OPEN NETWORKING
FOUNDATION

OF-CONFIG 1.2

OpenFlow Management and Configuration Protocol

ONF TS-016

ONF Document Type: OpenFlow Config
ONF Document Name: of-config-1.2

Disclaimer

Contents

1 Introduction

This document describes the motivation, scope, requirements, and specification of the standard configuration and management protocol of an operational context which is capable of containing an OpenFlow 1.3 (or previous versions) switch as described in Figure 1. This configuration and management protocol is referred to as OF-CONFIG and is a companion protocol to OpenFlow. This document specifies version 1.2 of OF-CONFIG.

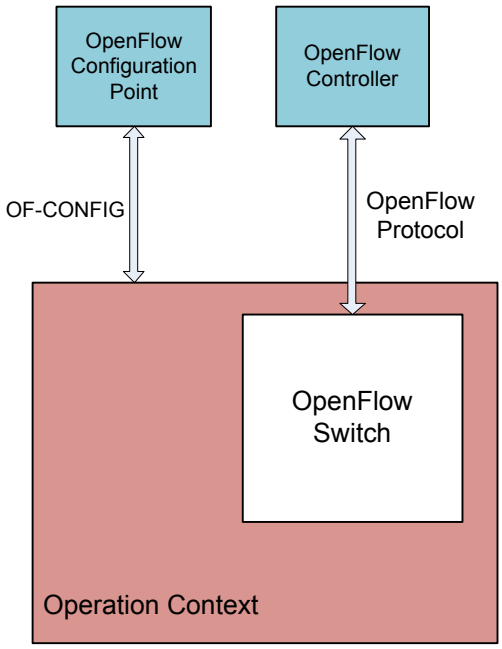


Figure 1: An OpenFlow Configuration Point communicates with an operational context which is capable of supporting an OpenFlow Switch using the OpenFlow Configuration and Management Protocol (OF-CONFIG)

The reader of this document is assumed to be familiar with the OpenFlow protocol and OpenFlow related concepts. Reading the OpenFlow whitepaper [2] and the OpenFlow Specification [1] is recommended prior to reading this document.

It is strongly recommended that switches which implement OF-CONFIG make changes to the OpenFlow logical switch described in this document via OF-CONFIG and limit changes to the OpenFlow logical switch via other methods (e.g. command line interfaces and other legacy management protocols). Future versions may better support out-of-band changes with detailed notification to the OpenFlow Configuration Point via OF-CONFIG.

2 Motivation

The OpenFlow protocol assumes that an OpenFlow switch (e.g. an Ethernet switch which supports the OpenFlow protocol) has been configured with various artifacts such as the IP addresses of OpenFlow controllers. The motivation for the OpenFlow Configuration Protocol (OF-CONFIG) is to enable the remote configuration of OpenFlow switches. While the OpenFlow protocol generally operates on a time-

scale of a flow (i.e. as flows are added and deleted), OF-CONFIG operates on a slower time-scale. An example is building forwarding tables and deciding forwarding actions which are done via Openflow protocol while enabling/disabling a port generally does not need to be done at the timescale of a flow and, hence, is done via OF-Config protocol.

OF-CONFIG defines an OpenFlow switch as an abstraction called an OpenFlow Logical Switch. The OF-CONFIG protocol enables configuration of essential artifacts of an OpenFlow Logical Switch so that an OpenFlow controller can communicate and control the OpenFlow Logical switch via the OpenFlow protocol.

OF-CONFIG introduces an operating context called an OpenFlow Capable Switch for one or more OpenFlow swithes. An OpenFlow Capable Switch is intended to be equivalent to an actual physical or virtual network element (e.g. an Ethernet switch) which is hosting one or more OpenFlow Logical Switches by partitioning a set of OpenFlow related resources such as ports and queues among the hosted OpenFlow Logical Switches . The OF-CONFIG protocol enables dynamic association of the OpenFlow related resources of an OpenFlow Capable Switch with specific OpenFlow Logical Switches which are being hosted on the OpenFlow Capable Switch. OF-CONFIG does not specify or report how the partitioning of resources on an OpenFlow Capable Switch is achieved. OF-CONFIG assumes that resources such as ports and queues are partitioned between multiple OpenFlow Logical Switches such that each OpenFlow Logical Switch can assume full control over the resources that is assigned to it.

OF-CONFIG 1.2 makes simplifying assumptions about the architecture of OpenFlow switches. The specification is deliberately decoupled from whether the switch supports virtualization models or specific hybrid operational models, for example.

The service which sends OF-CONFIG messages to an OpenFlow Capable Switch is called an OpenFlow Configuration Point. No assumptions are made about the nature of the OpenFlow Configuration Point. For example, it may be provided by software acting as an OpenFlow controller or it may by a service provided by a traditional network management framework. In some deployment contexts, the OpenFlow Configuration Point and OpenFlow controller may belong to different administrative entities, e.g., provider and customer, respectively. Interactions between the OpenFlow Configuration Points and OpenFlow controllers is outside the scope of OF-CONFIG 1.2, but is expected to be addressed in future versions of the specification.

Figure 2 shows the basic abstractions detailed in OF-CONFIG 1.2 and the lines indicate that the OpenFlow Configuration Points and OpenFlow Capable Switches communicate via OF-CONFIG. The configuration settings then take effect on targeted logical switch(es). OpenFlow Controllers and OpenFlow Logical Switches communicate via OpenFlow.

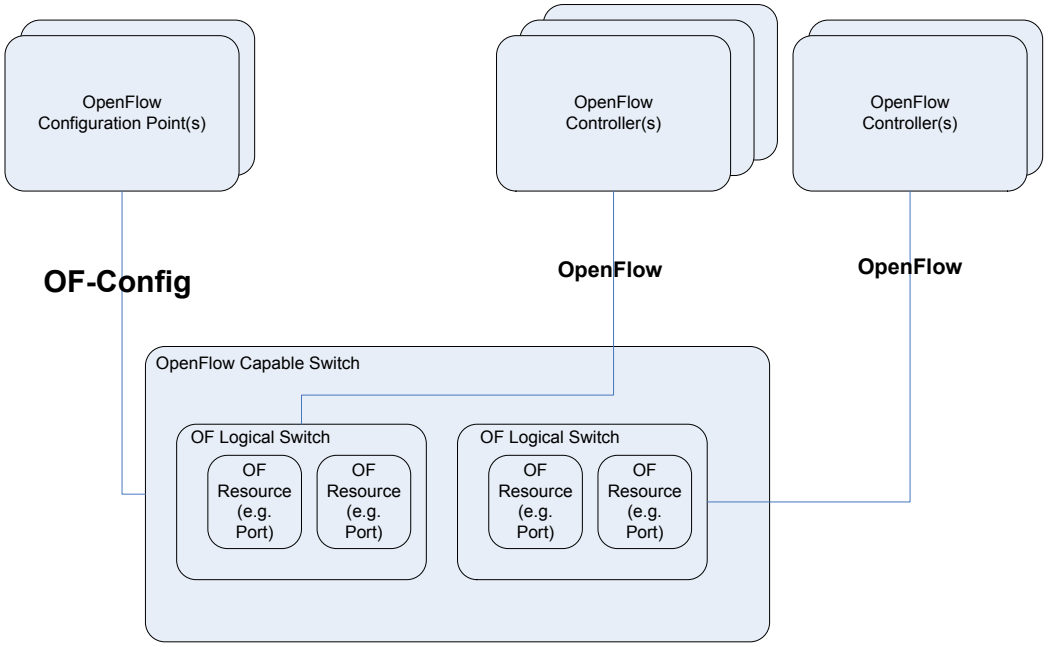


Figure 2: Relationship between components defined in this specification, the OF-CONFIG protocol and the OpenFlow protocol

A guiding principle in the development of this specification is to keep the protocol and schema simple and leverage existing protocols and schema models where possible. This helped in quick development of this specification and hopefully will also enable easier adoption, the motivation being to supplement the OpenFlow specification in a meaningful way to further drive the adoption of the software defined networking vision.

2.1 OF-CONFIG and OF-SWITCH

Although OF-CONFIG is considered a complementary protocol to the main OpenFlow switch specification (OF-SWITCH), it is useful to describe the differences that motivate the need for a separate protocol specification in ONF. The table below summarizes the key differences.

	OpenFlow	OF-CONFIG
Primary purpose	Modification of match-action rules effecting flows of network packets across an OpenFlow datapath	Remote configuration of possibly multiple OpenFlow datapaths on a physical or virtual platform
Terminology	Newer versions of the OpenFlow 1.3 specification adopt the term OpenFlow Logical Switch and distinguish it from the earlier “datapath” term. The terms OpenFlow Logical Swtich and OpenFlow Switch are interchangeable	In the OF-Config 1.1.1 specification the term OpenFlow Capable Switch introduces a new abstraction. OpenFlow Capable Switch = a new element OpenFlow Configuration Point = a new element
Transport	A bit-level protocol specified in the	An XML data model and operational

	OpenFlow standard currently supported over TCP, TLS, or SSL	behavior specified in the OF-Config standard bound to the NETCONF operations and transport standard for network device configuration and management
Protocol endpoints	1) An OpenFlow datapath, also referred to as OpenFlow Logical Switch (OFLS) 2) An OpenFlow Controller (OFC)	1) An OpenFlow Capable Switch (OFCS) able to instantiate one or more OpenFlow Logical Switches (i.e. OpenFlow datapaths) 2) An OpenFlow Configuration Point (OFCP)
Example usage	An OpenFlow Controller adds a flow modification to an OpenFlow datapath (OFLS) which allows Ethernet frames containing IP packets which originated from 192.168.3.10 and are coming in on the datapath's port 2 to be forwarded out on the datapath's port 14	An OpenFlow Configuration Point configures a particular OpenFlow Logical Switch (OF datapath) to be associated with a particular OpenFlow Controller

It should be noted that some properties of the OpenFlow Logical Switch (i.e. OpenFlow switch) are available and configurable via both OpenFlow and OF-CONFIG. Having multiple channels for some of the same data is considered appropriate since the primary purpose of OpenFlow Configuration Points and OpenFlow Controllers are quite different. The ONF strives for synchronization of the data models and semantics between the OpenFlow and OF-CONFIG standards prior to update of either standard.

3 Scope

OF-CONFIG 1.2 is focused on the following functions needed to configure an OpenFlow 1.3 logical switch:

- The assignment of one or more OpenFlow controllers to OpenFlow data planes
- The configuration of queues and ports
- The ability to remotely change some aspects of ports (e.g. up/down)
- Configuration of certificates for secure communication between the OpenFlow Logical Switches and OpenFlow Controllers
- Discovery of capabilities of an OpenFlow Logical Switch
- Configuration of a set of specific tunnel types such as IP-in-GRE, NV-GRE, VxLAN

New functionality introduced in OF-CONFIG 1.2 includes:

Instantiation of OpenFlow data planes (called OpenFlow Logical Switches)

Assignment of resources of an OpenFlow Capable Switch to one or more OpenFlow Logical Switches

Support for the emerging Negotiable Datapath Model (NDM) being developed in the ONF

- Versioning Support for negotiating which version(s) of OF-CONFIG are supported

Other functions and/or the description of their use have been improved.

While limited in scope, OF-CONFIG 1.2 lays the foundation on top of which various automated and more advanced configurations will be possible in future revisions. The ONF Configuration and Management working group will publish additional specifications for network operations, administration, and management (OAM), including, topology discovery, event management, and bootstrap of the OpenFlow capable network.

Note that even though this specification refers to OpenFlow 1.3, OF-CONFIG 1.2 supports previous OpenFlow versions, specifically, OpenFlow 1.0, 1.1 and 1.2.

4 Normative 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 RFC 2119 [3].

5 Terms

The following section lists several terms and definitions used in this document.

5.1 OpenFlow Capable Switch

An OpenFlow Capable switch is a physical or virtual switching device which can act an as operational context for an OpenFlow Logical Switch. OpenFlow Capable Switches contain and manage OpenFlow Resources which may be associated with an OpenFlow Logical Switch context.

5.2 OpenFlow Configuration Point

An OpenFlow Configuration Point configures one or more OpenFlow Capable Switches via the OpenFlow Configuration and Management Protocol (OF-CONFIG).

5.3 OpenFlow Logical Switch

An OpenFlow Logical Switch is a set of resources (e.g. ports) from an OpenFlow Capable Switch which can be associated with a specific OpenFlow Controller. An OpenFlow Logical switch is an instantiation of an OpenFlow Switch as specified in [1].

5.4 OpenFlow Resource

An OpenFlow Resource is a resource (e.g. port or queue) which is associated with an OpenFlow Capable Switch and may be associated with an OpenFlow Logical Switch.

5.4.1 OpenFlow Queue

An OpenFlow Queue is a queuing resource of an OpenFlow Logical Switch as described in the OpenFlow specification as the queue component of an OpenFlow switch.

5.4.2 OpenFlow Port

An OpenFlow Port is a forwarding interface of an OpenFlow Logical Switch as described in the OpenFlow specification as the port component of an OpenFlow switch. An Openflow Port may map to a physical port on a physical switch or a logical port on a physical or virtual switch.

5.5 OpenFlow Controller

An OpenFlow Controller is software which controls OpenFlow Logical Switches via the OpenFlow protocol.

5.6 NDM

A Negotiable Datapath Model (NDM) is an abstract switch model that describes specific switch forwarding behaviors controllable via the OpenFlow-Switch protocol. The NDM describes specific requirements for switch behavior so that implementers can perform optimizations or deliver more complex forwarding behaviors (beyond what can be scalably represented in a single OpenFlow table) than they could otherwise.

6 Requirements

This section describes requirements for the design of OF-CONFIG 1.2.

6.1 Requirements from the OpenFlow 1.3 Protocol Specification

The specification of version 1.3 of the OpenFlow protocol [1] includes explicit and implicit requirements for the configuration of OpenFlow switches. In [1] the term ‘configuration’ is used for two different kinds of operations: configuration using the OpenFlow protocol and configuration outside of the OpenFlow protocol. The first kind of configuration is dealt within [1]. OF-CONFIG 1.2 enables other configuration of OpenFlow switches. The specification of OF-CONFIG 1.2 is written with extensibility in mind. This includes versioning and backward compatibility.

6.1.1 Instantiation of one or more Openflow Data Planes on an Openflow Capable Switch

An OpenFlow capable switch is capable of hosting one or more OpenFlow data planes (also refered to as OpenFlow logical switch). Initially, the OpenFlow capable switch owns all the resources of the switch and does not have any data plane instantiated. Using the OF-CONFIG 1.2 protocol, OFCP can instantiate one or more OpenFlow data planes and can assign resources such as queues and ports to these OpenFlow data planes. Some of the resources like management port may not be assigned to any OpenFlow data plane.

6.1.2 Connection Setup to a Controller

Section 6.3 (Connection Setup) of [1] indicates that an OpenFlow switch must be able to initiate the connection to the OpenFlow controller and discusses the process of setting up a connection between the OpenFlow switch and an OpenFlow controller. The switch initiates the connection applying three parameters that need to be configured in advance. Note that OpenFlow 1.3 also allows the OpenFlow controller to optionally initiate the connection to the switch (described in [1]).

- the IP address of the controller
- the port number at the controller (optional) if the default OpenFlow transport port 6653 is not being used
- the transport protocol to use, either TLS or TCP
- the port number at the switch (optional) if controller-initiated connections are used

OF-CONFIG 1.2 must provide means for configuring these parameters. Note that in future, alternative mechanisms for discovering the OpenFlow controller may be supported.

6.1.3 Multiple Controllers

Section 6.3 of [1] discusses how a switch deals with multiple controllers simultaneously. This implicitly requires OF-CONFIG 1.2 to provide means for configuring multiple instances of the parameter set listed in 6.1.1 for specifying the connection setup to multiple controllers.

6.1.4 OpenFlow Logical Switches

The OpenFlow 1.3 protocol specifies various kinds of OpenFlow resources associated with an OpenFlow Logical Switch. The OF-CONFIG protocol must support the configuration of these OpenFlow resources associated with an OpenFlow Logical Switch. Examples of resources include queues and ports that have been assigned to an OpenFlow Logical Switch. It is assumed that OpenFlow Logical Switches have been instantiated out of band, for example, an administrator may have created them upfront. In addition, partitioning/assignment of OpenFlow resources amongst multiple OpenFlow switches that may exist in an OpenFlow Capable Switch has also been done out of band.

6.1.5 Connection Interruption

Section 6.4 of [1] discusses the choice of two modes the switch should immediately enter after losing contact with all controllers. The modes are

- fail secure mode
- fail standalone mode

OF-CONFIG protocol must provide means for configuring the mode to enter in such a case.

6.1.6 Encryption

Section 6.5 of [1] discusses encryption of connections to controllers that use TLS. It explicitly states "Each switch must be user-configurable with one certificate for authenticating the controller (controller certificate) and the other for authenticating to the controller (switch certificate)". Hence, OF-CONFIG must provide means for configuring a switch certificate and a controller certificate for each controller that is configured to use TLS.

6.1.7 Queues

Section A.3.6 of [1] describes the configuration of queues. Queue in [1] have three parameters that may be configurable:

- min-rate
- max-rate
- experimenter

OF-CONFIG 1.2 must provide means for configuring these parameters.

6.1.8 Ports

The OpenFlow protocol already contains methods to configure a limited amount of port parameters of OpenFlow switches. The OpenFlow protocol specification [1] does not explicitly require an external configuration means, and therefore we cannot derive the requirement for configuring ports from [1]. However, the configuration of ports is an essential step of configuring a network and thus a requirement for OF-CONFIG 1.2. Section A.3.4.3 of [1] defines the following parameters for port configuration:

- no-receive
- no-forward
- no-packetin
- admin-state

OF-CONFIG 1.2 must provide means for configuring these parameters.

Also defined in Section A.2.1 of the OpenFlow protocol specification [1] are port features. There are four sets of these features for current, advertised, supported, and peer-advertised features. Feature sets current, supported, and peer-advertised contain state information and cannot to be configured. Only advertised features could potentially be configured with the following parameters:

- speed

- duplex-mode
- copper-medium
- fiber-medium
- auto-negotiation
- pause
- asymmetric-pause

OF-CONFIG 1.2 must provide means for configuring these advertised features and for obtaining current, supported and peer-advertised state information for these features.

Section 4.4 of [1] defines logical ports that are higher level abstratcions and that may include encapsulation. In addition, logical ports support passing of meta data to the controller. These logical ports may be used in for example, datacenter scenarios for setting up virtual networks. OF-CONFIG 1.2 must support the configuration of these logical ports. However, the configuration of logical ports in OF-CONFIG 1.2 is limited to a small number of tunnels (specifically to IPinGRE, VxLAN and NVGRE) that may be used in datacenter scenarios like network virtualization. Future versions of OF-CONFIG will support configuration of additional types of tunnels.

6.1.9 Capability Discovery

OpenFlow 1.3 describes the various capabilities that an OpenFlow Logical Switch may implement eg there are several actions in OpenFlow 1.3 that are optional. While configuration of these capabilities is outside the scope of OF-CONFIG 1.2, it supports discovery of these capabilities. It is assumed that capabilities have been configured for OpenFlow Logical switches either as part of instantiation of these switches or through some out of band mechanisms.

6.1.10 Datapath ID

Section A.3.1 of [1] discusses the datapath ID of a switch. It is a 64-bit field with the lower 48 bit intended for the switch MAC address and the remaining 16 bit left to the switch operator. Although not explicitly requested by [1], OF-CONFIG should provide means for configuring the datapath ID.

6.2 Requirements for NDMs

OF-CONFIG 1.2 includes optional support for Negotiable Datapath Models (NDMs) [5]. An NDM is an abstract switch model that describes specific switch forwarding behaviors controllable via the OpenFlow-Switch protocol.

When a capable switch implements the NDM framework (which is an optional enhancement to OpenFlow), an OFCP and a capable switch agree on an NDM to be associated with a logical switch prior to sending control messages, such as flowmods, to the logical switch. This agreement may be implicit (i.e., each side is configured a priori) or negotiated when the control relationship is established.

NDMs are characterized by parameters related to table sizes or optional functionality. The NDM framework allows for implementations to have a range of flexibility in their parameters. Some

implementations may have no flexibility; others will allow some adjustment of parameters at the time the OFCP associates the NDM with a logical switch. NDM implementations that support parameter adjustment should also offer an RPC mechanism to allow the OFCP and the capable switch to determine the parameters in a specific situation.

The NDM framework simplifies the job of implementing an OpenFlow controller or OpenFlow agent for a switch. The NDM describes specific requirements for switch behavior so that implementers can perform optimizations or deliver more complex forwarding behaviors (beyond what can be represented in a single OpenFlow table) than they could otherwise.

The optional NDM manageability feature must support the following requirements:

1. The ability to query the capable switch about support for NDMs
2. The ability to query the capable switch for the set of available supported NDMs
3. The ability to associate a logical switch with a parameterized NDM
4. The ability to remove a parameterized NDM from a logical switch

6.3 Operational Requirements

The OF-CONFIG 1.2 must meet support the following scenarios:

1. OF-CONFIG 1.2 must support an OpenFlow Capable Switch being configured by multiple OpenFlow Configuration Points.
2. OF-CONFIG 1.2 must support an OpenFlow Configuration Point managing multiple OpenFlow Capable Switches.
3. OF-CONFIG 1.2 must support an OpenFlow Logical Switch being controlled by multiple OpenFlow Controllers.
4. OF-CONFIG 1.2 must support configuring ports and queues of an OpenFlow Capable Switch that have been assigned to an OpenFlow Logical Switch.
5. OF-CONFIG 1.2 must support discovery of capabilities of an OpenFlow Logical Switch.
6. OF-CONFIG 1.2 must support configuration of tunnels such as IP-in-GRE, NVGRE and VxLan that are represented as logical ports of an OpenFlow Logical Switch.

6.4 Requirements for the Switch Management Protocol

OF-CONFIG 1.2 defines a communication standard between an OpenFlow switch and an OpenFlow Configuration Point. It consists of a network management protocol specified in Section 7 and a data model defined in Section 8. This subsection specifies requirements for the network management protocol. Note that these requirements are a superset of the requirements that may be needed for the limited scope of configuration specified in this specifications. The intent for the below requirements is to future proof the protocol choice so that we are able to address the future scenarios without having to modify the protocol choice itself. The protocol must comply with the following requirements:

1. The protocol must be secure providing integrity, privacy, and authentication. Authentication of both ends, switch and configuration point, must be supported.
2. The protocol must support reliable transport of configuration requests and replies.
3. The protocol must support connection setup by the configuration point.
4. The protocol should support connection setup by the switch.
5. The protocol must be able to carry partial switch configurations.
6. The protocol must be able to carry bulk switch configurations.
7. The protocol must support the configuration point setting configuration data at the switch
8. The protocol must support the configuration point retrieving configuration data from the switch.
9. The protocol should support the configuration point retrieving status information from the switch.
10. The protocol must support creation, modification and deletion of configuration information at the switch.
11. The protocol must support reporting on the result of a successful configuration request.
12. The protocol must support reporting error codes for partially or completely failed configuration requests.
13. The protocol should support sending configuration requests independent of the completion of previous requests.
14. The protocol should support transaction capabilities including rollback per operation.
15. The protocol must provide means for asynchronous notifications from the switch to the configuration point. An example may be, even though this scenario is out of scope for OF-CONFIG 1.2, is if an administrator changes a configuration out of band, the switch may need to provide an appropriate notification to the OFCP.
16. The protocol should be extensible.
17. The protocol should support reporting its capabilities.

7 NETCONF as the Transport Protocol

The OF-CONFIG1.2 protocol provides a standard way to modify basic OpenFlow configuration for the operation of an OpenFlow logical switch within the context of an OpenFlow Capable Switch. At the same time, it provides vendors the ability to extend and innovate by providing new and improved configuration capabilities. To achieve these goals, OF-CONFIG 1.2 requires that devices supporting OF-CONFIG MUST implement the NETCONF protocol [4] as their transport protocol. This in turn implies as specified by the NETCONF specification that OpenFlow Capable Switches supporting OF-CONFIG must

implement SSH as a transport protocol. In addition, the OpenFlow Capable Switches implementing OF-CONFIG protocol may implement additional transports such as Web Services-Management or something else. Future versions of OF-CONFIG may specify binding to these additional transports.

NETCONF is a stable protocol that has been standardized for several years now. It is widely available on various platforms and achieves the needs for OF-CONFIG. NETCONF defines a set of operations on top of a messaging layer (RPC). The diagram below shows the various layers of the NETCONF protocol.

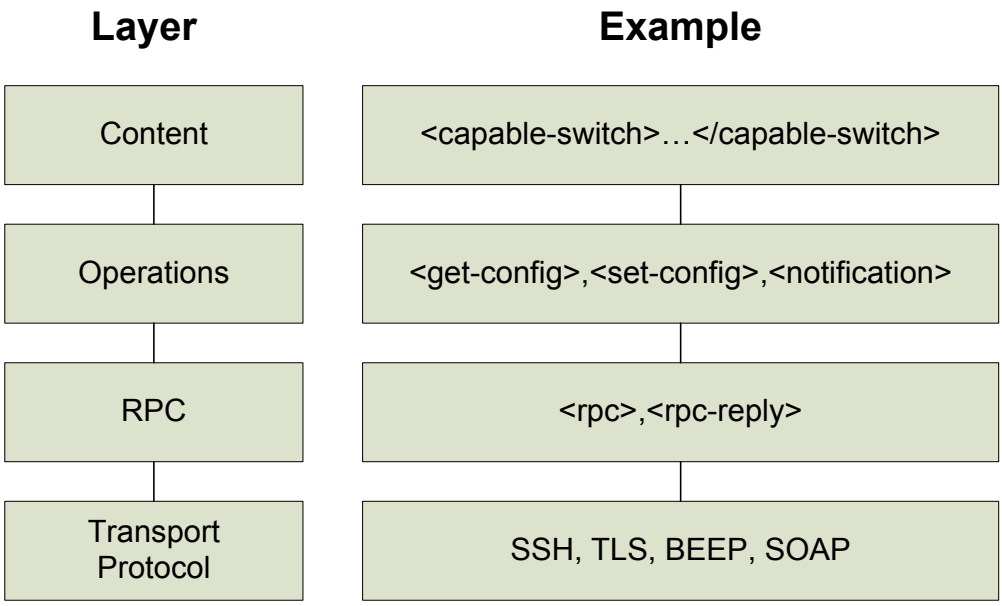


Figure 36 NETCONF Layers and Examples

The OpenFlow capable switches MUST support the schema as defined in this specification as the content layer in the above diagram. The schema currently covers basic configuration elements and will be extended in the next versions of this document.

The NETCONF protocol meets the OF0 CONFIG 1.2 requirements for communication between an OpenFlow Configuration Point and an OpenFlow switch as listed in Section 6.4. In addition, if future needs of OF-CONFIG are not met by the NETCONF protocol, NETCONF is extensible which will allow OF-CONFIG to extend NETCONF for its purpose.

1 It supports TLS as communication transport protocol (directly or with SOAP or BEEP in between) that can be used for providing integrity, privacy, and mutual authentication.

2 All specified transport mappings for NETCONF use TLS or TCP as underlying transport protocol and thus provide reliable transport.

3 The common way to establish a connection with NETCONF is from the Configuration Point (configuration point) to the managed device (switch).

4 The NETCONF standards support reversed configuration setup only if BEEP is used as transport protocol.

5 It supports partial switch configuration to the most fine-grain level.

6 It supports full switch configuration with a single operation.

7 It supports setting of configuration data.

8 It supports the retrieval of configuration data.

9 It supports the retrieval of (non-configuration) status data.

10 It supports creation, modification and deletion of configuration information.

11 It supports returning success codes after completing a configuration operation.

12 It supports support reporting error codes for partially or completely failed configuration requests.

13 It supports sending configuration requests independent of the completion of previous requests. Requests may be queued or processed concurrently at a switch. Each request has a request ID. Success or failure indications can be sent independently of other requests individually for each request ID.

14 It supports transaction capabilities including rollback per operation.

15 With its extension defined in RFC 5277 it supports asynchronous notifications from the managed device (switch) to the Configuration Point (configuration point).

16 It is extensible. New operations can be added and its support can be checked by capability retrieval.

17 It supports reporting its capabilities.

8 Data Model

This section specifies the data model for OF-CONFIG 1.2. Configurations of an OpenFlow Capable Switch or for portions of it are encoded in XML. The data model is structured into classes and attributes of classes. Each class is described in a separate sub-section by

1. a UML diagram giving an overview of the class,
2. an example for XML code encoding an instance of the class

The full XML schema is provided as a separate companion file. Normative for OF-CONFIG 1.2 is the XML schema and the normative constraints in the descriptions of the individual elements.

One of the design goals of the model is efficient and clear encoding of switch configurations in XML. Human readability is a strong feature of XML. But since the XML schema will mainly be created and parsed by the protocol entity, the ease of encoding and parsing was preferred over readability. This implies that in case of a trade-off between cleanness and simplicity of the XML-based configuration and

simplicity of the XML schema, usually cleanness and simplicity of the XML-based configuration has been preferred.

8.1 YANG Module

OF-CONFIG 1.2 has a companion YANG module, also distributed as a separate file to aid in implementation of the OF-CONFIG data model. It incorporates the XML schema specifications as well as the normative constraints though is not normative for this specification. The YANG module conforms to the normative constraints given in XML schema of 2013 and the additional explanations in this section. Most of the constraints that are given in the description of the XML schema are automatically enforced in the YANG module by syntax elements already built into the YANG language. Implementers that already use the NETCONF tools could profit by using the YANG module to reduce implementation time. Nevertheless, they need to ensure that all normative constraints are obeyed – including those that are not expressible by the YANG syntax.

8.2 Core Data Model

The following UML diagram describes the top-level classes of the data model.

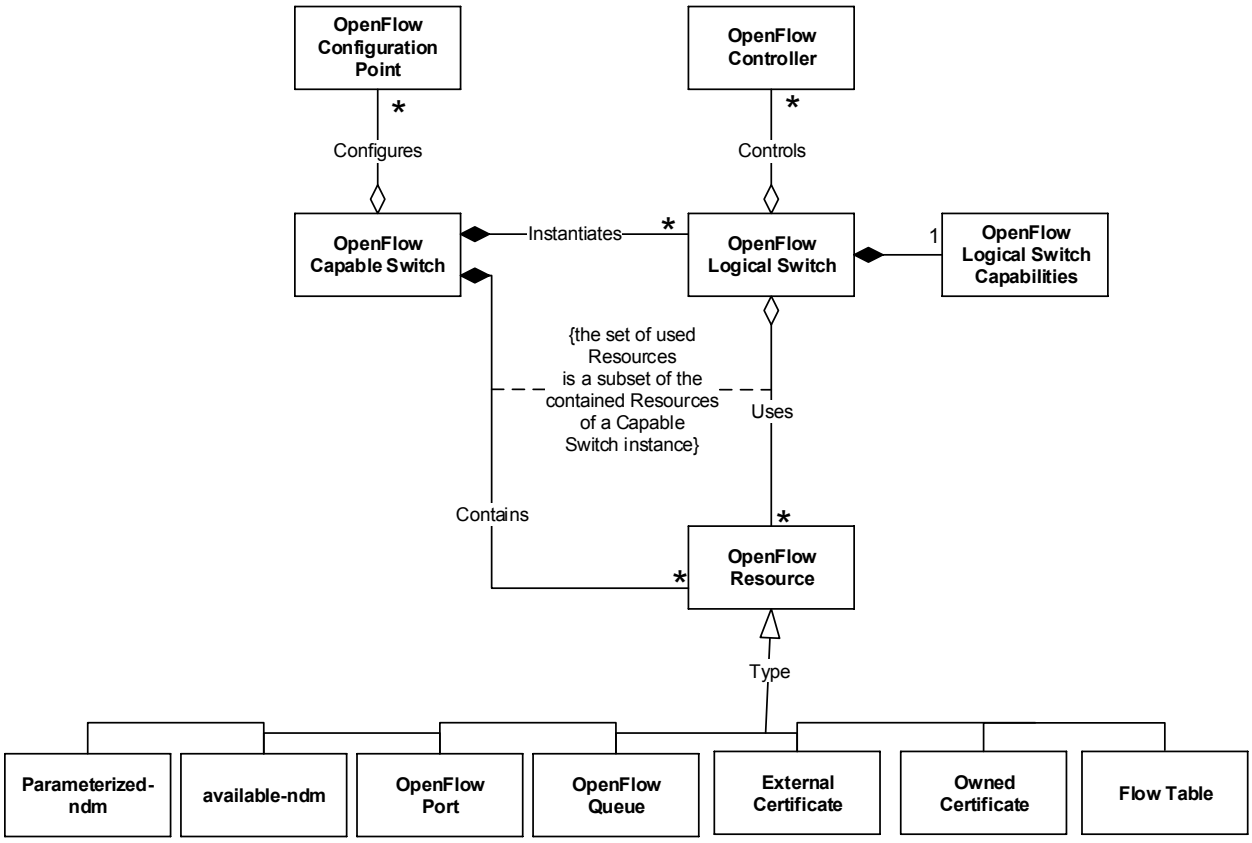


Figure 4: UML Class Diagram for OF-CONFIG Data Model

The core of the model is an OpenFlow Capable Switch that is configured by OpenFlow Configuration Points.

The switch contains a set of resources of different types. For OF-CONFIG 1.2, several types of resources are included in the model: OpenFlow Ports, OpenFlow Queues, External Certificate, Owned Certificate and Flow Table. More resource types may be added in future revisions of OF-CONFIG. OpenFlow resources can be made available for use to OpenFlow Logical Switches.

Instances of OpenFlow logical switches are contained within the OpenFlow Capable Switch. A set of OpenFlow Controllers is assigned to each OpenFlow logical switch.

The data model contains several identifiers, most of them encoded as an XML element <id>. Currently these IDs are defined as strings with required uniqueness in a certain context. Beyond uniqueness requirements, no further guidance is given on how to build these strings. This may be changed in the future. Particularly, the use of Universal Resource Names (URNs) is envisioned. This requires developing a naming scheme for URNs in OF-CONFIG and registering a URN namespace for the ONF. It is expected that recommendations for URN-based identifiers will be introduced by a future version of OF-CONFIG. Since URNs are represented as strings, such recommendations can be made compatible with identifiers in OF-CONFIG 1.2.

The UML models in this document are intended to represent high-level structure of the data model and may not reflect all details of each attribute. The full schema reference is in the companion YANG module and XML schema files.

8.3 OpenFlow Capable Switch

The OpenFlow Capable Switch serves as the root element for an OpenFlow configuration. It has relationships to

- OpenFlow Configuration Points that manage and particularly configure the OpenFlow Capable Switch,
- OpenFlow logical switches that are contained and instantiated within the OpenFlow Capable Switch,
- OpenFlow Resources contained in the OpenFlow Capable Switch that may be used by OpenFlow Logical Switches.

8.3.1 UML Diagram

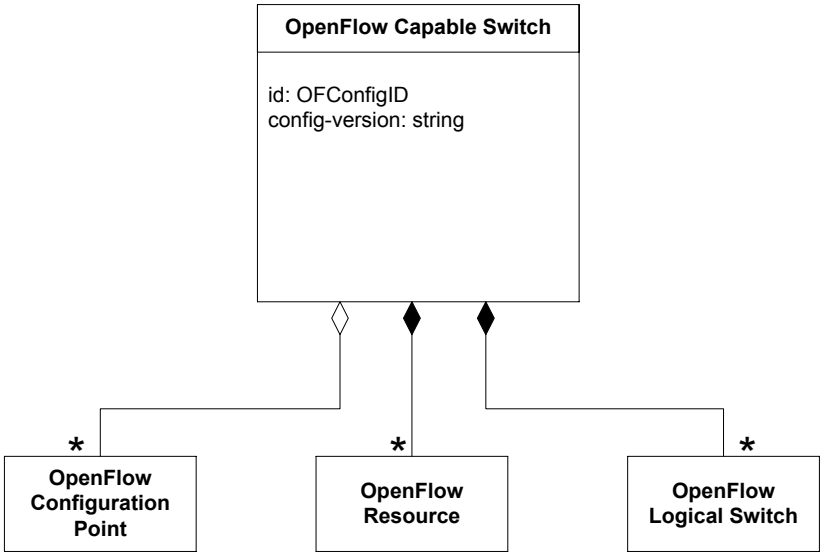


Figure 5: Data Model Diagram for OpenFlow Capable Switch

8.3.2 XML Example

```
<capable-switch>
  <id>CapableSwitch0</id>
  <configuration-points>
    ...
  </configuration-points>
  <resources>
    ...
  </resources>
  <logical-switches>
    ...
  </logical-switches>
</capable-switch>
```

8.4 OpenFlow Configuration Point

The Configuration Point is an entity that manages the switch using the OF-CONFIG protocol. Attributes of an OpenFlow Configuration Point allow the OpenFlow Capable Switches to identify a Configuration Point and specify which protocol is used for communication between Configuration Point and OpenFlow Capable Switch. The OpenFlow Capable Switch stores a list of Configuration Points that manage it or have managed it. An OpenFlow Configuration Point is to an OpenFlow Capable Switch what an OpenFlow Controller is to an OpenFlow Logical switch.

Instances of the Configuration Point class are used by switches to connect to a configuration point. Currently the only transport mapping that supports a connection set-up initiated by the switch to be configured is the mapping to the BEEP protocol (5). Other NETCONF transport mappings (6,7,8) may be

extended in the future to also support connection set-up in this direction. Nevertheless SSH is used as a default connection protocol because connection initiation by the switch is optional.

8.4.1 UML Diagram

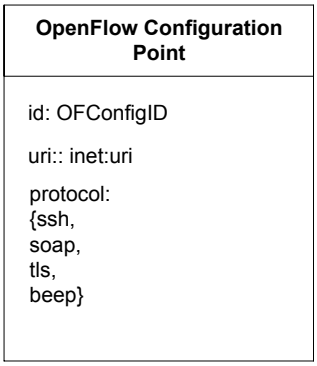


Figure 6: Data Model Diagram for an OpenFlow Configuration Point

8.4.2 XML Example

```
<configuration-point>
   <id>ConfigurationPoint1</id>
   <uri>uri0</uri>
   <protocol>ssh</protocol>
<configuration-point>
```

8.5 OpenFlow Logical Switch

The OpenFlow Logical Switch represents an instance of a logical switch that is available or can be made available on an OpenFlow Capable Switch. An OpenFlow Logical switch is a logical context which behaves as the datapath as described in the OpenFlow specification. The OpenFlow Logical Switch is connected to one or more OpenFlow Controllers via the OpenFlow protocol. It uses resources of the OpenFlow Capable Switch for realizing the capabilities offered via the OpenFlow protocol. The OpenFlow Logical Switch has relationships to

- OpenFlow Controllers that control the OpenFlow Capable Switch
- OpenFlow Resources that are available from the OpenFlow Capable Switch

8.5.1 UML Diagram

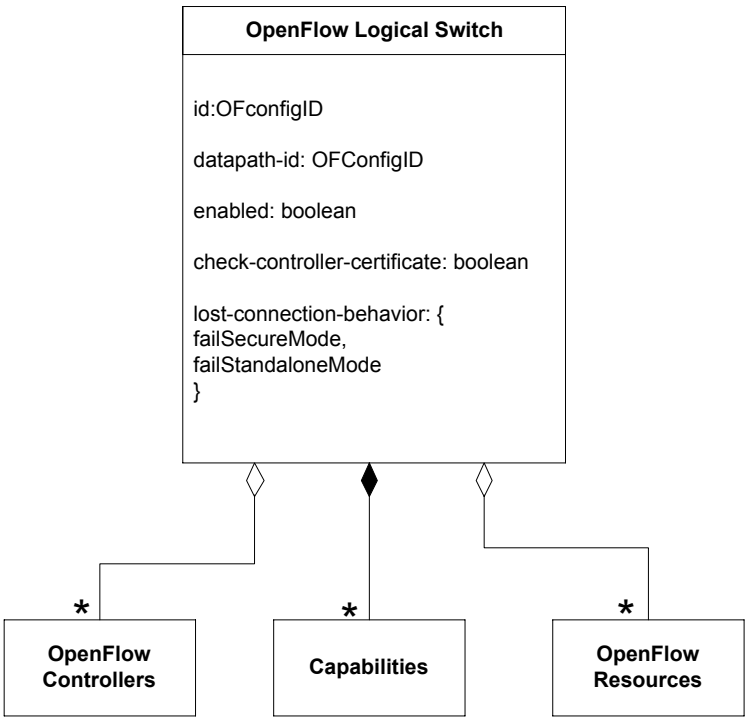


Figure 7: Data Model Diagram for an OpenFlow Logical Switch

8.5.2 XML Example

```
<logical-switch>
  <id>LogicalSwitch5</id>
    <capabilities>
        ...
    <capabilities>

  <datapath-id>datapath-id0</datapath-id>
  <enabled>true</enabled>
  <check-controller-certificate>false</check-controller-certificate>
  <lost-connection-behavior>failSecureMode</lost-connection-behavior>
  <controllers>
    ...
  </controllers>
  <resources>
    <port>port2</port>
    <port>port3</port>
    <queue>queue0</queue>
    <queue>queue1</queue>
    <certificate>ownedCertificate4</certificate>
    <flow-table>1</flow-table>
    <flow-table>2</flow-table>
```

```
    …
    <flow-table>255</flow-table>
  </resources>
</logical-switch>
```

8.6 Logical Switch Capabilities

8.6.1 UML Diagram

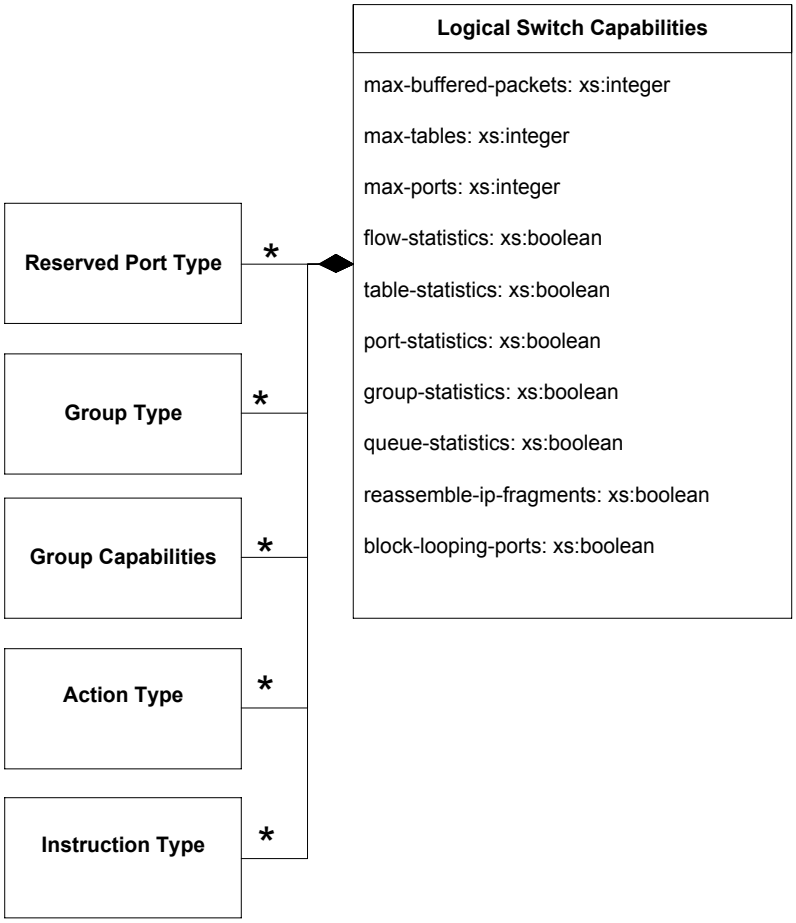


Figure 7: Data Model Diagram for an OpenFlow Logical Switch Capabilities

8.6.2 XML Example

```
<capabilities>
  <max-buffered-packets>512</max-buffered-packets>
  <max-tables>1024</max-tables>
  <max-ports>2048</max-ports>
  <flow-statistics>true</flow-statistics>
  <table-statistics>false</table-statistics>
  <port-statistics>true</port-statistics>
  <group-statistics>false</group-statistics>
  <queue-statistics>true</queue-statistics>
```

```
  <reassemble-ip-fragments>false</reassemble-ip-fragments>
  <block-looping-ports>false</block-looping-ports>
  <reserved-port-types>
    <type>all</type>
  </reserved-port-types>
  <group-types>
    <type>all</type>
  </group-types>
  <group-capabilities>
    <capability>select-weight</capability>
  </group-capabilities>
  <action-types>
    <type>output</type>
  </action-types>
  <instruction-types>
    <type>apply-actions</type>
    <type>write-actions</type>
  </instruction-types>
</capabilities>
```

8.7 OpenFlow Controller

The OpenFlow Controller class represents an entity that acts as OpenFlow Controller of an OpenFlow Logical Switch. Attributes of the class indicate the role of the controller and parameters of the OpenFlow connection to the controller. The port attribute should have a default value of 6653, the IANA-assigned port for OpenFlow. Note that normally, the OpenFlow switch initiates a connection to the controller and the local port attribute indicates the local ephemeral port that should be used at the switch. In the optional case where the controller initiates the connection, the local port attribute indicates the listening port on the switch (which should also be the IANA assigned port).

8.7.1 UML Diagram

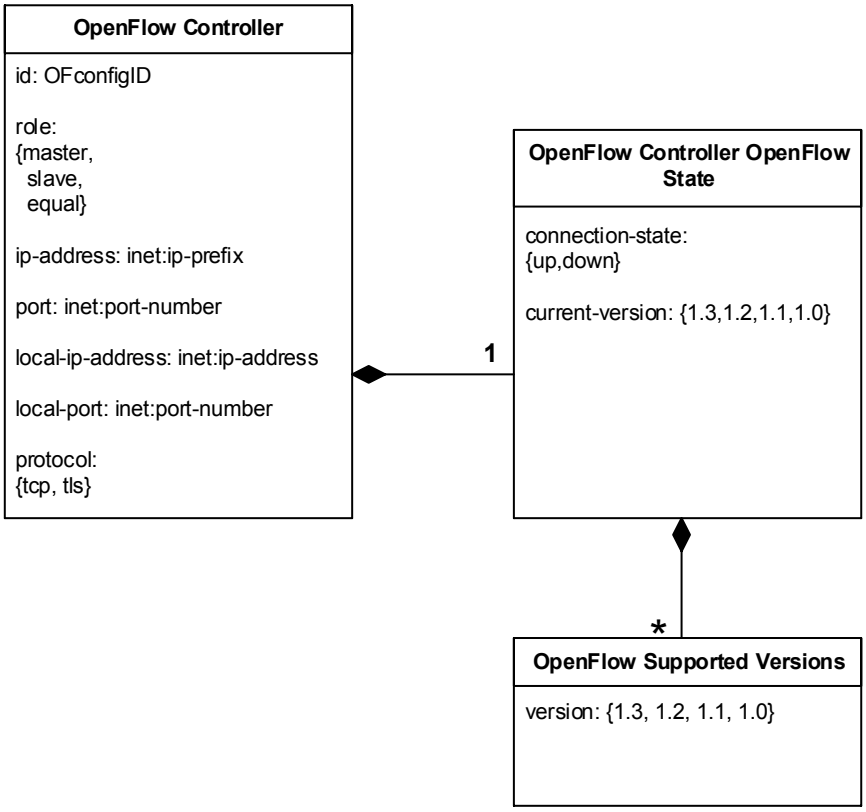


Figure 8: Data Model Diagram for an OpenFlow Controller

8.7.2 XML Example

```
<controller>
  <id>Controller3</id>
  <role>master</role>
  <ip-address>192.168.2.1/26</ip-address>
  <port>6633</port>
  <local-ip-address>192.168.2.129</local-ip-address>
  <local-port>32768</local-port>
  <protocol>tcp</protocol>
  <state>
    <connection-state>up</connection-state>
    <current-version>1.2</current-version>
    <supported-versions>
      <version>1.3</version>
      <version>1.0</version>
    </supported-versions>
  </state>
```

```
</controller>
```

8.8 OpenFlow Resource

OpenFlow Resource is a superclass of OpenFlow Port, OpenFlow Queue, Owned Certificate and External Certificate. The superclass contains the identifier attribute that is inherited by all subclasses in addition to their individual identifiers.

8.8.1 UML Diagram

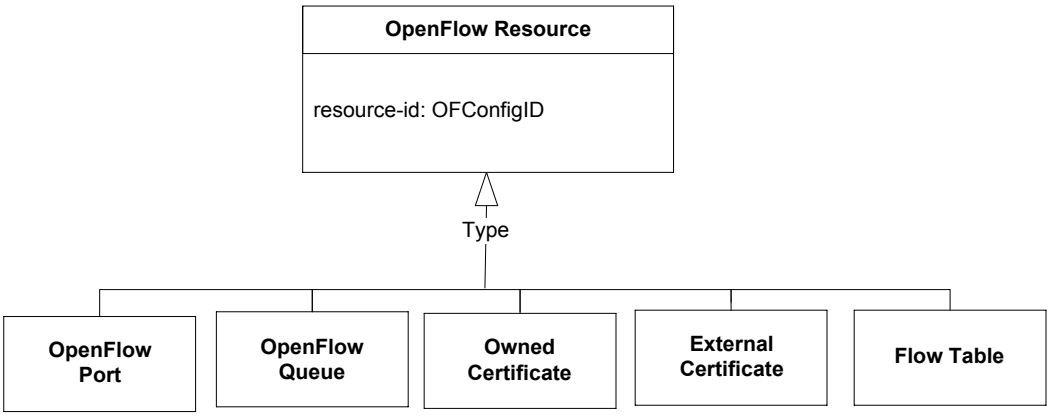


Figure 9: Data Model Diagram for an OpenFlow Resource

8.8.2 XML Example

The superclass is not instantiated.

8.9 OpenFlow Port

The OpenFlow Port is an instance of an OpenFlow resource. It may represent a physical port or a logical port. A logical port represents a tunnel endpoint as described in the OpenFlow protocol specification.

An OpenFlow Port contains a port configuration object and a port state object. A physical port contains a list of port feature objects. While there can't be more than one instance of the Port Configuration and the Port State, there may be multiple Port Features. In the case where a port represents a tunnel endpoint, then the port does not contain Port Feature objects, but a OpenFlow Tunnel object.

8.9.1 UML Diagram

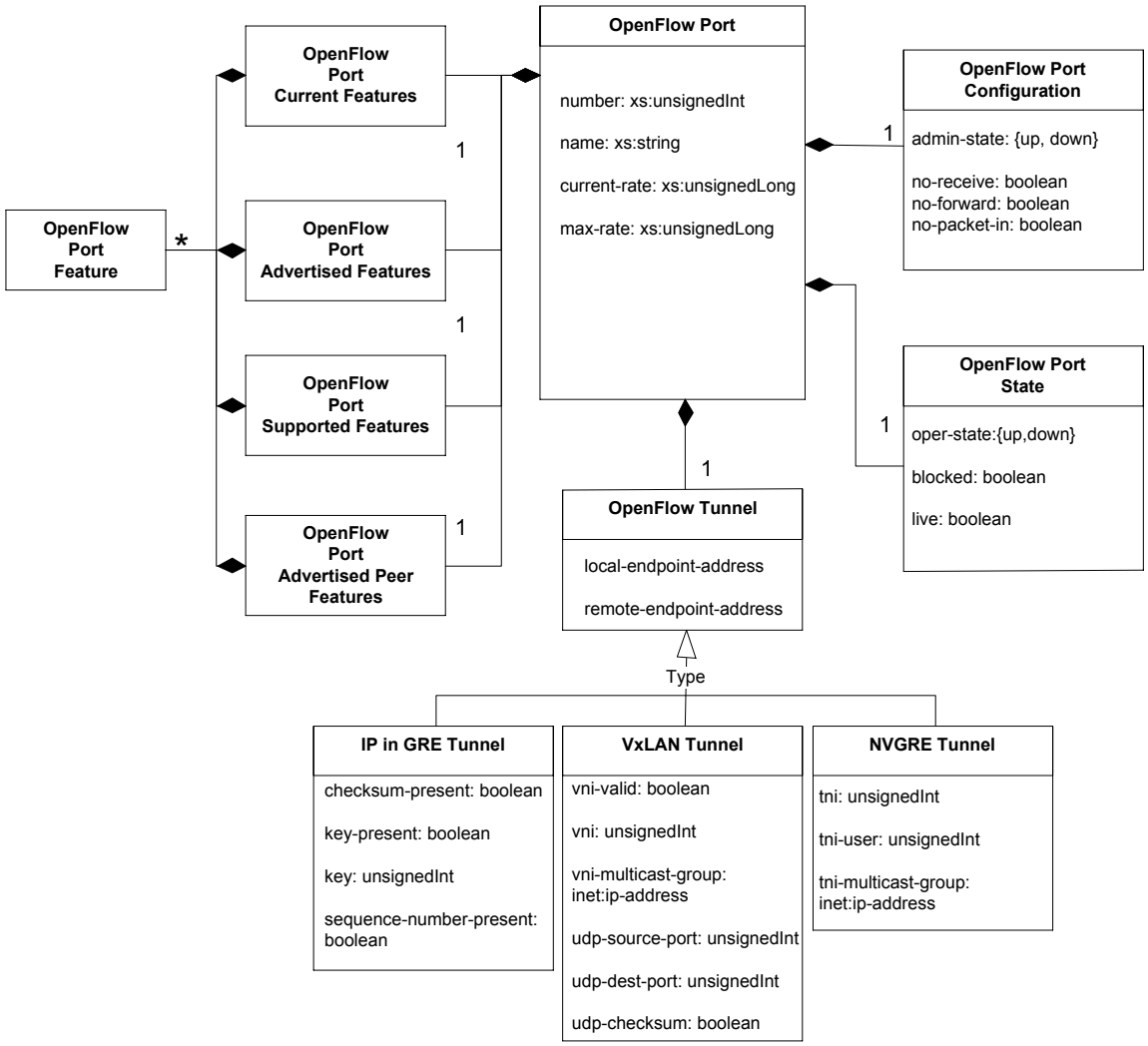


Figure 10: Data Model Diagram for an OpenFlow Port

8.9.2 XML Examples

```
<!-- Example for a physical port -->
<port>
  <resource-id>Port214748364</resource-id>
  <number>214748364</number>
  <name>name0</name>
  <current-rate>10000</current-rate>
  <max-rate>10000</max-rate>
  <configuration>
    <admin-state>up</admin-state>
    <no-receive>false</no-receive>
    <no-forward>false</no-forward>
```

```
    <no-packet-in>false</no-packet-in>
  </configuration>
  <state>
    <oper-state>up</oper-state>
    <blocked>false</blocked>
    <live>false</live>
  </state>
  <features>
    <current>
      ...
    </current>
    <advertised>
      ...
    </advertised>
    <supported>
      ...
    </supported>
    <advertised-peer>
      ...
    </advertised-peer>
  </features>
</port>

<!-- Example for a logical port representing a VxLAN tunnel -->
<port>
  <resource-id>LogicalPort14</resource-id>
  <number>14</number>
  <name>logicalPort14VxLAN</name>
  <max-rate>10000</max-rate>
  <configuration>
    <admin-state>up</admin-state>
    <no-receive>false</no-receive>
    <no-forward>false</no-forward>
    <no-packet-in>false</no-packet-in>
  </configuration>
  <state>
    <oper-state>up</oper-state>
    <blocked>false</blocked>
    <live>true</live>
  </state>
  <vxlan-tunnel>
    <local-endpoint-ipv4-address>
      192.0.2.9
    </local-endpoint-ipv4-address>
    <remote-endpoint-ipv4-address>
      192.0.2.112
    </remote-endpoint-ipv4-address>
    <vni-valid>true</vni-valid>
    <vni>15581985</vni>
    <udp-source-port>3804</udp-source-port>
    <udp-dest-port>4789</udp-dest-port>
    <udp-checksum>false</udp-checksum>
  </vxlan-tunnel>
</port>

<!-- Example for a logical port representing a NVGRE tunnel -->
```

```
<port>
  <resource-id>LogicalPort17</resource-id>
  <number>17</number>
  <name>logicalPort17NVGRE</name>
  <max-rate>1000</max-rate>
  <configuration>
    <admin-state>up</admin-state>
    <no-receive>false</no-receive>
    <no-forward>false</no-forward>
    <no-packet-in>false</no-packet-in>
  </configuration>
  <state>
    <oper-state>up</oper-state>
    <blocked>false</blocked>
    <live>true</live>
  </state>
  <nvgre-tunnel>
    <local-endpoint-ipv4-address>
      192.0.2.7
    </local-endpoint-ipv4-address>
    <remote-endpoint-ipv4-address>
      192.0.2.97
    </remote-endpoint-ipv4-address>
    <vsid>15581985</vsid>
    <flow-id>335</flow-id>
    </nvgre-tunnel>
</port>
```

8.10 OpenFlow Port Feature

OpenFlow Port Features include Port Rate, Port Medium, Port Pause, and Port Auto-Negotiate.The normative semantics of these features are described in the OpenFlow protocol specification.

8.10.1 UML Diagram

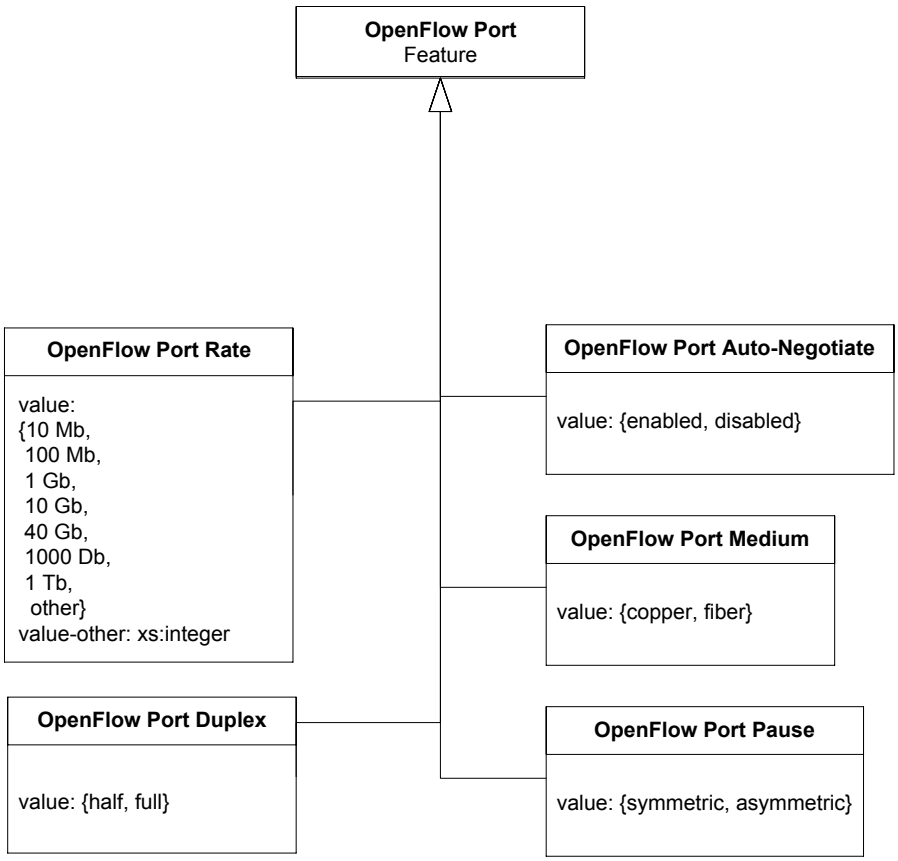


Figure 81: Data Model Diagram for an OpenFlow Port Feature

8.10.2 XML Example

```
<rate>10Mb-FD</rate>
<auto-negotiate>enabled</auto-negotiate>
<medium>copper</medium>
<pause>symmetric</pause>
```

8.11 OpenFlow Queue

The OpenFlow Queue is an instance of an OpenFlow resource. It contains list of queue properties. The OpenFlow Queue is a logical context which represents a queue as described in the OpenFlow protocol specification.

8.11.1 UML Diagram

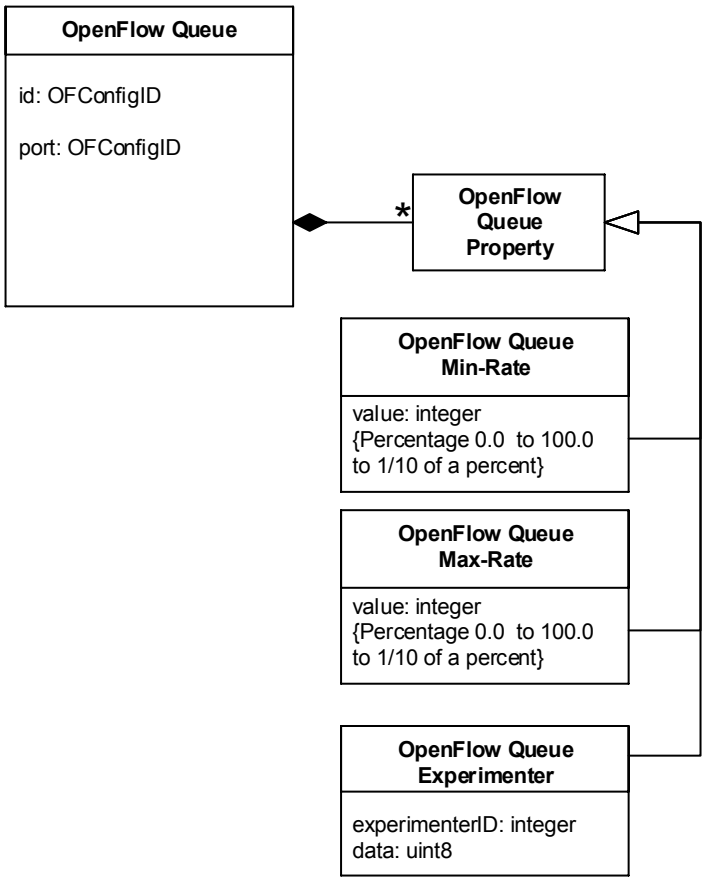


Figure 12: Data Model Diagram for an OpenFlow Queue

8.11.2 XML Example

```
<queue>
  <resource-id>Queue2</resource-id>
  <id>2</id>
  <port>4</port>
  <properties>
    <min-rate>10</min-rate>
    <max-rate>500</max-rate>
  </properties>
</queue>
```

8.12 External Certificate

Instances of an External Certificate contain a certificate that can be used by an OpenFlow Logical Switch for authenticating a controller when a TLS connection is established.

8.12.1 UML Diagram

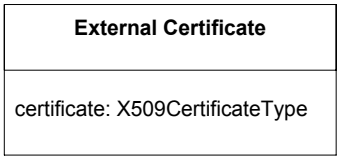


Figure 93: Data Model Diagram for a Certificate

8.12.2 XML Example

```
<external-certificate>
  <resource-id>ownedCertificate3</resource-id>
  <certificate>AEF134F56EDB667DFA4320AEF134F56EDB667DFA4320AEF134F
  56EDB667DFA4320AEF134F56EDB667DFA4320AEF134F56EDB667DFA4320
  ...
  AEF134F56EDB667DFA4320AEF134F56EDB667DFA4320AEF134F56EDB667
  DFA4320</certificate>
</external-certificate>
```

8.13 Owned Certificate

Instances of an Owned Certificate contain a certificate and a private key. It can be used by an OpenFlow Logical Switch for authenticating itself to a controller when a TLS connection is established.

8.13.1 UML Diagram

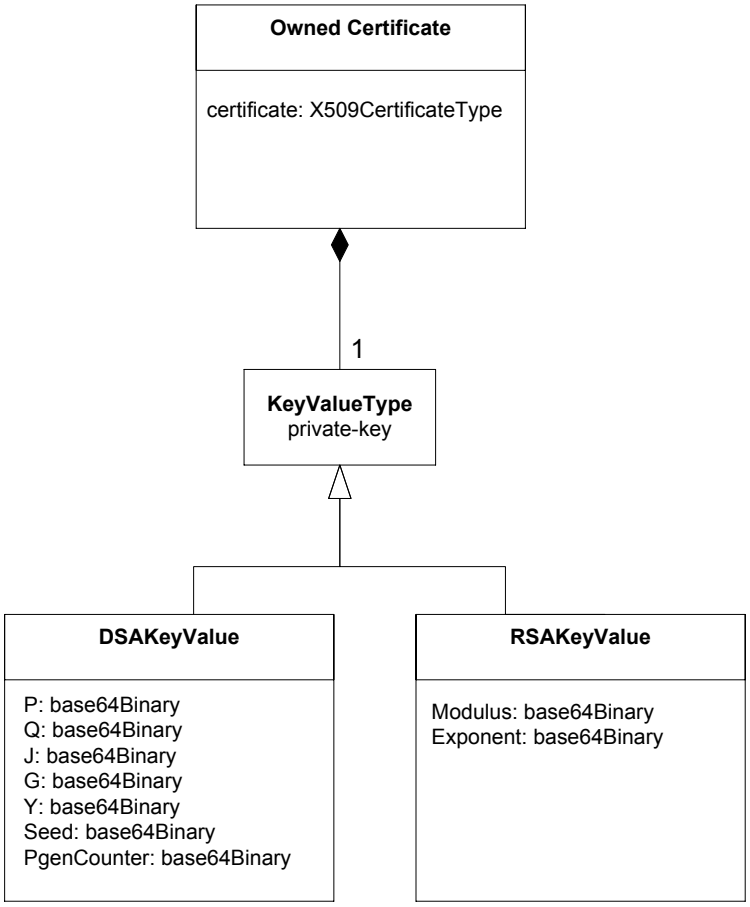


Figure 14: Data Model Diagram for Owned Certificate

8.13.2 XML Example

```
<owned-certificate>
  <resource-id>ownedCertificate3</resource-id>
  <certificate>AEF134F56EDB667DFA4320AEF134F56EDB667DFA4320AEF134F
  56EDB667DFA4320AEF134F56EDB667DFA4320AEF134F56EDB667DFA4320
  ...
  AEF134F56EDB667DFA4320AEF134F56EDB667DFA4320AEF134F56EDB667
  DFA4320</certificate>
  <private-key>
    <ds:RSAKeyValue>
      <ds:Modulus>CE45BAF6730F28CDB53534bC4323A333AAF555444DEED233232
      ...
      </ds:Modulus>
      <ds:Exponent>DFA4320AEF134F56EDB66786230900DFA3C6F4443234901234...
      </ds:Exponent>
  </private-key>
</owned-certificate>
```

8.14 OpenFlow Flow Table

The OpenFlow Flow Table is an instance of an OpenFlow resource. It contains list of flow table properties. The OpenFlow flow table is a logical context which represents a flow table as described in the OpenFlow protocol specification.

8.14.1 UML Diagram

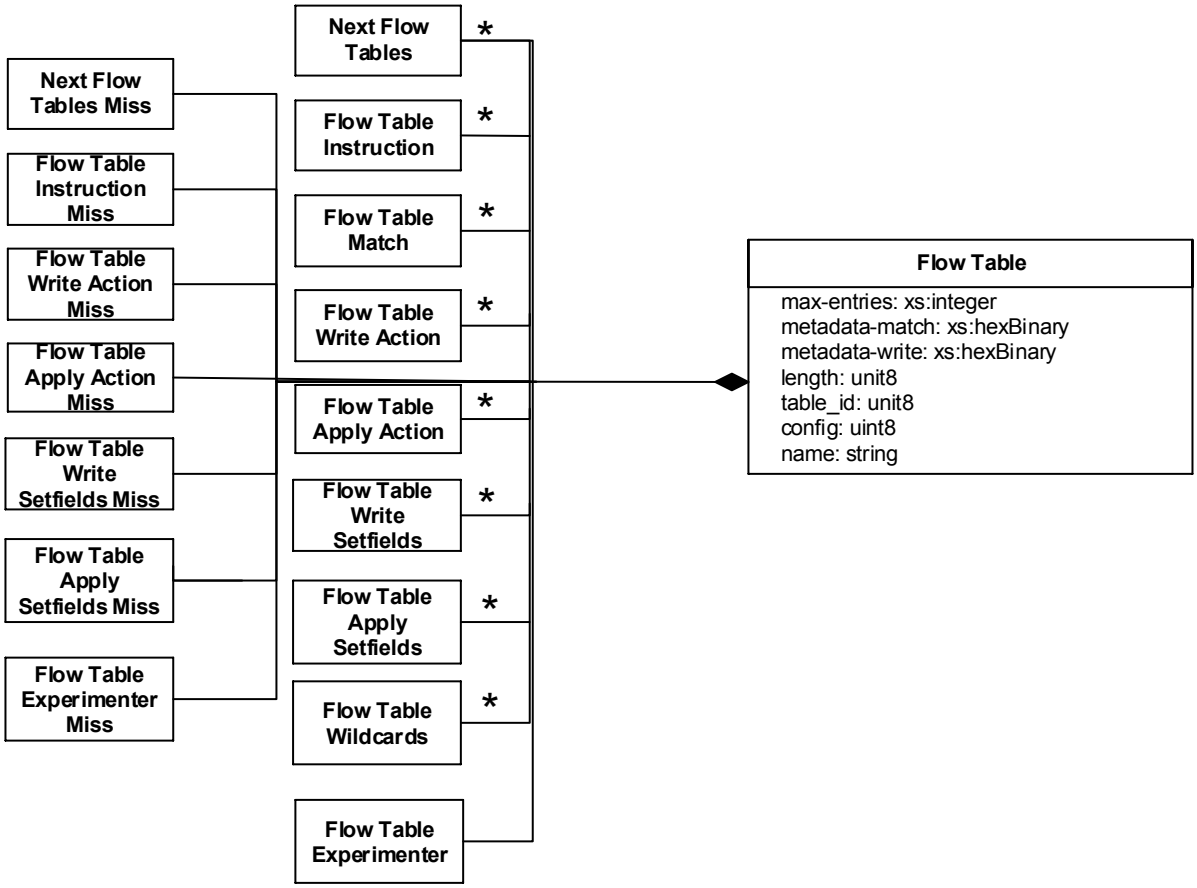


Figure 15: Data Model Diagram for Flow Table

8.14.2 XML Example

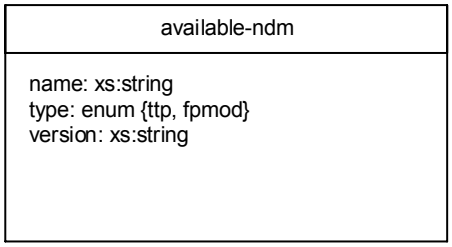
```
<flow-table>
  <resource-id>flowtable1</resource-id>
  <max-entries>255</max-entries>
  <next-tables>
    <table-id>100</table-id>
    <table-id>101</table-id>
  </next-tables>
  <instructions>
    <type>apply-actions</type>
    <type>clear-actions</type>
  </instructions>
  <matches>
    <type>input-port</type>
```

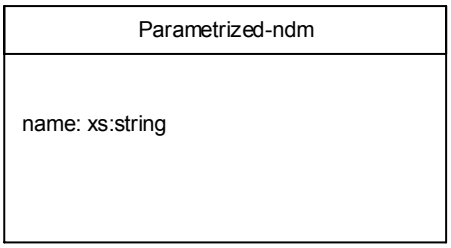
```
    <type>ethernet-dest</type>
  </matches>
  <write-actions>
    <type>output</type>
    <type>pop-mpls</type>
  </write-actions>
  <apply-actions>
    <type>output</type>
    <type>set-queue</type>
  </apply-actions>
  <write-setfields>
    <type>ethernet-dest</type>
  </write-setfields>
  <apply-setfields>
    <type>ethernet-dest</type>
  </apply-setfields>
  <wildcards>
    <type> udp-dest</type>
  </wildcards>
  <metadata-match>30</metadata-match>
</flow-table>
```

8.15 NDM

A Negotiable Datapath Model (NDM) is an abstract switch model that describes specific switch forwarding behaviors controllable via the OpenFlow-Switch protocol. When using the NDM framework (an optional enhancement to OpenFlow), an OFCP and a capable switch agree on an NDM to be associated with a logical switch prior to sending control messages, such as flowmods, to the logical switch. Note that Flow Table features described in Section 8.14 are generally not used with NDMs. Implementations are expected to extend the base XML schema with a set of NDM-specific data definitions (e.g. L2+L3). The details of the specific NDMs are outside the scope of this document. Refer to Appendix B in [5] for details.

8.15.1 UML Diagram





8.15.2 XML Example

This XML Example comes from a specific NDM supporting L2+L3 features as described in [XXX: reference to FAWG document.

```
<capable-switch xmlns="urn:onf:of111:config:yang"
                xmlns:ndm="urn:opennetworking.org:yang:ndm"
                xmlns:l2l3="urn:opennetworking.org:yang:ndm:l2l3">
  <logical-switches>
    <switch>
      <id>LogicalSwitch5</id>
      <resources>
        <ndm:ndm-implementation>
          <l2l3:l2l3>
            <l2l3:ingress-vlan-table-size>128</l2l3:ingress-vlan-table-
size>
            <l2l3:router-mac-table-size>128</l2l3:router-mac-table-size>
            <l2l3:l3-table-size>128</l2l3:l3-table-size>
            <l2l3:l2-table-size>128</l2l3:l2-table-size>
            <l2l3:egress-vlan-table-size>128</l2l3:egress-vlan-table-size>
          </l2l3:l2l3>
        </ndm:ndm-implementation>
      </resources>
    </switch>
  </logical-switches>
</capable-switch>
```

9 Binding to NETCONF

Below we specify the requirements and give examples of how the schema specified in section 8 and 2013 is bound to the NETCONF transport protocol.

9.1 Requirements

When implementing the XML schema defined in Section 8 and 2013 the following schemas are required in addition:

- ietf-yang-types.xsd found at http://www.yang-central.org/modules/xsd/ietf-yang-types.xsd
- ietf-inet-types.xsd found at http://www.cablelabs.com/specifications/XSD/ietf-inet-types.xsd

Those XML schemas define some basic datatypes that are used in the XML schema defined in this document.

A similar set is required when using the YANG model of Appendix B. There you need:

- ietf-yang-types.yang found at http://www.yang-central.org/modules/yang/ietf-yang-types.yang
- ietf-inet-types.yang found at http://www.yang-central.org/modules/yang/ietf-inet-types.yang

9.2 How the Data Model is Bound to NETCONF

NETCONF uses the XML encoding format for requests and responses. More specifically, it uses RPC-based communication model. It uses the `<rpc>` and `<rpc-reply>` elements as frames of NETCONF requests and responses. The content elements inside of `<rpc>` element must conform to the OpenFlow Configuraton XML schemas defined in this specification.

All NETCONF base protocol operations can be used to retrieve, configure, copy and delete OpenFlow Configuration data stores. These operations are defined in RFC6241. The commonly used operations are:

- edit-config
- get-config
- copy-config
- delete-config

9.2.1 edit-config

The `<edit-config>` operation loads all or part of a specified configuration to the specified target configuration. If the target configuration does not exist, it will be created. The “operation” attribute of elements in the `<config>` subtree specifies the type of operations to be performed on the element. NETCONF supports “create”, “replace”, “merge” and “delete”. The definition of these operations can be found RFC6241.

XML Example: Create a Capable-Switch Configuration

This XML example shows an edit-config operation to create a capable-switch configuration.

```
<?xmlversion="1.0" encoding="UTF-8"?>
<rpc message-id="1"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <target>
      <candidate/>
    </target>
    <default-operation>merge</default-operation>
    <test-option>set</test-option>
    <config>
      <capable-switch
        xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
        nc:operation="create"
        xmlns="urn:onf:of12:config:yang">
        <id>capable-switch-0</id>
        <logical-switches>
          <switch>
            <id>logic-switch-1</id>
            <datapath-id>11:11:11:11:11:11:11:11</datapath-id>
            <enabled>true</enabled>
            <controllers>
              <controller>
                <id>controller-0</id>
                <role>master</role>
                <ip-address>192.168.2.1</ip-address>
                <port>6633</port>
                <protocol>tcp</protocol>
              </controller>
            </controllers>
          </switch>
        </logical-switches>
      </capable-switch>
    </config>
  </edit-config>
</rpc>

<rpc-reply message-id="1"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

XML Example: Replace the ip-address Element of Controller

This XML example shows an edit-config operation to replace the `ip-address` element of controller.

```
<?xml version="1.0" encoding="UTF-8"?>
  <rpc message-id="1"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
    <target>
      <candidate/>
    </target>
    <default-operation>merge</default-operation>
    <config>
      <capable-switch xmlns="urn:onf:of12:config:yang">
        <logical-switches>
          <switch>
            <id>logic-switch-1</id>
            <controllers>
              <controller>
                <id>controller-0</id>
                <ip-address operation="replace">10.0.0.10</ip-address>
              </controller>
            </controllers>
          </switch>
        </logical-switches>
      </capable-switch>
    </config>
  </edit-config>
</rpc>

<rpc-reply message-id="1"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<ok/>
</rpc-reply>
```

RPC request must contain the key leave(s)(id element in this case) to uniquely identify the element being operated in the NETCONF datastore scope.

9.2.2 get-config

This operation is used to retrieve all or part of a specified configuration. The filter element identifies the portions of the OpenFlow configuration to retrieve. If this element is unspecified, the entire configuration is returned.

When issuing a NETCONF get request all elements in the requested sub-tree must be returned in the result. Those elements that can be modified by a NETCONF edit-config request or retrieved by a NETCONF get-config request are identified in the normative constraints which can be found in the description of each individual element.

XML Example: get-config

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <source>
      <running/>
    </source>
    <filter type="xpath" select="/capable-switch"/>
  </get-config>
</rpc>
```

```
<rpc-reply message-id="1"
   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <data>
     <capable-switch xmlns="urn:onf:of12:config:yang">
       <id>capable-switch-0</id>
       <logical-switches>
         <switch>
           <id>logic-switch-1</id>
           <datapath-id>11:11:11:11:11:11:11:11</datapath-id>
           <enabled>true</enabled>
           <controllers>
             <controller>
               <id>controller-0</id>
               <role>master</role>
               <ip-address>192.168.2.1</ip-address>
               <port>6633</port>
               <protocol>tcp</protocol>
             </controller>
           </controllers>
         </switch>
       </logical-switches>
     </capable-switch>
   </data>
</rpc-reply>
```

9.2.3 copy-config

This operation creates or replaces an entire configuration datastore with the contents of another complete configuration datastore. If the target datastore exists, it is overwritten. Otherwise, a new one is created, if allowed.

XML Example: copy-config

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1"
   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <copy-config>
     <target>
       <running/>
     </target>
     <source>
       <url>https://mydomain.com/of-config/new-config.xml</url>
     </source>
   </copy-config>
</rpc>


<rpc-reply message-id="1"
   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <ok/>
</rpc-reply>
```

9.2.4 delete-config

This operation deletes a configuration datastore. The <running>configuration datastore cannot be deleted.

XML Example: delete-config

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <delete-config>
     <target>
       <startup/>
     </target>
   </delete-config>
</rpc>


<rpc-reply message-id="1"
   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <ok/>
</rpc-reply>
```

9.3 RPC error

OpenFlow Configuration uses NETCONF <rpc-error> element(s) defined in RFC6241 to report operation failures. The <rpc-error> element(s) are sent in <rpc-reply> messages if an error occurs during the processing of an <rpc> request. The <rpc-reply> MAY contain multiple <rpc-error> elements. The <rpc-error>element includes the following information:

- error-type: Defines the conceptual layer of the error occurred.
- error-tag: contains a string to identifying the error condition.
- error-severity: contains a string to identifying the error severity.
- error-app-tag: contains a string to identifying the data-model-specific or implementation-specific error condition.
- error-path: contains the absolute XPath expression identifying the element path associated to the specific error being reported.
- error-message: contains error description suitable for human display
- error-info: contains data-model-specific error content

Detailed <rpc-error> definitions can be found in RFC 6241. Specific implementation may define implementation-specific error information and messages inside of error-info as sub-elements.

An example of <rpc-error> element in <rpc-reply> message:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101"
   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
```

```
   <rpc-error>
     <error-type>application</error-type>
     <error-tag> missing-element</error-tag>
     <error-severity>error</error-severity>
     <error-message xml:lang="en">
       expected key leaf in list
     </error-message>
     <error-info>
       <bad-element>id</bad-element>
       <error-number>383</error-number>
     </error-info>
   </rpc-error>
</rpc-reply>
```

Appendix A References

1. *OpenFlow Specification 1.3.* **Open Networking Foundation.** 2011.

2. *OpenFlow: enabling innovation in campus networks.* **McKeown, Nick, et al., et al.** 2008, ACM SIGCOMM Computer Communication Review, pp. 69-74.

3. **Bradner, S.** RFC 2119. *IETF.* [Online] March 1997. http://www.ietf.org/rfc/rfc2119.txt.

4. **Enns, et al., et al.** RFC 6241. *IETF.* [Online] June 2011. http://tools.ietf.org/rfc/rfc6241.txt.

5. OpenFlow Negotiable Datapath Models DRAFT v.0.6, available upon request, **Open Networking Foundation**, August 2013.

Appendix B Credits

Deepak Bansal, Stuart Bailey, Thomas Dietz, Carl Moberg, Juergen Quittek, Anantha Ramaiah, Anees Shaikh