OPEN NETWORKING
FOUNDATION

OpenFlow Management and Configuration Protocol (OF-Config 1.1.1)

Version 1.1.1

March 23, 2013

ONF TS-008

Contact: Deepak Bansal, Stuart Bailey, Thomas Dietz, Anees A. Shaikh

ONF Document Type: OpenFlow Config
ONF Document Name: of-config-1-1-1

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 1.1.1 of OF-CONFIG.

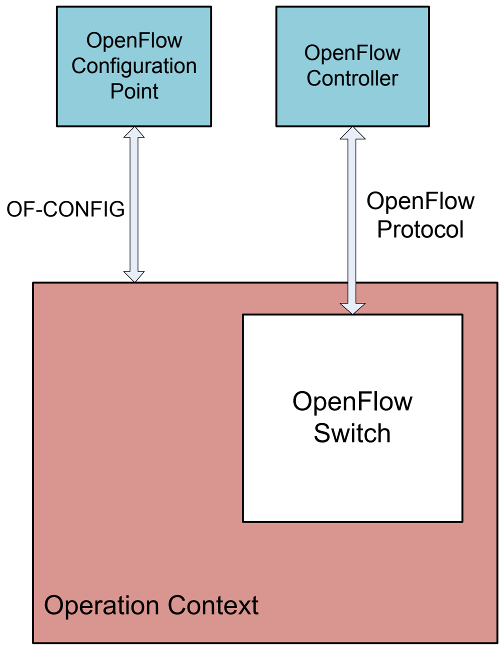

Figure 1: An OpenFlow Configuration Point communicates with an operational context which is capable of supporting an OpenFlow Switch using the OpenFlow Configuration and Management Protocol (OF-CONFIG)

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 switche via other methods (e.g. command line interfaces and other legacy management protocols). Future versions may better support other methods of change with detailed notification to the OpenFlow Configuration Point via OF-CONFIG.

2 Motivation

The OpenFlow protocol assumes that an OpenFlow datapath (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 datapaths. 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 does not need to be done at the timescale of a flow and hence, is done via OF-Config protocol.

OF-CONFIG frames an OpenFlow datapath 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 for one or more OpenFlow datapaths called an OpenFlow Capable Switch. 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 datapaths by partitioning a set of OpenFlow related resources such as ports and queues among the hosted OpenFlow datapaths. 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 amongst multiple OpenFlow Logical Switches such that each OpenFlow Logical Switch can assume full control over the resources that is assigned to it.

OF-CONFIG 1.1.1 makes simplifying assumptions about the architecture of OpenFlow switches. The specification is deliberately decoupled from whether the switch supports flowvisor or other virtualization models.

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 a service provided by software acting as an OpenFlow controller or it may by a service provided by a traditional network management framework. Any interaction between the OpenFlow Configuration Points and OpenFlow controllers is outside the scope of OF-CONFIG 1.1.1.

Figure 2 shows the basic abstractions detailed in OF-CONFIG 1.1.1 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 (i.e. datapaths) communicate via OpenFlow.

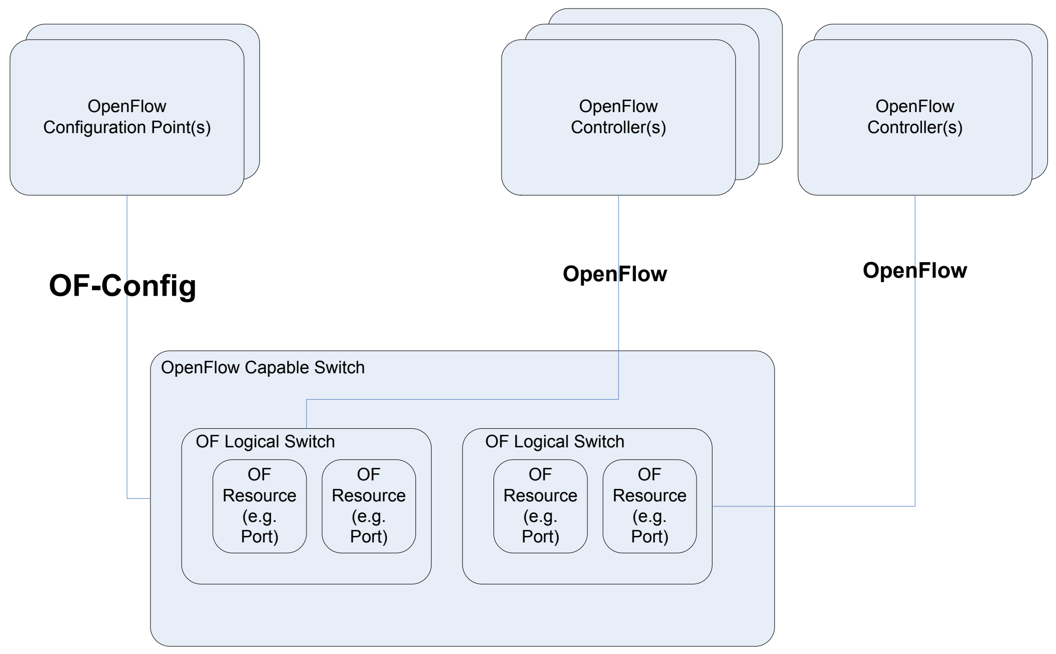

Figure 2: Relationship between components defined in this specification, the OF-CONFIG protocol and the OpenFlow protocol

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.

3 Scope

OF-CONFIG 1.1.1 is focused on the following functions needed to configure an OpenFlow 1.3 datapath:

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

Functionality introduced in OF-CONFIG 1.1.1 includes:

- Versioning Support

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

While limited in scope, OF-CONFIG 1.1.1 lays the foundation on top of which various automated and more advanced configurations will be possible in future revisions. Switch discovery, topology discovery, capability configuration, event triggers, instantiation of OpenFlow Logical Switches, assignment of resources such as ports and queues to OpenFlow Logical Switches, and bootstrap of the OpenFlow capable network are outside the scope of OF-CONFIG 1.1.1 protocol. These may be included in future versions.

Note that even though this specification refers to OpenFlow 1.3, OF-CONFIG 1.1.1 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 Datapath 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 datapath.

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 datapath. 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.

6 Requirements

This section describes requirements for the design of OF-CONFIG 1.1.1.

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 1.1.1 enables other configuration of OpenFlow switches. The specification of OF-CONFIG 1.1.1 is written with extensibility in mind. This includes versioning and backward compatibility.

6.1.1 Connection Setup to a Controller

Section 6.2 (Connection Setup) of (1) requires that an OpenFlow switch always 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:

- the IP address of the controller
- the port number at the controller
- the transport protocol to use, either TLS or TCP

OF-CONFIG 1.1.1 must provide means for configuring these parameters. Note that in future, alternative mechanisms for discovering the OpenFlow controller may be supported.

6.1.2 Multiple Controllers

Section 6.3 of (1) discusses how a switch deals with multiple controllers simultaneously. This implicitly requires OF-CONFIG 1.1.1 to provide means for configuring multiple instances of the parameter set listed in 6.1.1 for specifying the connection setup to multiple controllers.

6.1.3 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.

6.1.4 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.

6.1.5 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.

6.1.6 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 1.1.1 must provide means for configuring these parameters.

6.1.7 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 1.1.1. Section A.3.4.3 of (1) defines the following parameters for port configuration:

- no-recveive
- no-forward
- no-packetin
- admin-state

OF-CONFIG 1.1.1 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 1.1.1 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 1.1.1 must support the configuration of these logical ports. However, the configuration of logical ports in OF-CONFIG 1.1.1 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.

6.1.8 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 1.1.1, 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.

6.1.9 Datapath ID

Section A.3.1 of (1) discusses the datapath ID of a switch. It is a 64-bit filed 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 Operational Requirements

The OF-CONFIG 1.1.1 must meet support the following scenarios:

1. OF-CONFIG 1.1.1 must support an OpenFlow Capable Switch being configured by multiple OpenFlow Configuration Points.
2. OF-CONFIG 1.1.1 must support an OpenFlow Configuration Point managing multiple OpenFlow Capable Switches.
3. OF-CONFIG 1.1.1 must support an OpenFlow Logical Switch being controlled by multiple OpenFlow Controllers.
4. OF-CONFIG 1.1.1 must support configuring ports and queues of an OpenFlow Capable Switch that have been assigned to an OpenFlow Logical Switch.
5. OF-CONFIG 1.1.1 must support discovery of capabilities of an OpenFlow Logical Switch.
6. OF-CONFIG 1.1.1 must support configuration of tunnels such as IP-in-GRE, NVGRE and VxLan that are represented as logical ports of an OpenFlow Logical Switch.

6.3 Requirements for the Switch Management Protocol

OF-CONFIG 1.1.1 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 1.1.1, 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-CONFIG1.1.1 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-CONFIG1.1.1 requires that devices supporting OF-CONFIG1.1.1 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-CONFIG1.1.1 must implement SSH as a transport protocol. In addition, the OpenFlow Capable Switches implementing OF-CONFIG1.1.1 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-CONFIG1.1.1. NETCONF defines a set of operations on top of a messaging layer (RPC). The diagram below shows the various layers of the NETCONF protocol.

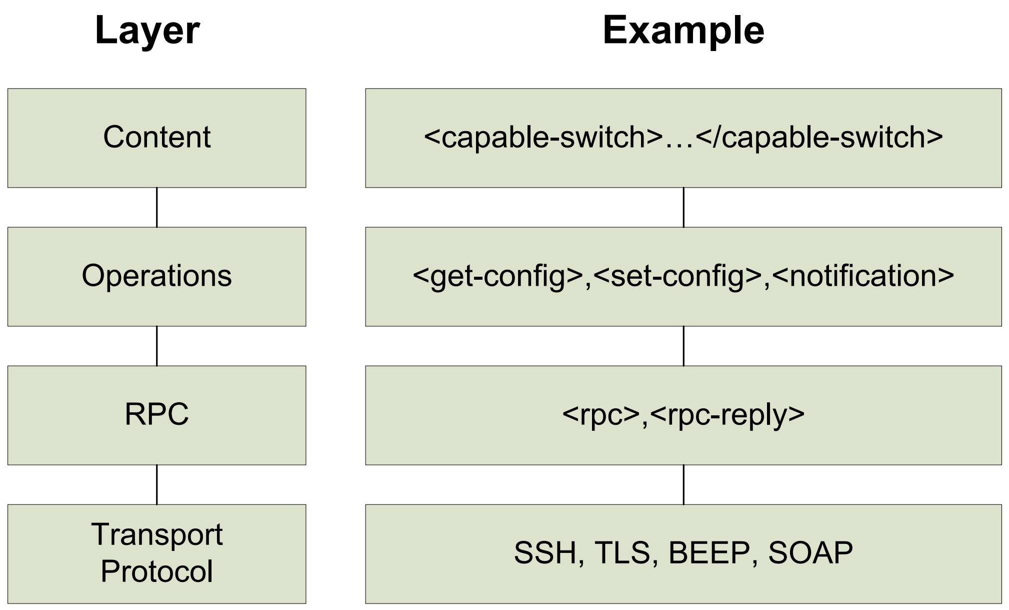

Figure 36 NETCONF Layers and Examples

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 OF-CONFIG 1.1.1 requirements for communication between an OpenFlow Configuration Point and an OpenFlow switch as listed in Section 6.3. 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 1.1.1. 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. a portion of an XML schema extracted from the normative XML schema in Appendix A, including normative constraints for instances of the class extending the XML schema by semantic specifications,
3. an example for XML code encoding an instance of the class

The full XML schema and the full YANG module are listed in Appendices A and B. Normative for OF-CONFIG 1.1.1 is the XML schema in Appendix A and the normative constraints in the descriptions of the individual elements. The YANG module in Appendix B incorporates the XML schema specifications as well as the normative constraints though is not normative for this specification.

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

In Appendix B you can find a YANG module that conforms to the normative constraints given in XML schema of Appendix A 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.

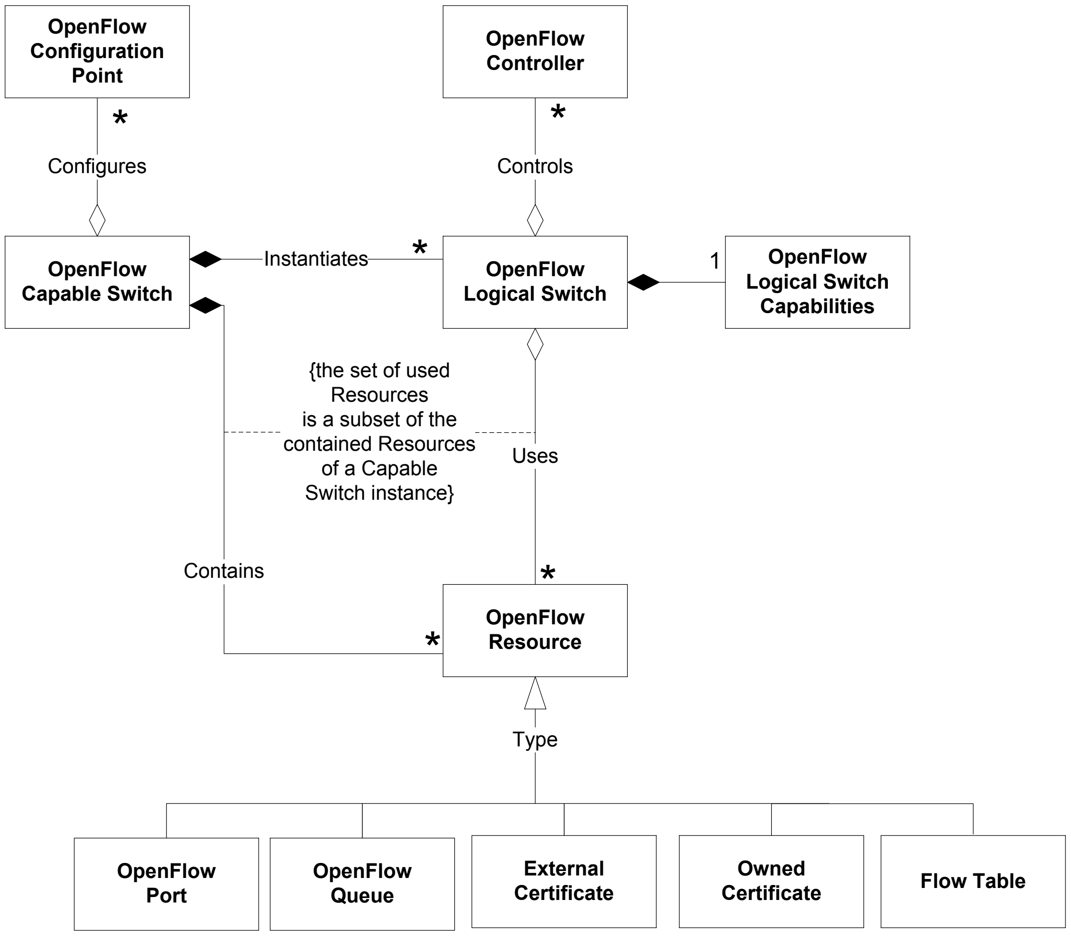

Figure 4: UML Class Diagram for OF-CONFIG Data Model

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 1.1.1, 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 1.1.1.

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

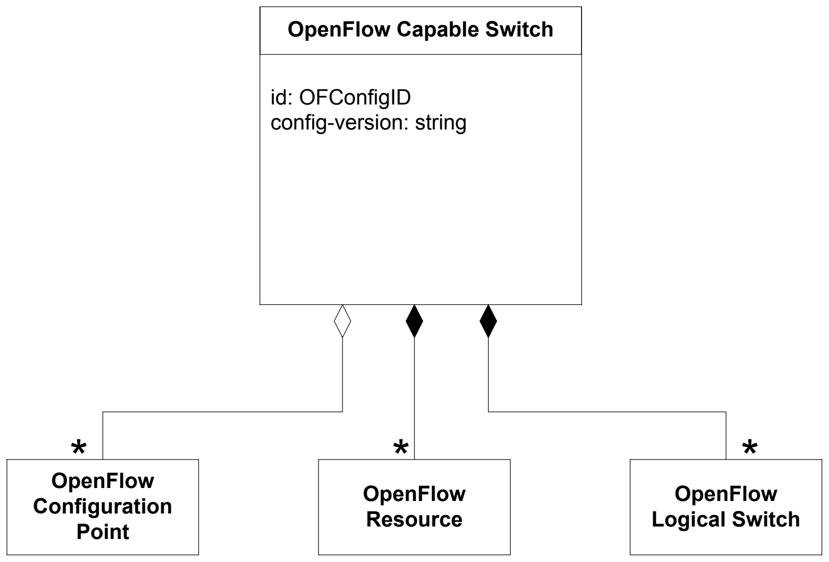

Figure 5: Data Model Diagram for OpenFlow Capable Switch

8.3.2 XML Schema

```
  <xs:element name="capable-switch">
    <xs:annotation>
      <xs:documentation>
        The OpenFlow Capable Switch serves as the root
        element for an OpenFlow configuration.  It contains OpenFlow logical
        switches and resources that can be assigned to logical
        switches.  It may have relations to OpenFlow Configuration
        Points.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="id"  type="inet:uri">
          <xs:annotation>
            <xs:documentation>
              A unique but locally arbitrary identifier that
              uniquely identifies a Capable Switch within the context of
              potential OpenFlow Configuration Points.  It MUST be
              persistent across reboots of the OpenFlow Capable Switch.

              This element MUST be present in the NETCONF data store.
              If this element is not present in a NETCONF &lt;edit-config&gt;
              operation 'create', 'merge' or 'replace' and the parent
              element does not exist, a 'data-missing' error is
              returned.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="config-version" minOccurs="0"  type="xs:string">
          <xs:annotation>
            <xs:documentation>
              The maximum supported OF-CONFIG version that is
              supported by the OpenFlow Capable Switch. For switches
              implementing this version of the OF-CONFIG protocol this
              MUST always be 1.1.1.

              This object can be used to identify the OF-CONFIG version
              a capable switch supports beginning with version 1.1.1 of
              OF-CONFIG. In addition the supported version can be
              determined by the namespace the OpenFlow Capable Switch
              returns to configuration request of an element (like
              capable-switch) that is present in all OF-CONFIG versions
              specified so far. This is the only possiblity to identify
              OF-CONFIG versions prior to OF-CONFIG 1.1.1.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="configuration-points" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="configuration-point" minOccurs="0"
 maxOccurs="unbounded">
                <xs:annotation>
```

```
                  <xs:documentation>
                    The list of all Configuration Points known to
                    the OpenFlow Capable Switch that may manage it using
                    OF-CONFIG.

                    The element 'id' of OFConfigurationType MUST be unique
                    within this list.
                  </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:group ref="OFConfigurationPointType"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
          <xs:key name="key_configuration-points_capable-
switch_configuration-point">
            <xs:selector xpath="of11-config:configuration-point"/>
            <xs:field xpath="of11-config:id"/>
          </xs:key>
        </xs:element>
        <xs:element name="resources" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              A lists containing all resources of the OpenFlow
              Capable Switch that can be used by OpenFlow Logical
              Switches.  Resources are listed here independent of their
              actual assignment to OpenFlow Logical Switches.  They may
              be available to be assigned to an OpenFlow Logical Switch
              or already in use by an OpenFlow Logical Switch.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="port" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>
                    The list contains all port resources of the
                    OpenFlow Capable Switch.

                    The element 'resource-id' of OFPortType MUST be unique
                    within this list.
                  </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:group ref="OFPortType"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="queue" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>
                    The list contains all queue resources of the
                    OpenFlow Capable Switch.

                    The element 'resource-id' of OFQueueType MUST be unique
                    within this list.
                  </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:group ref="OFQueueType"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="owned-certificate" minOccurs="0"
maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>
                    The list contains all owned certificate
                    resources of the OpenFlow Capable Switch.

                    The element 'resource-id' of OFOwnedCertificateType MUST
                    be unique within this list.
                  </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:group ref="OFOwnedCertificateType"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="external-certificate" minOccurs="0"
maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>
                    The list contains all external certificate
                    resources of the OpenFlow Capable Switch.

                    The element 'resource-id' of OFExternalCertificateType
                    MUST be unique within this list.
                  </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:group ref="OFExternalCertificateType"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="flow-table" minOccurs="0"
maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>
                    The list contains all flow table resources of
                    the OpenFlow Capable Switch.

                    The element 'resource-id' of OFFlowTableType MUST be
                    unique within this list.
                  </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:group ref="OFFlowTableType"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
          <xs:key name="key_resources_capable-switch_port">
            <xs:selector xpath="of11-config:port"/>
            <xs:field xpath="of11-config:resource-id"/>
          </xs:key>
          <xs:key name="key_resources_capable-switch_queue">
            <xs:selector xpath="of11-config:queue"/>
            <xs:field xpath="of11-config:resource-id"/>
          </xs:key>
          <xs:key name="key_resources_capable-switch_owned-certificate">
            <xs:selector xpath="of11-config:owned-certificate"/>
            <xs:field xpath="of11-config:resource-id"/>
          </xs:key>
          <xs:key name="key_resources_capable-switch_external-certificate">
            <xs:selector xpath="of11-config:external-certificate"/>
            <xs:field xpath="of11-config:resource-id"/>
          </xs:key>
          <xs:key name="key_resources_capable-switch_flow-table">
            <xs:selector xpath="of11-config:flow-table"/>
            <xs:field xpath="of11-config:resource-id"/>
          </xs:key>
        </xs:element>
        <xs:element name="logical-switches" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              This element contains a list of all OpenFlow
              Logical Switches available at the OpenFlow Capable
              Switch.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="switch" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>
                    The list of all OpenFlow Logical Switches on
                    the OpenFlow Capable Switch.

                    The element 'resource-id' of OFLogicalSwitchType MUST be
                    unique within this list.
                  </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:group ref="OFLogicalSwitchType"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
          <xs:key name="key_logical-switches_capable-switch_switch">
            <xs:selector xpath="of11-config:switch"/>
            <xs:field xpath="of11-config:id"/>
          </xs:key>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

8.3.3 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

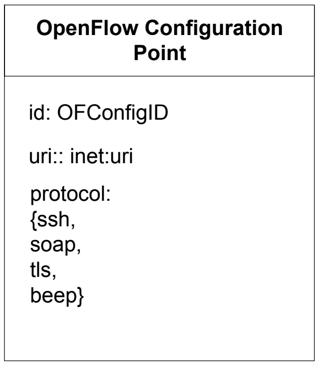

Figure 6: Data Model Diagram for an OpenFlow Configuration Point

8.4.2 XML Schema

```
  <xs:group name="OFConfigurationPointType">
    <xs:annotation>
      <xs:documentation>
        Representation of an OpenFlow Configuration Point.
        Instances of the Configuration Point class SHOULD be stored
        persistently across reboots of the OpenFlow Capable Switch.

        When a connection is established between an OpenFlow Capable
        Switch and a Configuration Point the switch  MUST store the
        connection information in an instance of the Configuration
        Point class. If such an instance does not exist, the OpenFlow
        Capable Switch MUST create an instance where it then stores
        the connection information.

        An OpenFlow Capable Switch that cannot initiate a connection
        to a configuration point does not have to implement the
        Configuration Point class. It SHOULD block attempts to write
        to instances of the Configuration Point class with NETCONF
        &lt;edit-config&gt; operations.

        NETCONF &lt;edit-config&gt; operations MUST be implemented as
        follows:

        * The 'id' element MUST be present at all &lt;edit-config&gt;
        operations to identify the configuration point.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="id"  type="OFConfigId">
        <xs:annotation>
          <xs:documentation>
            A unique but locally arbitrary identifier that
            identifies a Configuration Point within the context of an
            OpenFlow Capable Switch.

            This element MUST be present to identify the configuration
            point.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="uri"  type="inet:uri">
        <xs:annotation>
          <xs:documentation>
            A locator of the Configuration Point.  It
            identifies the location of the Configuration Point as a
            service resource and MUST include all information necessary
            for the OpenFlow Capable Switch to connect to the
            Configuration Point or re-connect to it should it become
            disconnected.  Such information MAY include, for example,
            protocol, fully qualified domain name, IP address, port
            number, etc.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="protocol"  type="OFConfigurationPointProtocolType">
        <xs:annotation>
          <xs:documentation>
            The transport protocol that the Configuration
            Point uses when communicating via NETCONF with the OpenFlow
            Capable Switch.

            This element is optional. If it is not present its value
            defaults to 'ssh'.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:simpleType name="OFConfigurationPointProtocolType">
    <xs:annotation>
      <xs:documentation>
        Possible protocols to connect ot an OF
        Configuration Point
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="ssh"/>
      <xs:enumeration value="soap"/>
      <xs:enumeration value="tls"/>
      <xs:enumeration value="beep"/>
    </xs:restriction>
  </xs:simpleType>
```

8.4.3 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 instant 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

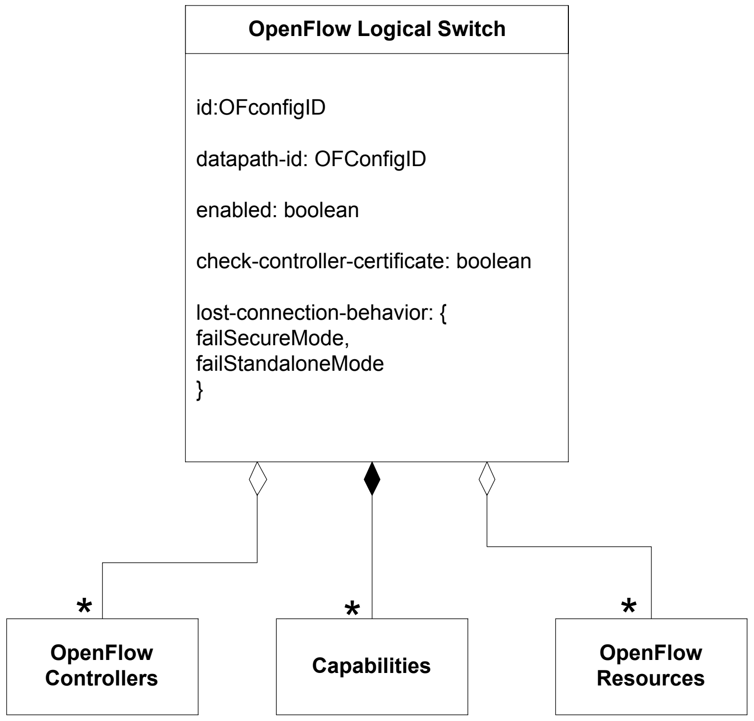

Figure 7: Data Model Diagram for an OpenFlow Logical Switch

8.5.2 XML Schema

```
  <xs:group name="OFLogicalSwitchType">
    <xs:annotation>
      <xs:documentation>
        This grouping specifies all properties of an
        OpenFlow Logical Switch.

        Elements of type OFLogicalSwitchType cannot be created or
        deleted with NETCONF &lt;edit-config&gt; operations 'create' or
        'delete'. The other NETCONF &lt;edit-config&gt; operations MUST be
        implemented as follows:

        * The 'id' element MUST be present at all &lt;edit-config&gt;
        operations to identify the OpenFlow Logical Switch.
        * If the operation is 'merge' or 'replace', and the element
        does not exist, a 'data-missing' error is returned. If the
        element exists its value is set to the value found in the
        XML RPC data.
        * If the operation is 'create', a 'operation-not-supported'
        error with type 'application' is returned.
        * If the operation is 'delete', 'operation-not-supported'
        error with type 'application' is returned.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="id"  type="OFConfigId">
```

```
        <xs:annotation>
          <xs:documentation>
            A unique but locally arbitrary identifier that
            identifies a Logical Switch within the context of an
            OpenFlow Capable Switch. It MUST be persistent across
            reboots of the OpenFlow Capable Switch.

            This element MUST be present to identify the OpenFlow
            Logical Switch.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="capabilities">
        <xs:annotation>
          <xs:documentation>
            This element contains all capability items that
            an OpenFlow Logical Switch MAY implement.

            This element and its children can only be retrieved by
            NETCONF &lt;get&gt; operation since it contain no configuration
            data.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:group ref="OFLogicalSwitchCapabilitiesType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="datapath-id"  type="datapath-id-type">
        <xs:annotation>
          <xs:documentation>
            The datapath identifier of the Logical Switch
            that uniquely identifies this Logical Switch within the
            context of all OpenFlow Controllers associated with the
            OpenFlow Logical Switch.  The datapath identifier is a
            string value that MUST be formatted as a sequence of 8
            2-digit hexadecimal numbers that are separated by colons,
            for example, '01:23:45:67:89:ab:cd:ef'.  When processing a
            datapath identifier, the case of the decimal digits MUST be
            ignored.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="enabled"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            This element indicates the administrative state
            of the OpenFlow Logical Switch.  A value of 'false' means
            the OpenFlow Logical Switch MUST NOT communicate with any
            OpenFlow Controllers, MUST NOT conduct any OpenFlow
            processing, and SHOULD NOT be utilizing computational or
            network resources of the underlying platform.

            This element is optional. If this element is not present it
            defaults to 'false'.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="check-controller-certificate"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            This element indicates the behavior of the
            OpenFlow Logical Switch when connecting to an OpenFlow
            Controller.

            If set to value 'false', the logical switch will connect to
            a controller without checking any controller certificate.

            If set to value 'true', then the logical switch will
            connect to a controller with element &lt;protocol&gt; set to
            'TLS', only if the controller provides a certificate that
            can be verified with one of the certificates stored in the
            list called external-certificates in the OpenFlow Capable
            Switch.

            If a certificate cannot be validated, the OpenFlow Logical
            Switch MUST terminate communication with the corresponding
            OpenFlow Controller, MUST NOT conduct any OpenFlow
            processing on requests of this OpenFlow controller, and
            SHOULD NOT further utilize any computational or network
            resources of for dealing with this connection.

            If set to value 'true', the OpenFlow Logical Switch MUST
            NOT connect to any OpenFlow Controller that does not
            provide a certificate. This implies that it cannot connect
            to an OpenFlow controller that has the value of element
            protocol set to 'TCP'. Only connections with protocol 'TLS'
            are possible in this case.

            This element is optional. If this element is not present it
            defaults to 'false'.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="lost-connection-behavior">
        <xs:annotation>
          <xs:documentation>
            This element indicates the the behavior of the
            OpenFlow Logical Switch in case it loses contact with all
            OpenFlow Controllers.  There are two alternative modes in
            such a case: fails secure mode and fail standalone mode as
            defined by the OpenFlow protocol specification version 1.2,
            section 6.4.  These are the only allowed values for this
            element. Default is the fail secure mode.
```

```
            This element is optional. If this element is not present it
            defaults to 'failSecureMode'.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="failSecureMode"/>
            <xs:enumeration value="failStandaloneMode"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="controllers">
        <xs:annotation>
          <xs:documentation>
            The list of controllers for this Logical switch.

            The element 'id' of OFControllerType MUST be unique within
            this list.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="controller" minOccurs="0"
maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
                  The list of OpenFlow Controllers that are
                  assigned to the OpenFlow Logical Switch.  The switch MUST
                  NOT connect to any OpenFlow Controller that is not
                  contained in this list.

                  NETCONF &lt;edit-config&gt; operations MUST be implemented
as
                  follows:

                  * The 'id' element MUST be present at all &lt;edit-
config&gt;
                  operations to identify the controller.
                  * If the operation is 'merge' or 'replace', the element
                  is created if it does not exist, and its value is set
                  to the value found in the XML RPC data.
                  * If the operation is 'create', the element is created if
                  it does not exist. If the element already exists, a
                  'data-exists' error is returned.
                  * If the operation is 'delete', the element is deleted if
                  it exists. If the element does not exist, a
                  'data-missing' error is returned.
                </xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:group ref="OFControllerType"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
        <xs:key name="key_controllers_controller">
          <xs:selector xpath="of11-config:controller"/>
          <xs:field xpath="of11-config:id"/>
        </xs:key>
      </xs:element>
      <xs:element name="resources">
        <xs:annotation>
          <xs:documentation>
            The list of identifiers of all resources of the
            OpenFlow Capable Switch that the OpenFlow Logical Switch
            has exclusive or non-exclusive access to.  A resource is
            identified by the value of its resource-identifier element.
            For each resource identifier value in this list, there MUST
            be an element with a matching resource identifier value in
            the resources list of the OpenFlow Capable Switch.

            Identifiers of this list are contained in elements
            indicating the type of resource: 'port', 'queue',
            'certificate', or 'flow-table'.  Depending on the type,
            different constraints apply.  These are specified in
            separate descriptions per type.

            At present the elements in this lists are not configurable
            and can only be retrieved by NETCONF &lt;get&gt; or &lt;get-
  config&gt;
            operations. Attemps to modify this element and its children
            with a NETCONF &lt;edit-config&gt; operation MUST result in an
            'operation-not-supported' error with type 'application'.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="port" minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
                  A resource identifier of a port of the
                  OpenFlow Capable Switch that the OpenFlow Logical Switch
                  has exclusive access to.

                  The elements in this list MUST refer to elements at the
                  following path:
                  /capable-switch/resources/port/resource-id

                  Elements in this list MUST be unique. This means each
                  port element can only be referenced once.
                </xs:documentation>
              </xs:annotation>
              <xs:simpleType>
                <xs:restriction base="inet:uri">
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="queue" minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
```

```
                  A resource identifier of a queue of the
                  OpenFlow Capable Switch that the OpenFlow Logical Switch
                  has exclusive access to.

                  The elements in this list MUST refer to elements at the
                  following path:
                  /capable-switch/resources/queue/resource-id

                  Elements in this list MUST be unique. This means each
                  queue element can only be referenced once.
                </xs:documentation>
              </xs:annotation>
              <xs:simpleType>
                <xs:restriction base="inet:uri">
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="certificate" minOccurs="0">
              <xs:annotation>
                <xs:documentation>
                  The resource identifier of the owned
                  certificate in the OpenFlow Capable Switch that the
                  OpenFlow Logical Switch uses to identify itself.  This
                  element MUST NOT occur more than once in an OpenFlow
                  Logical Switch's resource list.

                  If no such element is in an OpenFlow Logical Switch's
                  resource list, then the OpenFlow Logical Switch does not
                  authenticate itself towards an OpenFloe Controller with a
                  certificate.  If this element is present, then the
                  OpenFlow Logical Switch MUST provide this certificate for
                  authentication to an OpenFlow Controller when setting up
                  a TLS connection.

                  For TCP connections this element is irrelevant.

                  The element MUST refer to an element at the following
                  path:
                  /capable-switch/resources/owned-certificate/resource-id

                </xs:documentation>
              </xs:annotation>
              <xs:simpleType>
                <xs:restriction base="inet:uri">
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="flow-table" minOccurs="0"
 maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
                  A resource identifier of a flow table of the
                  OpenFlow Capable Switch that the OpenFlow Logical Switch
                  has exclusive access to.

                  The elements in this list MUST refer to elements at the
                  following path:
                  /capable-switch/resources/flow-table/resource-id

                  Elements in this list MUST be unique. This means each
                  flow-table element can only be referenced once.
                </xs:documentation>
              </xs:annotation>
              <xs:simpleType>
                <xs:restriction base="inet:uri">
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:simpleType name="datapath-id-type">
    <xs:annotation>
      <xs:documentation>
        The datapath-id type represents an OpenFlow
        datapath identifier.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
    <xs:pattern value="[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){7}"/>
    </xs:restriction>
  </xs:simpleType>
```

8.5.3 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

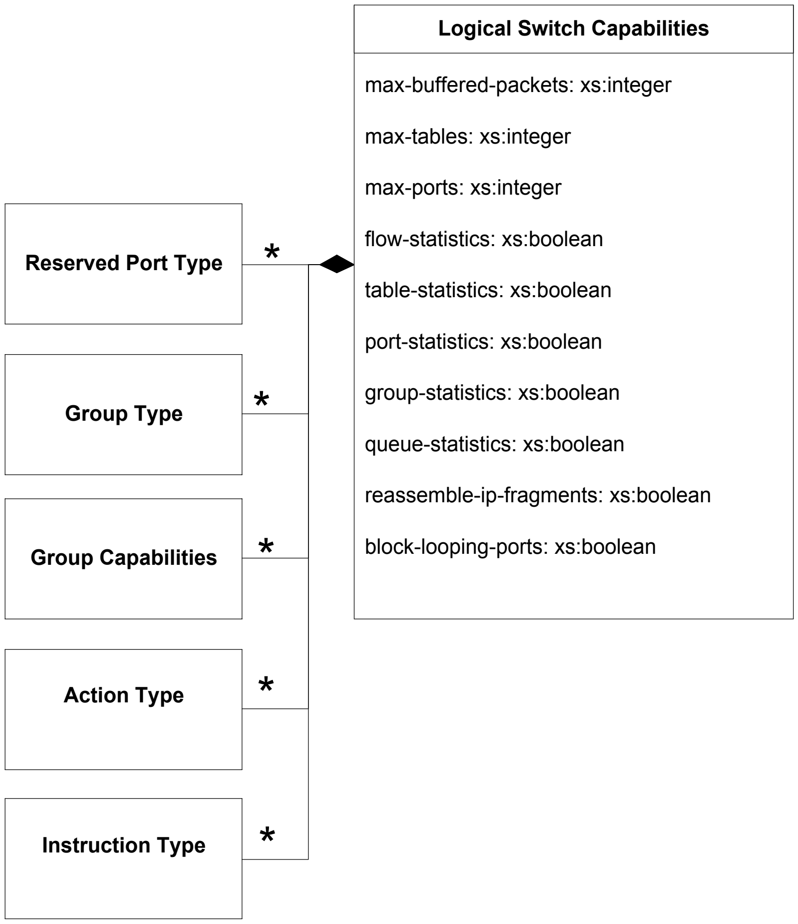
Logical Switch Capabilities
max-buffered-packets: xs:integer
max-tables: xs:integer
max-ports: xs:integer
flow-statistics: xs:boolean
table-statistics: xs:boolean
port-statistics: xs:boolean
group-statistics: xs:boolean
queue-statistics: xs:boolean
reassemble-ip-fragments: xs:boolean
block-looping-ports: xs:boolean
Reserved Port Type
Group Type
Group Capabilities
Action Type
Instruction Type
*
*
*
*
*

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

8.6.2 XML Schema

```
  <xs:group name="OFLogicalSwitchCapabilitiesType">
    <xs:annotation>
      <xs:documentation>
        This grouping specifies all properties of an
        OpenFlow logical switch's capabilities.

        Elements in the type OFLogicalSwitchCapabilitiesType are not
        configurable and can only be retrieved by NETCONF &lt;get&gt;
        operations. Attemps to modify this element and its children
        with a NETCONF &lt;edit-config&gt; operation MUST result in an
        'operation-not-supported' error with type 'application'.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="max-buffered-packets"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            The maximum number of packets the logical switch
            can buffer when sending packets to the controller using
            packet-in messages.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="max-tables"  type="xs:unsignedByte">
        <xs:annotation>
          <xs:documentation>
            The number of flow tables supported by the
            logical switch.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="max-ports"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            The number of flow tables supported by the
            logical switch.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="flow-statistics"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Specifies if the logical switch supports flow
            statistics.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="table-statistics"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Specifies if the logical switch supports table
            statistics.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="port-statistics"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Specifies if the logical switch supports port
            statistics.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="group-statistics"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
```

```
            Specifies if the logical switch supports group
            statistics.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="queue-statistics"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Specifies if the logical switch supports queue
            statistics.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="reassemble-ip-fragments"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Specifies if the logical switch supports
            reassemble IP fragments.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="block-looping-ports"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            'true' indicates that a switch protocol outside
            of OpenFlow, such as 802.1D Spanning Tree, will detect
            topology loops and block ports to prevent packet loops.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="reserved-port-types">
        <xs:annotation>
          <xs:documentation>
            Specify generic forwarding actions such as
            sending to the controller, flooding, or forwarding using
            non-OpenFlow methods, such as 'normal' switch processing.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="all"/>
                  <xs:enumeration value="controller"/>
                  <xs:enumeration value="table"/>
                  <xs:enumeration value="inport"/>
                  <xs:enumeration value="any"/>
                  <xs:enumeration value="normal"/>
                  <xs:enumeration value="flood"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="group-types">
        <xs:annotation>
          <xs:documentation>
            Specify the group types supported by the logical
            switch.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="all"/>
                  <xs:enumeration value="select"/>
                  <xs:enumeration value="indirect"/>
                  <xs:enumeration value="fast-failover"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="group-capabilities">
        <xs:annotation>
          <xs:documentation>
            Specify the group capabilities supported by the
            logical switch.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="capability" minOccurs="0"
maxOccurs="unbounded">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="select-weight"/>
                  <xs:enumeration value="select-liveness"/>
                  <xs:enumeration value="chaining"/>
                  <xs:enumeration value="chaining-check"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="action-types">
        <xs:annotation>
          <xs:documentation>
            Specify the action types supported by the
            logical switch.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
type="OFActionType"/>
```

```
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="instruction-types">
        <xs:annotation>
          <xs:documentation>
            Specify the instruction types supported by the
            logical switch.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
  type="OFInstructionType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>

<xs:simpleType name="OFActionType">
    <xs:annotation>
      <xs:documentation>
        The types of actions defined in OpenFlow Switch
        Specification versions 1.2, 1.3, and 1.3.1
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:enumeration value="output"/>
      <xs:enumeration value="copy-ttl-out"/>
      <xs:enumeration value="copy-ttl-in"/>
      <xs:enumeration value="set-mpls-ttl"/>
      <xs:enumeration value="dec-mpls-ttl"/>
      <xs:enumeration value="push-vlan"/>
      <xs:enumeration value="pop-vlan"/>
      <xs:enumeration value="push-mpls"/>
      <xs:enumeration value="pop-mpls"/>
      <xs:enumeration value="set-queue"/>
      <xs:enumeration value="group"/>
      <xs:enumeration value="set-nw-ttl"/>
      <xs:enumeration value="dec-nw-ttl"/>
      <xs:enumeration value="set-field"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="OFInstructionType">
    <xs:annotation>
      <xs:documentation>
        The types of instructions defined in OpenFlow
        Switch Specification versions 1.2, 1.3, and 1.3.1.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:enumeration value="apply-actions"/>
      <xs:enumeration value="clear-actions"/>
      <xs:enumeration value="write-actions"/>
      <xs:enumeration value="write-metadata"/>
      <xs:enumeration value="goto-table"/>
    </xs:restriction>
  </xs:simpleType>
```

8.6.3 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.

8.7.1 UML Diagram

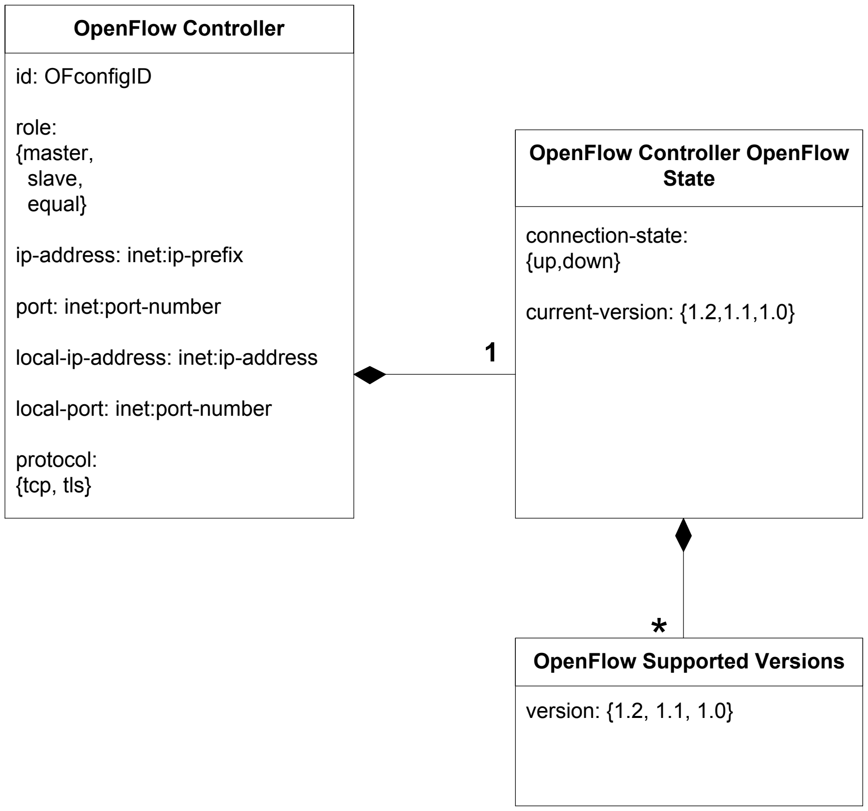
OpenFlow Controller
id: OFconfigID
role:
{master,
slave,
equal}
ip-address: inet:ip-prefix
port: inet:port-number
local-ip-address: inet:ip-address
local-port: inet:port-number
protocol:
{tcp, tls}
1
OpenFlow Controller OpenFlow State
connection-state:
{up,down}
current-version: {1.2,1.1,1.0}
*
OpenFlow Supported Versions
version: {1.2, 1.1, 1.0}

Figure 8: Data Model Diagram for an OpenFlow Controller

8.7.2 XML Schema

```
  <xs:group name="OFControllerType">
    <xs:annotation>
      <xs:documentation>
        This grouping specifies all properties of an
        OpenFlow Logical Switch Controller.

        NETCONF &lt;edit-config&gt; operations MUST be implemented as
        follows:

        * The 'id' element MUST be present at all &lt;edit-config&gt;
        operations to identify the controller.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="id"  type="OFConfigId">
        <xs:annotation>
          <xs:documentation>
            A unique but locally arbitrary identifier that
            uniquely identifies an OpenFlow Controller within the
            context of an OpenFlow Capable Switch.  It MUST be
            persistent across reboots of the OpenFlow Capable Switch.

            This element MUST be present to identify the OpenFlow
            controller.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="role">
        <xs:annotation>
          <xs:documentation>
            This element indicates the role of the OpenFlow
            Controller.  Semantics of these roles are specified in the
            OpenFlow specifications 1.0 - 1.3.1.  It is RECOMMENDED
            that the roles of controllers are not configured by
            OF-CONFIG 1.1.1 but determined using the OpenFlow protocol.
            OpenFlow Controllers configured by OF-CONFIG 1.1.1 have the
            default role 'equal'.  A role other than 'equal' MAY be
            assigned to a controller.  Roles 'slave' and 'equal' MAY be
            assigned to multiple controllers.  Role 'master' MUST NOT
            be assigned to more than one controller.

            This element is optional. If this element is not present it
            defaults to 'equal'.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="master"/>
            <xs:enumeration value="slave"/>
            <xs:enumeration value="equal"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="ip-address"  type="inet:ip-address">
        <xs:annotation>
          <xs:documentation>
            The IP address of the OpenFlow Controller.  This
            IP address is used by the OpenFlow Logical Switch when
            connecting to the OpenFlow Controller.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="port"  type="inet:port-number">
        <xs:annotation>
          <xs:documentation>
            The TCP port number at the OpenFlow Controller.
            This port number is used by the OpenFlow Logical Switch
            when connecting to the OpenFlow Controller using TCP or
            TLS.  The default value is 6633.

            This element is optional. If this element is not present it
            defaults to 6633.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="local-ip-address"  type="inet:ip-address">
        <xs:annotation>
          <xs:documentation>
            The local IP address of the OpenFlow Logical
            Switch when connecting to this OpenFlow Controller.  It is
            the source IP address of packets sent to this OpenFlow
            Controller.  If present, this element overrides any default
            IP address.


            This element is optional. Attempts to set this element to
            an IP address that cannot be used by the OpenFlow Logical
            Switch MUST result in an 'bad-element' error with type
            'application'. The &lt;error-info&gt; element MUST contain the
            name of this element in the &lt;bad-element&gt; element.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="local-port"  type="inet:port-number">
```

```
        <xs:annotation>
          <xs:documentation>
            The local TCP port number of the OpenFlow
            Logical Switch when connecting to this OpenFlow Controller.
            It is the source TCP port number of packets sent to this
            OpenFlow Controller.  If this element is not present, then
            the port number is chosen arbitrarily by the OpenFlow
            Logical Switch.

            This element is optional. Attempts to set this element to a
            port number that cannot be used by the OpenFlow Logical
            Switch MUST result in an 'bad-element' error with type
            'application'. The &lt;error-info&gt; element MUST contain the
            name of this element in the &lt;bad-element&gt; element.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="protocol">
        <xs:annotation>
          <xs:documentation>
            The default protocol tha the OpenFlow Logical
            Switch uses to connect to this OpenFlow Controller.  'tls'
            is the default value.

            This element is optional. If this element is not present it
            defaults to 'tls'.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="tcp"/>
            <xs:enumeration value="tls"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="state">
        <xs:annotation>
          <xs:documentation>
            This container holds connection state
            information that indicate the connection state of the
            OpenFlow Logical Switch and the OpenFlow protocol version
            used for the connection.

            Children of this element are not configurable and can only
            be retrieved by NETCONF &lt;get&gt; operations. Attemps to modify
            this element and its children with a NETCONF &lt;edit-config&gt;
            operation MUST result in an 'operation-not-supported' error
            with type 'application'.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="connection-state" minOccurs="0"
  type="OFUpDownStateType">
              <xs:annotation>
                <xs:documentation>
```

```
                  This object indicates the connections state of
                  the OpenFlow Logical Switch to this controller.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="current-version" minOccurs="0"
  type="OFOpenFlowVersionType">
              <xs:annotation>
                <xs:documentation>
                  This object indicates the version of the
                  OpenFlow protocol used between the OpenFlow Logical
                  Switch and this Controller.  If element connection-state
                  has value 'up', then this element indicates the actual
                  version in use.  If element connection-state has value
                  'down', then this element indicates the version number of
                  the last established connection with this OpenFlow
                  Controller.  The value of this element MAY be persistent
                  across reboots of the OpenFlow Logical Switch in such a
                  case.  If element connection-state has value 'down'and
                  there is no information about previous connections to
                  this OpenFlow controller, then this element is not
                  present or has the value '0'.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="supported-versions" minOccurs="0"
  maxOccurs="unbounded"  type="OFOpenFlowVersionType">
              <xs:annotation>
                <xs:documentation>
                  This list of elements includes one entry for
                  each OpenFlow protocol version that this OpenFlow
                  controller supports.  It SHOULD contain all
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="local-ip-address-in-use" minOccurs="0"
  type="inet:ip-address">
              <xs:annotation>
                <xs:documentation>
                  The local IP address of the OpenFlow Logical
                  Switch when connecting to this OpenFlow Controller.  It
                  is the source IP address of packets sent to this OpenFlow
                  Controller.  If present, this element overrides any
                  default IP address.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="local-port-in-use" minOccurs="0"
  type="inet:port-number">
              <xs:annotation>
                <xs:documentation>
                  The local TCP port number of the OpenFlow
                  Logical Switch.  If element connection-state has value
                  'up', then this element indicates the actual port number
                  in use.  If element connection-state has value 'down',
                  then this element indicates the port number used for the
                  last attempt to establish a connection with this OpenFlow
                  Controller.???
                  When connecting to this OpenFlow Controller, it is the
                  source TCP port number of packets sent to this OpenFlow
                  Controller.  If this element has its defaqult value 0,
                  then port number is chosen arbitrarily by the OpenFlow
                  Logical Switch.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:simpleType name="OFConfigId">
    <xs:annotation>
      <xs:documentation>
        Generic type of an identifier in OF-CONFIG
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="inet:uri">
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="OFOpenFlowVersionType">
    <xs:annotation>
      <xs:documentation>
        This enumeration contains the all OpenFlow
        versions released so far.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:enumeration value="not-applicable"/>
      <xs:enumeration value="1.0"/>
      <xs:enumeration value="1.0.1"/>
      <xs:enumeration value="1.1"/>
      <xs:enumeration value="1.2"/>
      <xs:enumeration value="1.3"/>
      <xs:enumeration value="1.3.1"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="OFUpDownStateType">
    <xs:annotation>
      <xs:documentation>
        Type to specify state information for a port or a
        connection.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:enumeration value="up"/>
      <xs:enumeration value="down"/>
    </xs:restriction>
  </xs:simpleType>
```

8.7.3 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>1.2</version>
      <version>1.1</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

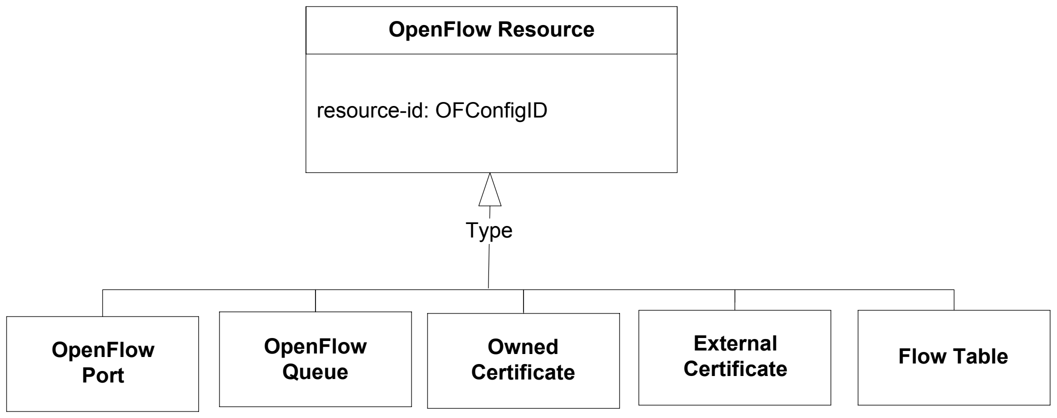

Figure 9: Data Model Diagram for an OpenFlow Resource

8.8.2 XML Schema

```
  <xs:group name="OFResourceType">
    <xs:annotation>
```

```
      <xs:documentation>
        This element specifies a generic OpenFlow resource
        that is used as a basis for specific resources. Even though
        this element is not used on its own the following rules for
        NETCONF operations MUST be obeyed also by elemnts using this
        element.

        NETCONF &lt;edit-config&gt; operations MUST be implemented as
        follows:

        * The 'id' element MUST be present at all &lt;edit-config&gt;
        operations to identify the resource.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="resource-id"  type="inet:uri">
        <xs:annotation>
          <xs:documentation>
            A unique but locally arbitrary identifier that
            uniquely identifies an OpenFlow Port within the context
            of an OpenFlow Logical Switch.  It MUST be persistent
            across reboots of the OpenFlow Capable Switch.

            This element MUST be present to identify the OpenFlow
            resource.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>
```

8.8.3 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 tunel 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 Port tunnel object.

8.9.1 UML Diagram

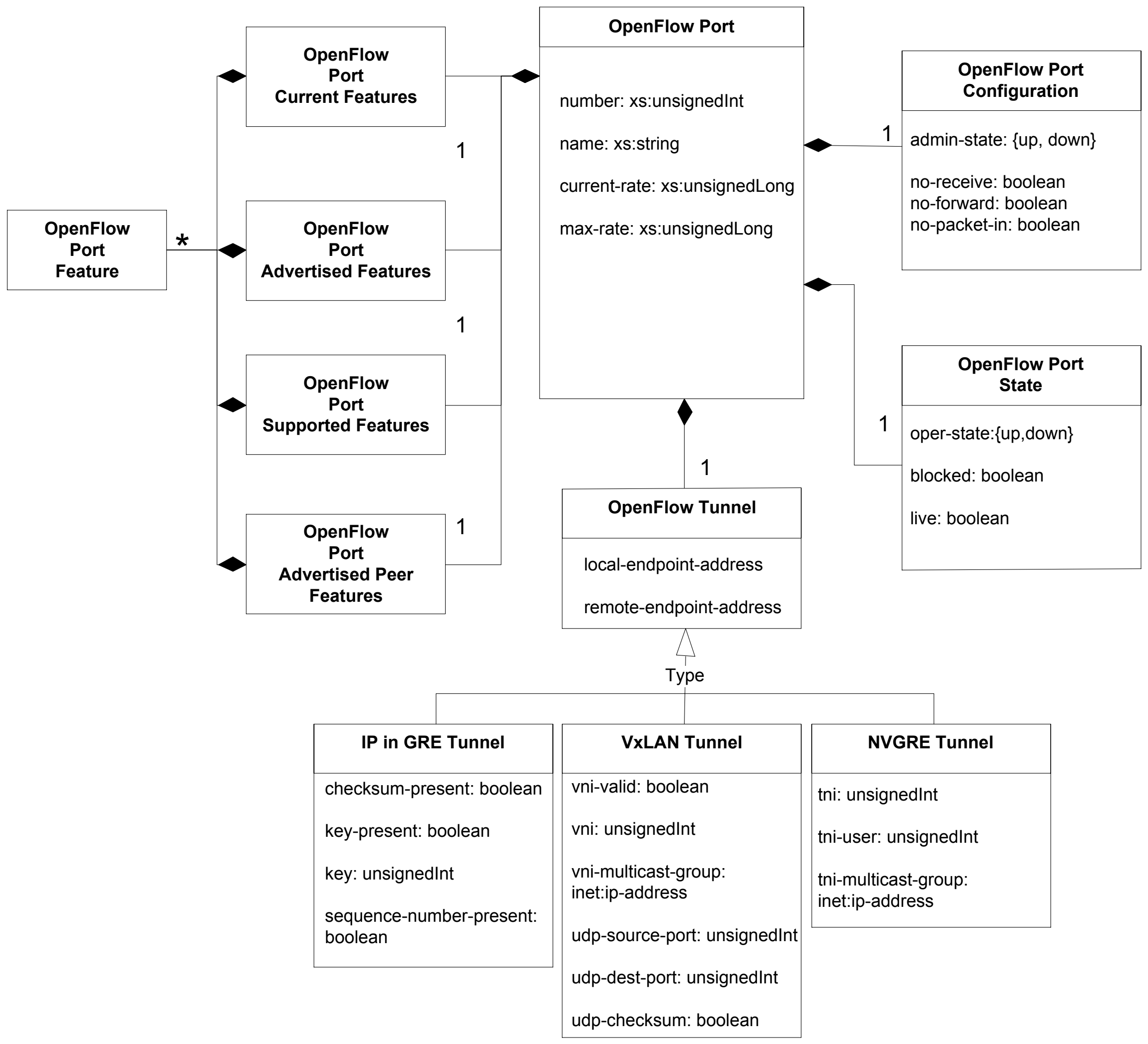

Figure 10: Data Model Diagram for an OpenFlow Port

8.9.2 XML Schema

```
  <xs:group name="OFPortType">
    <xs:annotation>
      <xs:documentation>
        This element specifies all properties of an
        OpenFlow resource of type OpenFlow Port. It represent a
        physical port or a logical port of the OpenFlow Capable
        Switch and can be assigned for exclusive use to an OpenFlow
        Logical Switch.  A logical port represents a tunnel endpoint
        as described in the OpenFlow protocol specification versions
        1.3 - 1.3.1.

        NETCONF &lt;edit-config&gt; operations MUST be implemented as
        follows:

        * The 'resource-id' element of OFResoureType MUST be present
        at all &lt;edit-config&gt; operations to identify the port.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFResourceType"/>
      <xs:element name="number"  type="xs:unsignedLong">
        <xs:annotation>
          <xs:documentation>
            This number identifies the OpenFlow Port to
            OpenFlow Controllers. It is assigned to an OpenFlow Port
            latest when the OpenFlow Port is associated with and
            OpenFlow Logical Switch.  If the OpenFlow Port is
            associated with an OpenFlow Logical Switch, this element
            MUST be unique within the context of the OpenFlow Logical
            Switch.

            OpenFlow Capable Switch implementations may choose to
            assign values to OpenFlow Ports that are unique within the
            context of the OpenFlow Logical Switch.  These numbers can
            be used independent of assignments to OpenFlow Logical
            Switches.

            Other implementations may assign values to this element
            only if the OpenFlow Port is assigned to an OpenFlow
            Logical Switch.  If no value is currently assigned to this
            element then this element MUST NOT be included in replies
            to NETCONF &lt;get&gt; requests. Since this element is not
            configurable with the NETCONF protocol it MUST NOT be
            included in replies to NETCONF &lt;get-config&gt; requests.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="name">
        <xs:annotation>
          <xs:documentation>
            This element assists OpenFlow Controllers in
            identifying OpenFlow Ports.

            This element is not to be set by the OP-CONFIG protocol,
            but it is set by the switch implementation.  It may be set
            at start-up time of an OpenFlow Capable Switch or when the
            OpenFlow Port is assigned to an OpenFlow Logical Switch.
            It MAY also be not set at all.  If this element is set to a
            value other than the empty string when being assigned to an
            OpenFlow Logical Switch, then the value of this element
            MUST be unique within the context of the OpenFlow Logical
            Switch.

            If no value or the empty string is currently assigned to
            this element then this element MUST not be included in
            replies to NETCONF &lt;get&gt; requests. Since this element is
            not configurable with the NETCONF protocol it MUST NOT be
            included in replies to NETCONF &lt;get-config&gt; requests.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:minLength value="1"/>
            <xs:maxLength value="16"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="current-rate"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            This element indicates the current bit rate of
            the port. Its values is to be provided in units of kilobit
            per second (kbps). This element is only valid if the
            element called 'rate' in the current Port Features has a
            value of 'other'.

            Since this element is not configurable with the NETCONF
            protocol it MUST NOT be included in replies to NETCONF
            &lt;get-config&gt; requests.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="max-rate"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            This element indicates the maximum bit rate of
            the port. Its values is to be provided in units of kilobit
            per second (kbps). This element is only valid if the
            element called 'rate' in the current Port Features has a
            value of 'other'.
            Since this element is not configurable with the NETCONF
            protocol it MUST NOT be included in replies to NETCONF
            &lt;get-config&gt; requests.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="configuration">
        <xs:annotation>
          <xs:documentation>
            This element represents the general
            adminitrative configuration of the OpenFlow Port.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="admin-state" minOccurs="0"
  type="OFUpDownStateType">
              <xs:annotation>
                <xs:documentation>
                  The administrative state of the port.  If
                  true, the port has been administratively brought down and
                  SHOULD not be used by OpenFlow.

                  This element is optional. If this element is not present
                  it defaults to 'up'.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="no-receive" minOccurs="0"  type="xs:boolean">
              <xs:annotation>
                <xs:documentation>
                  If true, packets received at this OpenFlow
                  port SHOULD be dropped.

                  This element is optional. If this element is not present
                  it defaults to 'false'.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="no-forward" minOccurs="0"  type="xs:boolean">
              <xs:annotation>
                <xs:documentation>
                  If true, packets forwarded to this OpenFlow
                  port SHOULD be dropped.

                  This element is optional. If this element is not present
                  it defaults to 'false'.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="no-packet-in" minOccurs="0"  type="xs:boolean">
              <xs:annotation>
                <xs:documentation>
                  If true, packets received on that port that
                  generate a table miss should never trigger a packet-in
                  message to the OpenFlow Controller.

                  This element is optional. If this element is not present
                  it defaults to 'false'.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="state">
        <xs:annotation>
          <xs:documentation>
            This element represents the general operational
            state of the OpenFlow Port.

            Children of this element are not configurable and can only be
            retrieved by NETCONF &lt;get&gt; operations. Attemps to modify
this
            element and its children with a NETCONF &lt;edit-config&gt;
            operation MUST result in an 'operation-not-supported' error
            with type 'application'.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="oper-state" minOccurs="0"
type="OFUpDownStateType">
              <xs:annotation>
                <xs:documentation>
                  If the value of this element is 'down', it
                  indicates that there is no physical link present.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="blocked" minOccurs="0"  type="xs:boolean">
              <xs:annotation>
                <xs:documentation>
                  If the value of this element is 'true', it
                  indicates that a switch protocol outside of OpenFlow,
                  such as 802.1D Spanning Tree, is preventing the use of
                  this OpenFlow port for OpenFlow flooding.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="live" minOccurs="0"  type="xs:boolean">
              <xs:annotation>
                <xs:documentation>
                  If the value of this element is 'true', it
                  indicates that this OpenFlow Port is live and can be used
                  for fast failover.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
```

```
    </xs:element>
    <xs:element name="features">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="current" minOccurs="0">
            <xs:annotation>
              <xs:documentation>
                The features (rates, duplex, etc.) of the
                port, that are currently in use.

                Children of this element are not configurable and can
                only be retrieved by NETCONF &lt;get&gt; operations.
Attemps to
                modify this element and its children with a NETCONF
                &lt;edit-config&gt; operation MUST result in an
                'operation-not-supported' error with type
                'application'.
              </xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:group ref="OFPortCurrentFeatureListType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="advertised" minOccurs="0">
            <xs:annotation>
              <xs:documentation>
                The features (rates, duplex, etc.) of the
                port, that are advertised to the peer port.

                NETCONF &lt;edit-config&gt; operations MUST be implemented
as
                follows:

                * The 'resource-id' element of OFResoureType MUST be
                present in the path or in the filter at all
                &lt;edit-config&gt; operations to identify the port.
                * If the operation is 'merge' or 'replace', the element
                is created if it does not exist, and its value is set
                to the value found in the XML RPC data.
                * If the operation is 'create', the element is created if
                it does not exist. If the element already exists, a
                'data-exists' error is returned.
                * If the operation is 'delete', the element is deleted if
                it exists. If the element does not exist, a
                'data-missing' error is returned.
              </xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:group ref="OFPortOtherFeatureListType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="supported" minOccurs="0">
```

```
            <xs:annotation>
              <xs:documentation>
                The features (rates, duplex, etc.) of the
                port, that are supported on the port.

                Children of this element are not configurable and can
                only be retrieved by NETCONF &lt;get&gt; operations.
Attemps to
                modify this element and its children with a NETCONF
                &lt;edit-config&gt; operation MUST result in an
                'operation-not-supported' error with type
                'application'.
              </xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:group ref="OFPortOtherFeatureListType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="advertised-peer" minOccurs="0">
            <xs:annotation>
              <xs:documentation>
                The features (rates, duplex, etc.) that are
                currently advertised by the peer port.

                Children of this element are not configurable and can
                only be retrieved by NETCONF &lt;get&gt; operations.
Attemps to
                modify this element and its children with a NETCONF
                &lt;edit-config&gt; operation MUST result in an
                'operation-not-supported' error with type
                'application'.
              </xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:group ref="OFPortOtherFeatureListType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:choice>
      <xs:annotation>
        <xs:documentation>
          Tunnels are modeled as logical ports.

          Elements in this choice are not configurable and can only
          be retrieved by NETCONF &lt;get&gt; operations. Attemps to modify
          this element and its children with a NETCONF &lt;edit-config&gt;
          operation MUST result in an 'operation-not-supported' error
          with type 'application'.

          Only elements from one choice must exist at a time.
        </xs:documentation>
      </xs:annotation>

      <xs:sequence>
        <xs:element name="tunnel">
          <xs:annotation>
            <xs:documentation>
              Properties of a basic IP-in-GRE tunnel.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:group ref="OFPortBaseTunnelType"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="ipgre-tunnel">
          <xs:annotation>
            <xs:documentation>
              Properties of a IP-in-GRE tunnel.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:group ref="OFPortIPGRETunnelType"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="vxlan-tunnel">
          <xs:annotation>
            <xs:documentation>
              Properties of a VxLAN tunnel.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:group ref="OFPortVXLANTunnelType"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="nvgre-tunnel">
          <xs:annotation>
            <xs:documentation>
              Properties of a NVGRE tunnel.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:group ref="OFPortNVGRETunnelType"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:choice>
  </xs:sequence>
</xs:group>

<xs:group name="OFPortBaseTunnelType">
  <xs:annotation>
    <xs:documentation>
      A group of common elements that are included
      in every supported tunnel type.  Tunnels are modeled as
      logical ports.

      One pair of local/remote endpoints must exist for a tunnel
      configuration.

      Only elements from one choice must exist at a time.
    </xs:documentation>
  </xs:annotation>

  <xs:sequence>
    <xs:choice>
      <xs:sequence>
        <xs:element name="local-endpoint-ipv4-adress"  type="inet:ipv4-
address">
          <xs:annotation>
            <xs:documentation>
              The IPv4 address of the local tunnel
              endpoint.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="remote-endpoint-ipv4-adress"  type="inet:ipv4-
address">
          <xs:annotation>
            <xs:documentation>
              The IPv4 address of the remote tunnel
              endpoint.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="local-endpoint-ipv6-adress"  type="inet:ipv6-
address">
          <xs:annotation>
            <xs:documentation>
              The IPv6 address of the local tunnel
              endpoint.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="remote-endpoint-ipv6-adress"  type="inet:ipv6-
address">
          <xs:annotation>
```

```
            <xs:documentation>
              The IPv6 address of the remote tunnel
              endpoint.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="local-endpoint-mac-adress"  type="yang:mac-
address">
          <xs:annotation>
            <xs:documentation>
              The MAC address of the local tunnel
              endpoint.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="remote-endpoint-mac-adress"  type="yang:mac-
address">
          <xs:annotation>
            <xs:documentation>
              The MAC address of the remote tunnel
              endpoint.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:choice>
  </xs:sequence>
</xs:group>

<xs:group name="OFPortIPGRETunnelType">
  <xs:annotation>
    <xs:documentation>
      Properties of a IP-in-GRE tunnel with key,
      checksum, and sequence number information.
    </xs:documentation>
  </xs:annotation>

  <xs:sequence>
    <xs:group ref="OFPortBaseTunnelType"/>
    <xs:element name="checksum-present"  type="xs:boolean">
      <xs:annotation>
        <xs:documentation>
          Indicates presence of the GRE checksum.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="key-present"  type="xs:boolean">
      <xs:annotation>
        <xs:documentation>
          Indicates presence of the GRE key.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="key"  type="xs:unsignedInt">
```

```
        <xs:annotation>
          <xs:documentation>
            The (optional) key of the GRE tunnel.  It MAY
            be used to set the OXM_OF_TUNNEL_ID match field metadata
            in the OpenFlow protocol
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="sequence-number-present"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Indicates presence of the GRE sequence
            number.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFPortVXLANTunnelType">
    <xs:annotation>
      <xs:documentation>
        Properties of a VxLAN tunnel.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFPortBaseTunnelType"/>
      <xs:element name="vni-valid"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Indicates how the corresponding flag should be
            set in packets sent on the tunnel.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="vni"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            Virtual network identifier assigned to all
            packets sent on the tunnel.  A VxLAN  implementation MAY
            use the this element to set the OXM_OF_TUNNEL_ID match
            field metadata in the OpenFlow protocol.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="vni-multicast-group"  type="inet:ip-address">
        <xs:annotation>
          <xs:documentation>
            If IP multicast is used to support broadcast
            on the tunnel this specifies the corresponding multicast
            IP address
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="udp-source-port"  type="inet:port-number">
```

```
        <xs:annotation>
          <xs:documentation>
            Specifies the outer UDP source port number.
            If this element is absent, the port number MAY be chosen
            dynamically.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="udp-dest-port"  type="inet:port-number">
        <xs:annotation>
          <xs:documentation>
            Specifies the outer UDP destination port
            number.  It is intended to reserve a port number for
            VxLAN at IANA.  As soon as this has been reserved, the
            reserved number SHOULD become the default value for this
            element.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="udp-checksum"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Boolean flag to indicate whether or not the
            outer UDP checksum should be set
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFPortNVGRETunnelType">
    <xs:annotation>
      <xs:documentation>
        Properties of a NVGRE tunnel.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFPortBaseTunnelType"/>
      <xs:element name="tni"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            Specifies the tenant network identifier
            assigned to all packets sent on the tunnel
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="tni-resv"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            Used to set the reserved user-defined bits of
            the GRE key field
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="tni-multicast-group"  type="inet:ip-address">
```

```
        <xs:annotation>
          <xs:documentation>
            If IP multicast is used to support broadcast
            on the tunnel this element specifies the corresponding
            multicast IP address
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>
```

8.9.3 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>3801</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>
    <tni>15581985</tni>
    <tni-resv>173</tni-resv>
  </nvgre-tunnel>
</port>
```

8.10 OpenFlow Port Feature

OpenFlow Port Features includePort 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

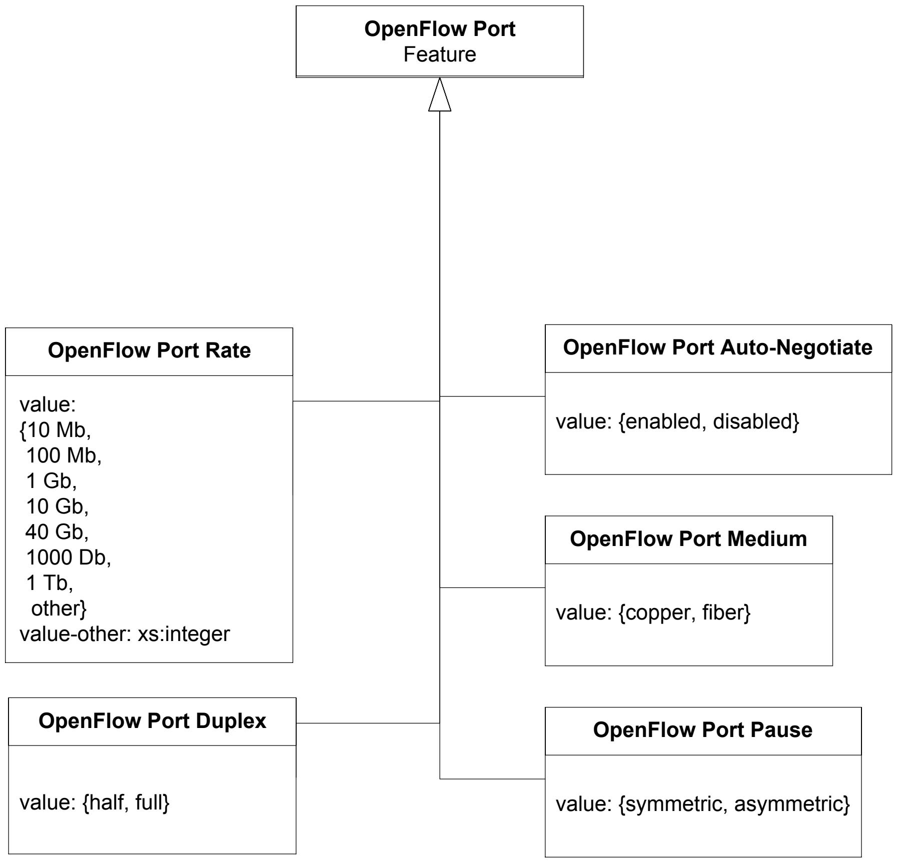

Figure 81: Data Model Diagram for an OpenFlow Port Feature

8.10.2 XML Schema

```
  <xs:group name="OFPortCurrentFeatureListType">
    <xs:annotation>
      <xs:documentation>
        The current features of a port.

        Elements in the type OFPortCurrentFeatureListType are not
        configurable and can only be retrieved by NETCONF &lt;get&gt;
        operations. Attemps to modify this element and its children
        with a NETCONF &lt;edit-config&gt; operation MUST result in an
        'operation-not-supported' error with type 'application'.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="rate"  type="OFPortRateType">
        <xs:annotation>
          <xs:documentation>
            The transmission rate that is currently used.
            The value MUST indicate a valid forwarding rate.

            The current Port Feature set MUST contain this element
            exactly once.  The other Port Feature sets MAY contain this
            element more than once.  If this element appears more than
            once in a Port Feature set than the value MUST be unique
            within the Port Feature set.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="auto-negotiate"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Specifies the administrative state of the
            forwarding rate auto-negotiation protocol at this OpenFlow
            Port.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="medium">
        <xs:annotation>
          <xs:documentation>
            This element MUST indicate a valid physical
            medium used by the OpenFlow Port.

            The current Port Feature set MUST contain this element
            exactly once. The other Port Feature sets MAY contain this
            element more than once. If this element appears more than
            once in a Port Feature set than the value MUST be unique
            within the Port Feature set.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="copper"/>
            <xs:enumeration value="fiber"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="pause">
        <xs:annotation>
          <xs:documentation>
            Specifies if pausing of transmission is
            supported at all and if yes if it is asymmetric or
            symmetric.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="unsupported"/>
            <xs:enumeration value="symmetric"/>
            <xs:enumeration value="asymmetric"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFPortOtherFeatureListType">
    <xs:annotation>
      <xs:documentation>
        The features of a port that are supported or
        advertised.

        If the elements in the OFPortOtherFeatureListType ares used
        as configurable elements the NETCONF &lt;edit-config&gt; operations
        MUST be implemented as follows:

        * The 'resource-id' element MUST be present in the path or in
        the filter at all &lt;edit-config&gt; operations to identify the
        resource.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.

        If elements in the type OFPortOtherFeatureListType are used
        in an non-configurable way, they only be retrieved by NETCONF
        &lt;get&gt; operations. Attemps to modify this element and its
        children with a NETCONF &lt;edit-config&gt; operation MUST result
        in an 'operation-not-supported' error with type
        'application'.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="rate"  type="OFPortRateType">
        <xs:annotation>
          <xs:documentation>
            The transmission rate that is supported or
            advertised. Multiple transmissions rates are allowed.

            At least one element MUST be present in the NETCONF data
            store. If none of this elements is are present in a NETCONF
            &lt;edit-config&gt; operation 'create', 'merge' or 'replace' and
            the parent element does not exist, a 'data-missing' error
            is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="auto-negotiate"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Specifies if auto-negotiation of transmission
            parameters is enabled for the port.

            This element is optional. If this element is not present it
            defaults to 'true'.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="medium">
        <xs:annotation>
          <xs:documentation>
            The transmission medium used by the port.
            Multiple media are allowed.

            At least one element MUST be present in the NETCONF data
            store. If none of this elements is are present in a NETCONF
            &lt;edit-config&gt; operation 'create', 'merge' or 'replace' and
            the parent element does not exist, a 'data-missing' error
            is returned.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="copper"/>
            <xs:enumeration value="fiber"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="pause">
        <xs:annotation>
          <xs:documentation>
            Specifies if pausing of transmission is
            supported at all and if yes if it is asymmetric or
            symmetric.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="unsupported"/>
            <xs:enumeration value="symmetric"/>
            <xs:enumeration value="asymmetric"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
```

```
  </xs:group>

  <xs:simpleType name="OFPortRateType">
    <xs:annotation>
      <xs:documentation>
        Type to specify the rate of a port including the
        duplex transmission feature. Possible rates are 10Mb, 100Mb,
        1Gb, 10Gb, 40Gb, 100Gb, 1Tb or other. Rates of 10Mb, 100Mb
        and 1 Gb can support half or full duplex transmission.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:enumeration value="10Mb-HD"/>
      <xs:enumeration value="10Mb-FD"/>
      <xs:enumeration value="100Mb-HD"/>
      <xs:enumeration value="100Mb-FD"/>
      <xs:enumeration value="1Gb-HD"/>
      <xs:enumeration value="1Gb-FD"/>
      <xs:enumeration value="10Gb"/>
      <xs:enumeration value="40Gb"/>
      <xs:enumeration value="100Gb"/>
      <xs:enumeration value="1Tb"/>
      <xs:enumeration value="other"/>
    </xs:restriction>
  </xs:simpleType>
```

8.10.3 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

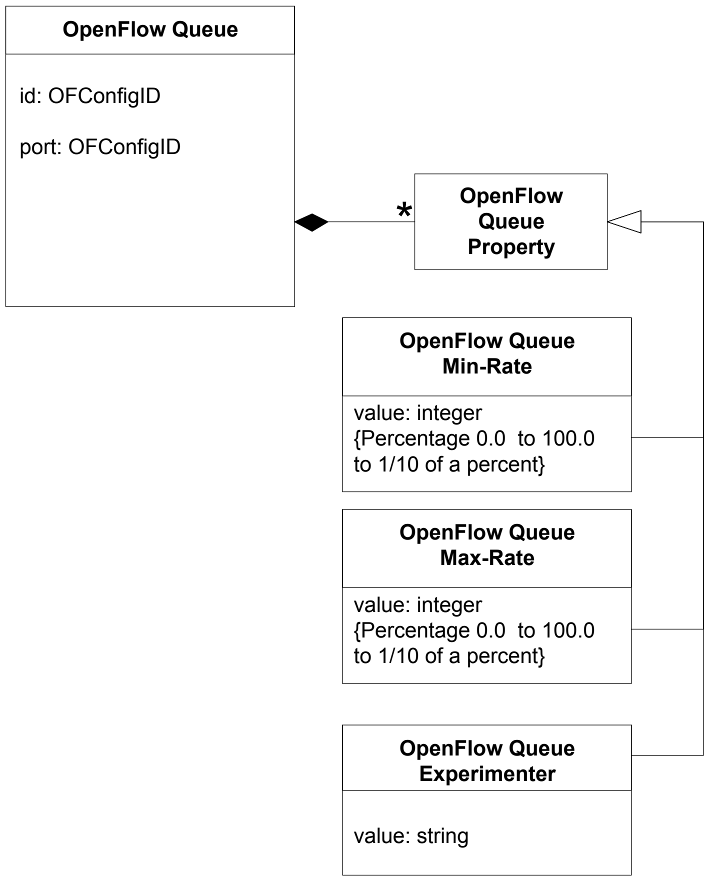
OpenFlow Queue
id: OFConfigID
port: OFConfigID
*
OpenFlow Queue Property
OpenFlow Queue Min-Rate
value: integer {Percentage 0.0 to 100.0 to 1/10 of a percent}
OpenFlow Queue Max-Rate
value: integer {Percentage 0.0 to 100.0 to 1/10 of a percent}
OpenFlow Queue Experimenter
value: string

Figure 12: Data Model Diagram for an OpenFlow Queue

8.11.2 XML Schema

```
  <xs:group name="OFQueueType">
    <xs:annotation>
      <xs:documentation>
        This grouping specifies all properties of a queue
        resource.

        NETCONF &lt;edit-config&gt; operations MUST be implemented as
        follows:

        * The 'resource-id' element of OFResoureType MUST be present
        at all &lt;edit-config&gt; operations to identify the port.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFResourceType"/>
      <xs:element name="id"  type="xs:unsignedLong">
        <xs:annotation>
          <xs:documentation>
            This id identifies the OpenFlow Queue to
            OpenFlow Controllers. It is assigned to an OpenFlow Queue
            latest when the OpenFlow Queue is associated with and
            OpenFlow Logical Switch.  If the OpenFlow Queue is
            associated with an OpenFlow Logical Switch, this element
            MUST be unique within the context of the OpenFlow Logical
            Switch.

            OpenFlow Capable Switch implementations may choose to
            assign values to OpenFlow Queues that are unique within the
            context of the OpenFlow Logical Switch.  These id can be
            used independent of assignments to OpenFlow Logical
            Switches.

            Other implementations may assign values to this element
            only if the OpenFlow Queue is assigned to an OpenFlow
            Logical Switch.  If no value is currently assigned to this
            element then this element MUST NOT be included in replies
            to NETCONF &lt;get&gt; requests. Since this element is not
            configurable with the NETCONF protocol it MUST NOT be
            included in replies to NETCONF &lt;get-config&gt; requests.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="port">
        <xs:annotation>
          <xs:documentation>
            Reference to port resources in the Capable
            Switch.

            This element associates an OpenFlow Queue with an OpenFlow
            Port. If the OpenFlow Queue is associated with an OpenFlow
            Logical Switch S and this element is present, then it MUST
            be set to the value of element resource-id of an OpenFlow
            Port which is associated with the OpenFlow Logical Switch
            S.

            The element MUST refer to an element at the following path:
            /capable-switch/resources/port/resource-id
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="inet:uri">
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="properties">
        <xs:annotation>
          <xs:documentation>
            The queue properties currently configured.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="min-rate" minOccurs="0"
  type="OFTenthOfAPercentType">
              <xs:annotation>
                <xs:documentation>
                  The minimal rate that is reserved for this
                  queue in 1/10 of a percent of the actual rate.

                  This element is optional. If not present a min-rate is
                  not set.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="max-rate" minOccurs="0"
  type="OFTenthOfAPercentType">
              <xs:annotation>
                <xs:documentation>
                  The maximum rate that is reserved for this
                  queue in 1/10 of a percent of the actual rate.

                  This element is optional. If not present the max-rate is
                  not set.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="experimenter" minOccurs="0"
  maxOccurs="unbounded"  type="xs:unsignedInt">
              <xs:annotation>
                <xs:documentation>
                  A list of experimenter identifiers of queue
                  properties used.

                  This element is optional.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:simpleType name="OFTenthOfAPercentType">
    <xs:annotation>
      <xs:documentation>
        This type defines a value in tenth of a percent.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:unsignedShort">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="1000"/>
    </xs:restriction>
  </xs:simpleType>
```

8.11.3 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>
    <experimenter>123498</experimenter>
    <experimenter>708</experimenter>
  </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

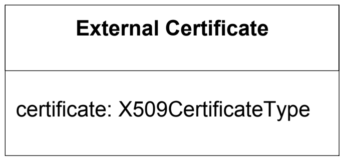

Figure 93: Data Model Diagram for a Certificate

8.12.2 XML Schema

```
  <xs:group name="OFExternalCertificateType">
    <xs:annotation>
      <xs:documentation>
        This grouping specifies a certificate that can be
        used by an OpenFlow Logical Switch for authenticating a
        controller when a TLS connection is established.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFResourceType"/>
      <xs:element name="certificate"  type="xs:string">
        <xs:annotation>
          <xs:documentation>
            An X.509 certificate in DER format base64
            encoded.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>
```

8.12.3 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

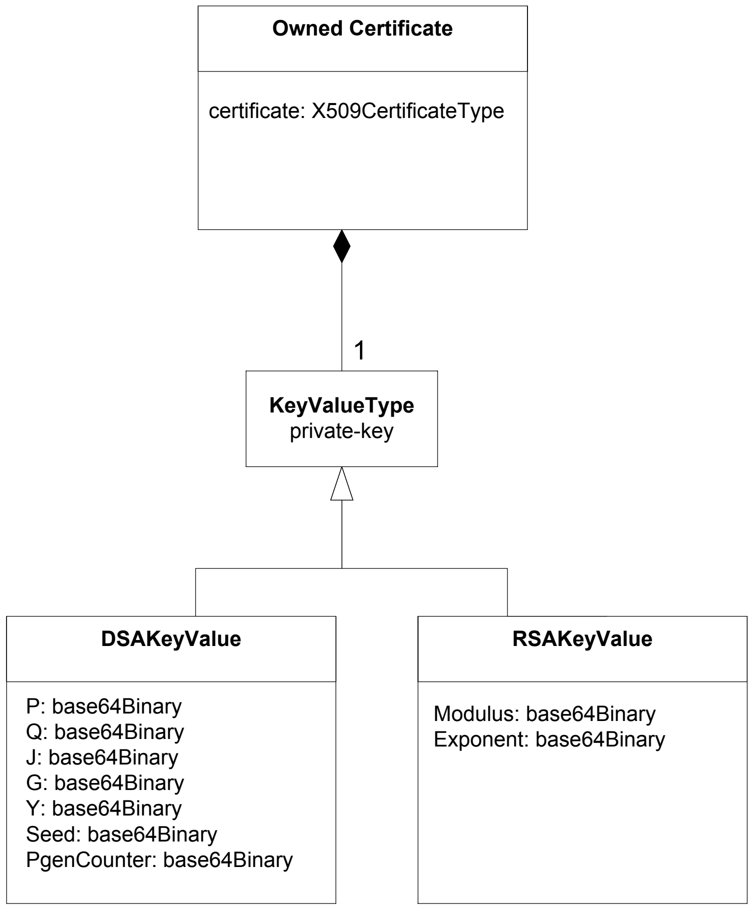

Figure 14: Data Model Diagram for Owned Certificate

8.13.2 XML Schema

```
  <xs:group name="OFOwnedCertificateType">
    <xs:annotation>
      <xs:documentation>
        This grouping specifies 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.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFResourceType"/>
      <xs:element name="certificate"  type="xs:string">
        <xs:annotation>
          <xs:documentation>
            An X.509 certificate in DER format base64
            encoded.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="private-key">
        <xs:annotation>
          <xs:documentation>
            This element contains the private key
            corresponding to the certificate. The private key is
            encoded as specified in XML-Signature Syntax and Processing
            (http://www.w3.org/TR/2001/PR-xmldsig-core-20010820/).
            Currently the specification only support DSA and RSA keys.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:group ref="KeyValueType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="KeyValueType">
    <xs:annotation>
      <xs:documentation>
        The KeyValue element contains a single public key
        that may be useful in validating the signature.

        NETCONF &lt;edit-config&gt; operations MUST be implemented as
        follows:

        * Exactly one of the elemenst 'DSAKeyValue' or 'RSAKeyValue'
        all &lt;edit-config&gt; operations.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:choice>
```

```
        <xs:sequence>
          <xs:element name="DSAKeyValue">
            <xs:complexType>
              <xs:sequence>
                <xs:group ref="DSAKeyValueType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:sequence>
          <xs:element name="RSAKeyValue">
            <xs:complexType>
              <xs:sequence>
                <xs:group ref="RSAKeyValueType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:choice>
    </xs:sequence>
  </xs:group>

  <xs:group name="DSAKeyValueType">
    <xs:annotation>
      <xs:documentation>
        DSA keys and the DSA signature algorithm are
        specified in 'FIPS PUB 186-2, Digital Signature Standard (DSS),
        U.S. Department of Commerce/National Institute of Standards and
        Technology,
        http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf'.
        DSA public key values can have the following fields:

        P
        a prime modulus meeting the requirements of the standard
        above
        Q
        an integer in the range 2**159 &lt; Q &lt; 2**160 which is a
        prime divisor of P-1
        G
        an integer with certain properties with respect to P and Q
        J
        (P - 1) / Q
        Y
        G**X mod P (where X is part of the private key and not made
        public)
        seed
        a DSA prime generation seed
        pgenCounter
        a DSA prime generation counter

        Parameter J is avilable for inclusion solely for efficiency as
        it is calculatable from P and Q. Parameters seed and
        pgenCounter are used in the DSA prime number generation
        algorithm specified in the above standard. As such, they are
        optional but MUST either both be present or both be absent.
        This prime generation algorithm is designed to provide
        assurance that a weak prime is not being used and it yields a P
        and Q value. Parameters P, Q, and G can be public and common to
        a group of users. They might be known from application context.
        As such, they are optional but P and Q MUST either both appear
        or both be absent. If all of P, Q, seed, and pgenCounter are
        present, implementations are not required to check if they are
        consistent and are free to use either P and Q or seed and
        pgenCounter. All parameters are encoded as base64 values.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="P"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element is optional. It MUST be present in
            the NETCONF data store, if the element 'Q' is present.

            If element 'Q' is present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and this element
            is missing, a 'data-missing' error is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Q"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element is optional. It MUST be present in
            the NETCONF data store, if the element 'P' is present.

            If element 'P' is present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and this element
            is missing, a 'data-missing' error is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="J"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element is optional.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="G"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element is optional.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Y"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element MUST be present in the NETCONF data
            store. If this element is not present in a NETCONF
            &lt;edit-config&gt; operation 'create', 'merge' or 'replace' and
            the parent element does not exist, a 'data-missing' error
            is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Seed"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element is optional. It MUST be present in
            the NETCONF data store, if the element 'PgenCounter' is
            present.

            If element 'PgenCounter' is present in a NETCONF
            &lt;edit-config&gt; operation 'create', 'merge' or 'replace' and
            this element is missing, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="PgenCounter"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element is optional. It MUST be present in
            the NETCONF data store, if the element 'Seed' is present.

            If element 'Seed' is present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and this element
            is missing, a 'data-missing' error is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="RSAKeyValueType">
    <xs:annotation>
      <xs:documentation>
        RSA key values have two fields: Modulus and
        Exponent.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="Modulus"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element MUST be present in the NETCONF data
            store. If this element is not present in a NETCONF
            &lt;edit-config&gt; operation 'create', 'merge' or 'replace' and
            the parent element does not exist, a 'data-missing' error
            is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Exponent"  type="xs:base64Binary">
        <xs:annotation>
```

```
          <xs:documentation>
            This element MUST be present in the NETCONF data
            store. If this element is not present in a NETCONF
            &lt;edit-config&gt; operation 'create', 'merge' or 'replace' and
            the parent element does not exist, a 'data-missing' error
            is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>
```

8.13.3 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

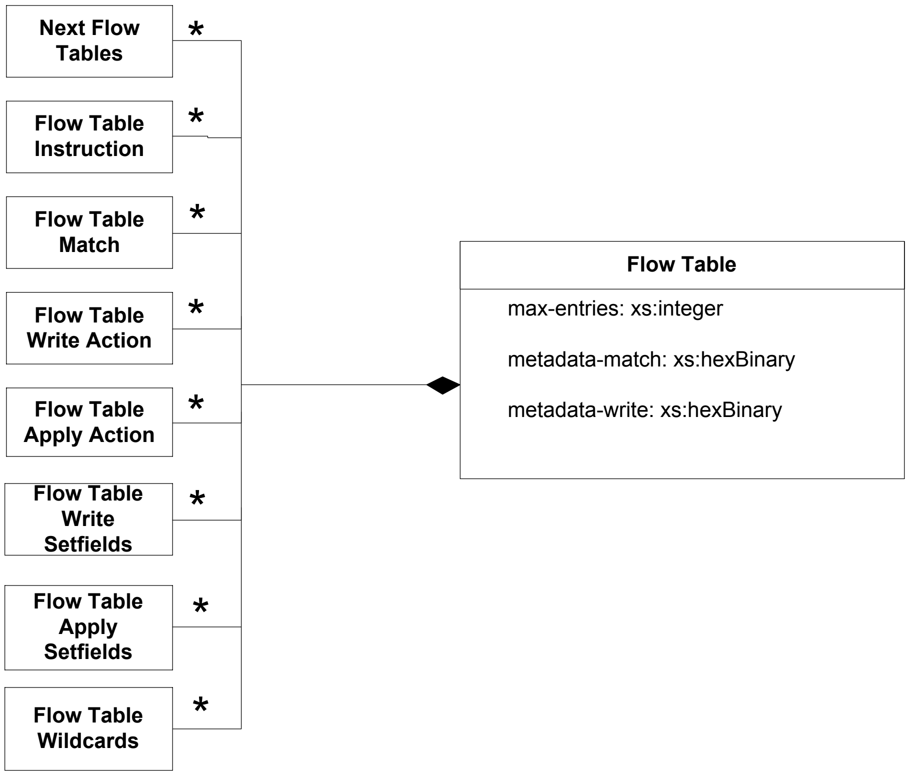

Figure 15: Data Model Diagram for Flow Table

8.14.2 XML Schema

```
  <xs:group name="OFFlowTableType">
    <xs:annotation>
      <xs:documentation>
        Representation of an OpenFlow Flow Table Resource.

        Elements in the type OFFlowTableType are not configurable and
        can only be retrieved by NETCONF &lt;get&gt; operations. Attemps to
        modify this element and its children with a NETCONF
        &lt;edit-config&gt; operation MUST result in an
        'operation-not-supported' error with type 'application'.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFResourceType"/>
      <xs:element name="max-entries"  type="xs:unsignedByte">
        <xs:annotation>
          <xs:documentation>
            The maximum number of flow entries supported by
            the flow table.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="next-tables">
```

```
        <xs:annotation>
          <xs:documentation>
            An array of resource-ids of all flow tables that
            can be directly reached from this table using the
            'goto-table' instruction.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="table-id" minOccurs="0" maxOccurs="unbounded"
type="inet:uri"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="instructions">
        <xs:annotation>
          <xs:documentation>
            The list of all instruction types supported by
            the flow table.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
type="OFInstructionType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="matches">
        <xs:annotation>
          <xs:documentation>
            The list of all match types supported by the
            flow table.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
type="OFMatchFieldType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="write-actions">
        <xs:annotation>
          <xs:documentation>
            The list of all write action types supported by
            the flow table.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
type="OFActionType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="apply-actions">
        <xs:annotation>
          <xs:documentation>
            The list of all apply action types supported by
            the flow table.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
  type="OFActionType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="write-setfields">
        <xs:annotation>
          <xs:documentation>
            The list of all 'set-field' action types
            supported by the table using write actions.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
  type="OFMatchFieldType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="apply-setfields">
        <xs:annotation>
          <xs:documentation>
            The list of all 'set-field' action types
            supported by the table using apply actions.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
  type="OFMatchFieldType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="wildcards">
        <xs:annotation>
          <xs:documentation>
            The list of all fields for which the table
            supports wildcarding.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
  type="OFMatchFieldType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="metadata-match"  type="hex-binary">
        <xs:annotation>
          <xs:documentation>
            This element indicates the bits of the metadata
            field on which the flow table can match.  It is represented
            as 64-bit integer in hexadecimal digits([0-9a-fA-F])
            format.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="metadata-write"  type="hex-binary">
        <xs:annotation>
          <xs:documentation>
            This element indicates the bits of the metadata
            field on which flow table can write using the
            'write-metadata' instruction.  It is represented as
            64-bit integer in hexadecimal digits([0-9a-fA-F]) format.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:simpleType name="OFMatchFieldType">
    <xs:annotation>
      <xs:documentation>
        The types of match field defined in OpenFlow
        Switch Specification versions 1.2, 1.3, and 1.3.1.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:enumeration value="input-port"/>
      <xs:enumeration value="physical-input-port"/>
      <xs:enumeration value="metadata"/>
      <xs:enumeration value="ethernet-dest"/>
      <xs:enumeration value="ethernet-src"/>
      <xs:enumeration value="ethernet-frame-type"/>
      <xs:enumeration value="vlan-id"/>
      <xs:enumeration value="vlan-priority"/>
      <xs:enumeration value="ip-dscp"/>
      <xs:enumeration value="ip-ecn"/>
      <xs:enumeration value="ip-protocol"/>
      <xs:enumeration value="ipv4-src"/>
      <xs:enumeration value="ipv4-dest"/>
      <xs:enumeration value="tcp-src"/>
      <xs:enumeration value="tcp-dest"/>
      <xs:enumeration value="udp-src"/>
      <xs:enumeration value="udp-dest"/>
      <xs:enumeration value="sctp-src"/>
      <xs:enumeration value="sctp-dest"/>
      <xs:enumeration value="icmpv4-type"/>
      <xs:enumeration value="icmpv4-code"/>
      <xs:enumeration value="arp-op"/>
      <xs:enumeration value="arp-src-ip-address"/>
      <xs:enumeration value="arp-target-ip-address"/>
      <xs:enumeration value="arp-src-hardware-address"/>
      <xs:enumeration value="arp-target-hardware-address"/>
      <xs:enumeration value="ipv6-src"/>
      <xs:enumeration value="ipv6-dest"/>
      <xs:enumeration value="ipv6-flow-label"/>
      <xs:enumeration value="icmpv6-type"/>
      <xs:enumeration value="icmpv6-code"/>
      <xs:enumeration value="ipv6-nd-target"/>
      <xs:enumeration value="ipv6-nd-source-link-layer"/>
      <xs:enumeration value="ipv6-nd-target-link-layer"/>
      <xs:enumeration value="mpls-label"/>
      <xs:enumeration value="mpls-tc"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="hex-binary">
    <xs:annotation>
      <xs:documentation>
        hex binary encoded string
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:base64Binary">
    </xs:restriction>
  </xs:simpleType>
```

8.14.3 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>
```

9 Binding to NETCONF

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

9.1 Requirements

When implementing the XML schema defined in Section 8 and Appendix A 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 XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:yin="urn:ietf:params:xml:schema:yang:yin:1"
           targetNamespace="urn:onf:of111:config:yang"
           xmlns="urn:onf:of111:config:yang"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified"
           version="2011-12-07"
           xml:lang="en"
          xmlns:yang="urn:ietf:params:xml:ns:yang:ietf-yang-types"
          xmlns:inet="urn:ietf:params:xml:ns:yang:ietf-inet-types"
          xmlns:of11-config="urn:onf:of111:config:yang">

  <xs:import namespace="urn:ietf:params:xml:ns:yang:ietf-yang-types"
             schemaLocation="ietf-yang-types.xsd"/>
  <xs:import namespace="urn:ietf:params:xml:ns:yang:ietf-inet-types"
             schemaLocation="ietf-inet-types.xsd"/>

  <xs:annotation>
    <xs:documentation>
      This schema was generated from the YANG module of-config1.1.1
      by pyang version 1.2.

      The schema describes an instance document consisting
      of the entire configuration data store, operational
      data, rpc operations, and notifications.
      This schema can thus NOT be used as-is to
      validate NETCONF PDUs.
    </xs:documentation>
  </xs:annotation>

  <xs:annotation>
    <xs:documentation>

      NETCONF Operational Considerations

      Elements that are configurable, optional and have a default
      value MAY be reported by replies to NETCONF &lt;get-config&gt;
      requests. All non-configurable values SHOULD be reported by
      replies to NETCONF &lt;get&gt; requests.

      Attemps to modify non-configurable elements with a NETCONF
      &lt;edit-config&gt; operation MUST result in an
      'operation-not-supported' error with type 'application'.

      When validating an &lt;edit-config&gt; operation the following
      errors MUST be detected:

      * Delete requests for non-existent data. In this case a
      'data-missing' error is returned.
      * Create requests for existent data. In this case a
      'data-exists' error is returned.
      * If the NETCONF operation creates data nodes under a
      'choice', any existing nodes from other branches are
      deleted.
    </xs:documentation>
  </xs:annotation>

  <!-- YANG typedefs -->
  <xs:simpleType name="OFConfigId">
    <xs:annotation>
      <xs:documentation>
        Generic type of an identifier in OF-CONFIG
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="inet:uri">
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="OFConfigurationPointProtocolType">
    <xs:annotation>
      <xs:documentation>
        Possible protocols to connect ot an OF
        Configuration Point
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:enumeration value="ssh"/>
      <xs:enumeration value="soap"/>
      <xs:enumeration value="tls"/>
      <xs:enumeration value="beep"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="OFOpenFlowVersionType">
    <xs:annotation>
      <xs:documentation>
        This enumeration contains the all OpenFlow
        versions released so far.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:enumeration value="not-applicable"/>
      <xs:enumeration value="1.0"/>
      <xs:enumeration value="1.0.1"/>
      <xs:enumeration value="1.1"/>
      <xs:enumeration value="1.2"/>
      <xs:enumeration value="1.3"/>
      <xs:enumeration value="1.3.1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="datapath-id-type">
    <xs:annotation>
      <xs:documentation>
        The datapath-id type represents an OpenFlow
        datapath identifier.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
    <xs:pattern value="[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){7}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="OFTenthOfAPercentType">
    <xs:annotation>
      <xs:documentation>
        This type defines a value in tenth of a percent.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:unsignedShort">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="1000"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="OFUpDownStateType">
    <xs:annotation>
      <xs:documentation>
        Type to specify state information for a port or a
        connection.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:enumeration value="up"/>
      <xs:enumeration value="down"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="OFPortRateType">
    <xs:annotation>
      <xs:documentation>
        Type to specify the rate of a port including the
        duplex transmission feature. Possible rates are 10Mb, 100Mb,
        1Gb, 10Gb, 40Gb, 100Gb, 1Tb or other. Rates of 10Mb, 100Mb
        and 1 Gb can support half or full duplex transmission.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:enumeration value="10Mb-HD"/>
      <xs:enumeration value="10Mb-FD"/>
      <xs:enumeration value="100Mb-HD"/>
      <xs:enumeration value="100Mb-FD"/>
      <xs:enumeration value="1Gb-HD"/>
      <xs:enumeration value="1Gb-FD"/>
      <xs:enumeration value="10Gb"/>
      <xs:enumeration value="40Gb"/>
      <xs:enumeration value="100Gb"/>
      <xs:enumeration value="1Tb"/>
      <xs:enumeration value="other"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="OFActionType">
```

```
    <xs:annotation>
      <xs:documentation>
        The types of actions defined in OpenFlow Switch
        Specification versions 1.2, 1.3, and 1.3.1
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:enumeration value="output"/>
      <xs:enumeration value="copy-ttl-out"/>
      <xs:enumeration value="copy-ttl-in"/>
      <xs:enumeration value="set-mpls-ttl"/>
      <xs:enumeration value="dec-mpls-ttl"/>
      <xs:enumeration value="push-vlan"/>
      <xs:enumeration value="pop-vlan"/>
      <xs:enumeration value="push-mpls"/>
      <xs:enumeration value="pop-mpls"/>
      <xs:enumeration value="set-queue"/>
      <xs:enumeration value="group"/>
      <xs:enumeration value="set-nw-ttl"/>
      <xs:enumeration value="dec-nw-ttl"/>
      <xs:enumeration value="set-field"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="OFInstructionType">
    <xs:annotation>
      <xs:documentation>
        The types of instructions defined in OpenFlow
        Switch Specification versions 1.2, 1.3, and 1.3.1.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:enumeration value="apply-actions"/>
      <xs:enumeration value="clear-actions"/>
      <xs:enumeration value="write-actions"/>
      <xs:enumeration value="write-metadata"/>
      <xs:enumeration value="goto-table"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="OFMatchFieldType">
    <xs:annotation>
      <xs:documentation>
        The types of match field defined in OpenFlow
        Switch Specification versions 1.2, 1.3, and 1.3.1.
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:enumeration value="input-port"/>
      <xs:enumeration value="physical-input-port"/>
      <xs:enumeration value="metadata"/>
      <xs:enumeration value="ethernet-dest"/>
      <xs:enumeration value="ethernet-src"/>
      <xs:enumeration value="ethernet-frame-type"/>
      <xs:enumeration value="vlan-id"/>
      <xs:enumeration value="vlan-priority"/>
      <xs:enumeration value="ip-dscp"/>
      <xs:enumeration value="ip-ecn"/>
      <xs:enumeration value="ip-protocol"/>
      <xs:enumeration value="ipv4-src"/>
      <xs:enumeration value="ipv4-dest"/>
      <xs:enumeration value="tcp-src"/>
      <xs:enumeration value="tcp-dest"/>
      <xs:enumeration value="udp-src"/>
      <xs:enumeration value="udp-dest"/>
      <xs:enumeration value="sctp-src"/>
      <xs:enumeration value="sctp-dest"/>
      <xs:enumeration value="icmpv4-type"/>
      <xs:enumeration value="icmpv4-code"/>
      <xs:enumeration value="arp-op"/>
      <xs:enumeration value="arp-src-ip-address"/>
      <xs:enumeration value="arp-target-ip-address"/>
      <xs:enumeration value="arp-src-hardware-address"/>
      <xs:enumeration value="arp-target-hardware-address"/>
      <xs:enumeration value="ipv6-src"/>
      <xs:enumeration value="ipv6-dest"/>
      <xs:enumeration value="ipv6-flow-label"/>
      <xs:enumeration value="icmpv6-type"/>
      <xs:enumeration value="icmpv6-code"/>
      <xs:enumeration value="ipv6-nd-target"/>
      <xs:enumeration value="ipv6-nd-source-link-layer"/>
      <xs:enumeration value="ipv6-nd-target-link-layer"/>
      <xs:enumeration value="mpls-label"/>
      <xs:enumeration value="mpls-tc"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="hex-binary">
    <xs:annotation>
      <xs:documentation>
        hex binary encoded string
      </xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:base64Binary">
    </xs:restriction>
  </xs:simpleType>

  <!-- YANG groupings -->

  <xs:group name="OFPortCurrentFeatureListType">
    <xs:annotation>
      <xs:documentation>
        The current features of a port.

        Elements in the type OFPortCurrentFeatureListType are not
        configurable and can only be retrieved by NETCONF &lt;get&gt;
        operations. Attemps to modify this element and its children
        with a NETCONF &lt;edit-config&gt; operation MUST result in an
        'operation-not-supported' error with type 'application'.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="rate"  type="OFPortRateType">
        <xs:annotation>
          <xs:documentation>
            The transmission rate that is currently used.
            The value MUST indicate a valid forwarding rate.

            The current Port Feature set MUST contain this element
            exactly once.  The other Port Feature sets MAY contain this
            element more than once.  If this element appears more than
            once in a Port Feature set than the value MUST be unique
            within the Port Feature set.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="auto-negotiate"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Specifies the administrative state of the
            forwarding rate auto-negotiation protocol at this OpenFlow
            Port.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="medium">
        <xs:annotation>
          <xs:documentation>
            This element MUST indicate a valid physical
            medium used by the OpenFlow Port.

            The current Port Feature set MUST contain this element
            exactly once. The other Port Feature sets MAY contain this
            element more than once. If this element appears more than
            once in a Port Feature set than the value MUST be unique
            within the Port Feature set.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="copper"/>
            <xs:enumeration value="fiber"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="pause">
        <xs:annotation>
          <xs:documentation>
            Specifies if pausing of transmission is
            supported at all and if yes if it is asymmetric or
            symmetric.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="unsupported"/>
            <xs:enumeration value="symmetric"/>
            <xs:enumeration value="asymmetric"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFPortOtherFeatureListType">
    <xs:annotation>
      <xs:documentation>
        The features of a port that are supported or
        advertised.

        If the elements in the OFPortOtherFeatureListType ares used
        as configurable elements the NETCONF &lt;edit-config&gt; operations
        MUST be implemented as follows:

        * The 'resource-id' element MUST be present in the path or in
        the filter at all &lt;edit-config&gt; operations to identify the
        resource.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.

        If elements in the type OFPortOtherFeatureListType are used
        in an non-configurable way, they only be retrieved by NETCONF
        &lt;get&gt; operations. Attemps to modify this element and its
        children with a NETCONF &lt;edit-config&gt; operation MUST result
        in an 'operation-not-supported' error with type
        'application'.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="rate"  type="OFPortRateType">
        <xs:annotation>
          <xs:documentation>
            The transmission rate that is supported or
            advertised. Multiple transmissions rates are allowed.

            At least one element MUST be present in the NETCONF data
            store. If none of this elements is are present in a NETCONF
            &lt;edit-config&gt; operation 'create', 'merge' or 'replace' and
            the parent element does not exist, a 'data-missing' error
            is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="auto-negotiate"  type="xs:boolean">
```

```
        <xs:annotation>
          <xs:documentation>
            Specifies if auto-negotiation of transmission
            parameters is enabled for the port.

            This element is optional. If this element is not present it
            defaults to 'true'.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="medium">
        <xs:annotation>
          <xs:documentation>
            The transmission medium used by the port.
            Multiple media are allowed.

            At least one element MUST be present in the NETCONF data
            store. If none of this elements is are present in a NETCONF
            &lt;edit-config&gt; operation 'create', 'merge' or 'replace' and
            the parent element does not exist, a 'data-missing' error
            is returned.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="copper"/>
            <xs:enumeration value="fiber"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="pause">
        <xs:annotation>
          <xs:documentation>
            Specifies if pausing of transmission is
            supported at all and if yes if it is asymmetric or
            symmetric.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="unsupported"/>
            <xs:enumeration value="symmetric"/>
            <xs:enumeration value="asymmetric"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="DSAKeyValueType">
```

```
    <xs:annotation>
      <xs:documentation>
        DSA keys and the DSA signature algorithm are
        specified in 'FIPS PUB 186-2, Digital Signature Standard (DSS),
        U.S. Department of Commerce/National Institute of Standards and
        Technology,
        http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf'.
        DSA public key values can have the following fields:

        P
        a prime modulus meeting the requirements of the standard
        above
        Q
        an integer in the range 2**159 &lt; Q &lt; 2**160 which is a
        prime divisor of P-1
        G
        an integer with certain properties with respect to P and Q
        J
        (P - 1) / Q
        Y
        G**X mod P (where X is part of the private key and not made
        public)
        seed
        a DSA prime generation seed
        pgenCounter
        a DSA prime generation counter

        Parameter J is avilable for inclusion solely for efficiency as
        it is calculatable from P and Q. Parameters seed and
        pgenCounter are used in the DSA prime number generation
        algorithm specified in the above standard. As such, they are
        optional but MUST either both be present or both be absent.
        This prime generation algorithm is designed to provide
        assurance that a weak prime is not being used and it yields a P
        and Q value. Parameters P, Q, and G can be public and common to
        a group of users. They might be known from application context.
        As such, they are optional but P and Q MUST either both appear
        or both be absent. If all of P, Q, seed, and pgenCounter are
        present, implementations are not required to check if they are
        consistent and are free to use either P and Q or seed and
        pgenCounter. All parameters are encoded as base64 values.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="P"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element is optional. It MUST be present in
            the NETCONF data store, if the element 'Q' is present.

            If element 'Q' is present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and this element
            is missing, a 'data-missing' error is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Q"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element is optional. It MUST be present in
            the NETCONF data store, if the element 'P' is present.

            If element 'P' is present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and this element
            is missing, a 'data-missing' error is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="J"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element is optional.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="G"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element is optional.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Y"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element MUST be present in the NETCONF data
            store. If this element is not present in a NETCONF
            &lt;edit-config&gt; operation 'create', 'merge' or 'replace' and
            the parent element does not exist, a 'data-missing' error
            is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Seed"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element is optional. It MUST be present in
            the NETCONF data store, if the element 'PgenCounter' is
            present.

            If element 'PgenCounter' is present in a NETCONF
            &lt;edit-config&gt; operation 'create', 'merge' or 'replace' and
            this element is missing, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="PgenCounter"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element is optional. It MUST be present in
            the NETCONF data store, if the element 'Seed' is present.

            If element 'Seed' is present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and this element
            is missing, a 'data-missing' error is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFPortBaseTunnelType">
    <xs:annotation>
      <xs:documentation>
        A group of common elements that are included
        in every supported tunnel type.  Tunnels are modeled as
        logical ports.

        One pair of local/remote endpoints must exist for a tunnel
        configuration.

        Only elements from one choice must exist at a time.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:choice>
        <xs:sequence>
          <xs:element name="local-endpoint-ipv4-adress"  type="inet:ipv4-
  address">
            <xs:annotation>
              <xs:documentation>
                The IPv4 address of the local tunnel
                endpoint.
              </xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="remote-endpoint-ipv4-adress"  type="inet:ipv4-
  address">
            <xs:annotation>
              <xs:documentation>
                The IPv4 address of the remote tunnel
                endpoint.
              </xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
        <xs:sequence>
          <xs:element name="local-endpoint-ipv6-adress"  type="inet:ipv6-
  address">
            <xs:annotation>
              <xs:documentation>
                The IPv6 address of the local tunnel
                endpoint.
              </xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="remote-endpoint-ipv6-adress"  type="inet:ipv6-
  address">
            <xs:annotation>
              <xs:documentation>
                The IPv6 address of the remote tunnel
                endpoint.
              </xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
        <xs:sequence>
          <xs:element name="local-endpoint-mac-adress"  type="yang:mac-
  address">
            <xs:annotation>
              <xs:documentation>
                The MAC address of the local tunnel
                endpoint.
              </xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="remote-endpoint-mac-adress"  type="yang:mac-
  address">
            <xs:annotation>
              <xs:documentation>
                The MAC address of the remote tunnel
                endpoint.
              </xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
      </xs:choice>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFPortIPGRETunnelType">
    <xs:annotation>
      <xs:documentation>
        Properties of a IP-in-GRE tunnel with key,
        checksum, and sequence number information.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFPortBaseTunnelType"/>
      <xs:element name="checksum-present"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Indicates presence of the GRE checksum.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="key-present"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Indicates presence of the GRE key.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="key"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            The (optional) key of the GRE tunnel.  It MAY
            be used to set the OXM_OF_TUNNEL_ID match field metadata
            in the OpenFlow protocol
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="sequence-number-present"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Indicates presence of the GRE sequence
            number.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFPortNVGRETunnelType">
    <xs:annotation>
      <xs:documentation>
        Properties of a NVGRE tunnel.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFPortBaseTunnelType"/>
      <xs:element name="tni"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            Specifies the tenant network identifier
            assigned to all packets sent on the tunnel
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="tni-resv"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            Used to set the reserved user-defined bits of
            the GRE key field
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="tni-multicast-group"  type="inet:ip-address">
        <xs:annotation>
          <xs:documentation>
            If IP multicast is used to support broadcast
            on the tunnel this element specifies the corresponding
            multicast IP address
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFQueueType">
    <xs:annotation>
      <xs:documentation>
        This grouping specifies all properties of a queue
        resource.

        NETCONF &lt;edit-config&gt; operations MUST be implemented as
        follows:

        * The 'resource-id' element of OFResoureType MUST be present
        at all &lt;edit-config&gt; operations to identify the port.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFResourceType"/>
      <xs:element name="id"  type="xs:unsignedLong">
        <xs:annotation>
          <xs:documentation>
            This id identifies the OpenFlow Queue to
            OpenFlow Controllers. It is assigned to an OpenFlow Queue
            latest when the OpenFlow Queue is associated with and
            OpenFlow Logical Switch.  If the OpenFlow Queue is
            associated with an OpenFlow Logical Switch, this element
            MUST be unique within the context of the OpenFlow Logical
            Switch.

            OpenFlow Capable Switch implementations may choose to
            assign values to OpenFlow Queues that are unique within the
            context of the OpenFlow Logical Switch.  These id can be
            used independent of assignments to OpenFlow Logical
            Switches.

            Other implementations may assign values to this element
            only if the OpenFlow Queue is assigned to an OpenFlow
            Logical Switch.  If no value is currently assigned to this
            element then this element MUST NOT be included in replies
            to NETCONF &lt;get&gt; requests. Since this element is not
            configurable with the NETCONF protocol it MUST NOT be
            included in replies to NETCONF &lt;get-config&gt; requests.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="port">
        <xs:annotation>
          <xs:documentation>
            Reference to port resources in the Capable
            Switch.

            This element associates an OpenFlow Queue with an OpenFlow
            Port. If the OpenFlow Queue is associated with an OpenFlow
            Logical Switch S and this element is present, then it MUST
            be set to the value of element resource-id of an OpenFlow
            Port which is associated with the OpenFlow Logical Switch
            S.

            The element MUST refer to an element at the following path:
            /capable-switch/resources/port/resource-id
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="inet:uri">
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="properties">
        <xs:annotation>
          <xs:documentation>
            The queue properties currently configured.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="min-rate" minOccurs="0"
  type="OFTenthOfAPercentType">
              <xs:annotation>
                <xs:documentation>
                  The minimal rate that is reserved for this
                  queue in 1/10 of a percent of the actual rate.

                  This element is optional. If not present a min-rate is
                  not set.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="max-rate" minOccurs="0"
  type="OFTenthOfAPercentType">
              <xs:annotation>
                <xs:documentation>
                  The maximum rate that is reserved for this
                  queue in 1/10 of a percent of the actual rate.

                  This element is optional. If not present the max-rate is
                  not set.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="experimenter" minOccurs="0"
  maxOccurs="unbounded"  type="xs:unsignedInt">
```

```
              <xs:annotation>
                <xs:documentation>
                  A list of experimenter identifiers of queue
                  properties used.

                  This element is optional.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFOwnedCertificateType">
    <xs:annotation>
      <xs:documentation>
        This grouping specifies 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.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFResourceType"/>
      <xs:element name="certificate"  type="xs:string">
        <xs:annotation>
          <xs:documentation>
            An X.509 certificate in DER format base64
            encoded.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="private-key">
        <xs:annotation>
          <xs:documentation>
            This element contains the private key
            corresponding to the certificate. The private key is
            encoded as specified in XML-Signature Syntax and Processing
            (http://www.w3.org/TR/2001/PR-xmldsig-core-20010820/).
            Currently the specification only support DSA and RSA keys.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:group ref="KeyValueType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFExternalCertificateType">
    <xs:annotation>
      <xs:documentation>
        This grouping specifies a certificate that can be
        used by an OpenFlow Logical Switch for authenticating a
        controller when a TLS connection is established.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFResourceType"/>
      <xs:element name="certificate"  type="xs:string">
        <xs:annotation>
          <xs:documentation>
            An X.509 certificate in DER format base64
            encoded.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFConfigurationPointType">
    <xs:annotation>
      <xs:documentation>
        Representation of an OpenFlow Configuration Point.
        Instances of the Configuration Point class SHOULD be stored
        persistently across reboots of the OpenFlow Capable Switch.

        When a connection is established between an OpenFlow Capable
        Switch and a Configuration Point the switch  MUST store the
        connection information in an instance of the Configuration
        Point class. If such an instance does not exist, the OpenFlow
        Capable Switch MUST create an instance where it then stores
        the connection information.

        An OpenFlow Capable Switch that cannot initiate a connection
        to a configuration point does not have to implement the
        Configuration Point class. It SHOULD block attempts to write
        to instances of the Configuration Point class with NETCONF
        &lt;edit-config&gt; operations.

        NETCONF &lt;edit-config&gt; operations MUST be implemented as
        follows:

        * The 'id' element MUST be present at all &lt;edit-config&gt;
        operations to identify the configuration point.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="id"  type="OFConfigId">
        <xs:annotation>
          <xs:documentation>
            A unique but locally arbitrary identifier that
            identifies a Configuration Point within the context of an
            OpenFlow Capable Switch.

            This element MUST be present to identify the configuration
            point.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="uri"  type="inet:uri">
        <xs:annotation>
          <xs:documentation>
            A locator of the Configuration Point.  It
            identifies the location of the Configuration Point as a
            service resource and MUST include all information necessary
            for the OpenFlow Capable Switch to connect to the
            Configuration Point or re-connect to it should it become
            disconnected.  Such information MAY include, for example,
            protocol, fully qualified domain name, IP address, port
            number, etc.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="protocol"  type="OFConfigurationPointProtocolType">
        <xs:annotation>
          <xs:documentation>
            The transport protocol that the Configuration
            Point uses when communicating via NETCONF with the OpenFlow
            Capable Switch.

            This element is optional. If it is not present its value
            defaults to 'ssh'.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="RSAKeyValueType">
    <xs:annotation>
      <xs:documentation>
        RSA key values have two fields: Modulus and
        Exponent.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="Modulus"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element MUST be present in the NETCONF data
            store. If this element is not present in a NETCONF
            &lt;edit-config&gt; operation 'create', 'merge' or 'replace' and
            the parent element does not exist, a 'data-missing' error
            is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Exponent"  type="xs:base64Binary">
        <xs:annotation>
          <xs:documentation>
            This element MUST be present in the NETCONF data
            store. If this element is not present in a NETCONF
            &lt;edit-config&gt; operation 'create', 'merge' or 'replace' and
            the parent element does not exist, a 'data-missing' error
            is returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFFlowTableType">
    <xs:annotation>
      <xs:documentation>
        Representation of an OpenFlow Flow Table Resource.

        Elements in the type OFFlowTableType are not configurable and
        can only be retrieved by NETCONF &lt;get&gt; operations. Attemps to
        modify this element and its children with a NETCONF
        &lt;edit-config&gt; operation MUST result in an
        'operation-not-supported' error with type 'application'.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFResourceType"/>
      <xs:element name="max-entries"  type="xs:unsignedByte">
        <xs:annotation>
          <xs:documentation>
            The maximum number of flow entries supported by
            the flow table.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="next-tables">
        <xs:annotation>
          <xs:documentation>
            An array of resource-ids of all flow tables that
            can be directly reached from this table using the
            'goto-table' instruction.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="table-id" minOccurs="0" maxOccurs="unbounded"
  type="inet:uri"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="instructions">
        <xs:annotation>
          <xs:documentation>
            The list of all instruction types supported by
            the flow table.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
  type="OFInstructionType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="matches">
        <xs:annotation>
          <xs:documentation>
            The list of all match types supported by the
            flow table.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
  type="OFMatchFieldType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="write-actions">
```

```
        <xs:annotation>
          <xs:documentation>
            The list of all write action types supported by
            the flow table.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
  type="OFActionType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="apply-actions">
        <xs:annotation>
          <xs:documentation>
            The list of all apply action types supported by
            the flow table.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
  type="OFActionType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="write-setfields">
        <xs:annotation>
          <xs:documentation>
            The list of all 'set-field' action types
            supported by the table using write actions.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
  type="OFMatchFieldType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="apply-setfields">
        <xs:annotation>
          <xs:documentation>
            The list of all 'set-field' action types
            supported by the table using apply actions.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
  type="OFMatchFieldType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="wildcards">
```

```
        <xs:annotation>
          <xs:documentation>
            The list of all fields for which the table
            supports wildcarding.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
type="OFMatchFieldType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="metadata-match"  type="hex-binary">
        <xs:annotation>
          <xs:documentation>
            This element indicates the bits of the metadata
            field on which the flow table can match.  It is represented
            as 64-bit integer in hexadecimal digits([0-9a-fA-F])
            format.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="metadata-write"  type="hex-binary">
        <xs:annotation>
          <xs:documentation>
            This element indicates the bits of the metadata
            field on which flow table can write using the
            'write-metadata' instruction.  It is represented as
            64-bit integer in hexadecimal digits([0-9a-fA-F]) format.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFLogicalSwitchType">
    <xs:annotation>
      <xs:documentation>
        This grouping specifies all properties of an
        OpenFlow Logical Switch.

        Elements of type OFLogicalSwitchType cannot be created or
        deleted with NETCONF &lt;edit-config&gt; operations 'create' or
        'delete'. The other NETCONF &lt;edit-config&gt; operations MUST be
        implemented as follows:

        * The 'id' element MUST be present at all &lt;edit-config&gt;
        operations to identify the OpenFlow Logical Switch.
        * If the operation is 'merge' or 'replace', and the element
        does not exist, a 'data-missing' error is returned. If the
        element exists its value is set to the value found in the
        XML RPC data.
        * If the operation is 'create', a 'operation-not-supported'
        error with type 'application' is returned.
        * If the operation is 'delete', 'operation-not-supported'
        error with type 'application' is returned.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="id"  type="OFConfigId">
        <xs:annotation>
          <xs:documentation>
            A unique but locally arbitrary identifier that
            identifies a Logical Switch within the context of an
            OpenFlow Capable Switch. It MUST be persistent across
            reboots of the OpenFlow Capable Switch.

            This element MUST be present to identify the OpenFlow
            Logical Switch.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="capabilities">
        <xs:annotation>
          <xs:documentation>
            This element contains all capability items that
            an OpenFlow Logical Switch MAY implement.

            This element and its children can only be retrieved by
            NETCONF &lt;get&gt; operation since it contain no configuration
            data.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:group ref="OFLogicalSwitchCapabilitiesType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="datapath-id"  type="datapath-id-type">
        <xs:annotation>
          <xs:documentation>
            The datapath identifier of the Logical Switch
            that uniquely identifies this Logical Switch within the
            context of all OpenFlow Controllers associated with the
            OpenFlow Logical Switch.  The datapath identifier is a
            string value that MUST be formatted as a sequence of 8
            2-digit hexadecimal numbers that are separated by colons,
            for example, '01:23:45:67:89:ab:cd:ef'.  When processing a
            datapath identifier, the case of the decimal digits MUST be
            ignored.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="enabled"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            This element indicates the administrative state
            of the OpenFlow Logical Switch.  A value of 'false' means
            the OpenFlow Logical Switch MUST NOT communicate with any
            OpenFlow Controllers, MUST NOT conduct any OpenFlow
            processing, and SHOULD NOT be utilizing computational or
            network resources of the underlying platform.

            This element is optional. If this element is not present it
            defaults to 'false'.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="check-controller-certificate"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            This element indicates the behavior of the
            OpenFlow Logical Switch when connecting to an OpenFlow
            Controller.

            If set to value 'false', the logical switch will connect to
            a controller without checking any controller certificate.

            If set to value 'true', then the logical switch will
            connect to a controller with element &lt;protocol&gt; set to
            'TLS', only if the controller provides a certificate that
            can be verified with one of the certificates stored in the
            list called external-certificates in the OpenFlow Capable
            Switch.

            If a certificate cannot be validated, the OpenFlow Logical
            Switch MUST terminate communication with the corresponding
            OpenFlow Controller, MUST NOT conduct any OpenFlow
            processing on requests of this OpenFlow controller, and
            SHOULD NOT further utilize any computational or network
            resources of for dealing with this connection.

            If set to value 'true', the OpenFlow Logical Switch MUST
            NOT connect to any OpenFlow Controller that does not
            provide a certificate. This implies that it cannot connect
            to an OpenFlow controller that has the value of element
            protocol set to 'TCP'. Only connections with protocol 'TLS'
            are possible in this case.

            This element is optional. If this element is not present it
            defaults to 'false'.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="lost-connection-behavior">
        <xs:annotation>
          <xs:documentation>
            This element indicates the the behavior of the
            OpenFlow Logical Switch in case it loses contact with all
            OpenFlow Controllers.  There are two alternative modes in
            such a case: fails secure mode and fail standalone mode as
            defined by the OpenFlow protocol specification version 1.2,
            section 6.4.  These are the only allowed values for this
            element. Default is the fail secure mode.

            This element is optional. If this element is not present it
            defaults to 'failSecureMode'.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="failSecureMode"/>
            <xs:enumeration value="failStandaloneMode"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="controllers">
        <xs:annotation>
          <xs:documentation>
            The list of controllers for this Logical switch.

            The element 'id' of OFControllerType MUST be unique within
            this list.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="controller" minOccurs="0"
  maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
                  The list of OpenFlow Controllers that are
                  assigned to the OpenFlow Logical Switch.  The switch MUST
                  NOT connect to any OpenFlow Controller that is not
                  contained in this list.

                  NETCONF &lt;edit-config&gt; operations MUST be implemented
  as
                  follows:

                  * The 'id' element MUST be present at all &lt;edit-
  config&gt;
                  operations to identify the controller.
                  * If the operation is 'merge' or 'replace', the element
                  is created if it does not exist, and its value is set
                  to the value found in the XML RPC data.
                  * If the operation is 'create', the element is created if
                  it does not exist. If the element already exists, a
                  'data-exists' error is returned.
                  * If the operation is 'delete', the element is deleted if
                  it exists. If the element does not exist, a
                  'data-missing' error is returned.
                </xs:documentation>
              </xs:annotation>
              <xs:complexType>
```

```
              <xs:sequence>
                <xs:group ref="OFControllerType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
      <xs:key name="key_controllers_controller">
        <xs:selector xpath="of11-config:controller"/>
        <xs:field xpath="of11-config:id"/>
      </xs:key>
    </xs:element>
    <xs:element name="resources">
      <xs:annotation>
        <xs:documentation>
          The list of identifiers of all resources of the
          OpenFlow Capable Switch that the OpenFlow Logical Switch
          has exclusive or non-exclusive access to.  A resource is
          identified by the value of its resource-identifier element.
          For each resource identifier value in this list, there MUST
          be an element with a matching resource identifier value in
          the resources list of the OpenFlow Capable Switch.

          Identifiers of this list are contained in elements
          indicating the type of resource: 'port', 'queue',
          'certificate', or 'flow-table'.  Depending on the type,
          different constraints apply.  These are specified in
          separate descriptions per type.

          At present the elements in this lists are not configurable
          and can only be retrieved by NETCONF &lt;get&gt; or &lt;get-
config&gt;
          operations. Attemps to modify this element and its children
          with a NETCONF &lt;edit-config&gt; operation MUST result in an
          'operation-not-supported' error with type 'application'.
        </xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:element name="port" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
              <xs:documentation>
                A resource identifier of a port of the
                OpenFlow Capable Switch that the OpenFlow Logical Switch
                has exclusive access to.

                The elements in this list MUST refer to elements at the
                following path:
                /capable-switch/resources/port/resource-id

                Elements in this list MUST be unique. This means each
                port element can only be referenced once.
              </xs:documentation>
            </xs:annotation>
            <xs:simpleType>
              <xs:restriction base="inet:uri">
```

```
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="queue" minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
                  A resource identifier of a queue of the
                  OpenFlow Capable Switch that the OpenFlow Logical Switch
                  has exclusive access to.

                  The elements in this list MUST refer to elements at the
                  following path:
                  /capable-switch/resources/queue/resource-id

                  Elements in this list MUST be unique. This means each
                  queue element can only be referenced once.
                </xs:documentation>
              </xs:annotation>
              <xs:simpleType>
                <xs:restriction base="inet:uri">
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="certificate" minOccurs="0">
              <xs:annotation>
                <xs:documentation>
                  The resource identifier of the owned
                  certificate in the OpenFlow Capable Switch that the
                  OpenFlow Logical Switch uses to identify itself.  This
                  element MUST NOT occur more than once in an OpenFlow
                  Logical Switch's resource list.

                  If no such element is in an OpenFlow Logical Switch's
                  resource list, then the OpenFlow Logical Switch does not
                  authenticate itself towards an OpenFloe Controller with a
                  certificate.  If this element is present, then the
                  OpenFlow Logical Switch MUST provide this certificate for
                  authentication to an OpenFlow Controller when setting up
                  a TLS connection.

                  For TCP connections this element is irrelevant.

                  The element MUST refer to an element at the following
                  path:
                  /capable-switch/resources/owned-certificate/resource-id

                </xs:documentation>
              </xs:annotation>
              <xs:simpleType>
                <xs:restriction base="inet:uri">
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="flow-table" minOccurs="0"
  maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
                  A resource identifier of a flow table of the
                  OpenFlow Capable Switch that the OpenFlow Logical Switch
                  has exclusive access to.

                  The elements in this list MUST refer to elements at the
                  following path:
                  /capable-switch/resources/flow-table/resource-id

                  Elements in this list MUST be unique. This means each
                  flow-table element can only be referenced once.
                </xs:documentation>
              </xs:annotation>
              <xs:simpleType>
                <xs:restriction base="inet:uri">
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="KeyValueType">
    <xs:annotation>
      <xs:documentation>
        The KeyValue element contains a single public key
        that may be useful in validating the signature.

        NETCONF &lt;edit-config&gt; operations MUST be implemented as
        follows:

        * Exactly one of the elemenst 'DSAKeyValue' or 'RSAKeyValue'
        all &lt;edit-config&gt; operations.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:choice>
        <xs:sequence>
          <xs:element name="DSAKeyValue">
            <xs:complexType>
              <xs:sequence>
                <xs:group ref="DSAKeyValueType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:sequence>
          <xs:element name="RSAKeyValue">
            <xs:complexType>
              <xs:sequence>
                <xs:group ref="RSAKeyValueType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:choice>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFLogicalSwitchCapabilitiesType">
    <xs:annotation>
      <xs:documentation>
        This grouping specifies all properties of an
        OpenFlow logical switch's capabilities.

        Elements in the type OFLogicalSwitchCapabilitiesType are not
        configurable and can only be retrieved by NETCONF &lt;get&gt;
        operations. Attemps to modify this element and its children
        with a NETCONF &lt;edit-config&gt; operation MUST result in an
        'operation-not-supported' error with type 'application'.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="max-buffered-packets"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            The maximum number of packets the logical switch
            can buffer when sending packets to the controller using
            packet-in messages.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="max-tables"  type="xs:unsignedByte">
        <xs:annotation>
          <xs:documentation>
            The number of flow tables supported by the
            logical switch.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="max-ports"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            The number of flow tables supported by the
            logical switch.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="flow-statistics"  type="xs:boolean">
```

```
        <xs:annotation>
          <xs:documentation>
            Specifies if the logical switch supports flow
            statistics.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="table-statistics"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Specifies if the logical switch supports table
            statistics.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="port-statistics"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Specifies if the logical switch supports port
            statistics.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="group-statistics"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Specifies if the logical switch supports group
            statistics.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="queue-statistics"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Specifies if the logical switch supports queue
            statistics.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="reassemble-ip-fragments"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Specifies if the logical switch supports
            reassemble IP fragments.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="block-looping-ports"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            'true' indicates that a switch protocol outside
            of OpenFlow, such as 802.1D Spanning Tree, will detect
            topology loops and block ports to prevent packet loops.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="reserved-port-types">
        <xs:annotation>
          <xs:documentation>
            Specify generic forwarding actions such as
            sending to the controller, flooding, or forwarding using
            non-OpenFlow methods, such as 'normal' switch processing.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="all"/>
                  <xs:enumeration value="controller"/>
                  <xs:enumeration value="table"/>
                  <xs:enumeration value="inport"/>
                  <xs:enumeration value="any"/>
                  <xs:enumeration value="normal"/>
                  <xs:enumeration value="flood"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="group-types">
        <xs:annotation>
          <xs:documentation>
            Specify the group types supported by the logical
            switch.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="all"/>
                  <xs:enumeration value="select"/>
                  <xs:enumeration value="indirect"/>
                  <xs:enumeration value="fast-failover"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="group-capabilities">
        <xs:annotation>
          <xs:documentation>
            Specify the group capabilities supported by the
            logical switch.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
```

```
          <xs:sequence>
            <xs:element name="capability" minOccurs="0"
maxOccurs="unbounded">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="select-weight"/>
                  <xs:enumeration value="select-liveness"/>
                  <xs:enumeration value="chaining"/>
                  <xs:enumeration value="chaining-check"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="action-types">
        <xs:annotation>
          <xs:documentation>
            Specify the action types supported by the
            logical switch.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
type="OFActionType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="instruction-types">
        <xs:annotation>
          <xs:documentation>
            Specify the instruction types supported by the
            logical switch.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded"
type="OFInstructionType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFPortType">
    <xs:annotation>
      <xs:documentation>
        This element specifies all properties of an
        OpenFlow resource of type OpenFlow Port. It represent a
        physical port or a logical port of the OpenFlow Capable
        Switch and can be assigned for exclusive use to an OpenFlow
        Logical Switch.  A logical port represents a tunnel endpoint
        as described in the OpenFlow protocol specification versions
        1.3 - 1.3.1.
        NETCONF &lt;edit-config&gt; operations MUST be implemented as
        follows:

        * The 'resource-id' element of OFResoureType MUST be present
        at all &lt;edit-config&gt; operations to identify the port.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFResourceType"/>
      <xs:element name="number"  type="xs:unsignedLong">
        <xs:annotation>
          <xs:documentation>
            This number identifies the OpenFlow Port to
            OpenFlow Controllers. It is assigned to an OpenFlow Port
            latest when the OpenFlow Port is associated with and
            OpenFlow Logical Switch.  If the OpenFlow Port is
            associated with an OpenFlow Logical Switch, this element
            MUST be unique within the context of the OpenFlow Logical
            Switch.

            OpenFlow Capable Switch implementations may choose to
            assign values to OpenFlow Ports that are unique within the
            context of the OpenFlow Logical Switch.  These numbers can
            be used independent of assignments to OpenFlow Logical
            Switches.

            Other implementations may assign values to this element
            only if the OpenFlow Port is assigned to an OpenFlow
            Logical Switch.  If no value is currently assigned to this
            element then this element MUST NOT be included in replies
            to NETCONF &lt;get&gt; requests. Since this element is not
            configurable with the NETCONF protocol it MUST NOT be
            included in replies to NETCONF &lt;get-config&gt; requests.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="name">
        <xs:annotation>
          <xs:documentation>
            This element assists OpenFlow Controllers in
            identifying OpenFlow Ports.

            This element is not to be set by the OP-CONFIG protocol,
            but it is set by the switch implementation.  It may be set
            at start-up time of an OpenFlow Capable Switch or when the
            OpenFlow Port is assigned to an OpenFlow Logical Switch.
            It MAY also be not set at all.  If this element is set to a
            value other than the empty string when being assigned to an
            OpenFlow Logical Switch, then the value of this element
            MUST be unique within the context of the OpenFlow Logical
            Switch.

            If no value or the empty string is currently assigned to
            this element then this element MUST not be included in
            replies to NETCONF &lt;get&gt; requests. Since this element is
            not configurable with the NETCONF protocol it MUST NOT be
            included in replies to NETCONF &lt;get-config&gt; requests.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:minLength value="1"/>
            <xs:maxLength value="16"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="current-rate"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            This element indicates the current bit rate of
            the port. Its values is to be provided in units of kilobit
            per second (kbps). This element is only valid if the
            element called 'rate' in the current Port Features has a
            value of 'other'.

            Since this element is not configurable with the NETCONF
            protocol it MUST NOT be included in replies to NETCONF
            &lt;get-config&gt; requests.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="max-rate"  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>
            This element indicates the maximum bit rate of
            the port. Its values is to be provided in units of kilobit
            per second (kbps). This element is only valid if the
            element called 'rate' in the current Port Features has a
            value of 'other'.

            Since this element is not configurable with the NETCONF
            protocol it MUST NOT be included in replies to NETCONF
            &lt;get-config&gt; requests.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="configuration">
        <xs:annotation>
          <xs:documentation>
            This element represents the general
            adminitrative configuration of the OpenFlow Port.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="admin-state" minOccurs="0"
  type="OFUpDownStateType">
              <xs:annotation>
                <xs:documentation>
                  The administrative state of the port.  If
                  true, the port has been administratively brought down and
                  SHOULD not be used by OpenFlow.

                  This element is optional. If this element is not present
                  it defaults to 'up'.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="no-receive" minOccurs="0"  type="xs:boolean">
              <xs:annotation>
                <xs:documentation>
                  If true, packets received at this OpenFlow
                  port SHOULD be dropped.

                  This element is optional. If this element is not present
                  it defaults to 'false'.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="no-forward" minOccurs="0"  type="xs:boolean">
              <xs:annotation>
                <xs:documentation>
                  If true, packets forwarded to this OpenFlow
                  port SHOULD be dropped.

                  This element is optional. If this element is not present
                  it defaults to 'false'.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="no-packet-in" minOccurs="0"  type="xs:boolean">
              <xs:annotation>
                <xs:documentation>
                  If true, packets received on that port that
                  generate a table miss should never trigger a packet-in
                  message to the OpenFlow Controller.

                  This element is optional. If this element is not present
                  it defaults to 'false'.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="state">
        <xs:annotation>
```

```
          <xs:documentation>
            This element represents the general operational
            state of the OpenFlow Port.

            Children of this element are not configurable and can only be
            retrieved by NETCONF &lt;get&gt; operations. Attemps to modify
this
            element and its children with a NETCONF &lt;edit-config&gt;
            operation MUST result in an 'operation-not-supported' error
            with type 'application'.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="oper-state" minOccurs="0"
type="OFUpDownStateType">
              <xs:annotation>
                <xs:documentation>
                  If the value of this element is 'down', it
                  indicates that there is no physical link present.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="blocked" minOccurs="0"  type="xs:boolean">
              <xs:annotation>
                <xs:documentation>
                  If the value of this element is 'true', it
                  indicates that a switch protocol outside of OpenFlow,
                  such as 802.1D Spanning Tree, is preventing the use of
                  this OpenFlow port for OpenFlow flooding.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="live" minOccurs="0"  type="xs:boolean">
              <xs:annotation>
                <xs:documentation>
                  If the value of this element is 'true', it
                  indicates that this OpenFlow Port is live and can be used
                  for fast failover.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="features">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="current" minOccurs="0">
              <xs:annotation>
                <xs:documentation>
                  The features (rates, duplex, etc.) of the
                  port, that are currently in use.

                  Children of this element are not configurable and can
                  only be retrieved by NETCONF &lt;get&gt; operations.
```

```
Attemps to
                modify this element and its children with a NETCONF
                &lt;edit-config&gt; operation MUST result in an
                'operation-not-supported' error with type
                'application'.
              </xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:group ref="OFPortCurrentFeatureListType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="advertised" minOccurs="0">
            <xs:annotation>
              <xs:documentation>
                The features (rates, duplex, etc.) of the
                port, that are advertised to the peer port.

                NETCONF &lt;edit-config&gt; operations MUST be implemented
as
                follows:

                * The 'resource-id' element of OFResoureType MUST be
                present in the path or in the filter at all
                &lt;edit-config&gt; operations to identify the port.
                * If the operation is 'merge' or 'replace', the element
                is created if it does not exist, and its value is set
                to the value found in the XML RPC data.
                * If the operation is 'create', the element is created if
                it does not exist. If the element already exists, a
                'data-exists' error is returned.
                * If the operation is 'delete', the element is deleted if
                it exists. If the element does not exist, a
                'data-missing' error is returned.
              </xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:group ref="OFPortOtherFeatureListType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="supported" minOccurs="0">
            <xs:annotation>
              <xs:documentation>
                The features (rates, duplex, etc.) of the
                port, that are supported on the port.

                Children of this element are not configurable and can
                only be retrieved by NETCONF &lt;get&gt; operations.
Attemps to
                modify this element and its children with a NETCONF
                &lt;edit-config&gt; operation MUST result in an
                'operation-not-supported' error with type
                'application'.
              </xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:group ref="OFPortOtherFeatureListType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="advertised-peer" minOccurs="0">
            <xs:annotation>
              <xs:documentation>
                The features (rates, duplex, etc.) that are
                currently advertised by the peer port.

                Children of this element are not configurable and can
                only be retrieved by NETCONF &lt;get&gt; operations.
 Attemps to
                modify this element and its children with a NETCONF
                &lt;edit-config&gt; operation MUST result in an
                'operation-not-supported' error with type
                'application'.
              </xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:group ref="OFPortOtherFeatureListType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:choice>
      <xs:annotation>
        <xs:documentation>
          Tunnels are modeled as logical ports.

          Elements in this choice are not configurable and can only
          be retrieved by NETCONF &lt;get&gt; operations. Attemps to modify
          this element and its children with a NETCONF &lt;edit-config&gt;
          operation MUST result in an 'operation-not-supported' error
          with type 'application'.

          Only elements from one choice must exist at a time.
        </xs:documentation>
      </xs:annotation>

      <xs:sequence>
        <xs:element name="tunnel">
          <xs:annotation>
            <xs:documentation>
              Properties of a basic IP-in-GRE tunnel.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
```

```
              <xs:group ref="OFPortBaseTunnelType"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="ipgre-tunnel">
          <xs:annotation>
            <xs:documentation>
              Properties of a IP-in-GRE tunnel.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:group ref="OFPortIPGRETunnelType"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="vxlan-tunnel">
          <xs:annotation>
            <xs:documentation>
              Properties of a VxLAN tunnel.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:group ref="OFPortVXLANTunnelType"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="nvgre-tunnel">
          <xs:annotation>
            <xs:documentation>
              Properties of a NVGRE tunnel.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:group ref="OFPortNVGRETunnelType"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:choice>
  </xs:sequence>
</xs:group>

<xs:group name="OFResourceType">
  <xs:annotation>
    <xs:documentation>
      This element specifies a generic OpenFlow resource
      that is used as a basis for specific resources. Even though
```

```
        this element is not used on its own the following rules for
        NETCONF operations MUST be obeyed also by elemnts using this
        element.

        NETCONF &lt;edit-config&gt; operations MUST be implemented as
        follows:

        * The 'id' element MUST be present at all &lt;edit-config&gt;
        operations to identify the resource.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="resource-id"  type="inet:uri">
        <xs:annotation>
          <xs:documentation>
            A unique but locally arbitrary identifier that
            uniquely identifies an OpenFlow Port within the context
            of an OpenFlow Logical Switch.  It MUST be persistent
            across reboots of the OpenFlow Capable Switch.

            This element MUST be present to identify the OpenFlow
            resource.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFPortVXLANTunnelType">
    <xs:annotation>
      <xs:documentation>
        Properties of a VxLAN tunnel.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:group ref="OFPortBaseTunnelType"/>
      <xs:element name="vni-valid"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Indicates how the corresponding flag should be
            set in packets sent on the tunnel.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="vni"  type="xs:unsignedInt">
```

```
        <xs:annotation>
          <xs:documentation>
            Virtual network identifier assigned to all
            packets sent on the tunnel.  A VxLAN  implementation MAY
            use the this element to set the OXM_OF_TUNNEL_ID match
            field metadata in the OpenFlow protocol.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="vni-multicast-group"  type="inet:ip-address">
        <xs:annotation>
          <xs:documentation>
            If IP multicast is used to support broadcast
            on the tunnel this specifies the corresponding multicast
            IP address
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="udp-source-port"  type="inet:port-number">
        <xs:annotation>
          <xs:documentation>
            Specifies the outer UDP source port number.
            If this element is absent, the port number MAY be chosen
            dynamically.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="udp-dest-port"  type="inet:port-number">
        <xs:annotation>
          <xs:documentation>
            Specifies the outer UDP destination port
            number.  It is intended to reserve a port number for
            VxLAN at IANA.  As soon as this has been reserved, the
            reserved number SHOULD become the default value for this
            element.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="udp-checksum"  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>
            Boolean flag to indicate whether or not the
            outer UDP checksum should be set
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="OFControllerType">
    <xs:annotation>
      <xs:documentation>
        This grouping specifies all properties of an
        OpenFlow Logical Switch Controller.

        NETCONF &lt;edit-config&gt; operations MUST be implemented as
        follows:

        * The 'id' element MUST be present at all &lt;edit-config&gt;
        operations to identify the controller.
        * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.
      </xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="id"  type="OFConfigId">
        <xs:annotation>
          <xs:documentation>
            A unique but locally arbitrary identifier that
            uniquely identifies an OpenFlow Controller within the
            context of an OpenFlow Capable Switch.  It MUST be
            persistent across reboots of the OpenFlow Capable Switch.

            This element MUST be present to identify the OpenFlow
            controller.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="role">
        <xs:annotation>
          <xs:documentation>
            This element indicates the role of the OpenFlow
            Controller.  Semantics of these roles are specified in the
            OpenFlow specifications 1.0 - 1.3.1.  It is RECOMMENDED
            that the roles of controllers are not configured by
            OF-CONFIG 1.1.1 but determined using the OpenFlow protocol.
            OpenFlow Controllers configured by OF-CONFIG 1.1.1 have the
            default role 'equal'.  A role other than 'equal' MAY be
            assigned to a controller.  Roles 'slave' and 'equal' MAY be
            assigned to multiple controllers.  Role 'master' MUST NOT
            be assigned to more than one controller.

            This element is optional. If this element is not present it
            defaults to 'equal'.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="master"/>
            <xs:enumeration value="slave"/>
            <xs:enumeration value="equal"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="ip-address"  type="inet:ip-address">
        <xs:annotation>
          <xs:documentation>
            The IP address of the OpenFlow Controller.  This
            IP address is used by the OpenFlow Logical Switch when
            connecting to the OpenFlow Controller.

            This element MUST be present in the NETCONF data store.
            If this element is not present in a NETCONF &lt;edit-config&gt;
            operation 'create', 'merge' or 'replace' and the parent
            element does not exist, a 'data-missing' error is
            returned.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="port"  type="inet:port-number">
        <xs:annotation>
          <xs:documentation>
            The TCP port number at the OpenFlow Controller.
            This port number is used by the OpenFlow Logical Switch
            when connecting to the OpenFlow Controller using TCP or
            TLS.  The default value is 6633.

            This element is optional. If this element is not present it
            defaults to 6633.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="local-ip-address"  type="inet:ip-address">
        <xs:annotation>
          <xs:documentation>
            The local IP address of the OpenFlow Logical
            Switch when connecting to this OpenFlow Controller.  It is
            the source IP address of packets sent to this OpenFlow
            Controller.  If present, this element overrides any default
            IP address.


            This element is optional. Attempts to set this element to
            an IP address that cannot be used by the OpenFlow Logical
            Switch MUST result in an 'bad-element' error with type
            'application'. The &lt;error-info&gt; element MUST contain the
            name of this element in the &lt;bad-element&gt; element.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="local-port"  type="inet:port-number">
        <xs:annotation>
          <xs:documentation>
            The local TCP port number of the OpenFlow
            Logical Switch when connecting to this OpenFlow Controller.
            It is the source TCP port number of packets sent to this
            OpenFlow Controller.  If this element is not present, then
            the port number is chosen arbitrarily by the OpenFlow
            Logical Switch.
            This element is optional. Attempts to set this element to a
            port number that cannot be used by the OpenFlow Logical
            Switch MUST result in an 'bad-element' error with type
            'application'. The &lt;error-info&gt; element MUST contain the
            name of this element in the &lt;bad-element&gt; element.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="protocol">
        <xs:annotation>
          <xs:documentation>
            The default protocol tha the OpenFlow Logical
            Switch uses to connect to this OpenFlow Controller.  'tls'
            is the default value.

            This element is optional. If this element is not present it
            defaults to 'tls'.
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="tcp"/>
            <xs:enumeration value="tls"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="state">
        <xs:annotation>
          <xs:documentation>
            This container holds connection state
            information that indicate the connection state of the
            OpenFlow Logical Switch and the OpenFlow protocol version
            used for the connection.

            Children of this element are not configurable and can only
            be retrieved by NETCONF &lt;get&gt; operations. Attemps to modify
            this element and its children with a NETCONF &lt;edit-config&gt;
            operation MUST result in an 'operation-not-supported' error
            with type 'application'.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="connection-state" minOccurs="0"
  type="OFUpDownStateType">
              <xs:annotation>
                <xs:documentation>
                  This object indicates the connections state of
                  the OpenFlow Logical Switch to this controller.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="current-version" minOccurs="0"
  type="OFOpenFlowVersionType">
              <xs:annotation>
                <xs:documentation>
                  This object indicates the version of the
                  OpenFlow protocol used between the OpenFlow Logical
                  Switch and this Controller.  If element connection-state
                  has value 'up', then this element indicates the actual
                  version in use.  If element connection-state has value
                  'down', then this element indicates the version number of
                  the last established connection with this OpenFlow
                  Controller.  The value of this element MAY be persistent
                  across reboots of the OpenFlow Logical Switch in such a
                  case.  If element connection-state has value 'down'and
                  there is no information about previous connections to
                  this OpenFlow controller, then this element is not
                  present or has the value '0'.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="supported-versions" minOccurs="0"
  maxOccurs="unbounded"  type="OFOpenFlowVersionType">
              <xs:annotation>
                <xs:documentation>
                  This list of elements includes one entry for
                  each OpenFlow protocol version that this OpenFlow
                  controller supports.  It SHOULD contain all
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="local-ip-address-in-use" minOccurs="0"
  type="inet:ip-address">
              <xs:annotation>
                <xs:documentation>
                  The local IP address of the OpenFlow Logical
                  Switch when connecting to this OpenFlow Controller.  It
                  is the source IP address of packets sent to this OpenFlow
                  Controller.  If present, this element overrides any
                  default IP address.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="local-port-in-use" minOccurs="0"
  type="inet:port-number">
              <xs:annotation>
                <xs:documentation>
                  The local TCP port number of the OpenFlow
                  Logical Switch.  If element connection-state has value
                  'up', then this element indicates the actual port number
                  in use.  If element connection-state has value 'down',
                  then this element indicates the port number used for the
                  last attempt to establish a connection with this OpenFlow
                  Controller.???
                  When connecting to this OpenFlow Controller, it is the
                  source TCP port number of packets sent to this OpenFlow
                  Controller.  If this element has its defaqult value 0,
                  then port number is chosen arbitrarily by the OpenFlow
                  Logical Switch.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:element name="capable-switch">
    <xs:annotation>
      <xs:documentation>
        The OpenFlow Capable Switch serves as the root
        element for an OpenFlow configuration.  It contains logical
        switches and resources that can be assigned to logical
        switches.  It may have relations to OpenFlow Configuration
        Points.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="id"  type="inet:uri">
          <xs:annotation>
            <xs:documentation>
              A unique but locally arbitrary identifier that
              uniquely identifies a Capable Switch within the context of
              potential OpenFlow Configuration Points.  It MUST be
              persistent across reboots of the OpenFlow Capable Switch.

              This element MUST be present in the NETCONF data store.
              If this element is not present in a NETCONF &lt;edit-config&gt;
              operation 'create', 'merge' or 'replace' and the parent
              element does not exist, a 'data-missing' error is
              returned.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="config-version" minOccurs="0"  type="xs:string">
          <xs:annotation>
            <xs:documentation>
              The maximum supported OF-CONFIG version that is
              supported by the OpenFlow Capable Switch. For switches
              implementing this version of the OF-CONFIG protocol this
              MUST always be 1.1.1.

              This object can be used to identify the OF-CONFIG version
              a capable switch supports beginning with version 1.1.1 of
              OF-CONFIG. In addtion the supported version can be
              determined by the namespace the OpenFlow Capable Switch
              returns to configuration request of an element (like
              capable-switch) that is present in all OF-CONFIG versions
              specified so far. This is the only possiblity to identify
              OF-CONFIG versions prior to OF-CONFIG 1.1.1.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="configuration-points" minOccurs="0">
          <xs:complexType>
```

```
          <xs:sequence>
            <xs:element name="configuration-point" minOccurs="0"
maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
                  The list of all Configuration Points known to
                  the OpenFlow Capable Switch that may manage it using
                  OF-CONFIG.

                  The element 'id' of OFConfigurationType MUST be unique
                  within this list.
                </xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:group ref="OFConfigurationPointType"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
        <xs:key name="key_configuration-points_capable-
switch_configuration-point">
          <xs:selector xpath="of11-config:configuration-point"/>
          <xs:field xpath="of11-config:id"/>
        </xs:key>
      </xs:element>
      <xs:element name="resources" minOccurs="0">
        <xs:annotation>
          <xs:documentation>
            A lists containing all resources of the OpenFlow
            Capable Switch that can be used by OpenFlow Logical
            Switches.  Resources are listed here independent of their
            actual assignment to OpenFlow Logical Switches.  They may
            be available to be assigned to an OpenFlow Logical Switch
            or already in use by an OpenFlow Logical Switch.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="port" minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
                  The list contains all port resources of the
                  OpenFlow Capable Switch.

                  The element 'resource-id' of OFPortType MUST be unique
                  within this list.
                </xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:group ref="OFPortType"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="queue" minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
                  The list contains all queue resources of the
                  OpenFlow Capable Switch.

                  The element 'resource-id' of OFQueueType MUST be unique
                  within this list.
                </xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:group ref="OFQueueType"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="owned-certificate" minOccurs="0"
maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
                  The list contains all owned certificate
                  resources of the OpenFlow Capable Switch.

                  The element 'resource-id' of OFOwnedCertificateType MUST
                  be unique within this list.
                </xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:group ref="OFOwnedCertificateType"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="external-certificate" minOccurs="0"
maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
                  The list contains all external certificate
                  resources of the OpenFlow Capable Switch.

                  The element 'resource-id' of OFExternalCertificateType
                  MUST be unique within this list.
                </xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:group ref="OFExternalCertificateType"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="flow-table" minOccurs="0"
maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
                  The list contains all flow table resources of
                  the OpenFlow Capable Switch.
                  The element 'resource-id' of OFFlowTableType MUST be
                  unique within this list.
                </xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:group ref="OFFlowTableType"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
        <xs:key name="key_resources_capable-switch_port">
          <xs:selector xpath="of11-config:port"/>
          <xs:field xpath="of11-config:resource-id"/>
        </xs:key>
        <xs:key name="key_resources_capable-switch_queue">
          <xs:selector xpath="of11-config:queue"/>
          <xs:field xpath="of11-config:resource-id"/>
        </xs:key>
        <xs:key name="key_resources_capable-switch_owned-certificate">
          <xs:selector xpath="of11-config:owned-certificate"/>
          <xs:field xpath="of11-config:resource-id"/>
        </xs:key>
        <xs:key name="key_resources_capable-switch_external-certificate">
          <xs:selector xpath="of11-config:external-certificate"/>
          <xs:field xpath="of11-config:resource-id"/>
        </xs:key>
        <xs:key name="key_resources_capable-switch_flow-table">
          <xs:selector xpath="of11-config:flow-table"/>
          <xs:field xpath="of11-config:resource-id"/>
        </xs:key>
      </xs:element>
      <xs:element name="logical-switches" minOccurs="0">
        <xs:annotation>
          <xs:documentation>
            This element contains a list of all OpenFlow
            Logical Switches available at the OpenFlow Capable
            Switch.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="switch" minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>
                  The list of all OpenFlow Logical Switches on
                  the OpenFlow Capable Switch.

                  The element 'resource-id' of OFLogicalSwitchType MUST be
                  unique within this list.
                </xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
```

```
                    <xs:group ref="OFLogicalSwitchType"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
          <xs:key name="key_logical-switches_capable-switch_switch">
            <xs:selector xpath="of11-config:switch"/>
            <xs:field xpath="of11-config:id"/>
          </xs:key>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Appendix B YANG Specification

```
module of-config1.1.1 {
  namespace "urn:onf:of111:config:yang";
  prefix of11-config;

  import ietf-yang-types { prefix yang; }
  import ietf-inet-types { prefix inet; }

  organization "ONF Config Management Group";

  contact "tbd";

  description "

    NETCONF Operational Considerations

    Elements that are configurable, optional and have a default
    value MAY be reported by replies to NETCONF <get-config>
    requests. All non-configurable values SHOULD be reported by
    replies to NETCONF <get> requests.

     Attemps to modify non-configurable elements with a NETCONF
     <edit-config> operation MUST result in an
     'operation-not-supported' error with type 'application'.

     When validating an <edit□config> operation the following
     errors MUST be detected:

     * Delete requests for non-existent data. In this case a
       'data-missing' error is returned.
     * Create requests for existent data. In this case a
       'data-exists' error is returned.
     * If the NETCONF operation creates data nodes under a
        'choice', any existing nodes from other branches are
        deleted.";
  revision 2011-12-07 {
    description "First Version";

    reference "tbd";
  }

/*****************************************************************
 * Features
 *****************************************************************/

/*****************************************************************
 * Type definitions
 *****************************************************************/

  typedef OFConfigId {
    type inet:uri;
    description "Generic type of an identifier in OF-CONFIG";
  }

  typedef OFConfigurationPointProtocolType {
    type enumeration {
      enum "ssh";
      enum "soap";
      enum "tls";
      enum "beep";
    }
    description "Possible protocols to connect ot an OF
      Configuration Point";
  }

  typedef OFOpenFlowVersionType {
    type enumeration {
      enum "not-applicable";
      enum "1.0";
      enum "1.0.1";
      enum "1.1";
      enum "1.2";
      enum "1.3";
      enum "1.3.1";
    }
    description "This enumeration contains the all OpenFlow
      versions released so far.";
  }

  typedef datapath-id-type {
    type string {
      pattern '[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){7}';
    }
    description "The datapath-id type represents an OpenFlow
      datapath identifier.";
  }

  typedef OFTenthOfAPercentType {
    type uint16 {
      range "0..1000";
    }
    units "1/10 of a percent";
    description "This type defines a value in tenth of a percent.";
  }

  typedef OFUpDownStateType {
    type enumeration {
      enum up;
      enum down;
    }
    description "Type to specify state information for a port or a
      connection.";
  }

  typedef OFPortRateType {
    type enumeration {
      enum 10Mb-HD;
      enum 10Mb-FD;
      enum 100Mb-HD;
      enum 100Mb-FD;
      enum 1Gb-HD;
      enum 1Gb-FD;
      enum 10Gb;
      enum 40Gb;
      enum 100Gb;
      enum 1Tb;
      enum other;
    }
    description "Type to specify the rate of a port including the
      duplex transmission feature. Possible rates are 10Mb, 100Mb,
      1Gb, 10Gb, 40Gb, 100Gb, 1Tb or other. Rates of 10Mb, 100Mb
      and 1 Gb can support half or full duplex transmission.";
  }

  typedef OFActionType {
    type enumeration {
      enum output;
      enum copy-ttl-out;
      enum copy-ttl-in;
      enum set-mpls-ttl;
      enum dec-mpls-ttl;
      enum push-vlan;
      enum pop-vlan;
      enum push-mpls;
      enum pop-mpls;
      enum set-queue;
      enum group;
      enum set-nw-ttl;
      enum dec-nw-ttl;
      enum set-field;
    }
    description "The types of actions defined in OpenFlow Switch
      Specification versions 1.2, 1.3, and 1.3.1";
  }

  typedef OFInstructionType {
    type enumeration {
      enum apply-actions;
      enum clear-actions;
      enum write-actions;
      enum write-metadata;
      enum goto-table;
    }
    description "The types of instructions defined in OpenFlow
      Switch Specification versions 1.2, 1.3, and 1.3.1.";
  }

  typedef OFMatchFieldType {
    type enumeration {
      enum input-port;
      enum physical-input-port;
      enum metadata;
      enum ethernet-dest;
      enum ethernet-src;
      enum ethernet-frame-type;
      enum vlan-id;
      enum vlan-priority;
      enum ip-dscp;
      enum ip-ecn;
      enum ip-protocol;
      enum ipv4-src;
      enum ipv4-dest;
      enum tcp-src;
      enum tcp-dest;
      enum udp-src;
      enum udp-dest;
      enum sctp-src;
      enum sctp-dest;
      enum icmpv4-type;
      enum icmpv4-code;
      enum arp-op;
      enum arp-src-ip-address;
      enum arp-target-ip-address;
      enum arp-src-hardware-address;
      enum arp-target-hardware-address;
      enum ipv6-src;
      enum ipv6-dest;
      enum ipv6-flow-label;
      enum icmpv6-type;
      enum icmpv6-code;
      enum ipv6-nd-target;
      enum ipv6-nd-source-link-layer;
      enum ipv6-nd-target-link-layer;
      enum mpls-label;
      enum mpls-tc;
    }
    description "The types of match field defined in OpenFlow
      Switch Specification versions 1.2, 1.3, and 1.3.1.";
  }

  typedef hex-binary {
    type binary;
    description "hex binary encoded string";
    reference "http://www.w3.org/TR/2004/REC-xmlschema-2-
  20041028/datatypes.html#hexBinary";
  }

/*****************************************************************
 * Groupings
 *****************************************************************/

  grouping OFConfigurationPointType {
    description "Representation of an OpenFlow Configuration Point.
      Instances of the Configuration Point class SHOULD be stored
      persistently across reboots of the OpenFlow Capable Switch.

      When a connection is established between an OpenFlow Capable
      Switch and a Configuration Point the switch  MUST store the
      connection information in an instance of the Configuration
      Point class. If such an instance does not exist, the OpenFlow
      Capable Switch MUST create an instance where it then stores
      the connection information.

      An OpenFlow Capable Switch that cannot initiate a connection
      to a configuration point does not have to implement the
      Configuration Point class. It SHOULD block attempts to write
      to instances of the Configuration Point class with NETCONF
      <edit-config> operations.

      NETCONF <edit-config> operations MUST be implemented as
      follows:

      * The 'id' element MUST be present at all <edit-config>
        operations to identify the configuration point.
      * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
      * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data-exists' error is returned.
      * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data-missing'
        error is returned.";
    leaf id {
      type OFConfigId;
      mandatory true;
      description "A unique but locally arbitrary identifier that
        identifies a Configuration Point within the context of an
        OpenFlow Capable Switch.

        This element MUST be present to identify the configuration
        point.";
    }
    leaf uri {
      type inet:uri;
      mandatory true;
      description "A locator of the Configuration Point.  It
        identifies the location of the Configuration Point as a
        service resource and MUST include all information necessary
        for the OpenFlow Capable Switch to connect to the
        Configuration Point or re-connect to it should it become
        disconnected.  Such information MAY include, for example,
        protocol, fully qualified domain name, IP address, port
        number, etc.

        This element MUST be present in the NETCONF data store.
        If this element is not present in a NETCONF <edit-config>
        operation 'create', 'merge' or 'replace' and the parent
        element does not exist, a 'data-missing' error is
        returned.";
    }
    leaf protocol {
      type OFConfigurationPointProtocolType;
      default "ssh";
      description "The transport protocol that the Configuration
        Point uses when communicating via NETCONF with the OpenFlow
        Capable Switch.

        This element is optional. If it is not present its value
        defaults to 'ssh'.";
      reference "The mappings of NETCONF to different transport
        protocols are defined in RFC 6242 for SSH, RFC 4743 for
        SOAP, RFC 4744 for BEEP, and RFC 5539 for TLS";
    }
  }

  grouping OFLogicalSwitchType {
    description "This grouping specifies all properties of an
      OpenFlow Logical Switch.

      Elements of type OFLogicalSwitchType cannot be created or
      deleted with NETCONF <edit-config> operations 'create' or
      'delete'. The other NETCONF <edit-config> operations MUST be
      implemented as follows:

      * The 'id' element MUST be present at all <edit-config>
        operations to identify the OpenFlow Logical Switch.
      * If the operation is 'merge' or 'replace', and the element
        does not exist, a 'data-missing' error is returned. If the
        element exists its value is set to the value found in the
        XML RPC data.
      * If the operation is 'create', a 'operation-not-supported'
        error with type 'application' is returned.
      * If the operation is 'delete', 'operation-not-supported'
        error with type 'application' is returned.";
    leaf id {
      type OFConfigId;
      mandatory true;
      description "A unique but locally arbitrary identifier that
        identifies a Logical Switch within the context of an
        OpenFlow Capable Switch. It MUST be persistent across
        reboots of the OpenFlow Capable Switch.

        This element MUST be present to identify the OpenFlow
        Logical Switch.";
    }
    container capabilities {
      config false;
      description "This element contains all capability items that
        an OpenFlow Logical Switch MAY implement.

        This element and its children can only be retrieved by
        NETCONF <get> operation since it contain no configuration
        data.";
      uses OFLogicalSwitchCapabilitiesType;
    }
    leaf datapath-id {
      type datapath-id-type;
      mandatory true;
      description "The datapath identifier of the Logical Switch
        that uniquely identifies this Logical Switch within the
        context of all OpenFlow Controllers associated with the
        OpenFlow Logical Switch.  The datapath identifier is a
        string value that MUST be formatted as a sequence of 8
        2-digit hexadecimal numbers that are separated by colons,
        for example, '01:23:45:67:89:ab:cd:ef'.  When processing a
        datapath identifier, the case of the decimal digits MUST be
        ignored.

        This element MUST be present in the NETCONF data store.
        If this element is not present in a NETCONF <edit-config>
        operation 'create', 'merge' or 'replace' and the parent
        element does not exist, a 'data-missing' error is
        returned.";
    }
    leaf enabled {
      type boolean;
      default false;
      description "This element indicates the administrative state
        of the OpenFlow Logical Switch.  A value of 'false' means
        the OpenFlow Logical Switch MUST NOT communicate with any
        OpenFlow Controllers, MUST NOT conduct any OpenFlow
        processing, and SHOULD NOT be utilizing computational or
        network resources of the underlying platform.

        This element is optional. If this element is not present it
        defaults to 'false'.";
    }
    leaf check-controller-certificate {
      type boolean;
      default false;
      description "This element indicates the behavior of the
        OpenFlow Logical Switch when connecting to an OpenFlow
        Controller.

        If set to value 'false', the logical switch will connect to
        a controller without checking any controller certificate.

        If set to value 'true', then the logical switch will
        connect to a controller with element <protocol> set to
        'TLS', only if the controller provides a certificate that
        can be verified with one of the certificates stored in the
        list called external-certificates in the OpenFlow Capable
        Switch.

        If a certificate cannot be validated, the OpenFlow Logical
        Switch MUST terminate communication with the corresponding
        OpenFlow Controller, MUST NOT conduct any OpenFlow
        processing on requests of this OpenFlow controller, and
        SHOULD NOT further utilize any computational or network
        resources of for dealing with this connection.

        If set to value 'true', the OpenFlow Logical Switch MUST
        NOT connect to any OpenFlow Controller that does not
        provide a certificate. This implies that it cannot connect
        to an OpenFlow controller that has the value of element
        protocol set to 'TCP'. Only connections with protocol 'TLS'
        are possible in this case.

        This element is optional. If this element is not present it
        defaults to 'false'.";
    }
    leaf lost-connection-behavior {
      type enumeration {
        enum failSecureMode;
        enum failStandaloneMode;
      }
      default failSecureMode;
      description "This element indicates the the behavior of the
        OpenFlow Logical Switch in case it loses contact with all
        OpenFlow Controllers.  There are two alternative modes in
        such a case: fails secure mode and fail standalone mode as
        defined by the OpenFlow protocol specification version 1.2,
        section 6.4.  These are the only allowed values for this
        element. Default is the fail secure mode.

        This element is optional. If this element is not present it
        defaults to 'failSecureMode'.";
    }
    container controllers {
      description "The list of controllers for this Logical switch.

        The element 'id' of OFControllerType MUST be unique within
        this list.";
      list controller {
        key "id";
        description "The list of OpenFlow Controllers that are
          assigned to the OpenFlow Logical Switch.  The switch MUST
          NOT connect to any OpenFlow Controller that is not
          contained in this list.

          NETCONF <edit-config> operations MUST be implemented as
          follows:

          * The 'id' element MUST be present at all <edit-config>
            operations to identify the controller.
          * If the operation is 'merge' or 'replace', the element
```

```
            is created if it does not exist, and its value is set
            to the value found in the XML RPC data.
          * If the operation is 'create', the element is created if
            it does not exist. If the element already exists, a
            'data□exists' error is returned.
          * If the operation is 'delete', the element is deleted if
            it exists. If the element does not exist, a
            'data□missing' error is returned.";
        uses OFControllerType;
      }
    }
    container resources {
      description "The list of identifiers of all resources of the
        OpenFlow Capable Switch that the OpenFlow Logical Switch
        has exclusive or non-exclusive access to.  A resource is
        identified by the value of its resource-identifier element.
        For each resource identifier value in this list, there MUST
        be an element with a matching resource identifier value in
        the resources list of the OpenFlow Capable Switch.

        Identifiers of this list are contained in elements
        indicating the type of resource: 'port', 'queue',
        'certificate', or 'flow-table'.  Depending on the type,
        different constraints apply.  These are specified in
        separate descriptions per type.

        At present the elements in this lists are not configurable
        and can only be retrieved by NETCONF <get> or <get-config>
        operations. Attemps to modify this element and its children
        with a NETCONF <edit-config> operation MUST result in an
        'operation-not-supported' error with type 'application'.";
      leaf-list port {
        type leafref {
          path "/capable-switch/resources/port/resource-id";
        }
        description "A resource identifier of a port of the
          OpenFlow Capable Switch that the OpenFlow Logical Switch
          has exclusive access to.

          The elements in this list MUST refer to elements at the
          following path:
            /capable-switch/resources/port/resource-id

          Elements in this list MUST be unique. This means each
          port element can only be referenced once.";
      }
      leaf-list queue {
        type leafref {
          path "/capable-switch/resources/queue/resource-id";
        }
        description "A resource identifier of a queue of the
          OpenFlow Capable Switch that the OpenFlow Logical Switch
          has exclusive access to.

          The elements in this list MUST refer to elements at the
          following path:
            /capable-switch/resources/queue/resource-id

          Elements in this list MUST be unique. This means each
          queue element can only be referenced once.";
      }
      leaf certificate {
        type leafref {
          path "/capable-switch/resources/owned-certificate/resource-id";
        }
        description "The resource identifier of the owned
          certificate in the OpenFlow Capable Switch that the
          OpenFlow Logical Switch uses to identify itself.  This
          element MUST NOT occur more than once in an OpenFlow
          Logical Switch's resource list.

          If no such element is in an OpenFlow Logical Switch's
          resource list, then the OpenFlow Logical Switch does not
          authenticate itself towards an OpenFloe Controller with a
          certificate.  If this element is present, then the
          OpenFlow Logical Switch MUST provide this certificate for
          authentication to an OpenFlow Controller when setting up
          a TLS connection.

          For TCP connections this element is irrelevant.

          The element MUST refer to an element at the following
          path:
            /capable-switch/resources/owned-certificate/resource-id
          ";
      }
      leaf-list flow-table {
        type leafref {
          path "/capable-switch/resources/flow-table/resource-id";
        }
        description "A resource identifier of a flow table of the
          OpenFlow Capable Switch that the OpenFlow Logical Switch
          has exclusive access to.

          The elements in this list MUST refer to elements at the
          following path:
            /capable-switch/resources/flow-table/resource-id

          Elements in this list MUST be unique. This means each
          flow-table element can only be referenced once.";
      }
    }
  }

  grouping OFLogicalSwitchCapabilitiesType {
    description "This grouping specifies all properties of an
      OpenFlow logical switch's capabilities.

      Elements in the type OFLogicalSwitchCapabilitiesType are not
      configurable and can only be retrieved by NETCONF <get>
      operations. Attemps to modify this element and its children
      with a NETCONF <edit-config> operation MUST result in an
```

```
          'operation-not-supported' error with type 'application'.";
      leaf max-buffered-packets {
        type uint32;
        description "The maximum number of packets the logical switch
          can buffer when sending packets to the controller using
          packet-in messages.";
      }
      leaf max-tables {
        type uint8;
        description "The number of flow tables supported by the
          logical switch.";
      }
      leaf max-ports {
        type uint32;
        description "The number of flow tables supported by the
          logical switch.";
      }
      leaf flow-statistics {
        type boolean;
        default false;
        description "Specifies if the logical switch supports flow
          statistics.";
      }
      leaf table-statistics {
        type boolean;
        default false;
        description "Specifies if the logical switch supports table
          statistics.";
      }
      leaf port-statistics {
        type boolean;
        default false;
        description "Specifies if the logical switch supports port
          statistics.";
      }
      leaf group-statistics {
        type boolean;
        default false;
        description "Specifies if the logical switch supports group
          statistics.";
      }
      leaf queue-statistics {
        type boolean;
        default false;
        description "Specifies if the logical switch supports queue
          statistics.";
      }
      leaf reassemble-ip-fragments {
        type boolean;
        default false;
        description "Specifies if the logical switch supports
          reassemble IP fragments.";
      }
      leaf block-looping-ports {
        type boolean;
        default false;
        description "'true' indicates that a switch protocol outside
          of OpenFlow, such as 802.1D Spanning Tree, will detect
          topology loops and block ports to prevent packet loops.";
      }
      container reserved-port-types {
        description "Specify generic forwarding actions such as
          sending to the controller, flooding, or forwarding using
          non-OpenFlow methods, such as 'normal' switch processing.";
        reference "The types of reserved ports are defined in
           OpenFlow Switch Specification versions 1.2, 1.3, and
           1.3.1.";
        leaf-list type {
          type enumeration {
            enum all;
            enum controller;
            enum table;
            enum inport;
            enum any;
            enum normal;
            enum flood;
          }
        }
      }
      container group-types {
        description "Specify the group types supported by the logical
          switch.";
        reference "The types of groups are defined in OpenFlow Switch
          Specification versions 1.2, 1.3, and 1.3.1.";
        leaf-list type {
          type enumeration {
            enum all;
            enum select;
            enum indirect;
            enum fast-failover;
          }
        }
      }
      container group-capabilities {
        description "Specify the group capabilities supported by the
          logical switch.";
        reference "The types of group capability are defined in
          OpenFlow Switch Specification versions 1.2, 1.3, and
          1.3.1.";
        leaf-list capability {
          type enumeration {
            enum select-weight;
            enum select-liveness;
            enum chaining;
            enum chaining-check;
          }
        }
      }
      container action-types {
        description "Specify the action types supported by the
          logical switch.";
        leaf-list type {
          type OFActionType;
        }
      }
      container instruction-types {
        description "Specify the instruction types supported by the
          logical switch.";
        leaf-list type {
          type OFInstructionType;
        }
      }
    }

    grouping OFControllerType {
      description "This grouping specifies all properties of an
        OpenFlow Logical Switch Controller.

        NETCONF <edit-config> operations MUST be implemented as
        follows:

        * The 'id' element MUST be present at all <edit-config>
          operations to identify the controller.
        * If the operation is 'merge' or 'replace', the element is
          created if it does not exist, and its value is set to the
          value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
          does not exist. If the element already exists, a
          'data□exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
          exists. If the element does not exist, a 'data□missing'
          error is returned.";
      leaf id {
        type OFConfigId;
        mandatory true;
        description "A unique but locally arbitrary identifier that
          uniquely identifies an OpenFlow Controller within the
          context of an OpenFlow Capable Switch.  It MUST be
          persistent across reboots of the OpenFlow Capable Switch.

          This element MUST be present to identify the OpenFlow
          controller.";
      }
      leaf role {
        type enumeration {
          enum master;
          enum slave;
          enum equal;
        }
        default equal;
        description "This element indicates the role of the OpenFlow
          Controller.  Semantics of these roles are specified in the
          OpenFlow specifications 1.0 - 1.3.1.  It is RECOMMENDED
          that the roles of controllers are not configured by
          OF-CONFIG 1.1.1 but determined using the OpenFlow protocol.
          OpenFlow Controllers configured by OF-CONFIG 1.1.1 have the
          default role 'equal'.  A role other than 'equal' MAY be
          assigned to a controller.  Roles 'slave' and 'equal' MAY be
          assigned to multiple controllers.  Role 'master' MUST NOT
          be assigned to more than one controller.

          This element is optional. If this element is not present it
          defaults to 'equal'.";
      }
      leaf ip-address {
        type inet:ip-address;
        mandatory true;
        description "The IP address of the OpenFlow Controller.  This
          IP address is used by the OpenFlow Logical Switch when
          connecting to the OpenFlow Controller.

          This element MUST be present in the NETCONF data store.
          If this element is not present in a NETCONF <edit-config>
          operation 'create', 'merge' or 'replace' and the parent
          element does not exist, a 'data-missing' error is
          returned.";
      }
      leaf port {
        type inet:port-number;
        default 6633;
        description "The TCP port number at the OpenFlow Controller.
          This port number is used by the OpenFlow Logical Switch
          when connecting to the OpenFlow Controller using TCP or
          TLS.  The default value is 6633.

          This element is optional. If this element is not present it
          defaults to 6633.";
      }
      leaf local-ip-address {
        type inet:ip-address;
        description "The local IP address of the OpenFlow Logical
          Switch when connecting to this OpenFlow Controller.  It is
          the source IP address of packets sent to this OpenFlow
          Controller.  If present, this element overrides any default
          IP address.


          This element is optional. Attempts to set this element to
          an IP address that cannot be used by the OpenFlow Logical
          Switch MUST result in an 'bad-element' error with type
          'application'. The <error-info> element MUST contain the
          name of this element in the <bad-element> element.";
      }
      leaf local-port {
        type inet:port-number;
        description "The local TCP port number of the OpenFlow
          Logical Switch when connecting to this OpenFlow Controller.
          It is the source TCP port number of packets sent to this
          OpenFlow Controller.  If this element is not present, then
          the port number is chosen arbitrarily by the OpenFlow
          Logical Switch.

          This element is optional. Attempts to set this element to a
          port number that cannot be used by the OpenFlow Logical
          Switch MUST result in an 'bad-element' error with type
          'application'. The <error-info> element MUST contain the
          name of this element in the <bad-element> element.";
      }
      leaf protocol {
        type enumeration {
          enum "tcp";
          enum "tls";
        }
        default "tls";
        description "The default protocol tha the OpenFlow Logical
          Switch uses to connect to this OpenFlow Controller.  'tls'
          is the default value.

          This element is optional. If this element is not present it
          defaults to 'tls'.";
      }
      container state {
        config false;
        description "This container holds connection state
          information that indicate the connection state of the
          OpenFlow Logical Switch and the OpenFlow protocol version
          used for the connection.

          Children of this element are not configurable and can only
          be retrieved by NETCONF <get> operations. Attemps to modify
          this element and its children with a NETCONF <edit-config>
          operation MUST result in an 'operation-not-supported' error
          with type 'application'.";
        leaf connection-state {
          type OFUpDownStateType;
          description "This object indicates the connections state of
            the OpenFlow Logical Switch to this controller.";
        }
        leaf current-version {
          type OFOpenFlowVersionType;
          description "This object indicates the version of the
            OpenFlow protocol used between the OpenFlow Logical
            Switch and this Controller.  If element connection-state
            has value 'up', then this element indicates the actual
            version in use.  If element connection-state has value
            'down', then this element indicates the version number of
            the last established connection with this OpenFlow
            Controller.  The value of this element MAY be persistent
            across reboots of the OpenFlow Logical Switch in such a
            case.  If element connection-state has value 'down'and
            there is no information about previous connections to
            this OpenFlow controller, then this element is not
            present or has the value '0'.";
        }
        leaf-list supported-versions {
          type OFOpenFlowVersionType;
          description "This list of elements includes one entry for
            each OpenFlow protocol version that this OpenFlow
            controller supports.  It SHOULD contain all";
        }
        leaf local-ip-address-in-use {
          type inet:ip-address;
          description "The local IP address of the OpenFlow Logical
            Switch when connecting to this OpenFlow Controller.  It
            is the source IP address of packets sent to this OpenFlow
            Controller.  If present, this element overrides any
            default IP address.";
        }
        leaf local-port-in-use {
          type inet:port-number;
          description "The local TCP port number of the OpenFlow
            Logical Switch.  If element connection-state has value
            'up', then this element indicates the actual port number
            in use.  If element connection-state has value 'down',
            then this element indicates the port number used for the
            last attempt to establish a connection with this OpenFlow
            Controller.???
            When connecting to this OpenFlow Controller, it is the
            source TCP port number of packets sent to this OpenFlow
            Controller.  If this element has its defaqult value 0,
            then port number is chosen arbitrarily by the OpenFlow
            Logical Switch.";
        }
      }
    }

    grouping OFResourceType {
      description "This element specifies a generic OpenFlow resource
        that is used as a basis for specific resources. Even though
        this element is not used on its own the following rules for
        NETCONF operations MUST be obeyed also by elemnts using this
        element.

        NETCONF <edit-config> operations MUST be implemented as
        follows:

        * The 'id' element MUST be present at all <edit-config>
          operations to identify the resource.
        * If the operation is 'merge' or 'replace', the element is
          created if it does not exist, and its value is set to the
          value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
          does not exist. If the element already exists, a
          'data□exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
          exists. If the element does not exist, a 'data□missing'
          error is returned.";
      leaf resource-id {
        type inet:uri;
        mandatory true;
        description "A unique but locally arbitrary identifier that
          uniquely identifies an OpenFlow Port within the context
          of an OpenFlow Logical Switch.  It MUST be persistent
          across reboots of the OpenFlow Capable Switch.

          This element MUST be present to identify the OpenFlow
          resource.";
      }
    }

    grouping OFPortBaseTunnelType {
      description "A group of common elements that are included
        in every supported tunnel type.  Tunnels are modeled as
        logical ports.

        One pair of local/remote endpoints must exist for a tunnel
        configuration.

        Only elements from one choice must exist at a time.";
      choice endpoints {
        mandatory true;
        case v4-endpoints {
          leaf local-endpoint-ipv4-adress {
            type inet:ipv4-address;
            description "The IPv4 address of the local tunnel
              endpoint.";
          }
          leaf remote-endpoint-ipv4-adress {
            type inet:ipv4-address;
            description "The IPv4 address of the remote tunnel
              endpoint.";
          }
        }
        case v6-endpoints {
          leaf local-endpoint-ipv6-adress {
            type inet:ipv6-address;
            description "The IPv6 address of the local tunnel
              endpoint.";
          }
          leaf remote-endpoint-ipv6-adress {
            type inet:ipv6-address;
            description "The IPv6 address of the remote tunnel
              endpoint.";
          }
        }
        case mac-endpoints {
          leaf local-endpoint-mac-adress {
            type yang:mac-address;
            description "The MAC address of the local tunnel
              endpoint.";
          }
          leaf remote-endpoint-mac-adress {
            type yang:mac-address;
            description "The MAC address of the remote tunnel
              endpoint.";
          }
        }
      }
    }

    grouping OFPortIPGRETunnelType {
      description "Properties of a IP-in-GRE tunnel with key,
        checksum, and sequence number information.";
      uses OFPortBaseTunnelType;
      leaf checksum-present {
        type boolean;
        default true;
        description "Indicates presence of the GRE checksum.";
      }
      leaf key-present {
        type boolean;
        default true;
        description "Indicates presence of the GRE key.";
      }
      leaf key {
        when "../key-present='true'" {
          description "This element is only relevant if element
            key-present of this IP GRE Tunnel has value 'true'.";
        }
        type uint32;
        mandatory true;
        description "The (optional) key of the GRE tunnel.  It MAY
          be used to set the OXM_OF_TUNNEL_ID match field metadata
          in the OpenFlow protocol";
      }
      leaf sequence-number-present {
        type boolean;
        default false;
        description "Indicates presence of the GRE sequence
          number.";
      }
    }

    grouping OFPortVXLANTunnelType {
      description "Properties of a VxLAN tunnel.";
      uses OFPortBaseTunnelType;
      leaf vni-valid {
        type boolean;
        default true;
        description "Indicates how the corresponding flag should be
          set in packets sent on the tunnel.";
      }
      leaf vni {
        type uint32;
        description "Virtual network identifier assigned to all
          packets sent on the tunnel.  A VxLAN  implementation MAY
          use the this element to set the OXM_OF_TUNNEL_ID match
          field metadata in the OpenFlow protocol.";
      }
      leaf vni-multicast-group {
        type inet:ip-address;
        description "If IP multicast is used to support broadcast
          on the tunnel this specifies the corresponding multicast
          IP address";
      }
      leaf udp-source-port {
        type inet:port-number;
        description "Specifies the outer UDP source port number.
```

```
          If this element is absent, the port number MAY be chosen
          dynamically.";
      }
      leaf udp-dest-port {
        type inet:port-number;
        description "Specifies the outer UDP destination port
          number.  It is intended to reserve a port number for
          VxLAN at IANA.  As soon as this has been reserved, the
          reserved number SHOULD become the default value for this
          element.";
      }
      leaf udp-checksum {
        type boolean;
        default false;
        description "Boolean flag to indicate whether or not the
          outer UDP checksum should be set";
      }
    }

    grouping OFPortNVGRETunnelType {
      description "Properties of a NVGRE tunnel.";
      uses OFPortBaseTunnelType;
      leaf tni {
        type uint32;
        description "Specifies the tenant network identifier
          assigned to all packets sent on the tunnel";
      }
      leaf tni-resv {
        type uint32;
        description "Used to set the reserved user-defined bits of
          the GRE key field";
      }
      leaf tni-multicast-group {
        type inet:ip-address;
        description "If IP multicast is used to support broadcast
          on the tunnel this element specifies the corresponding
          multicast IP address";
      }
    }

    grouping OFPortType {
      description "This element specifies all properties of an
        OpenFlow resource of type OpenFlow Port. It represent a
        physical port or a logical port of the OpenFlow Capable
        Switch and can be assigned for exclusive use to an OpenFlow
        Logical Switch.  A logical port represents a tunnel endpoint
        as described in the OpenFlow protocol specification versions
        1.3 - 1.3.1.

        NETCONF <edit-config> operations MUST be implemented as
        follows:

        * The 'resource-id' element of OFResoureType MUST be present
          at all <edit-config> operations to identify the port.
        * If the operation is 'merge' or 'replace', the element is
          created if it does not exist, and its value is set to the
          value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
          does not exist. If the element already exists, a
          'data‑exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
          exists. If the element does not exist, a 'data‑missing'
          error is returned.";
      uses OFResourceType;
      leaf number {
        type uint64;
        config false;
        description "This number identifies the OpenFlow Port to
          OpenFlow Controllers. It is assigned to an OpenFlow Port
          latest when the OpenFlow Port is associated with and
          OpenFlow Logical Switch.  If the OpenFlow Port is
          associated with an OpenFlow Logical Switch, this element
          MUST be unique within the context of the OpenFlow Logical
          Switch.

          OpenFlow Capable Switch implementations may choose to
          assign values to OpenFlow Ports that are unique within the
          context of the OpenFlow Logical Switch.  These numbers can
          be used independent of assignments to OpenFlow Logical
          Switches.

          Other implementations may assign values to this element
          only if the OpenFlow Port is assigned to an OpenFlow
          Logical Switch.  If no value is currently assigned to this
          element then this element MUST NOT be included in replies
          to NETCONF <get> requests. Since this element is not
          configurable with the NETCONF protocol it MUST NOT be
          included in replies to NETCONF <get-config> requests.";
      }
      leaf name {
        type string { length "1..16"; }
        config false;
        description "This element assists OpenFlow Controllers in
          identifying OpenFlow Ports.

          This element is not to be set by the OP-CONFIG protocol,
          but it is set by the switch implementation.  It may be set
          at start-up time of an OpenFlow Capable Switch or when the
          OpenFlow Port is assigned to an OpenFlow Logical Switch.
          It MAY also be not set at all.  If this element is set to a
          value other than the empty string when being assigned to an
          OpenFlow Logical Switch, then the value of this element
          MUST be unique within the context of the OpenFlow Logical
          Switch.

          If no value or the empty string is currently assigned to
          this element then this element MUST not be included in
          replies to NETCONF <get> requests. Since this element is
          not configurable with the NETCONF protocol it MUST NOT be
          included in replies to NETCONF <get-config> requests.";
      }
      leaf current-rate {
        when "../features/current/rate='other'" {
          description "This element is only valid if the element rate
            of the current features has value 'other'.";
        }
        type uint32;
        units "kbit/s";
        config false;
        description "This element indicates the current bit rate of
          the port. Its values is to be provided in units of kilobit
          per second (kbps). This element is only valid if the
          element called 'rate' in the current Port Features has a
          value of 'other'.

          Since this element is not configurable with the NETCONF
          protocol it MUST NOT be included in replies to NETCONF
          <get-config> requests.";
      }
      leaf max-rate {
        when "../features/current/rate='other'" {
          description "This element is only valid if the element rate
            of the current features has value 'other'.";
        }
        type uint32;
        units "kbit/s";
        config false;
        description "This element indicates the maximum bit rate of
          the port. Its values is to be provided in units of kilobit
          per second (kbps). This element is only valid if the
          element called 'rate' in the current Port Features has a
          value of 'other'.

          Since this element is not configurable with the NETCONF
          protocol it MUST NOT be included in replies to NETCONF
          <get-config> requests.";
      }
      container configuration {
        description "This element represents the general
          adminitrative configuration of the OpenFlow Port.";
        leaf admin-state {
          type OFUpDownStateType;
          default 'up';
          description "The administrative state of the port.  If
            true, the port has been administratively brought down and
            SHOULD not be used by OpenFlow.

            This element is optional. If this element is not present
            it defaults to 'up'.";
        }
        leaf no-receive {
          type boolean;
          default false;
          description "If true, packets received at this OpenFlow
            port SHOULD be dropped.

            This element is optional. If this element is not present
            it defaults to 'false'.";
        }
        leaf no-forward {
          type boolean;
          default false;
          description "If true, packets forwarded to this OpenFlow
            port SHOULD be dropped.

            This element is optional. If this element is not present
            it defaults to 'false'.";
        }
        leaf no-packet-in {
          type boolean;
          default false;
          description "If true, packets received on that port that
            generate a table miss should never trigger a packet-in
            message to the OpenFlow Controller.

            This element is optional. If this element is not present
            it defaults to 'false'.";
        }
      }
      container state {
        config false;
        description "This element represents the general operational
        state of the OpenFlow Port.

        Children of this element are not configurable and can only be
        retrieved by NETCONF <get> operations. Attemps to modify this
        element and its children with a NETCONF <edit-config>
        operation MUST result in an 'operation-not-supported' error
        with type 'application'.";
        leaf oper-state {
          type OFUpDownStateType;
          description "If the value of this element is 'down', it
            indicates that there is no physical link present.";
        }
        leaf blocked {
          type boolean;
          description "If the value of this element is 'true', it
            indicates that a switch protocol outside of OpenFlow,
            such as 802.1D Spanning Tree, is preventing the use of
            this OpenFlow port for OpenFlow flooding.";
        }
        leaf live {
          type boolean;
          description "If the value of this element is 'true', it
            indicates that this OpenFlow Port is live and can be used
            for fast failover.";
        }
      }
      container features {
        container current {
          uses OFPortCurrentFeatureListType;
          config false;
          description "The features (rates, duplex, etc.) of the
            port, that are currently in use.
            Children of this element are not configurable and can
            only be retrieved by NETCONF <get> operations. Attemps to
            modify this element and its children with a NETCONF
            <edit-config> operation MUST result in an
            'operation-not-supported' error with type
            'application'.";
        }
        container advertised {
          uses OFPortOtherFeatureListType;
          description "The features (rates, duplex, etc.) of the
            port, that are advertised to the peer port.

            NETCONF <edit-config> operations MUST be implemented as
            follows:

            * The 'resource-id' element of OFResoureType MUST be
              present in the path or in the filter at all
              <edit-config> operations to identify the port.
            * If the operation is 'merge' or 'replace', the element
              is created if it does not exist, and its value is set
              to the value found in the XML RPC data.
            * If the operation is 'create', the element is created if
              it does not exist. If the element already exists, a
              'data-exists' error is returned.
            * If the operation is 'delete', the element is deleted if
              it exists. If the element does not exist, a
              'data-missing' error is returned.";
        }
        container supported {
          uses OFPortOtherFeatureListType;
          config false;
          description "The features (rates, duplex, etc.) of the
            port, that are supported on the port.

            Children of this element are not configurable and can
            only be retrieved by NETCONF <get> operations. Attemps to
            modify this element and its children with a NETCONF
            <edit-config> operation MUST result in an
            'operation-not-supported' error with type
            'application'.";
        }
        container advertised-peer {
          uses OFPortOtherFeatureListType;
          config false;
          description "The features (rates, duplex, etc.) that are
            currently advertised by the peer port.

            Children of this element are not configurable and can
            only be retrieved by NETCONF <get> operations. Attemps to
            modify this element and its children with a NETCONF
            <edit-config> operation MUST result in an
            'operation-not-supported' error with type
            'application'.";
        }
      }
      choice tunnel-type {
        description "Tunnels are modeled as logical ports.

          Elements in this choice are not configurable and can only
          be retrieved by NETCONF <get> operations. Attemps to modify
          this element and its children with a NETCONF <edit-config>
          operation MUST result in an 'operation-not-supported' error
          with type 'application'.

          Only elements from one choice must exist at a time.";
        container tunnel {
          description "Properties of a basic IP-in-GRE tunnel.";
          uses OFPortBaseTunnelType;
        }
        container ipgre-tunnel {
          description "Properties of a IP-in-GRE tunnel.";
          uses OFPortIPGRETunnelType;
        }
        container vxlan-tunnel {
          description "Properties of a VxLAN tunnel.";
          uses OFPortVXLANTunnelType;
        }
        container nvgre-tunnel {
          description "Properties of a NVGRE tunnel.";
          uses OFPortNVGRETunnelType;
        }
      }
    }

    grouping OFQueueType {
      description "This grouping specifies all properties of a queue
        resource.

        NETCONF <edit-config> operations MUST be implemented as
        follows:

        * The 'resource-id' element of OFResoureType MUST be present
          at all <edit-config> operations to identify the port.
        * If the operation is 'merge' or 'replace', the element is
          created if it does not exist, and its value is set to the
          value found in the XML RPC data.
        * If the operation is 'create', the element is created if it
          does not exist. If the element already exists, a
          'data‐exists' error is returned.
        * If the operation is 'delete', the element is deleted if it
          exists. If the element does not exist, a 'data‐missing'
          error is returned.";
      uses OFResourceType;
      leaf id {
        type uint64;
        mandatory true;
        description "This id identifies the OpenFlow Queue to
          OpenFlow Controllers. It is assigned to an OpenFlow Queue
          latest when the OpenFlow Queue is associated with and
          OpenFlow Logical Switch.  If the OpenFlow Queue is
          associated with an OpenFlow Logical Switch, this element
          MUST be unique within the context of the OpenFlow Logical
          Switch.

          OpenFlow Capable Switch implementations may choose to
          assign values to OpenFlow Queues that are unique within the
          context of the OpenFlow Logical Switch.  These id can be
          used independent of assignments to OpenFlow Logical
          Switches.

          Other implementations may assign values to this element
          only if the OpenFlow Queue is assigned to an OpenFlow
          Logical Switch.  If no value is currently assigned to this
          element then this element MUST NOT be included in replies
          to NETCONF <get> requests. Since this element is not
          configurable with the NETCONF protocol it MUST NOT be
          included in replies to NETCONF <get-config> requests.";
      }
      leaf port {
        type leafref {
          path "/capable-switch/resources/port/resource-id";
        }
        description "Reference to port resources in the Capable
          Switch.

          This element associates an OpenFlow Queue with an OpenFlow
          Port. If the OpenFlow Queue is associated with an OpenFlow
          Logical Switch S and this element is present, then it MUST
          be set to the value of element resource-id of an OpenFlow
          Port which is associated with the OpenFlow Logical Switch
          S.

          The element MUST refer to an element at the following path:
            /capable-switch/resources/port/resource-id";
      }
      container properties {
        description "The queue properties currently configured.";
        leaf min-rate {
          type OFTenthOfAPercentType;
          description "The minimal rate that is reserved for this
            queue in 1/10 of a percent of the actual rate.

            This element is optional. If not present a min-rate is
            not set.";
        }
        leaf max-rate {
          type OFTenthOfAPercentType;
          description "The maximum rate that is reserved for this
            queue in 1/10 of a percent of the actual rate.

            This element is optional. If not present the max-rate is
            not set.";
        }
        leaf-list experimenter {
          type uint32;
          description "A list of experimenter identifiers of queue
            properties used.
            This element is optional.";
        }
      }
    }

    grouping OFPortCurrentFeatureListType {
      description "The current features of a port.

        Elements in the type OFPortCurrentFeatureListType are not
        configurable and can only be retrieved by NETCONF <get>
        operations. Attemps to modify this element and its children
        with a NETCONF <edit-config> operation MUST result in an
        'operation-not-supported' error with type 'application'.";
      leaf rate {
        type OFPortRateType;
        description "The transmission rate that is currently used.
          The value MUST indicate a valid forwarding rate.

          The current Port Feature set MUST contain this element
          exactly once.  The other Port Feature sets MAY contain this
          element more than once.  If this element appears more than
          once in a Port Feature set than the value MUST be unique
          within the Port Feature set.";
      }
      leaf auto-negotiate {
        type boolean;
        description "Specifies the administrative state of the
          forwarding rate auto-negotiation protocol at this OpenFlow
          Port.";
      }
      leaf medium {
        type enumeration {
          enum copper;
          enum fiber;
        }
        description "This element MUST indicate a valid physical
          medium used by the OpenFlow Port.

          The current Port Feature set MUST contain this element
          exactly once. The other Port Feature sets MAY contain this
          element more than once. If this element appears more than
          once in a Port Feature set than the value MUST be unique
          within the Port Feature set.";
      }
      leaf pause {
        type enumeration {
          enum unsupported;
          enum symmetric;
          enum asymmetric;
        }
        description "Specifies if pausing of transmission is
          supported at all and if yes if it is asymmetric or
          symmetric.";
      }
    }
```

```
grouping OFPortOtherFeatureListType {
  description "The features of a port that are supported or
    advertised.

    If the elements in the OFPortOtherFeatureListType ares used
    as configurable elements the NETCONF <edit-config> operations
    MUST be implemented as follows:

    * The 'resource-id' element MUST be present in the path or in
      the filter at all <edit-config> operations to identify the
      resource.
    * If the operation is 'merge' or 'replace', the element is
      created if it does not exist, and its value is set to the
      value found in the XML RPC data.
    * If the operation is 'create', the element is created if it
      does not exist. If the element already exists, a
      'data□exists' error is returned.
    * If the operation is 'delete', the element is deleted if it
      exists. If the element does not exist, a 'data□missing'
      error is returned.

    If elements in the type OFPortOtherFeatureListType are used
    in an non-configurable way, they only be retrieved by NETCONF
    <get> operations. Attemps to modify this element and its
    children with a NETCONF <edit-config> operation MUST result
    in an 'operation-not-supported' error with type
    'application'.";
  leaf-list rate {
    type OFPortRateType;
    min-elements 1;
    description "The transmission rate that is supported or
      advertised. Multiple transmissions rates are allowed.

      At least one element MUST be present in the NETCONF data
      store. If none of this elements is are present in a NETCONF
      <edit-config> operation 'create', 'merge' or 'replace' and
      the parent element does not exist, a 'data-missing' error
      is returned.";
  }
  leaf auto-negotiate {
    type boolean;
    default true;
    description "Specifies if auto-negotiation of transmission
      parameters is enabled for the port.

      This element is optional. If this element is not present it
      defaults to 'true'.";
  }
  leaf-list medium {
    type enumeration {
      enum copper;
      enum fiber;
    }
    min-elements 1;
    description "The transmission medium used by the port.
```

```
        Multiple media are allowed.

        At least one element MUST be present in the NETCONF data
        store. If none of this elements is are present in a NETCONF
        <edit-config> operation 'create', 'merge' or 'replace' and
        the parent element does not exist, a 'data-missing' error
        is returned.";
    }
    leaf pause {
      type enumeration {
        enum unsupported;
        enum symmetric;
        enum asymmetric;
      }
      mandatory true;
      description "Specifies if pausing of transmission is
        supported at all and if yes if it is asymmetric or
        symmetric.

        This element MUST be present in the NETCONF data store.
        If this element is not present in a NETCONF <edit-config>
        operation 'create', 'merge' or 'replace' and the parent
        element does not exist, a 'data-missing' error is
        returned.";
    }
  }

  grouping OFExternalCertificateType {
    description "This grouping specifies a certificate that can be
      used by an OpenFlow Logical Switch for authenticating a
      controller when a TLS connection is established.";
    uses OFResourceType;
    leaf certificate {
      type string;
      mandatory true;
      description "An X.509 certificate in DER format base64
        encoded.

        This element MUST be present in the NETCONF data store.
        If this element is not present in a NETCONF <edit-config>
        operation 'create', 'merge' or 'replace' and the parent
        element does not exist, a 'data-missing' error is
        returned.";
    }
  }

  grouping OFOwnedCertificateType {
    description "This grouping specifies 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.";
    uses OFResourceType;
    leaf certificate {
      type string;
      mandatory true;
      description "An X.509 certificate in DER format base64
```

```
        encoded.

        This element MUST be present in the NETCONF data store.
        If this element is not present in a NETCONF <edit-config>
        operation 'create', 'merge' or 'replace' and the parent
        element does not exist, a 'data-missing' error is
        returned.";
    }
    container private-key {
      uses KeyValueType;
      description "This element contains the private key
        corresponding to the certificate. The private key is
        encoded as specified in XML-Signature Syntax and Processing
        (http://www.w3.org/TR/2001/PR-xmldsig-core-20010820/).
        Currently the specification only support DSA and RSA keys.

        This element MUST be present in the NETCONF data store.
        If this element is not present in a NETCONF <edit-config>
        operation 'create', 'merge' or 'replace' and the parent
        element does not exist, a 'data-missing' error is
        returned.";
    }
  }

  grouping KeyValueType {
    description "The KeyValue element contains a single public key
      that may be useful in validating the signature.

      NETCONF <edit-config> operations MUST be implemented as
      follows:

      * Exactly one of the elemenst 'DSAKeyValue' or 'RSAKeyValue'
        all <edit-config> operations.
      * If the operation is 'merge' or 'replace', the element is
        created if it does not exist, and its value is set to the
        value found in the XML RPC data.
      * If the operation is 'create', the element is created if it
        does not exist. If the element already exists, a
        'data exists' error is returned.
      * If the operation is 'delete', the element is deleted if it
        exists. If the element does not exist, a 'data missing'
        error is returned.";
    choice key-type {
      mandatory true;
      case dsa {
        container DSAKeyValue {
          uses DSAKeyValueType;
        }
      }
      case rsa {
        container RSAKeyValue {
          uses RSAKeyValueType;
        }
      }
    }
  }
```

```
grouping DSAKeyValueType {
  description "DSA keys and the DSA signature algorithm are
  specified in 'FIPS PUB 186-2, Digital Signature Standard (DSS),
  U.S. Department of Commerce/National Institute of Standards and
  Technology,
  http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf'.
  DSA public key values can have the following fields:

  P
      a prime modulus meeting the requirements of the standard
      above
  Q
      an integer in the range 2**159 < Q < 2**160 which is a
      prime divisor of P-1
  G
      an integer with certain properties with respect to P and Q
  J
      (P - 1) / Q
  Y
      G**X mod P (where X is part of the private key and not made
      public)
  seed
      a DSA prime generation seed
  pgenCounter
      a DSA prime generation counter

  Parameter J is avilable for inclusion solely for efficiency as
  it is calculatable from P and Q. Parameters seed and
  pgenCounter are used in the DSA prime number generation
  algorithm specified in the above standard. As such, they are
  optional but MUST either both be present or both be absent.
  This prime generation algorithm is designed to provide
  assurance that a weak prime is not being used and it yields a P
  and Q value. Parameters P, Q, and G can be public and common to
  a group of users. They might be known from application context.
  As such, they are optional but P and Q MUST either both appear
  or both be absent. If all of P, Q, seed, and pgenCounter are
  present, implementations are not required to check if they are
  consistent and are free to use either P and Q or seed and
  pgenCounter. All parameters are encoded as base64 values.";
  leaf P {
    when "count(../Q) != 0";
    type binary;
    mandatory true;
    description "This element is optional. It MUST be present in
      the NETCONF data store, if the element 'Q' is present.

      If element 'Q' is present in a NETCONF <edit-config>
      operation 'create', 'merge' or 'replace' and this element
      is missing, a 'data-missing' error is returned.";
  }
  leaf Q {
    when "count(../P) != 0";
    type binary;
    mandatory true;
```

```
      description "This element is optional. It MUST be present in
        the NETCONF data store, if the element 'P' is present.

        If element 'P' is present in a NETCONF <edit-config>
        operation 'create', 'merge' or 'replace' and this element
        is missing, a 'data-missing' error is returned.";
    }
    leaf J {
      type binary;
      description "This element is optional.";
    }
    leaf G {
      type binary;
      description "This element is optional.";
    }
    leaf Y {
      type binary;
      mandatory true;
      description "This element MUST be present in the NETCONF data
        store. If this element is not present in a NETCONF
        <edit-config> operation 'create', 'merge' or 'replace' and
        the parent element does not exist, a 'data-missing' error
        is returned.";
    }
    leaf Seed {
      when "count(../PgenCounter) != 0";
      type binary;
      mandatory true;
      description "This element is optional. It MUST be present in
        the NETCONF data store, if the element 'PgenCounter' is
        present.

        If element 'PgenCounter' is present in a NETCONF
        <edit-config> operation 'create', 'merge' or 'replace' and
        this element is missing, a 'data-missing' error is
        returned.";
    }
    leaf PgenCounter {
      when "count(../Seed) != 0";
      type binary;
      mandatory true;
      description "This element is optional. It MUST be present in
        the NETCONF data store, if the element 'Seed' is present.

        If element 'Seed' is present in a NETCONF <edit-config>
        operation 'create', 'merge' or 'replace' and this element
        is missing, a 'data-missing' error is returned.";
    }
  }

  grouping RSAKeyValueType {
    description "RSA key values have two fields: Modulus and
      Exponent.";
    leaf Modulus {
      type binary;
      mandatory true;
      description "This element MUST be present in the NETCONF data
        store. If this element is not present in a NETCONF
        <edit-config> operation 'create', 'merge' or 'replace' and
        the parent element does not exist, a 'data-missing' error
        is returned.";
    }
    leaf Exponent {
      type binary;
      mandatory true;
      description "This element MUST be present in the NETCONF data
        store. If this element is not present in a NETCONF
        <edit-config> operation 'create', 'merge' or 'replace' and
        the parent element does not exist, a 'data-missing' error
        is returned.";
    }
  }

  grouping OFFlowTableType {
    description "Representation of an OpenFlow Flow Table Resource.

      Elements in the type OFFlowTableType are not configurable and
      can only be retrieved by NETCONF <get> operations. Attemps to
      modify this element and its children with a NETCONF
      <edit-config> operation MUST result in an
      'operation-not-supported' error with type 'application'.";
    uses OFResourceType;
    leaf max-entries {
      type uint8;
      description "The maximum number of flow entries supported by
        the flow table.";
    }
    container next-tables {
      leaf-list table-id {
        type inet:uri;
      }
      description "An array of resource-ids of all flow tables that
        can be directly reached from this table using the
        'goto-table' instruction.";
    }
    container instructions {
      leaf-list type {
        type OFInstructionType;
      }
      description "The list of all instruction types supported by
        the flow table.";
    }
    container matches {
      leaf-list type {
        type OFMatchFieldType;
      }
      description "The list of all match types supported by the
        flow table.";
    }
    container write-actions {
      leaf-list type {
        type OFActionType;
      }
      description "The list of all write action types supported by
        the flow table.";
    }
    container apply-actions {
      leaf-list type {
        type OFActionType;
      }
      description "The list of all apply action types supported by
        the flow table.";
    }
    container write-setfields {
      leaf-list type {
        type OFMatchFieldType;
      }
      description "The list of all 'set-field' action types
        supported by the table using write actions.";
    }
    container apply-setfields {
      leaf-list type {
        type OFMatchFieldType;
      }
      description "The list of all 'set-field' action types
        supported by the table using apply actions.";
    }
    container wildcards {
      leaf-list type {
        type OFMatchFieldType;
      }
      description "The list of all fields for which the table
        supports wildcarding.";
    }
    leaf metadata-match {
      type hex-binary;
      description "This element indicates the bits of the metadata
        field on which the flow table can match.  It is represented
        as 64-bit integer in hexadecimal digits([0-9a-fA-F])
        format.";
    }
    leaf metadata-write {
      type hex-binary;
      description "This element indicates the bits of the metadata
        field on which flow table can write using the
        'write-metadata' instruction.  It is represented as
        64-bit integer in hexadecimal digits([0-9a-fA-F]) format.";
    }
  }

/*****************************************************************
 * Main container
 *****************************************************************/

  container capable-switch {
    description "The OpenFlow Capable Switch serves as the root
      element for an OpenFlow configuration.  It contains logical
      switches and resources that can be assigned to logical
      switches.  It may have relations to OpenFlow Configuration
      Points.";
    leaf id {
      type inet:uri;
      mandatory true;
      description "A unique but locally arbitrary identifier that
        uniquely identifies a Capable Switch within the context of
        potential OpenFlow Configuration Points.  It MUST be
        persistent across reboots of the OpenFlow Capable Switch.

        This element MUST be present in the NETCONF data store.
        If this element is not present in a NETCONF <edit-config>
        operation 'create', 'merge' or 'replace' and the parent
        element does not exist, a 'data-missing' error is
        returned.";
    }
    leaf config-version {
      type string;
      config false;
      description "The maximum supported OF-CONFIG version that is
        supported by the OpenFlow Capable Switch. For switches
        implementing this version of the OF-CONFIG protocol this
        MUST always be 1.1.1.

        This object can be used to identify the OF-CONFIG version
        a capable switch supports beginning with version 1.1.1 of
        OF-CONFIG. In addtion the supported version can be
        determined by the namespace the OpenFlow Capable Switch
        returns to configuration request of an element (like
        capable-switch) that is present in all OF-CONFIG versions
        specified so far. This is the only possiblity to identify
        OF-CONFIG versions prior to OF-CONFIG 1.1.1.";
    }
    container configuration-points {
      list configuration-point {
        key "id";
        description "The list of all Configuration Points known to
          the OpenFlow Capable Switch that may manage it using
          OF-CONFIG.

          The element 'id' of OFConfigurationType MUST be unique
          within this list.";
        uses OFConfigurationPointType;
      }
    }
    container resources {
      description "A lists containing all resources of the OpenFlow
        Capable Switch that can be used by OpenFlow Logical
        Switches.  Resources are listed here independent of their
        actual assignment to OpenFlow Logical Switches.  They may
        be available to be assigned to an OpenFlow Logical Switch
        or already in use by an OpenFlow Logical Switch.";
      list port {
        must "features/current/rate != 'other' or " +
          "(count(current-rate) = 1 and count(max-rate) = 1 and " +
          " current-rate > 0 and max-rate > 0)" {
```

```
          error-message "current-rate and max-rate must be
            specified and greater than 0 if rate equals 'other'";
          description "current-rate and max-rate can only be
            present if rate = 'other', see corresponding leaf
            descriptions. If rate = 'other', then both leafs must
            be set to values greater than zero.";
        }
        key "resource-id";
        description "The list contains all port resources of the
          OpenFlow Capable Switch.

          The element 'resource-id' of OFPortType MUST be unique
          within this list.";
        uses OFPortType;
      }
      list queue {
        key "resource-id";
        description "The list contains all queue resources of the
          OpenFlow Capable Switch.

          The element 'resource-id' of OFQueueType MUST be unique
          within this list.";
        uses OFQueueType;
      }
      list owned-certificate {
        key "resource-id";
        description "The list contains all owned certificate
          resources of the OpenFlow Capable Switch.

          The element 'resource-id' of OFOwnedCertificateType MUST
          be unique within this list.";
        uses OFOwnedCertificateType;
      }
      list external-certificate {
        key "resource-id";
        description "The list contains all external certificate
          resources of the OpenFlow Capable Switch.

          The element 'resource-id' of OFExternalCertificateType
          MUST be unique within this list.";
        uses OFExternalCertificateType;
      }
      list flow-table {
        key "resource-id";
        description "The list contains all flow table resources of
          the OpenFlow Capable Switch.

          The element 'resource-id' of OFFlowTableType MUST be
          unique within this list.";
        uses OFFlowTableType;
      }
    }
    container logical-switches {
      description "This element contains a list of all OpenFlow
        Logical Switches available at the OpenFlow Capable
        Switch.";
      list switch {
        key "id";
        description "The list of all OpenFlow Logical Switches on
          the OpenFlow Capable Switch.

          The element 'resource-id' of OFLogicalSwitchType MUST be
          unique within this list.";
        uses OFLogicalSwitchType;
      }
    }
  }

}
```

Appendix C Bibliography

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.