues>
       </atomic>
     </dataTypeDef>
     <!-- Data Type Definition for the group -->
     <dataTypeDef>
       <name>GroupCounterType</name>
       <synopsis>Counters per group</synopsis>
       <struct>
         <component componentID="1">
           <name>ReferenceCount</name>
           <synopsis>Flow Entries</synopsis>
           <typeRef>uint32</typeRef>
         </component>
         <component componentID="2">
```

```
            <name>PacketCount</name>
            <synopsis>Packet Count</synopsis>
            <typeRef>uint64</typeRef>
          </component>
          <component componentID="3">
            <name>ByteCount</name>
            <synopsis>Byte Count</synopsis>
            <typeRef>uint64</typeRef>
          </component>
        </struct>
      </dataTypeDef>
      <dataTypeDef>
        <name>GroupBucketExecuteType</name>
        <synopsis>To determine which Action Bucket(s) should be
         executed</synopsis>
        <atomic>
          <baseType>uchar</baseType>
          <specialValues>
            <specialValue value="1">
              <name>all</name>
              <synopsis>Execute all buckets in the group. This group is
              used for multicast or broadcast forwarding. The packet is
              effectively cloned for each bucket; one packet is processed
              for each bucket of the group. If a bucket directs a packet
              explicitly out the ingress port, this packet clone is
              dropped. If the controller writer wants to forward out the
              ingress port, the group should include an extra bucket
              which includes an output action to the OFPP_IN_PORT virtual
              port.</synopsis>
            </specialValue>
            <specialValue value="2">
              <name>select</name>
              <synopsis>Execute one bucket in the group. Packets are sent
              to a single bucket in the group, based on a switch-computed
              selection algorithm (e.g. hash on some user-configured
              tuple or simple round robin). All configuration and state
              for the selection algorithm is external to OpenFlow. When a
              port speciffied in a bucket in a select group goes down,
              the switch may restrict bucket selection to the remaining
              set (those with forwarding actions to live ports) instead
              of dropping packets destined to that port. This behavior
              may reduce the disruption of a downed link or switch.
              </synopsis>
            </specialValue>
            <specialValue value="3">
              <name>indirect</name>
              <synopsis>Execute the one defined bucket in this group.
              Allows multiple flows or groups to point to a common group
              identifier, supporting faster, more efficient convergence
              (e.g. next hops for IP forwarding). This group type is
              effectively identical to an all group with one bucket.
              </synopsis>
            </specialValue>
            <specialValue value="4">
              <name>fastfailover</name>
              <synopsis>Execute the first live bucket. Each action bucket
              is associated with a specific port and/or group that
              controls its liveness. Enables the switch to change
              forwarding without requiring a round trip to the
              controller. If no buckets are live, packets are dropped.
              This group type must implement a liveness
              mechanism.</synopsis>
            </specialValue>
          </specialValues>
        </atomic>
      </dataTypeDef>
      <dataTypeDef>
        <name>BucketCounterType</name>
        <synopsis>Counters per bucket</synopsis>
        <struct>
          <component componentID="1">
            <name>PacketCount</name>
            <synopsis>Packet Count</synopsis>
            <typeRef>uint64</typeRef>
          </component>
          <component componentID="2">
            <name>ByteCount</name>
            <synopsis>Byte Count</synopsis>
            <typeRef>uint64</typeRef>
          </component>
        </struct>
      </dataTypeDef>
      <dataTypeDef>
        <name>GroupTableEntry</name>
        <synopsis>A Row of the Group Table</synopsis>
        <struct>
          <component componentID="1">
            <name>GroupID</name>
            <synopsis>Group Identifier uniquely identifying the group
            </synopsis>
            <typeRef>uint32</typeRef>
          </component>
          <component componentID="2">
            <name>GroupType</name>
            <synopsis>The group type to determine which action bucket
            will be executed.</synopsis>
```

```
           <typeRef>GroupBucketExecuteType</typeRef>
         </component>
         <component componentID="3">
           <name>GroupCounters</name>
           <synopsis>Counters per group</synopsis>
           <typeRef>GroupCounterType</typeRef>
         </component>
         <component componentID="4">
           <name>ActionBuckets</name>
           <synopsis>An ordered list of action buckets. Each action
           bucket is a set of actions similar to a flow table</synopsis>
           <array>
             <typeRef>ActionBucket</typeRef>
           </array>
         </component>
       </struct>
     </dataTypeDef>
     <dataTypeDef>
       <name>ActionBucket</name>
       <synopsis>An Action Bucket</synopsis>
       <struct>
         <component componentID="1">
           <name>Weight</name>
           <synopsis>Relative weight of bucket. Only defined for select
           groups.</synopsis>
           <typeRef>uint16</typeRef>
         </component>
         <component componentID="2">
           <name>WatchPort</name>
           <synopsis>Port whose state affects whether this bucket is
           live. Required for fast failover group</synopsis>
           <typeRef>uint32</typeRef>
         </component>
         <component componentID="3">
           <name>WatchGroup</name>
           <synopsis>Group whose state affects whether this group is
           live. Only required for fast failover groups</synopsis>
           <typeRef>uint32</typeRef>
         </component>
         <component componentID="4">
           <name>Actions</name>
           <synopsis>Actions for this bucket</synopsis>
           <typeRef>Actions</typeRef>
         </component>
         <component componentID="5">
           <name>BucketCounter</name>
           <synopsis>A counter for this bucket</synopsis>
           <typeRef>BucketCounterType</typeRef>
         </component>
       </struct>
     </dataTypeDef>
     <!-- Data Type Definition for ports -->
     <dataTypeDef>
       <name>PortNumberType</name>
       <synopsis>Port Number values</synopsis>
       <atomic>
         <baseType>uint32</baseType>
         <specialValues>
           <specialValue value="0xfffffff8">
             <name>InPort</name>
             <synopsis>Sent the packet out the input port. This virtual
             port must be explicitly used in order to send back out of
             the input port</synopsis>
           </specialValue>
           <specialValue value="0xfffffff9">
             <name>Table</name>
             <synopsis>Submit the packet to the first flow table. NBL
             This destination port can only be used in packet-out
             messages</synopsis>
           </specialValue>
           <specialValue value="0xfffffffa">
             <name>Normal</name>
             <synopsis>Process with normal L2/L3 switching</synopsis>
           </specialValue>
           <specialValue value="0xfffffffb">
             <name>Flood</name>
             <synopsis>Send the packet to all physical ports in VLAN,
             except input port and those blocked or link down</synopsis>
           </specialValue>
           <specialValue value="0xfffffffc">
             <name>All</name>
             <synopsis>Send the packet to all physical ports, except
             input port.</synopsis>
           </specialValue>
           <specialValue value="0xfffffffd">
             <name>Controller</name>
             <synopsis>Send the packet to the controller.</synopsis>
           </specialValue>
           <specialValue value="0xfffffffe">
             <name>Local</name>
             <synopsis>Local openflow "port".</synopsis>
           </specialValue>
           <specialValue value="0xffffffff">
             <name>Any</name>
             <synopsis>Wildcard port used only for flow mod (delete) and
             flow stats requests. Selects all flows regardless of output
             port (including flows with no output port).</synopsis>
           </specialValue>
         </specialValues>
       </atomic>
     </dataTypeDef>
     <dataTypeDef>
       <name>PortConfigurationType</name>
       <synopsis>Types of configuration for the OpenFlow port</synopsis>
       <struct>
         <component componentID="1">
           <name>PortDown</name>
           <synopsis>Port is administatively down</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="2">
           <name>NoReceiving</name>
           <synopsis>Drop all packets received by this port</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="3">
           <name>NoForwarding</name>
           <synopsis>Drop packets forwarded to the port</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="4">
           <name>NoPacket_In</name>
           <synopsis>Do not send packet-in messages for port</synopsis>
           <typeRef>boolean</typeRef>
         </component>
       </struct>
     </dataTypeDef>
     <dataTypeDef>
       <name>PortStateType</name>
       <synopsis>Current State of the port</synopsis>
       <struct>
         <component componentID="1">
           <name>LinkDown</name>
           <synopsis>No physical link present</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="2">
           <name>PortBlocked</name>
           <synopsis>Port is blocked</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="3">
           <name>PortLive</name>
           <synopsis>Live for Fast Failover Group</synopsis>
           <typeRef>boolean</typeRef>
         </component>
       </struct>
     </dataTypeDef>
     <dataTypeDef>
       <name>PortFeaturesType</name>
       <synopsis>Port Features</synopsis>
       <struct>
         <component componentID="1">
           <name>SpeedTypes</name>
           <synopsis>Types of Speed supported</synopsis>
           <struct>
             <component componentID="1">
               <name>10MB_HD</name>
               <synopsis>10 Mb half-duplex rate support.</synopsis>
               <typeRef>boolean</typeRef>
             </component>
             <component componentID="2">
               <name>10MB_FD</name>
               <synopsis>10 Mb full-duplex rate support.</synopsis>
               <typeRef>boolean</typeRef>
             </component>
             <component componentID="3">
               <name>100MB_HD</name>
               <synopsis>100 Mb half-duplex rate support.</synopsis>
               <typeRef>boolean</typeRef>
             </component>
             <component componentID="4">
               <name>100MB_FD</name>
               <synopsis>100 Mb full-duplex rate support.</synopsis>
               <typeRef>boolean</typeRef>
             </component>
             <component componentID="5">
               <name>1GB_HD</name>
               <synopsis>1 Gb half-duplex rate support.</synopsis>
               <typeRef>boolean</typeRef>
             </component>
             <component componentID="6">
               <name>1GB_FD</name>
               <synopsis>1 Gb full-duplex rate support.</synopsis>
               <typeRef>boolean</typeRef>
             </component>
             <component componentID="7">
               <name>10GB_FD</name>
               <synopsis>10 Gb full-duplex rate support.</synopsis>
               <typeRef>boolean</typeRef>
             </component>
             <component componentID="8">
```

```
               <name>40GB_FD</name>
               <synopsis>40 Gb full-duplex rate support.</synopsis>
               <typeRef>boolean</typeRef>
             </component>
             <component componentID="9">
               <name>100GB_FD</name>
               <synopsis>100 Gb full-duplex rate support.</synopsis>
               <typeRef>boolean</typeRef>
             </component>
             <component componentID="10">
               <name>1TB_FD</name>
               <synopsis>1 Tb full-duplex rate support.</synopsis>
               <typeRef>boolean</typeRef>
             </component>
             <component componentID="11">
               <name>Other</name>
               <synopsis>Other rate, not listed.</synopsis>
               <typeRef>boolean</typeRef>
             </component>
           </struct>
         </component>
         <component componentID="2">
           <name>MediumConnected</name>
           <synopsis>Medium Connected to the port</synopsis>
           <struct>
             <component componentID="1">
               <name>Copper</name>
               <synopsis>Copper Medium</synopsis>
               <typeRef>boolean</typeRef>
             </component>
             <component componentID="2">
               <name>Fiber</name>
               <synopsis>Fiber Medium</synopsis>
               <typeRef>boolean</typeRef>
             </component>
           </struct>
         </component>
         <component componentID="3">
           <name>Auto</name>
           <synopsis>Auto-negotiation</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="4">
           <name>PauseTypes</name>
           <synopsis>Pause Types supported of the port</synopsis>
           <struct>
             <component componentID="1">
               <name>Pause</name>
               <synopsis>Pause</synopsis>
               <typeRef>boolean</typeRef>
             </component>
             <component componentID="2">
               <name>AsymmetricPause</name>
               <synopsis>Asymmetric pause</synopsis>
               <typeRef>boolean</typeRef>
             </component>
           </struct>
         </component>
       </struct>
     </dataTypeDef>
     <dataTypeDef>
       <name>PortCounterType</name>
       <synopsis>Counter per port</synopsis>
       <struct>
         <component componentID="1">
           <name>ReceivedPackets</name>
           <synopsis>Packets Received</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="2">
           <name>TransmittedPackets</name>
           <synopsis>Packets Transmitted</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="3">
           <name>ReceivedBytes</name>
           <synopsis>Bytes Received</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="4">
           <name>TransmittedBytes</name>
           <synopsis>Bytes Transmitted</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="5">
           <name>ReceivedDrops</name>
           <synopsis>Drops Received</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="6">
           <name>TransmitDrops</name>
           <synopsis>Transmit Drops</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="7">
           <name>RecieveErrors</name>
           <synopsis>Errors in reception</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="8">
           <name>TransmitErrors</name>
           <synopsis>Errors in transmittion</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="9">
           <name>ReceivedFrameAlignmentErrors</name>
           <synopsis>Frame Alignment Errors received</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="10">
           <name>ReceiveOverrunErrors</name>
           <synopsis>Received Overrun Errors</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="11">
           <name>ReceivedCRCErrors</name>
           <synopsis>CRC Errors in received packets</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="12">
           <name>Collisions</name>
           <synopsis>Collisions</synopsis>
           <typeRef>uint64</typeRef>
         </component>
       </struct>
     </dataTypeDef>
     <!-- Data Type definitions for Queues -->
     <dataTypeDef>
       <name>QueuePropertyType</name>
       <synopsis>Property type for a queue</synopsis>
       <atomic>
         <baseType>uint16</baseType>
         <specialValues>
           <specialValue value="0">
             <name>None</name>
             <synopsis>No property defined</synopsis>
           </specialValue>
           <specialValue value="1">
             <name>MinimumRate</name>
             <synopsis>Minimum datarate guaranteed</synopsis>
           </specialValue>
         </specialValues>
       </atomic>
     </dataTypeDef>
     <dataTypeDef>
       <name>QueueArrayPropertiesType</name>
       <synopsis>Type Definition for property</synopsis>
       <struct>
         <component componentID="1">
           <name>Property</name>
           <synopsis>One of Queue Priority Types</synopsis>
           <typeRef>QueuePropertyType</typeRef>
         </component>
         <component componentID="2">
           <name>Length</name>
           <synopsis>Length of property</synopsis>
           <typeRef>uint32</typeRef>
         </component>
       </struct>
     </dataTypeDef>
     <dataTypeDef>
       <name>QueueCounterType</name>
       <synopsis>Counters per queue</synopsis>
       <struct>
         <component componentID="1">
           <name>TransmitPackets</name>
           <synopsis>Packets Transmitted</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="2">
           <name>TransmitBytes</name>
           <synopsis>Bytes Transmitted</synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="3">
           <name>TransimtOverrunErrors</name>
           <synopsis>Overrun Errors</synopsis>
           <typeRef>uint64</typeRef>
         </component>
       </struct>
     </dataTypeDef>
   </dataTypeDefs>
   <metadataDefs>
     <metadataDef>
       <name>IngressPort</name>
       <synopsis>The Ingress port the packet has arrived from</synopsis>
       <metadataID>1024</metadataID>
       <typeRef>uint32</typeRef>
     </metadataDef>
     <metadataDef>
       <name>InPhyPort</name>
       <synopsis>The Port Index of the Physical interface the frame
```

```
          entered the switch</synopsis>
          <metadataID>1025</metadataID>
          <typeRef>uint32</typeRef>
        </metadataDef>
        <metadataDef>
          <name>PacketID</name>
          <synopsis>The PacketID metadata is used to uniquelly identify a
          packet within the Flow Table or the Group Table to continue
          processing it after it has been returned from an OFActionLFB.
          </synopsis>
          <metadataID>1026</metadataID>
          <typeRef>uint32</typeRef>
        </metadataDef>
        <metadataDef>
          <name>ActionIndex</name>
          <synopsis>The Action Index metadata is used to point the row in
          the array in an Action LFB </synopsis>
          <metadataID>1027</metadataID>
          <typeRef>uint32</typeRef>
        </metadataDef>
        <metadataDef>
          <name>GroupIndex</name>
          <synopsis>The Group index metadata is used to point to the row of
          the array in an Group LFB</synopsis>
          <metadataID>1028</metadataID>
          <typeRef>uint32</typeRef>
        </metadataDef>
        <metadataDef>
          <name>LFBClassIDMetadata</name>
          <synopsis>The LFBClassID</synopsis>
          <metadataID>1029</metadataID>
          <typeRef>uint32</typeRef>
        </metadataDef>
        <metadataDef>
          <name>QueueIDMetadata</name>
          <synopsis>The Queue ID</synopsis>
          <metadataID>1030</metadataID>
          <typeRef>uint32</typeRef>
        </metadataDef>
        <metadataDef>
          <name>BufferID</name>
          <synopsis>The Buffer ID of a stored packet in the switch requried
          for the PacketOut message</synopsis>
          <metadataID>1031</metadataID>
          <typeRef>uint32</typeRef>
        </metadataDef>
        <metadataDef>
          <name>RedirectReason</name>
```

```
       <synopsis>The reason the packet was redirected to the controller
       </synopsis>
       <metadataID>1032</metadataID>
       <atomic>
         <baseType>uchar</baseType>
         <specialValues>
           <specialValue value="0">
             <name>NoMatch</name>
             <synopsis>No match on the Flow Table (table miss)
             </synopsis>
           </specialValue>
           <specialValue value="1">
             <name>Action</name>
             <synopsis>Specific Output to controller action.</synopsis>
           </specialValue>
         </specialValues>
       </atomic>
     </metadataDef>
     <metadataDef>
       <name>FlowTableID</name>
       <synopsis>The FlowTable ID the packet was sent to the controller
       from</synopsis>
       <metadataID>1033</metadataID>
       <typeRef>uchar</typeRef>
     </metadataDef>
     <metadataDef>
       <name>ActionListMetadata</name>
       <synopsis>The Action List that may come along with the packet in
       a PacketOut message</synopsis>
       <metadataID>1034</metadataID>
       <typeRef>octetstring</typeRef>
     </metadataDef>
   </metadataDefs>
   <LFBClassDefs>
     <!-- OpenFlow Switch LFB -->
     <LFBClassDef LFBClassID="1024">
       <name>OFSwitch</name>
       <synopsis>Similar to the FEProtocol and FEObject LFB, the
       OpenFlowSwitch LFB contains information required for the OpenFlow
       protocol.</synopsis>
       <version>1.1</version>
       <components>
         <component componentID="1" access="read-only">
           <name>DatapathID</name>
           <synopsis>Datapath unique ID. The lower 48-bits are for a MAC
            address, while the upper 16-bits are implementer-defined.
            </synopsis>
           <typeRef>uint64</typeRef>
         </component>
         <component componentID="4" access="read-write">
           <name>MissSendLen</name>
           <synopsis>Max bytes of new flow that datapath should send to
           the controller.</synopsis>
           <typeRef>uint16</typeRef>
         </component>
         <component componentID="5" access="read-write">
           <name>HandleFragments</name>
           <synopsis>if true drop fragments. If false no special
           handling.</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="6" access="read-write">
           <name>ReassembleFragments</name>
           <synopsis>If true, reassemble fragments</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="7" access="read-write">
           <name>InvalidTTLtoController</name>
           <synopsis>Send packets with invalid TTL ie. 0 or 1 to
           controller</synopsis>
           <typeRef>boolean</typeRef>
         </component>
         <component componentID="8" access="read-only">
           <name>SwitchDescription</name>
           <synopsis>Information about the switch</synopsis>
           <typeRef>SwitchDescriptionType</typeRef>
         </component>
         <component componentID="9" access="read-write">
           <name>Ports</name>
           <synopsis>The Ports that this switch has. It is an array of
           the Port Numbers</synopsis>
           <array>
             <typeRef>uint32</typeRef>
           </array>
         </component>
       </components>
       <capabilities>
         <capability componentID="31">
           <name>FlowStatistics</name>
           <synopsis>Whether the switch keep flow statistics</synopsis>
           <typeRef>boolean</typeRef>
         </capability>
         <capability componentID="32">
           <name>TableStatistics</name>
           <synopsis>Whether the switch keep table statistics</synopsis>
           <typeRef>boolean</typeRef>
         </capability>
         <capability componentID="33">
           <name>PortStatistics</name>
           <synopsis>Whether the switch keep port statistics</synopsis>
           <typeRef>boolean</typeRef>
         </capability>
         <capability componentID="34">
           <name>GroupStatistics</name>
           <synopsis>Whether the switch keep group statistics</synopsis>
           <typeRef>boolean</typeRef>
         </capability>
         <capability componentID="35">
           <name>IPReassembly</name>
           <synopsis>Whether the switch can reassemble IP fragments
           </synopsis>
           <typeRef>boolean</typeRef>
         </capability>
         <capability componentID="36">
           <name>QueueStats</name>
           <synopsis>Whether the switch keeps queue statistics
           </synopsis>
           <typeRef>boolean</typeRef>
         </capability>
         <capability componentID="37">
           <name>ARPMatchIP</name>
           <synopsis>Whether the switch matches IP addresses in APR
           packets</synopsis>
           <typeRef>boolean</typeRef>
         </capability>
         <capability componentID="38">
           <name>ActionsSupported</name>
           <synopsis>What actions are supported</synopsis>
           <array>
             <atomic>
               <baseType>ActionType</baseType>
               <rangeRestriction>
                 <allowedRange max="65534" min="0"/>
               </rangeRestriction>
             </atomic>
             <contentKey contentKeyID="1">
               <contentKeyField>ActionType</contentKeyField>
             </contentKey>
           </array>
         </capability>
         <capability componentID="39">
           <name>MaxBufferedPackets</name>
           <synopsis>Maximum packets buffered at once.</synopsis>
           <typeRef>uint32</typeRef>
         </capability>
         <capability componentID="40">
           <name>TablesSupported</name>
           <synopsis>Number of tables supported by the datapath
           </synopsis>
           <typeRef>uchar</typeRef>
         </capability>
       </capabilities>
       <events baseID="61">
         <event eventID="1">
           <name>PortAdded</name>
           <synopsis>This event is sent when a port is added</synopsis>
           <eventTarget>
             <eventField>Ports</eventField>
           </eventTarget>
           <eventCreated/>
         </event>
         <event eventID="2">
           <name>PortDeleted</name>
           <synopsis>This event is sent when a port is deleted
           </synopsis>
           <eventTarget>
             <eventField>Ports</eventField>
           </eventTarget>
           <eventDeleted/>
         </event>
         <event eventID="3">
           <name>PortModified</name>
           <synopsis>This event is sent when a port is modified
           </synopsis>
           <eventTarget>
             <eventField>Ports</eventField>
           </eventTarget>
           <eventChanged/>
         </event>
       </events>
     </LFBClassDef>
     <!--FlowTable LFB -->
     <LFBClassDef LFBClassID="1025">
       <name>OFFlowTables</name>
       <synopsis>An LFB that houses all OpenFlow Flow Tables residing in
        the switch</synopsis>
       <version>1.1</version>
       <inputPorts>
         <inputPort group="true">
           <name>InputPort</name>
           <synopsis>An Input port that expects packets from an OFPort
           LFB</synopsis>
           <expectation>
             <frameExpected>
               <ref>Arbitrary</ref>
             </frameExpected>
             <metadataExpected>
               <ref>IngressPort</ref>
               <ref>InPhyPort</ref>
             </metadataExpected>
           </expectation>
         </inputPort>
         <inputPort group="true">
           <name>PacketReturn</name>
           <synopsis>A port that expects the packet to be returned from
           an OFActionLFB. If the OFActionLFB is the OFQueueLFB then the
           QueueID metadata is expected as well.</synopsis>
           <expectation>
             <frameExpected>
               <ref>Arbitrary</ref>
             </frameExpected>
             <metadataExpected>
               <ref>PacketID</ref>
               <ref dependency="optional">QueueID</ref>
             </metadataExpected>
           </expectation>
         </inputPort>
         <inputPort group="false">
           <name>RedirectPacketIn</name>
           <synopsis>A port that expects a packet from the controller
           </synopsis>
           <expectation>
             <frameExpected>
               <ref>Arbitrary</ref>
             </frameExpected>
             <metadataExpected>
               <ref>ActionList</ref>
               <ref>IngressPort</ref>
             </metadataExpected>
           </expectation>
         </inputPort>
         <inputPort group="false">
           <name>RedirectBufferIn</name>
           <synopsis>A port that expects a Buffer ID index from the
           controller</synopsis>
           <expectation>
             <metadataExpected>
               <ref>BufferID</ref>
               <ref>ActionList</ref>
               <ref>IngressPort</ref>
             </metadataExpected>
           </expectation>
         </inputPort>
       </inputPorts>
       <outputPorts>
         <outputPort group="true">
           <name>OutputPort</name>
           <synopsis>A port that produces packets leaving the flow table
           to go to the OFOutputAction (to be sent to an output port)
           </synopsis>
           <product>
             <frameProduced>
               <ref>Arbitrary</ref>
             </frameProduced>
             <metadataProduced>
               <one-of>
                 <metadataSet>
                   <ref>IngressPort</ref>
                   <ref>InPhyPort</ref>
                 </metadataSet>
                 <metadataSet>
                   <ref>IngressPort</ref>
                   <ref>InPhyPort</ref>
                   <ref>QueueID</ref>
                 </metadataSet>
               </one-of>
             </metadataProduced>
           </product>
         </outputPort>
         <outputPort group="true">
           <name>GroupPort</name>
           <synopsis>A port that produces packets leaving the flow table
           to go to the OFGroupTable.</synopsis>
           <product>
             <frameProduced>
               <ref>Arbitrary</ref>
             </frameProduced>
             <metadataProduced>
               <one-of>
                 <metadataSet>
                   <ref>IngressPort</ref>
                   <ref>InPhyPort</ref>
                   <ref>GroupIndex</ref>
                 </metadataSet>
                 <metadataSet>
                   <ref>IngressPort</ref>
                   <ref>InPhyPort</ref>
                   <ref>QueueID</ref>
```

```
                  <ref>GroupIndex</ref>
                </metadataSet>
              </one-of>
            </metadataProduced>
          </product>
        </outputPort>
        <outputPort group="true">
          <name>ActionPort</name>
          <synopsis>A port that sends the packet to an OFActionLFB
          </synopsis>
          <product>
            <frameProduced>
              <ref>Arbitrary</ref>
            </frameProduced>
            <metadataProduced>
              <one-of>
                <metadataSet>
                  <ref>LFBClassIDMetadata</ref>
                  <ref>PacketID</ref>
                </metadataSet>
                <metadataSet>
                  <ref>LFBClassIDMetadata</ref>
                  <ref>PacketID</ref>
                  <ref>ActionIndex</ref>
                </metadataSet>
              </one-of>
            </metadataProduced>
          </product>
        </outputPort>
        <outputPort group="false">
          <name>RedirectPacketOut</name>
          <synopsis>A port that sends a packet or part of it to the
          controller</synopsis>
          <product>
            <frameProduced>
              <ref>Arbitrary</ref>
            </frameProduced>
            <metadataProduced>
              <one-of>
                <metadataSet>
                  <ref>IngressPort</ref>
                  <ref>InPhyPort</ref>
                  <ref>RedirectReason</ref>
                  <ref>FlowTableID</ref>
                </metadataSet>
                <metadataSet>
                  <ref>IngressPort</ref>
                  <ref>InPhyPort</ref>
```

```
                   <ref>RedirectReason</ref>
                   <ref>FlowTableID</ref>
                   <ref>BufferID</ref>
                 </metadataSet>
               </one-of>
             </metadataProduced>
           </product>
         </outputPort>
       </outputPorts>
       <components>
         <component componentID="1" access="read-write">
           <name>FlowTables</name>
           <synopsis>Flow entries inside the FlowTable LFB</synopsis>
           <array>
             <struct>
               <component componentID="1">
                 <name>FlowEntries</name>
                 <synopsis>An array of Flow Entries</synopsis>
                 <array>
                   <typeRef>FlowEntry</typeRef>
                 </array>
               </component>
               <component componentID="2">
                 <name>FlowTableCounter</name>
                 <synopsis>A counter for each Flow Table</synopsis>
                 <typeRef>TableCounterType</typeRef>
               </component>
               <component componentID="3">
                 <name>MissBehaviour</name>
                 <synopsis>What should the FlowTable do if a miss occurs
                 </synopsis>
                 <typeRef>FlowTableMissConfigType</typeRef>
               </component>
             </struct>
           </array>
         </component>
         <component componentID="2" access="read-write">
           <name>ApplyActionList</name>
           <synopsis>Table of actions for each flow</synopsis>
           <array>
             <typeRef>Actions</typeRef>
           </array>
         </component>
         <component componentID="3" access="read-write">
           <name>WriteActions</name>
           <synopsis>Table of Actions to write to the ActionSet
           </synopsis>
           <array>
```

```
             <typeRef>Actions</typeRef>
           </array>
         </component>
         <component componentID="4" access="read-write">
           <name>WriteMetadataTable</name>
           <synopsis>The write MetaDataTable</synopsis>
           <array>
             <typeRef>WriteMetadataTableType</typeRef>
           </array>
         </component>
         <component componentID="5" access="read-write">
           <name>GotoFlowTable</name>
           <synopsis>Containing the FlowTable IDs this flow should go
           to.</synopsis>
           <array>
             <typeRef>uint32</typeRef>
           </array>
         </component>
         <component componentID="6" access="read-write">
           <name>GroupTable</name>
           <synopsis>Table of group indeces to point a packet to
           </synopsis>
           <array>
             <typeRef>uint32</typeRef>
           </array>
         </component>
       </components>
       <events baseID="61">
         <event eventID="1">
           <name>FlowRemoved</name>
           <synopsis>If a CE subscribes to this event, it will send an
           event when a flow is removed.</synopsis>
           <eventTarget>
             <eventField>FlowEntries</eventField>
             <eventSubscript>FlowEntry</eventSubscript>
           </eventTarget>
           <eventDeleted/>
           <eventReports>
             <eventReport>
               <eventField>FlowTableID</eventField>
             </eventReport>
             <eventReport>
               <eventField>FlowEntries</eventField>
               <eventSubscript>FlowEntry</eventSubscript>
               <eventField>Cookie</eventField>
             </eventReport>
             <eventReport>
               <eventField>FlowEntries</eventField>
               <eventSubscript>FlowEntry</eventSubscript>
               <eventField>MatchFields</eventField>
             </eventReport>
             <eventReport>
               <eventField>FlowEntries</eventField>
               <eventSubscript>FlowEntry</eventSubscript>
               <eventField>Timeouts</eventField>
               <eventSubscript>IdleTimeout</eventSubscript>
             </eventReport>
             <eventReport>
               <eventField>FlowEntries</eventField>
               <eventSubscript>FlowEntry</eventSubscript>
               <eventField>Priority</eventField>
             </eventReport>
           </eventReports>
         </event>
       </events>
     </LFBClassDef>
     <!-- GroupTable LFB -->
     <LFBClassDef LFBClassID="1026">
       <name>OFGroupTable</name>
       <synopsis>The OpenFlow Group Tables</synopsis>
       <version>1.1</version>
       <inputPorts>
         <inputPort group="true">
           <name>PacketIn</name>
           <synopsis>A port to expect packets, the GroupIndex metadata,
           the IngressPort and InPhyPort and optionally the QueueID if
           the packet has already been through the OFActionSetQueue LFB.
           </synopsis>
           <expectation>
             <frameExpected>
               <ref>Arbitrary</ref>
             </frameExpected>
             <metadataExpected>
               <ref>GroupIndex</ref>
               <ref>IngressPort</ref>
               <ref>InPhyPort</ref>
               <ref dependency="optional">QueueID</ref>
             </metadataExpected>
           </expectation>
         </inputPort>
         <inputPort group="true">
           <name>PacketReturn</name>
           <synopsis>A port that expects the packet to be returned from
           an OFActionLFB. If the OFActionLFB is the OFQueueLFB then the
           QueueID metadata is expected as well.</synopsis>
           <expectation>
```

```
             <frameExpected>
               <ref>Arbitrary</ref>
             </frameExpected>
             <metadataExpected>
               <ref>PacketID</ref>
               <ref dependency="optional">QueueID</ref>
             </metadataExpected>
           </expectation>
         </inputPort>
       </inputPorts>
       <outputPorts>
         <outputPort group="true">
           <name>PacketOut</name>
           <synopsis>The port to return the packet to caller</synopsis>
           <product>
             <frameProduced>
               <ref>Arbitrary</ref>
             </frameProduced>
           </product>
         </outputPort>
         <outputPort group="true">
           <name>ActionPort</name>
           <synopsis>A port that sends the packet to an OFActionLFB
           </synopsis>
           <product>
             <frameProduced>
               <ref>Arbitrary</ref>
             </frameProduced>
             <metadataProduced>
               <one-of>
                 <metadataSet>
                   <ref>LFBClassIDMetadata</ref>
                   <ref>LFBInstanceIDMetadata</ref>
                 </metadataSet>
                 <metadataSet>
                   <ref>LFBClassIDMetadata</ref>
                   <ref>LFBInstanceIDMetadata</ref>
                   <ref>ActionIndex</ref>
                 </metadataSet>
               </one-of>
             </metadataProduced>
           </product>
         </outputPort>
       </outputPorts>
       <components>
         <component componentID="1">
           <name>GroupTable</name>
           <synopsis>The group table</synopsis>
```

```
          <array>
            <typeRef>GroupTableEntry</typeRef>
          </array>
        </component>
      </components>
    </LFBClassDef>
    <!-- Port LFB -->
    <LFBClassDef LFBClassID="1027">
      <name>OFPort</name>
      <synopsis>Input or Output port of an OpenFlow switch</synopsis>
      <version>1.1</version>
      <inputPorts>
        <inputPort group="false">
          <name>PacketOut</name>
          <synopsis>The input port of the Port LFB from the
          OFActionOutput LFB to output packets.</synopsis>
          <expectation>
            <frameExpected>
              <ref>Arbitrary</ref>
            </frameExpected>
            <metadataExpected>
              <ref dependency="optional">QueueID</ref>
            </metadataExpected>
          </expectation>
        </inputPort>
      </inputPorts>
      <outputPorts>
        <outputPort group="false">
          <name>PacketIn</name>
          <synopsis>Sends a packet to the OFFlowTables that is received
          by the OFPort LFB.</synopsis>
          <product>
            <frameProduced>
              <ref>Arbitrary</ref>
            </frameProduced>
            <metadataProduced>
              <ref>IngressPort</ref>
              <ref>InPhyPort</ref>
            </metadataProduced>
          </product>
        </outputPort>
        <outputPort group="true">
          <name>QueueOut</name>
          <synopsis>Sends a packet to the OFQueue LFB to be processed
          and output from the switch</synopsis>
          <product>
            <frameProduced>
              <ref>Arbitrary</ref>
```

```
             </frameProduced>
           </product>
         </outputPort>
       </outputPorts>
       <components>
         <component componentID="1" access="read-only">
           <name>PortNumber</name>
           <synopsis>The port number uniquely identifies a port within a
           switch.</synopsis>
           <typeRef>PortNumberType</typeRef>
         </component>
         <component componentID="2" access="read-only">
           <name>IEEEMAC</name>
           <synopsis>MAC Address of the port</synopsis>
           <typeRef>IEEEMAC</typeRef>
         </component>
         <component componentID="3" access="read-only">
           <name>Name</name>
           <synopsis>Human readable name of the port</synopsis>
           <typeRef>string[16]</typeRef>
         </component>
         <component componentID="4" access="read-write">
           <name>Configuration</name>
           <synopsis>Configuration of the port</synopsis>
           <typeRef>PortConfigurationType</typeRef>
         </component>
         <component componentID="5" access="read-only">
           <name>State</name>
           <synopsis>State of the OpenFlow Switch</synopsis>
           <typeRef>PortState</typeRef>
         </component>
         <component componentID="6" access="read-only">
           <name>CurrentFeatures</name>
           <synopsis>Current features of the port</synopsis>
           <typeRef>PortFeaturesType</typeRef>
         </component>
         <component componentID="7" access="read-write">
           <name>Advertised</name>
           <synopsis>Features advertised by the port</synopsis>
           <typeRef>PortFeaturesType</typeRef>
         </component>
         <component componentID="8" access="read-only">
           <name>CurrentSpeed</name>
           <synopsis>Current port bitrate in kbps</synopsis>
           <typeRef>uint32</typeRef>
         </component>
         <component componentID="9" access="read-only">
           <name>MaximumSpeed</name>
```

```
            <synopsis>Maximum port bitrate in kbps</synopsis>
            <typeRef>uint32</typeRef>
          </component>
          <component componentID="10" access="read-only">
            <name>PortCounter</name>
            <synopsis>Counters for the port</synopsis>
            <typeRef>PortCounterType</typeRef>
          </component>
        </components>
        <capabilities>
          <capability componentID="31">
            <name>Supported</name>
            <synopsis>Features Supported by the port</synopsis>
            <typeRef>PortFeaturesType</typeRef>
          </capability>
          <capability componentID="32">
            <name>Peer</name>
            <synopsis>Features advertised by the peer</synopsis>
            <typeRef>PortFeaturesType</typeRef>
          </capability>
        </capabilities>
      </LFBClassDef>
      <!-- Queue LFB -->
      <LFBClassDef LFBClassID="1028">
        <name>OFQueue</name>
        <synopsis>A queue LFB that can be attached to a port and be used
        to map flows on it. Flows mapped to a queue will be treated
        according to that queue’s configuration</synopsis>
        <version>1.1</version>
        <inputPorts>
          <inputPort group="true">
            <name>PacketIn</name>
            <synopsis>An input port that expects any kind of frame.
            </synopsis>
            <expectation>
              <frameExpected>
                <ref>Arbitrary</ref>
              </frameExpected>
            </expectation>
          </inputPort>
        </inputPorts>
        <components>
          <component componentID="1" access="read-only">
            <name>QueueID</name>
            <synopsis>ID for the specific queue</synopsis>
            <typeRef>uint32</typeRef>
          </component>
          <component componentID="2" access="read-write">
            <name>Properties</name>
            <synopsis>List of queue properties</synopsis>
            <array>
              <typeRef>QueueArrayPropertiesType</typeRef>
            </array>
          </component>
          <component componentID="3" access="read-only">
            <name>QueueCounter</name>
            <synopsis>Counters for the queue</synopsis>
            <typeRef>QueueCounterType</typeRef>
          </component>
        </components>
      </LFBClassDef>
      <!-- OFRedirectIn LFB -->
      <LFBClassDef LFBClassID="1029">
        <name>OFRedirectIn</name>
        <synopsis>The OFRedirectIn LFB abstracts the process for the
        controller to inject data packets into the switch to input data
        packets into the data path.</synopsis>
        <version>1.1</version>
        <derivedFrom>RedirectIn</derivedFrom>
        <outputPorts>
          <outputPort group="false">
            <name>PacketIn</name>
            <synopsis>An output port that sends a packet in the data
            path</synopsis>
            <product>
              <frameProduced>
                <ref>Arbitrary</ref>
              </frameProduced>
              <metadataProduced>
                <ref>ActionList</ref>
                <ref>IngressPort</ref>
              </metadataProduced>
            </product>
          </outputPort>
          <outputPort group="false">
            <name>BufferIn</name>
            <synopsis>An output port that sends only the buffer id to
            locate a buffered packet</synopsis>
            <product>
              <frameProduced>
                <ref>Null</ref>
              </frameProduced>
              <metadataProduced>
                <ref>BufferID</ref>
                <ref>ActionList</ref>
                <ref>IngressPort</ref>
              </metadataProduced>
            </product>
          </outputPort>
        </outputPorts>
      </LFBClassDef>
      <!-- OFRedirectOut LFB -->
      <LFBClassDef LFBClassID="1030">
        <name>OFRedirectOut</name>
        <synopsis>The OFRedirectOut LFB abstracts the process for the
        switch to deliver data packets to the controller</synopsis>
        <version>1.1</version>
        <inputPorts>
          <inputPort group="true">
            <name>Outgoing</name>
            <synopsis>The input port expects either the whole packet to
            be sent to the controller or part of it along with the buffer
            ID</synopsis>
            <expectation>
              <frameExpected>
                <ref>Arbitrary</ref>
              </frameExpected>
              <metadataExpected>
                <ref>IngressPort</ref>
                <ref>InPhyPort</ref>
                <ref>RedirectReason</ref>
                <ref dependency="optional">FlowTableID</ref>
                <ref dependency="optional">BufferID</ref>
              </metadataExpected>
            </expectation>
          </inputPort>
        </inputPorts>
      </LFBClassDef>
      <!-- Action LFBs -->
      <LFBClassDef LFBClassID="1031">
        <name>OFAction</name>
        <synopsis>An LFB that performs one specific action on a packet in
        the OpenFlow switch. The OFActionLFB expects any kind of packet
        and as metadata the FlowTableInstanceID to know from which Flow
        Table the packet has arrived from and the Action Index to specify
        the row in the Action Table, if there is an Action table.
        </synopsis>
        <version>1.1</version>
        <inputPorts>
          <inputPort group="true">
            <name>PacketIn</name>
            <synopsis>An input port that gets the packet to perform the
            action on. Expects the ClassID of the LFB that calls it to
            know to which LFB to return it to. Can accept calls from the
```

```
           OFFlowTables or the OFGroupLFB.</synopsis>
           <expectation>
             <frameExpected>
               <ref>Arbitrary</ref>
             </frameExpected>
             <metadataExpected>
               <ref>PacketID</ref>
               <ref>LFBClassIDMetadata</ref>
               <ref dependency="optional">ActionIndex</ref>
               <ref dependency="optional">IngressPort</ref>
               <ref dependency="optional">InPhyPort</ref>
               <ref dependency="optional">QueueID</ref>
             </metadataExpected>
           </expectation>
         </inputPort>
       </inputPorts>
       <outputPorts>
         <outputPort group="true">
           <name>PacketOut</name>
           <synopsis>The output port from which the packet will be send
           back to the LFB (OFFlowTables or OFGroupTable) from which it
           came from.</synopsis>
           <product>
             <frameProduced>
               <ref>Arbitrary</ref>
             </frameProduced>
             <metadataProduced>
               <ref>PacketID</ref>
             </metadataProduced>
           </product>
         </outputPort>
       </outputPorts>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1032">
       <name>OFActionOutput</name>
       <synopsis>An LFB that performs the Output Action</synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <outputPorts>
         <outputPort group="true">
           <name>PortOutput</name>
           <synopsis>Send a copy of the packet to the specified port
           </synopsis>
           <product>
             <frameProduced>
               <ref>Arbitrary</ref>
             </frameProduced>
           </product>
         </outputPort>
         <outputPort group="true">
           <name>All</name>
           <synopsis>Send the packet out all standard ports, but not to
           the ingress port or ports configured not to forward
           </synopsis>
           <product>
             <frameProduced>
               <ref>Arbitrary</ref>
             </frameProduced>
           </product>
         </outputPort>
         <outputPort group="false">
           <name>Controller</name>
           <synopsis>Send the packet to the controller</synopsis>
           <product>
             <frameProduced>
               <ref>Arbitrary</ref>
             </frameProduced>
           </product>
         </outputPort>
         <outputPort group="false">
           <name>Table</name>
           <synopsis>Submit the packet to the first flow table so that
           the packet can be processed through the regular OpenFlow
           pipeline. Only valid in the action set of a packet-out
           message</synopsis>
           <product>
             <frameProduced>
               <ref>Arbitrary</ref>
             </frameProduced>
           </product>
         </outputPort>
         <outputPort group="true">
           <name>InPort</name>
           <synopsis>Send the packet out the ingress port.</synopsis>
           <product>
             <frameProduced>
               <ref>Arbitrary</ref>
             </frameProduced>
           </product>
         </outputPort>
         <outputPort group="false">
           <name>Local</name>
           <synopsis>Send the packet to the switch’s local networking
           stack</synopsis>
           <product>
             <frameProduced>
```

```
               <ref>Arbitrary</ref>
             </frameProduced>
           </product>
         </outputPort>
         <outputPort group="false">
           <name>Normal</name>
           <synopsis>Process the packet using the traditional
           non-OpenFlow pipeline of the switch.</synopsis>
           <product>
             <frameProduced>
               <ref>Arbitrary</ref>
             </frameProduced>
           </product>
         </outputPort>
         <outputPort group="true">
           <name>Flood</name>
           <synopsis>Flood the packet using the normal pipeline of the
           switch.</synopsis>
           <product>
             <frameProduced>
               <ref>Arbitrary</ref>
             </frameProduced>
           </product>
         </outputPort>
       </outputPorts>
       <components>
         <component componentID="1" access="read-write">
           <name>OutputActionTable</name>
           <synopsis>Output to switch port</synopsis>
           <array>
             <struct>
               <component componentID="1">
                 <name>Port</name>
                 <synopsis>The port to send the packet out</synopsis>
                 <typeRef>PortNumberType</typeRef>
               </component>
               <component componentID="2">
                 <name>MaxLength</name>
                 <synopsis>If the port is the controller sets the
                 maximum number of bytes to send.</synopsis>
                 <typeRef>uint16</typeRef>
               </component>
             </struct>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1033">
```

```
       <name>OFActionSetVLANVID</name>
       <synopsis>An LFB that performs the Set VLANID Action</synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>SetVLANVIDActionTable</name>
           <synopsis>Set the 802.1q VLAN ID</synopsis>
           <array>
             <typeRef>uint16</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1034">
       <name>OFActionSetVLANPriority</name>
       <synopsis>An LFB that performs the Set VLAN Priority Action
       </synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>SetVLANPriorityActionTable</name>
           <synopsis>Set the 802.1q VLAN Priority</synopsis>
           <array>
             <typeRef>uchar</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1035">
       <name>OFActionSetMACSource</name>
       <synopsis>An LFB that performs the Set MAC Source Action
       </synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>SetMACSourceActionTable</name>
           <synopsis>Set MAC source address</synopsis>
           <array>
             <typeRef>IEEEMAC</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1036">
       <name>OFActionSetMACDestination</name>
       <synopsis>An LFB that performs the Set MAC Destionation Action
       </synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>SetMACDestinationActionTable</name>
           <synopsis>Set MAC destination address</synopsis>
           <array>
             <typeRef>IEEEMAC</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1037">
       <name>OFActionSetIPSource</name>
       <synopsis>An LFB that performs the Set IP Source Action
       </synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>SetIPSourceActionTable</name>
           <synopsis>Set the IP source address</synopsis>
           <array>
             <typeRef>IPv4Addr</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1038">
       <name>OFActionSetIPDestination</name>
       <synopsis>An LFB that performs the Set IP Destination Action
       </synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>SetIPDestinationActionTable</name>
           <synopsis>Set the IP destination address</synopsis>
           <array>
             <typeRef>IPv4Addr</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1039">
       <name>OFActionSetIPTOS</name>
       <synopsis>An LFB that performs the Set VLANID Action</synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>SetIPTOSActionTable</name>
           <synopsis>Set IP ToS field</synopsis>
           <array>
             <typeRef>uchar</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1040">
       <name>OFActionSetIPECN</name>
       <synopsis>An LFB that performs the Set IP ECN Action</synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>SetIPECNActionTable</name>
           <synopsis>Set IP ECN field</synopsis>
           <array>
             <typeRef>uchar</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1041">
       <name>OFActionSetTCPSource</name>
       <synopsis>An LFB that performs the Set TCP/UDP/SCTP Source port
       Action</synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>SetTCPSourceActionTable</name>
           <synopsis>Sets TCP/UDP/SCTP source port</synopsis>
           <array>
             <typeRef>uint16</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1042">
       <name>OFActionSetTCPDestination</name>
       <synopsis>An LFB that performs the Set TCP/UDP/SCTP Destination
        port Action</synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>SetTCPDestinationActionTable</name>
           <synopsis>Sets TCP/UDP/SCTP destination port</synopsis>
           <array>
             <typeRef>uint16</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1043">
       <name>OFActionCopyTTLOut</name>
       <synopsis>An LFB that performs the copy TTL outwards Action
       </synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1044">
       <name>OFActionCopyTTLIn</name>
       <synopsis>An LFB that performs the copy TTL inwards Action
       </synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1045">
       <name>OFActionSetMPLSLabel</name>
       <synopsis>An LFB that performs the Set MPLS Label Action
       </synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>SetMPLSLabelActionTable</name>
           <synopsis>Sets MPLS Label Table</synopsis>
           <array>
             <typeRef>uint32</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1046">
       <name>OFActionSetMPLSTC</name>
       <synopsis>An LFB that performs the Set MPLS Traffic Class Action
       </synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
```

```
       <components>
         <component componentID="1" access="read-write">
           <name>SetMPLSTCActionTable</name>
           <synopsis>Sets MPLS Traffic Class Table</synopsis>
           <array>
             <typeRef>uchar</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1047">
       <name>OFActionSetMPLSTTL</name>
       <synopsis>An LFB that performs the Set MPLS TTL Action</synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>SetMPLSTTLTable</name>
           <synopsis>Sets MPLS TTL Table</synopsis>
           <array>
             <typeRef>uchar</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1048">
       <name>OFActionDecrementMPLSTTL</name>
       <synopsis>An LFB that performs the decrementation of the MPLS TTL
       Action</synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1049">
       <name>OFActionPushVLan</name>
       <synopsis>An LFB that performs the Push VLAN Action</synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>PushVLANTable</name>
           <synopsis>Push VLAN Table</synopsis>
           <array>
             <typeRef>uint16</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1050">
```

```
        <name>OFActionPopVLAN</name>
        <synopsis>An LFB that performs the Pop VLAN Action</synopsis>
        <version>1.1</version>
        <derivedFrom>OFAction</derivedFrom>
      </LFBClassDef>
      <LFBClassDef LFBClassID="1051">
        <name>OFActionPushMPLS</name>
        <synopsis>An LFB that performs the Push MPLS Action</synopsis>
        <version>1.1</version>
        <derivedFrom>OFAction</derivedFrom>
        <components>
          <component componentID="1" access="read-write">
            <name>PushMPLSTable</name>
            <synopsis>Push MPLS Table</synopsis>
            <array>
              <typeRef>uint16</typeRef>
            </array>
          </component>
        </components>
      </LFBClassDef>
      <LFBClassDef LFBClassID="1052">
        <name>OFActionPopMPLS</name>
        <synopsis>An LFB that performs the Pop MPLS Action</synopsis>
        <version>1.1</version>
        <derivedFrom>OFAction</derivedFrom>
        <components>
          <component componentID="1" access="read-write">
            <name>PopMPLSTable</name>
            <synopsis>Pop MPLS Table</synopsis>
            <array>
              <typeRef>uint16</typeRef>
            </array>
          </component>
        </components>
      </LFBClassDef>
      <LFBClassDef LFBClassID="1053">
        <name>OFActionSetQueue</name>
        <synopsis>An LFB that performs the Set Queue Action</synopsis>
        <version>1.1</version>
        <derivedFrom>OFAction</derivedFrom>
        <outputPorts>
          <outputPort group="true">
            <name>QueuePacketOut</name>
            <synopsis>The output port from which the packet will be send
            back to the Flow Table/GroupTable from which it came from.
            </synopsis>
            <product>
              <frameProduced>
```

```
               <ref>Arbitrary</ref>
             </frameProduced>
             <metadataProduced>
               <ref>PacketID</ref>
               <ref>QueueID</ref>
             </metadataProduced>
           </product>
         </outputPort>
       </outputPorts>
       <components>
         <component componentID="1" access="read-write">
           <name>SetQueueTable</name>
           <synopsis>Sets Queue Table</synopsis>
           <array>
             <typeRef>uint32</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1054">
       <name>OFActionSetIPTTL</name>
       <synopsis>An LFB that performs the Set IP TTL Action</synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
       <components>
         <component componentID="1" access="read-write">
           <name>SetIPTTLActionTable</name>
           <synopsis>Sets IP TTL Table</synopsis>
           <array>
             <typeRef>uchar</typeRef>
           </array>
         </component>
       </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="1055">
       <name>OFActionDecrementIPTTL</name>
       <synopsis>An LFB that performs the decrementation of the IP TTL
       Action</synopsis>
       <version>1.1</version>
       <derivedFrom>OFAction</derivedFrom>
     </LFBClassDef>
   </LFBClassDefs>
 </LFBLibrary>
```

OpenFlow XML Library

7. Acknowledgements

The authors would like to thank Ahmad N. Quttoum, Zoltan Lajos Kis, Joel Halpern and especially Jamal Hadi Salim, for discussions which helped shape this document.

8. IANA Considerations

(TBD)

9. Security Considerations

TBD

10. References

10.1. Normative References

[I-D.ietf-forces-lfb-lib]
Wang, W., Haleplidis, E., Ogawa, K., Li, C., and J. Halpern, "ForCES Logical Function Block (LFB) Library", draft-ietf-forces-lfb-lib-08 (work in progress), February 2012.

[McKeown] "McKeown, N., Anderson, T., Balakrishnan, H., et al, "OpenFlow: enabling innovation in campus networks", ACM SIGCOMM Computer Communication Review. 2008, 38(2): 69-74.", <http://www.OpenFlow.org/documents/OpenFlow-spec-v1.1.0.pdf>.

[OpenFlowSpec1.1]
http://www.OpenFlow.org/, "The OpenFlow 1.1 Specification.", <http://www.OpenFlow.org/documents/OpenFlow-spec-v1.1.0.pdf>.

[RFC3654] Khosravi, H. and T. Anderson, "Requirements for Separation of IP Control and Forwarding", RFC 3654, November 2003.

[RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework", RFC 3746, April 2004.

[RFC5810] Doria, A., Hadi Salim, J., Haas, R., Khosravi, H., Wang, W., Dong, L., Gopal, R., and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification", RFC 5810, March 2010.

[RFC5811] Hadi Salim, J. and K. Ogawa, "SCTP-Based Transport Mapping Layer (TML) for the Forwarding and Control Element Separation (ForCES) Protocol", RFC 5811, March 2010.

[RFC5812] Halpern, J. and J. Hadi Salim, "Forwarding and Control Element Separation (ForCES) Forwarding Element Model", RFC 5812, March 2010.

[RFC5813] Haas, R., "Forwarding and Control Element Separation (ForCES) MIB", RFC 5813, March 2010.

[RFC6053] Haleplidis, E., Ogawa, K., Wang, W., and J. Hadi Salim, "Implementation Report for Forwarding and Control Element Separation (ForCES)", RFC 6053, November 2010.

10.2. Informative References

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

Authors' Addresses

Evangelos Haleplidis
University of Patras
Department of Electrical & Computer Engineering
Patras, 26500
Greece

Email: ehalep@ece.upatras.gr

Omar Cherkaoui
University of Quebec in Montreal
Montreal,
Canada

Email: cherkaoui.omar@uqam.ca

Susan Hares
Huawei
USA

Email: shares@ndzh.com

Weiming Wang
Zhejiang Gongshang University
18 Xuezheng Str., Xiasha University Town
Hangzhou, 310018
P.R.China

Phone: +86-571-28877721
Email: wmwang@zjgsu.edu.cn