OPEN NETWORKING
FOUNDATION

Conformance Test Specification for OpenFlow Switch Specification 1.0.1

June 13, 2013

Disclaimer

Contents

Figures

Tables

Test Cases

1 Introduction

This document defines the requirements and corresponding test procedures that determine the conformance of an OpenFlow 1.0.1 enabled switch. Requirements are derived from the OpenFlow Switch Specification 1.0.0 and the subsequent Errata v1.0.1 available on the ONF website at www.opennetworking.org.

Vendors may refer to these requirements and test procedures during development of their product. Consumers may use these requirements and test results to determine the viability of products for inclusion within their network infrastructure. Test tool manufacturers may use these requirements and test procedures in development of their testing products.

Requirements and test procedures to determine conformance for any changes, clarifications or additions to the main 1.0.0 specification beyond Errata 1.0.1 will be covered in addendums to this document.

This document does not cover requirement and test procedures for extensions outside of the main specification.

Requirements and test procedures to determine conformance for any major specification release beyond 1.0 (1.1, 1.2, etc.…) will be covered in a separate document.

This document does not include requirements or test procedures to validate security, interoperability or performance.

2 Glossary

- Action: An operation that forwards the packet to a port or modifies the packet.
- Byte: An 8-bit octet.
- Controller: Test Framework Controller interacting with DUT using the OpenFlow protocol.
- Control Plane: Software responsible for controlling the Data Plane.
- Control Plane Connection: The TCP connection between the DUT and the Controller Software.
- Controller Software: Implementation of the Control Plane.
- Data Plane: Functionality within a Network Device responsible for packet switching, filtering, and related management.
- Data Plane Port: A physical port where packets enter and exit the Data Plane of the DUT.
- DUT: Device Under Test.
- Egress Port: Data Plane port on which the data packets exit the DUT.
- Flow: A communications interaction between a pair or more endpoints identified by a n-tuple consisting of Layer 2-4 header information.
- Flow Action: An Action associated with a Flow Rule.
- Flow Entry/Flow Rule: an element in a flow table used to match and process packets. It contains a set of match fields for matching packets, a priority for matching precedence, a set of counters to track packets, and a set of instructions to apply.
- Flow Statistics: Performance indicators for a flow.
- Flow Table: The Forwarding Table in a Networking Device that defines how the device should process the flow.
- Hybrid: Control Plane that simultaneously supports OpenFlow and Non-OpenFlow control
- Ingress Port: Data Plane port on which the data packets enters the DUT.
- Layer 2: Functionality and protocols associated with network switching.
- Layer 3: Functionality and protocols associated with network routing.
- Local: Indicates a non-OpenFlow function native to the DUT.
- Match: Outcome when an inbound packet conforms to a Flow Entry in the Flow Table.
- Match Field: A field against which a packet is matched, including packet headers, the ingress port and the metadata value. A match field may be wildcarded (match any value) and in some cases bitmasked.
- OpenFlow: ONF standard protocol that enable OpenFlow Controllers to control Networking Devices in an SDN architecture.
- OpenFlow Controller: An SDN Controller that uses OpenFlow as the interface between the Control and Data Planes.
- OpenFlow Pipeline: A chain of OpenFlow processing elements in a DUT . Often used to distinguish from the Local processing elements.
- OpenFlow Switch: Networking Device that supports OpenFlow protocol.
- Packet: An Ethernet frame, including header and payload.

- Port: Where packets enter and exit the OpenFlow pipeline. May be a physical port, a logical port defined by the switch, or a reserved port defined by the OpenFlow protocol.
- TCP Port: A number assigned to user sessions and server applications in an IP network. Port numbers, which are standardized by the Internet Assigned Numbers Authority (IANA), reside in the header area of the TCP packet.

3 Conformance Requirements and Definitions

Official conformance testing will be performed as outlined by the ONF Conformance Testing Program <Insert Link>.

Usage of the OpenFlow Trademark is outlined in the ONF Trademark Policies located at https://www.opennetworking.org/membership/onf-documents.

In some cases, test cases described in this document are mutually exclusive, OPTIONAL or only relevant for some implementations. This section outlines the valid combinations of test cases required to achieve conformance.

In some cases, the methods of validation are not fully described and may be left up to the tester or test tool developer.

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.

3.1 Conformance Profiles

Many hardware implementations cannot support all features within the standard.

Support of some applications does not require the use of all 12 match fields described in the OpenFlow Switch Specification 1.0.0.

While we believe the intention of the specification was to require all match fields, there was sufficient ambiguity to allow other interpretations.

Due to these issues, several profiles were defined to specify required match fields to support the most common applications.

3.1.1 Full Profile

To be considered fully conformant with the OpenFlow Switch Specification 1.0.0 and the subsequent OpenFlow Switch Errata 1.0.1 the implementation MUST satisfy the requirements of all test cases that indicate “MANDATORY” for “All” or “Full” profiles.

To satisfy the Full profile requirements, the device MUST be able to match all 12 fields individually and simultaneously under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.

3.1.2 Layer2 Profile

To be considered conformant with the Layer 2 Profile, the implementation MUST satisfy the requirements of all test cases that indicate "MANDATORY" for "All" or "L2" profiles. Refer to Table 1 in Test Suite 50 for specific match field requirements.

3.1.3 Layer3 Profile

To be considered conformant with the Layer 3 Profile, the implementation MUST satisfy the requirements of all test cases that indicate "MANDATORY" for "All" or "L3" profiles. Refer to Table 2 in Test Suite 50 for specific match field requirements.

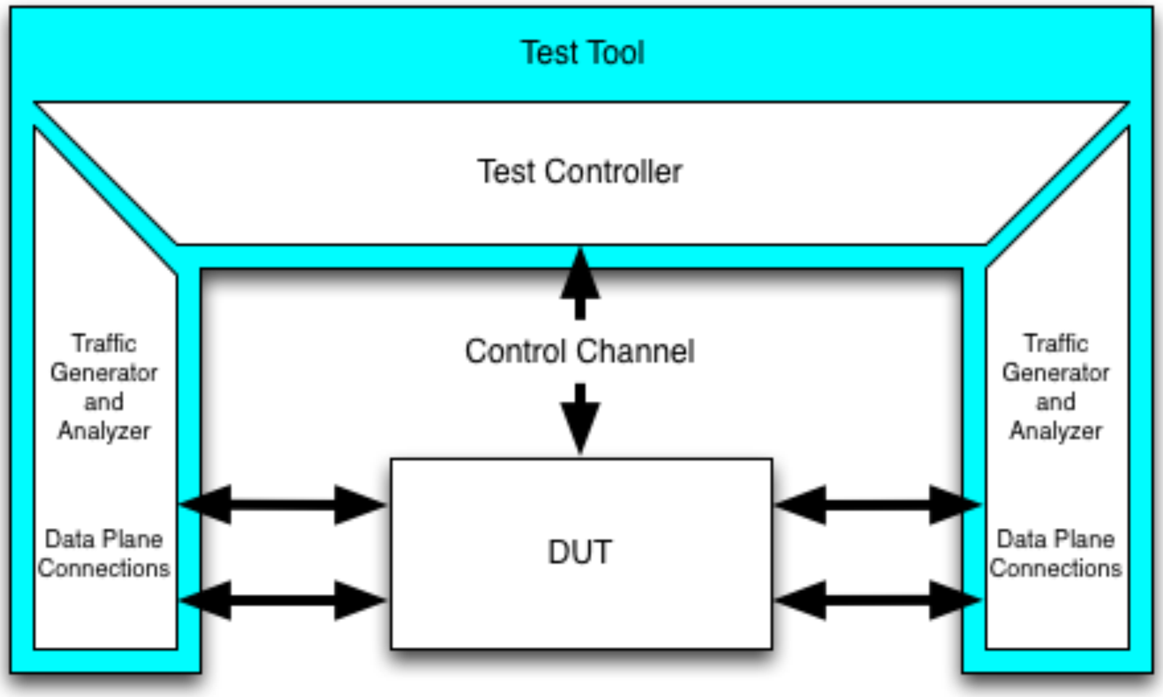
4 Test Bed Configuration

The primary testbed will consist of

1. A test controller with a single control channel connection to the DUT.
2. The test controller should have the ability to perform a packet trace and decode OpenFlow 1.0 packets.
3. A traffic generator/analyzer with a minimum of 4 ports compatible with the DUT for data plane connections.

A backup test controller MAY be used for some tests, but is not required.
Each test case will describe the number and type of connections required.

Figure 1: Test Bed Diagram



5 Test Case Template

Test Suite X: <Suite Title>

<Brief description of test suite.
<Describe any special cases or dependencies.>
<Describe relevant profile dependencies>

Test case X.Y: <Test Case Title>

Test Number	X.Y
Test Title	Group/Subgroup/Title
Test Purpose	Brief description of test purpose
Specification Reference	Reference document, section and p.. (Include Specification Wording when useful for clarity)
Profile Status	List all relevant profiles and the OPTIONAL or MANDATORY status for each. Ex. [OPTIONAL | MANDATORY] for [All | [Full | L2 | L3]] Profiles
Requirements	Brief description of requirements that DUT MUST satisfy
Topology	Describe topology or reference diagram
Methodology	Describe test procedure and methodology
Results	Description of the format in which to display results. Ex. (Pass or Fail)
Remarks	Description of any particular observations that might affect results

6 Test Cases

Test Suite 10: Basic Sanity Checks

Basic Sanity Checks verifies establishment of a control channel, verifies behavior when the control channel is lost.

Special cases:

1. Control channel port and encryption:
4 methods of control channel establishment are tested (Tests 10.20, 10.30, 10.40, 10.50). At least one of the four MUST be supported for the device to be considered conformant.

2. Control channel failure:
Errata 1.0.1 adds support for Fail-Secure Mode in the case of a control channel failure. After loss of the control channel, the switch MUST enter either Emergency Mode (Tests 10.90, 10.100, 10.110) or Fail-Secure Mode (Test 10.120) to be considered conformant. The test cases are labeled as either MANDATORY for Emergency Mode or MANDATORY for Fail-Secure Mode.

3. Backup Controllers:

As backup controllers are optional, so are test cases 10.60 and 10.70.

Profiles:
All profiles MUST pass all test cases from 10.10 to 10.150, except the 3 named exceptions / under the previous named constraints.

Test case 10.10: Startup behavior with established control channel

Test Number	10.10
Test Title	Connection Setup / Establish Control Channel / Switch Startup
Test Purpose	Verify the startup mode, verify no packets are forwarded
Specification Reference	OpenFlow Switch Errata v1.0.1 4.3 Controller Connection Failure Behavior, p. 8
Profile Status	MANDATORY for All Profiles
Requirements	At first startup, the DUT MUST not forward any data plane packets
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections on the DUT.
Methodology	Startup switch, Configure and connect the Primary-controller on the DUT. Send packets to data plane, verify packets are not forwarded on the data plane.
Results	Pass or Fail
Remarks	On initial startup, the switch should not have any emergency rules or flows installed. It should not default to layer 2 forwarding.

Test case 10.20: Configure and establish control channel

Test Number	10.20
Test Title	Connection Setup/ Establish Control Channel / Encrypted, Unencrypted, Default & non-Default TCP port Combinations
Test Purpose	Test all methods of control channel establishment
Specification Reference	OpenFlow Switch Specification 1.0.0 4.2 Connection setup / p. 12 4.4 Encryption / p. 13
Profile Status	1 of 4 [10.20a | 10.20b | 10.20c | 10.20d] is MANDATORY for All Profiles
Requirements	A control channel MUST be established between the DUT and reference controller using at least 1 of 4 methods.
Topology	Control-plane connection between DUT and reference controller
Methodology	Follow methodology for each sub-test (10.20a-d).
Results	Pass or Fail
Remarks	

Test case 10.20a: Use default tcp port

Test Number	10.20a
Test Title	Connection Setup/ Establish Control Channel/ TCP using default port 6633
Test Purpose	Test unencrypted control channel establishment on default TCP port

Specification Reference	OpenFlow Switch Specification 1.0.0 / 4.2 Connection setup / p. 12 *The switch must be able to establish the communication at a user-configurable (but otherwise fixed) IP address, using a user-specified port.*
Profile Status	
Requirements	A control channel MUST be established between the DUT and reference controller without encryption on the default TCP port 6633.
Topology	Control-plane connection between DUT and reference controller
Methodology	Reference controller must be running and reachable at configured IP and Port 6633. Configure DUT to connect with reference controller using unencrypted TCP. If required, manually configure switch to connect to controller using TCP port 6633.
Results	Pass or Fail or Not Tested
Remarks	

Test case 10.20b: Use non-default tcp port

Test Number	10.20b
Test Title	Connection Setup/ Establish Control Channel/ TCP using non-default port
Test Purpose	Test unencrypted control channel establishment on non-default TCP port
Specification Reference	OpenFlow Switch Specification 1.0.0 / 4.2 Connection setup / p. 12 *The switch must be able to establish the communication at a user-configurable (but otherwise fixed) IP address, using a user-specified port.*
Profile Status	OPTIONAL
Requirements	A control channel MUST be established between the DUT and reference controller without encryption on a non-default TCP port.
Topology	Control-plane connection between DUT and reference controller
Methodology	Reference controller must be running and reachable at configured IP and Port. Configure DUT to connect with reference controller using unencrypted TCP. Manually configure switch to connect to controller using previously configured TCP port.
Results	Pass or Fail or Not Tested
Remarks	

Test case 10.20c: Use TLS with default tcp port

Test Number	10.20c
Test Title	Connection Setup/ Establish Control Channel/ TLS using default port 6633
Test Purpose	Test encrypted control channel establishment on default TCP port
Specification Reference	OpenFlow Switch Specification 1.0.0 / 4.4 Encryption/ p. 13 *The TLS connection is initiated by the switch on startup to the controller's server, which is located by default on TCP port 6633*

Profile Status	OPTIONAL
Requirements	A control channel MUST be established between the DUT and reference controller with encryption on the default TCP port 6633.
Topology	Control-plane connection between DUT and reference controller
Methodology	Reference controller must be running and reachable at configured IP and Port 6633. Configure DUT to connect with reference controller using encrypted TLS. If required, manually configure switch to connect to controller using TCP port 6633.
Results	Pass or Fail or Not Tested
Remarks	

Test case 10.20d: Use TLS with non-default tcp port

Test Number	10.20c
Test Title	Connection Setup/ Establish Control Channel/ TLS using non-default port.
Test Purpose	Test encrypted control channel establishment on non-default TCP port
Specification Reference	OpenFlow Switch Specification 1.0.0 / 4.4 Encryption/ p. 13 *The TLS connection is initiated by the switch on startup to the controller's server, which is located by default on TCP port 6633*
Profile Status	OPTIONAL
Requirements	A control channel MUST be established between the DUT and reference controller with encryption on a non-default TCP port.
Topology	Control-plane connection between DUT and reference controller
Methodology	Reference controller must be running and reachable at configured IP and non-default Port. Configure DUT to connect with reference controller using encrypted TLS.
Results	Pass or Fail or Not Tested
Remarks	

Test case 10.30 Supported version announcement

Test Number	10.30
Test Title	Connection Setup / Establish control channel / Supported version announcement
Test Purpose	Check the Switch reports the correct version to the controller
Specification Reference	OpenFlow Switch Specification 1.0.0 / 4.2 Connection Setup / p. 12 *When an OpenFlow connection is first established, each side of the connection must immediately send an OFPT_HELLO message with the version field set to the highest OpenFlow protocol version supported by the sender.*
Profile Status	MANDATORY for All Profiles
Requirements	The DUT MUST announce the correct protocol version 1.0
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Verify version field in Hello message.
Results	Pass or Fail

Remarks	

Test case 10.40: Supported version negotiation

Test Number	10.40
Test Title	Connection Setup / Establish control channel / Supported version negotiation
Test Purpose	Check the Switch negotiates the correct version with the controller
Specification Reference	OpenFlow Switch Specification 1.0.0 / 4.2 Connection Setup / p. 12 *Upon receipt of this message, the recipient may calculate the OpenFlow protocol version to be used as the smaller of the version number that it sent and the one that it received.*
Profile Status	MANDATORY for ALL Profiles
Requirements	The DUT MUST negotiate the correct version to use on the control channel
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Verify switch negotiates the correct version with the controller.
Results	Pass or Fail
Remarks	In this case, the controller announces the correct version, so negotiation to version 1.0 should succeed.

Test case 10.50: No common version negotiated

Test Number	10.50
Test Title	Error messages / Connection Setup / Establish control channel / No common version negotiated
Test Purpose	Verify the switch reports the correct error message and terminates the connection when no common version can be negotiated with the controller.
Specification Reference	OpenFlow Switch Specification 1.0.0/4.2 Connection Setup/p. 12 *if the negotiated version is supported by the recipient, then the connection proceeds. Otherwise, the recipient must reply with an OFPT_ERROR message with a type field of OFPET_HELLO_FAILED, a code field of OFPHFC_COMPATIBLE, and optionally an ASCII string explaining the situation in data, and then terminate the connection.*
Profile Status	MANDATORY for All Profiles
Requirements	The DUT MUST handle version negotiation as described in the specification.
Topology	Control-plane connection between DUT and reference controller
Methodology	Configure and connect the Primary-controller on the DUT. The controller sends an unsupported version, which prevents version negotiation from succeeding. The Error message is verified in packet traces or controller logs.
Results	Pass or Fail
Remarks	

Test case 10.60: Echo timeout triggering connection attempt to Backup-controller

Test Number	10.60

Test Title	Connection interruption / Primary control channel lost / Echo timeout triggering connection attempt to Backup-controller
Test Purpose	Verify switch tries to contact Backup-controller after losing connection to Primary-controller.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.3 Connection interruption, p. 12 *in the case that a switch loses contact with the controller, as a result of an echo request timeout, TLS session timeout, or other disconnection, it should attempt to contact one or more backup controllers. The ordering of the controller IP addresses is not specified by the protocol.*
Profile Status	OPTIONAL
Requirements	If supported and configured, the DUT MUST contact a Backup-controller after losing connection with the Primary-controller
Topology	Control-plane connection between DUT and reference controller
Methodology	Configure Primary-controller and Backup-controller on the DUT. Fail the Primary-controller connection by echo request timeout. Verify the device tries to connect to the Backup-controller. This can be done by packet trace or established connection to Backup-controller.
Results	Pass or Fail or Not Tested
Remarks	

Test case 10.70: TLS Session timeout triggering connection attempt to Backup-controller

Test Number	10.70
Test Title	Connection interruption / Primary control channel lost / TLS Session timeout triggering connection attempt to Backup-controller
Test Purpose	Verify switch tries to contact Backup-controller after losing connection to Primary-controller.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.3 Connection interruption, p. 12 *in the case that a switch loses contact with the controller, as a result of an echo request timeout, TLS session timeout, or other disconnection, it should attempt to contact one or more backup controllers. The ordering of the controller IP addresses is not specified by the protocol.*
Profile Status	OPTIONAL
Requirements	If supported and configured, the DUT MUST contact a Backup-controller after losing connection with the Primary-controller
Topology	Control-plane connection between DUT and reference controller
Methodology	Configure Primary-controller and Backup-controller on the DUT. Fail the Primary-controller connection by TLS Session timeout. Verify the device tries to connect to the Backup-controller. This can be done by packet trace or established connection to Backup-controller.
Results	Pass or Fail or Not Tested
Remarks	

Test case 10.80: Losing the control channel triggers connection attempts

Test Number	10.80

Test Title	Connection interruption / Primary control channel lost / Losing the control channel triggers connection attempts
Test Purpose	Verify switch tries to reconnect after losing control channel, check whether retries and backoff are applied as configured on the DUT.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.3 Connection interruption, p. 12 *if some number of attempts to contact a controller (zero or more) fail, the switch must enter "emergency mode" and immediately reset the current TCP connection.*
Profile Status	MANDATORY for ALL Profiles
Requirements	After losing the control channel, the DUT MUST try to re-establish a connection with the controller
Topology	Control-plane connection between DUT and reference controller
Methodology	Configure and connect the Primary-controller on the DUT. Fail the Primary-controller connection. Verify the device attempts to re-connect to the controller. Verify the frequency of reconnection attempts. Verify with packet trace.
Results	Pass or Fail
Remarks	Method for generating control channel failure is unspecified

Test case 10.90: Losing the control channel triggers emergency mode

Test Number	10.90
Test Title	Connection interruption / Primary control channel lost / Losing the control channel triggers emergency mode
Test Purpose	Verify switch activates emergency rules after losing control channel connections.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.3 Connection interruption, p. 12 *In emergency mode, the matching process is dictated by the emergency flow table entries (those marked with the emergency bit when added to the switch*
Profile Status	MANDATORY for Emergency Mode.
Requirements	After losing the control channel, the DUT MUST activate the emergency rule set.
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections on the DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create emergency flow table entries. Verify with data plane traffic that emergency flow table entries are not active. Fail control channel. Verify with data plane traffic that emergency flow entries are activated.
Results	Pass or Fail
Remarks	

Test case 10.100: Emergency mode removes standard flow entries

Test Number	10.100
Test Title	Connection interruption / Primary control channel lost / Emergency mode removes standard flow entries
Test Purpose	Verify switch deletes all normal flow entries when emergency mode is activated

Specification Reference	OpenFlow Switch Specification 1.0.0, 4.3 Connection interruption, p. 13 *All normal entries are deleted when entering emergency mode*
Profile Status	MANDATORY for Emergency Mode.
Requirements	After activating the emergency rule set all normal entries in the flow table MUST be deleted.
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections on the DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create normal and emergency flow table entries. Verify with data plane traffic that normal entries are active and emergency flow table entries are not active. Fail control channel. Verify with data plane traffic that emergency flow entries are now active, and normal entries are inactive. Check the flow table to verify normal entries are deleted.
Results	Pass or Fail
Remarks	

Test case 10.110: Emergency rules after control channel reconnection

Test Number	10.110
Test Title	Connection interruption / Primary control channel lost / Emergency rules after control channel reconnection
Test Purpose	Verify switch keeps the emergency rules active after reconnection to a controller.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.3 Connection interruption, p. 13 *Upon connecting to a controller again, the emergency flow entries remain. The controller then has the option of deleting all flow entries, if desired.*
Profile Status	MANDATORY for Emergency Mode.
Requirements	After reconnection to the controller, the emergency rule set MUST stay active.
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections on the DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create emergency flow table entries. Fail control channel. After reconnection of the control channel, verify with data plane traffic that the emergency flow table entries stay active.
Results	Pass or Fail
Remarks	

Test case 10.120: Fail secure mode

Test Number	10.120
Test Title	Connection interruption / Primary control channel lost / Fail secure mode
Test Purpose	Verify switch keeps the normal flow table active after losing the control channel. Verify the entries time out as expected. Verify flow table entries stay active after reconnection.
Specification	OpenFlow Switch Errata 1.0.1, 4.3 Controller Connection Failure

Reference	Behavior, p. 8 *In "fail secure mode", the only change to switch behavior is that packets and messages destined to the controllers are dropped. Flow entries should continue to expire according to their timeouts in "fail secure mode*
Profile Status	MANDATORY for Fail-Secure Mode
Requirements	After losing the control channel, normal flow entries MUST stay active and time out as expected. After reconnection, flow entries MUST stay active and time out as expected.
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections on the DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create normal flow table entries with different timeouts. Fail control channel. Verify with data plane traffic that flow table entries stay active and time out as expected. After reconnection of the control channel, verify with data plane traffic that the flow table entries left in the flow table stay active and time out as expected.
Results	Pass or Fail
Remarks	

Test Suite 20: Basic OpenFlow protocol messages

Test suite 20 checks for implementation of the basic protocol messages. We only verify that the messages do not generate error messages on the DUT, we do not check for correct responses or implementation. Detailed checks follow later in the respective test groups.

Special cases:
None

Profiles:
All profiles MUST pass all test cases from 20.10 to 20.110.

Test case 20.10: Verify Features Request / Reply is implemented

Test Number	20.10
Test Title	Controller to Switch messages / Features / Verify Features Request/Reply is implemented.
Test Purpose	Verify that a basic Features Request generates a Features Reply.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.1.1 Controller to Switch, p. 10. *Features: Upon Transport Layer Security (TLS) session establishment, the controller sends a features request message to the switch. The switch must reply with a features reply that specifies the capabilities supported by the switch*
Profile Status	MANDATORY for ALL Profiles
Requirements	Generate a Features Reply in response to a Features Request
Topology	Control-plane connection between DUT and reference controller
Methodology	Configure and connect the Primary-controller on the DUT. Send ofpt_features_request to the DUT; verify ofpt_features_reply is returned.

Results	Pass or Fail
Remarks	

Test case 20.20: Verify basic Config Request is implemented

Test Number	20.20
Test Title	Controller to Switch messages / Configuration / Verify basic Config Request is implemented
Test Purpose	Verify that a basic Get Config Request does not generate an error.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.1.1 Controller to Switch, p. 10. *Configuration: The controller is able to set and query configuration parameters in the switch. The switch only responds to a query from the controller.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Generate a Config Reply in response to a Config Request
Topology	Control-plane connection between DUT and reference controller
Methodology	Configure and connect the Primary-controller on the DUT. Send ofpt_get_config_request to the DUT, verify ofpt_get_config_reply is returned.
Results	Pass or Fail
Remarks	

Test case 20.30: Verify basic Modify state Add message is implemented

Test Number	20.30
Test Title	Controller to Switch messages / Modify state / Verify basic Modify state Add message is implemented
Test Purpose	Verify that a basic Flow ADD request does not generate an error.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.1.1 Controller to Switch, p. 10. *Modify-State: Modify-State messages are sent by the controller to manage state on the switches. Their primary purpose is to add/delete and modify Flows in the Flow tables and to set switch port properties*
Profile Status	MANDATORY for ALL Profiles
Requirements	Modify Flow Add implemented
Topology	Control-plane connection between DUT and reference controller
Methodology	Configure and connect the Primary-controller on the DUT. Send ofpt_flow_mod command ofpfc_add to the DUT, verify no Error is returned
Results	Pass or Fail
Remarks	

Test case 20.40: Verify basic Modify state Delete message is implemented

Test Number	20.40
Test Title	Controller to Switch messages / Modify state / Verify basic Modify state Delete message is implemented
Test Purpose	Verify that a basic Flow Delete request does not generate an error.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.1.1 Controller to Switch, p. 10. *Modify-State: Modify-State messages are sent by the controller to manage state on the switches. Their primary purpose is to add/delete and modify Flows in the Flow tables and to set switch port properties*

Profile Status	MANDATORY for ALL Profiles
Requirements	Modify Flow Delete implemented
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Send ofpt_flow_mod command ofpfc_delete to the DUT, verify no Error is returned
Results	Pass or Fail
Remarks	

Test case 20.50: Verify basic Modify Flow Modify message is implemented

Test Number	20.50
Test Title	Controller to Switch messages / Modify state / Verify basic Modify State Modify message is implemented
Test Purpose	Verify that a basic Modify State Modify request does not generate an error.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.1.1 Controller to Switch, p. 10. *Modify-State: Modify-State messages are sent by the controller to manage state on the switches. Their primary purpose is to add/delete and modify Flows in the Flow tables and to set switch port properties*
Profile Status	MANDATORY for ALL Profiles
Requirements	Modify Flow Delete implemented
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Send ofpt_flow_mod command ofpfc_modify to the DUT, verify no Error is returned
Results	Pass or Fail
Remarks	

Test case 20.60: Verify basic Read state is implemented

Test Number	20.60
Test Title	Controller to Switch messages / Read state / Verify basic Read state is implemented
Test Purpose	Verify that a basic Read state request does not generate an error.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.1.1 Controller to Switch, p. 10. *Read-State: Read-State messages are used by the controller to collect statistics from the switch's flow-tables, ports and the individual flow entries*
Profile Status	MANDATORY for ALL Profiles
Requirements	Read state is implemented
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Send ofpt_stats_request to the DUT. Verify ofpt_stats_reply is received
Results	Pass or Fail
Remarks	

Test case 20.70: Verify basic send packet is implemented

Test Number	20.70
Test Title	Controller to Switch messages / Send packet / Verify basic send packet

	is implemented
Test Purpose	Verify that a basic send packet request does not generate an error.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.1.1 Controller to Switch, p. 10. *Send-Packet: These are used by the controller to send packets out of a specified port on the switch*
Profile Status	MANDATORY for ALL Profiles
Requirements	Send packet implemented
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection on the DUT.
Methodology	Configure and connect the Primary-controller on DUT. Send ofpt_packet message to DUT. Verify no error is returned.
Results	Pass or Fail
Remarks	

Test case 20.80: Verify basic barrier request-reply is implemented

Test Number	20.80
Test Title	Controller to Switch messages / Barrier / Verify basic barrier request-reply is implemented
Test Purpose	Verify that a basic barrier request does not generate an error.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.1.1 Controller to Switch, p. 10. *Barrier: Barrier request/reply messages are used by the controller to ensure message dependencies have been met or to receive notifications for completed operations.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Basic barrier request-reply implemented
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Send basic barrier request to the DUT; verify no Error is returned.
Results	Pass or Fail
Remarks	

Test case 20.90: Packet_in generation

Test Number	20.90
Test Title	Asynchronous Messages / Packet_in generation
Test Purpose	Verify that non matched data plane packets generate a packet_in to the controller
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.1.2 Asynchronous, p. 10. *Packet-in: For all packets that do not have a matching Flow entry, a packet-in event is sent to the controller.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Packet_in is implemented
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection on the DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Send packet to the DUT data plane port; verify the controller receives a packet_in.
Results	Pass or Fail
Remarks	

Test case 20.100: Verify basic Hello messages are implemented

Test Number	20.100
Test Title	Symmetric Messages / Hello / Verify basic Hello messages are implemented
Test Purpose	Verify basic Hello message generation with correct version field
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.1.3 Symmetric, p. 11. *Hello: Hello messages are exchanged between the switch and controller upon connection startup.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Hello message is implemented
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Verify from controller log or packet trace that a ofpt_hello message is generated, and the version field correctly populated
Results	Pass or Fail
Remarks	

Test case 20.110: Verify Echo Reply messages are implemented

Test Number	20.110
Test Title	Symmetric Messages / Echo / Verify Echo Reply messages are implemented
Test Purpose	Verify basic Echo Reply generation
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.1.3 Symmetric, p. 11. *Hello: Echo: Echo request/reply messages can be sent from either the switch or the controller, and must return an echo reply. They can be used to indicate the latency, bandwidth, and/or liveness of a controller-switch connection*
Profile Status	MANDATORY for ALL Profiles
Requirements	ofpt_echo_request/reply is implemented
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Send ofpt_echo_request. Verify from controller log or packet trace that ofpt_echo_reply message is generated.
Results	Pass or Fail
Remarks	

Test Suite 30: Spanning Tree

Test suite 30 checks for implementation of Spanning Tree Protocol related functions. Most of these tests are not required for conformance. Only port state and config messages that have a possible application outside of STP are required for conformance.

Special cases:
Only the following test cases are MANDATORY and tested: 30.40 Port Down.

Profiles:
All profiles MUST pass test case 30.40.

Test case 30.10: Flood control port mod message

Test Number	30.10
Test Title	Spanning Tree / No Spanning Tree / Flood control port mod message
Test Purpose	Verify Controller is able to control flooding with port mod messages
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.5 Spanning Tree, p. 13. *Switches that do not support 802.1D spanning tree must allow the controller to specify the port status for packet flooding through the port-mod messages*
Profile Status	OPTIONAL
Requirements	Port mod ofppc_no_flood flag is implemented
Topology	Control-plane connection between DUT and reference controller. At least 4 data plane connections to the DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Move a subset of ports into the flood group, create flow entry with flood action, generate matching data plane packet. Verify only ports in the flood group output packet. Verify all ports in the flood group output the packet.
Results	Pass or Fail or Not Tested
Remarks	For Example: 4 data plane ports - 1 input port, 2 output ports in the flood group, 1 output port not in flood group.

Test case 30.20: Port config bits

Test Number	30.20
Test Title	Spanning Tree / Config / Port config bits
Test Purpose	Verify Controller is able to read the current Spanning Tree state
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.5 Spanning Tree, p. 13. *The port config bits indicate whether a port has been administratively brought down, options for handling 802.1D spanning tree packets, and how to handle incoming and outgoing packets. These bits, configured over multiple switches, enable an OpenFlow network to safely flood packets along either a custom or 802.1D spanning tree; When OFPPFL_NO_STP is 0, STP controls the OFPPFL_NO_FLOOD and OFPPFL_STP_* bits directly. OFPPFL_NO_FLOOD is set to 0 when the STP port state is Forwarding, otherwise to 1. The bits in OFPPFL_STP_MASK are set to one of the other OFPPFL_STP_* values according to the current STP port state.*
Profile Status	OPTIONAL
Requirements	Port mod OFPPFL_STP_* and OFPPFL_NO_* config bits are implemented
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Send ofpt_features_request and verify port config bits are set according to the DUT config
Results	Pass or Fail or Not Tested
Remarks	

Test case 30.40: Port administratively down

Test Number	30.40
Test Title	Spanning Tree / Config / Port administratively down
Test Purpose	Verify Controller is able to bring port up and down
Specification	OpenFlow Switch Specification 1.0.0, 5.2.1 Port Structures, p. 17.

Reference	*OFPPC_PORT_DOWN =1<<0, /* Port is administratively down.*
Profile Status	MANDATORY for ALL Profiles
Requirements	DUT is able to get and set port state "administratively down"
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to the DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Verify current port state; if port is down bring port up. Send port down message; verify port is down. Send port up message; verify port is up again.
Results	Pass or Fail
Remarks	Port must end in a port_up state for subsequent tests.

Test case 30.50: Disable 802.1D Spanning Tree

Test Number	30.50
Test Title	Spanning Tree / Config / Disable 802.1D Spanning Tree
Test Purpose	Verify Controller is able to enable and disable Spanning Tree on port
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.2.1 Port Structures, p. 17. *OFPPC_NO_STP=1<<1, /* Disable 802.1D spanning tree on port*
Profile Status	OPTIONAL
Requirements	DUT is able to get and set port state "Disable 802.1D "
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Verify current port state; if 802.1D is enabled, disable it. Using features_request, verify that ofppc_stp bit=0. Enable Spanning Tree. Using features_request, verify that ofppc_stp bit=1. Disable Spanning Tree. Using features_request, verify that ofppc_stp bit=0.
Results	Pass or Fail or Not Tested
Remarks	Testing of non-OpenFlow Spanning Tree implementation is out of scope.

Test case 30.60: Drop all except 802.1D

Test Number	30.60
Test Title	Spanning Tree / Config / Drop all except 802.1D
Test Purpose	Verify Controller is able to enable and disable OFPPC_NO_RECV
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.2.1 Port Structures, p. 17. *OFPPC_NO_RECV=1<<2, /* Drop all packets except 802.1D spanning tree packets.*
Profile Status	OPTIONAL
Requirements	DUT is able to get and set port state "OFPPC_NO_RECV "
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Verify current port state for OFPPC_NO_RECV; if OFPPC_NO_RECV is enabled, disable it. Verify on data plane port that Spanning Tree packets are received, all other packet types are also received. Enable OFPPC_NO_RECV. Verify on data plane port that Spanning Tree

	packets are still received, all other packet types are now dropped. Disable OFPPC_NO_RECV again. Verify on data plane port that Spanning Tree packets are still received, and all other packet types are received again.
Results	Pass or Fail or Not Tested
Remarks	

Test case 30.70: Forward all except 802.1D

Test Number	30.70
Test Title	Spanning Tree / Config / Forward all except 802.1D
Test Purpose	Verify Controller is able to enable and disable OFPPC_NO_RECV_STP
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.2.1 Port Structures, p. 17. *OFPPC_NO_RECV_STP=1<3, Drop received 802.1D STP packets.*
Profile Status	OPTIONAL
Requirements	DUT is able to get and set port state "OFPPC_NO_RECV_STP", and all 802.1D packets on the port are dropped, all other packets are forwarded
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Verify current port state for OFPPC_NO_RECV _STP; if OFPPC_NO_RECV _STP is enabled, disable it. Verify on data plane port that Spanning Tree packets are received, all other packet types are also received. Enable OFPPC_NO_RECV. Verify on data plane port that Spanning Tree packets are dropped, all other packet types are still received. Disable OFPPC_NO_RECV again. Verify on data plane port that Spanning Tree packets are now received, and all other packet types are still received.
Results	Pass or Fail or Not Tested
Remarks	

Test case 30.80: Flood control port mod message

Test Number	30.80
Test Title	Spanning Tree / No Spanning Tree / Flood control port mod message
Test Purpose	Verify Controller is able to control flooding with port mod messages
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.5 Spanning Tree, p. 13. *Switches that do not support 802.1D spanning tree must allow the controller to specify the port status for packet flooding through the port-mod messages*
Profile Status	OPTIONAL
Requirements	Port mod ofppc_no_flood flag is implemented
Topology	Control-plane connection between DUT and reference controller. At least 4 data plane connections to the DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Move a subset of ports into the flood group, create flow entry with flood action, generate matching data plane packet. Verify only ports in the flood group output packet. Verify all ports in the flood group output the

	packet.
Results	Pass or Fail or Not Tested
Remarks	For Example: 4 data plane ports - 1 input port, 2 output ports in the flood group, 1 output port not in flood group.

Test case 30.90: Drop all egress packets on port

Test Number	30.90
Test Title	Spanning Tree / Config / Drop all egress packets on port.
Test Purpose	Verify Controller is able to enable and disable OFPPC_NO_FWD
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.2.1 Port Structures, p. 17. *OFPPC_NO_FWD=1<<5, Drop packets forwarded to port.*
Profile Status	OPTIONAL
Requirements	DUT is able to get and set port state OFPPC_NO_FWD, and all packets on the port are dropped.
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Verify current port state for OFPPC_NO_FWD; if OFPPC_NO_FWD is enabled, disable it. Create Flow Rule forwarding to one port. Verify on data plane that packets are forwarded to that port. Enable OFPPC_NO_FWD. Verify on data plane that packets are now dropped. Disable OFPPC_NO_FWD again. Verify on data plane that packets are forwarded again.
Results	Pass or Fail
Remarks	The DUT must end in state OFPPC_NO_FWD=0 for subsequent tests.

Test case 30.100: No Packet_in

Test Number	30.100
Test Title	Spanning Tree / Config / No Packet_in
Test Purpose	Verify Controller is able to enable and disable OFPPC_NO_PACKET_IN
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.2.1 Port Structures, p. 17. *OFPPC_NO_FWD=1<<5, Drop packets forwarded to port.*
Profile Status	OPTIONAL
Requirements	DUT is able to get and set port state OFPPC_NO_FWD, and all packets on the port are dropped.
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Verify current port state for OFPPC_NO_PACKET_IN; if OFPPC_NO_PACKET_IN is enabled, disable it. Verify on control-plane connection that packets reaching this port generate packet_in messages. Enable OFPPC_NO_PACKET_IN. Verify on control-plane connection that packets reaching the port do not generate packet_in messages. Disable OFPPC_NO_PACKET_IN again. Verify on control-plane connection that packets are now generating packet_in again.

Results	Pass or Fail
Remarks	The DUT must end in state OFPPC_NO_PACKET_IN=0 for subsequent tests.

Test case 30.110: STP classification

Test Number	30.110
Test Title	Spanning Tree / Hybrid / STP classification
Test Purpose	Verify DUT is able to process STP locally first.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.5 Spanning Tree, p. 13. *If spanning tree is supported, process packets locally first. OpenFlow switches may OPTIONALLY support 802.1D Spanning Tree Protocol. Those switches that do support it are expected to process all 802.1D packets locally before performing flow lookup.*
Profile Status	OPTIONAL
Requirements	DUT is HYBRID and able to run a local Spanning Tree.
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a Flow Rule forwarding STP packets from port 1 to port 2. Verify STP packets do not trigger a flow match as the local STP processes them first.
Results	Pass or Fail or Not Tested
Remarks	

Test case 30.120: STP features reply

Test Number	30.120
Test Title	Spanning Tree / Hybrid / STP features reply
Test Purpose	Verify a DUT that implements STP sets the OFPC_STP bit in the ’capabilities’ field of its OFPT_FEATURES_REPLY message
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.5 Spanning Tree, p. 13. *A switch that implements STP must set the OFPC_STP bit in the ’capabilities’ field of its OFPT_FEATURES_REPLY message*
Profile Status	OPTIONAL
Requirements	The DUT MUST set the OFPC_STP bit if it supports Spanning Tree.
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Verify the OFPC_STP bit in the ’capabilities’ field of the DUT’s OFPT_FEATURES_REPLY message is set by checking the controller log or packet trace.
Results	Pass or Fail or Not Tested
Remarks	

Test case 30.130: STP on all physical ports

Test Number	30.130
Test Title	Spanning Tree / Hybrid / STP on all physical ports
Test Purpose	Verify a DUT that implements Local STP supports STP on all physical ports

Specification Reference	OpenFlow Switch Specification 1.0.0, 4.5 Spanning Tree, p. 13. *A switch that implements STP must make it available on all of its physical ports, but it need not implement it on virtual ports (e.g. OFPP_LOCAL)*
Profile Status	OPTIONAL
Requirements	A switch that implements Local STP MUST make it available on all of its physical ports
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Activate Local STP on all available physical ports.
Results	Pass or Fail or Not Tested
Remarks	

Test case 30.140: Flood along STP topology

Test Number	30.140
Test Title	Spanning Tree / Hybrid / Flood along STP topology
Test Purpose	Verify a DUT that implements Local STP floods packets only along the locally determined STP topology
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.5 Spanning Tree, p. 13. *Port status, as specified by the spanning tree protocol, is then used to limit packets forwarded to the OFP_FLOOD port to only those ports along the spanning tree*
Profile Status	OPTIONAL
Requirements	A switch that implements STP locally MUST adapt the ofppc_flood status of ports to the external STP topology
Topology	Control-plane connection between DUT and reference controller. At least 4 data plane ports connected to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Activate Local STP on all available physical ports. Force Local STP to disable at least one port (one method would be to create a loop between two ports). Use controller to send packets to the OFP_FLOOD port. Verify packets are only forwarded along the Local STP topology.
Results	Pass or Fail or Not Tested
Remarks	

Test case 30.150: STP triggers port_update message

Test Number	30.150
Test Title	Spanning Tree / Hybrid / STP triggers port_update message
Test Purpose	Verify a DUT that implements Local STP reports port state changes caused by Local STP back to the controller
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.5 Spanning Tree, p. 13. *Port changes as a result of the spanning tree are sent to the controller via port-update messages*
Profile Status	OPTIONAL
Requirements	Local STP reports port state changes caused by STP back to the controller
Topology	Control-plane connection between DUT and reference controller. At least 4 data plane ports connected to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Activate

	Local STP on all available physical ports. Force Local STP topology change so STP port state on the DUT changes (one method would be to create a loop between two ports). Verify port_update message is sent to the controller.
Results	Pass or Fail or Not Tested
Remarks	

Test case 30.160: OFP_ALL or explicit out_port override STP

Test Number	30.160
Test Title	Spanning Tree / Hybrid / OFP_ALL or explicit out_port override STP
Test Purpose	Verify OFP_ALL and explicit out_port actions ignore Local STP generated port state when forwarding packets.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.5 Spanning Tree, p. 13. *Note that forward actions that specify the outgoing port or OFP_ALL ignore the port status set by the spanning tree protocol*
Profile Status	OPTIONAL
Requirements	A switch MUST forward packets to OFP_ALL or explicit set port, ignoring the port status set by the spanning tree protocol
Topology	Control-plane connection between DUT and reference controller. At least 4 data plane ports connected to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Activate Local STP on all available physical ports. Verify at least one of the connected ports is blocked by spanning tree (one method would be to create a loop between two ports). Create a flow with target OFP_ALL; send a matching packet, verify it gets output from the blocked port. Create a flow explicitly forwarding a packet to the blocked port, send matching packet, verify packet gets output from the blocked port.
Results	Pass or Fail or Not Tested
Remarks	

Test case 30.170: Enable – Disable STP per port

Test Number	30.170
Test Title	Spanning Tree / Hybrid / Enable – Disable STP per port
Test Purpose	Verify the DUT allows enabling and disabling Local STP per port, and changes forwarding behavior accordingly
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.5 Spanning Tree, p. 13. *The switch must support disabling spanning tree per port. Packets received on ports that are disabled by spanning tree must follow the normal flow table processing path*
Profile Status	OPTIONAL
Requirements	When a Local STP port state changes from STP enabled to ATP disabled, the packets received on that port MUST be processed by the normal flow table processing path.
Topology	Control-plane connection between DUT and reference controller. At least 4 data plane ports connected to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Activate Local STP on all available physical ports. Send STP packets; they should not generate packet_in events as they are processed by the

	external STP. Disable STP on one port; verify STP packets from this port now generate packet_in events.
Results	Pass or Fail or Not Tested
Remarks	

Test Suite 40: Flow modification messages

Test suite 40 deals with all the requirements for adding, editing, deleting and removing a flow.

Special cases:
In some instances 40.40, 40.50 and 40.60 may not be applicable and are OPTIONAL. For example, a software switch might be able to have every possible port number available.

Tests 40.70 and 40.130 deal with Emergency Mode and are OPTIONAL.

Profiles:
All profiles MUST pass all tests from 40.10 to 40.230. With the exceptions (40.40 to 40.70 and 40.130) named above.

Test case 40.10: Overlap checking

Test Number	40.10
Test Title	Flow Table Modification Messages / ADD / Overlap checking
Test Purpose	Verify that overlap checking generates an error when the controller attempts to add an overlapping flow to the flow table.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *For ADD requests with the OFPFF_CHECK_OVERLAP flag set, the switch must first check for any overlapping flow entries. Two flow entries overlap if a single packet may match both, and both entries have the same priority. If an overlap conflict exists between an existing flow entry and the ADD request, the switch must refuse the addition and respond with an ofp_error_msg with OFPET_FLOW_MOD_FAILED type and OFPFMFC_OVERLAP code*
Profile Status	MANDATORY for ALL Profiles
Requirements	DUT implements overlap checking
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Add flow 1 into flow table. Try to add overlapping flow with “check overlap” flag set into flow table. Verify the correct error message is returned. Verify flow is not entered into flow table.
Results	Pass or Fail
Remarks	

Test case 40.20: No overlap checking

Test Number	40.20
Test Title	Flow Table Modification Messages / ADD / No overlap checking
Test Purpose	Verify that no overlap checking does not generate an error when the

	controller attempts to add an overlapping flow to the flow table.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *For valid (non-overlapping) ADD requests, or those with no overlap checking, the switch must insert the flow entry at the lowest numbered table for which the switch supports all wildcards set in the flow_match structure, and for which the priority would be observed during the matching process."*
Profile Status	MANDATORY for ALL Profiles
Requirements	DUT implements adding overlapping flows
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Add flow 1 into flow table. Add overlapping flow with “check overlap” flag not set into flow table. Verify no error message is returned. Verify flow is entered into flow table.
Results	Pass or Fail
Remarks	

Test case 40.30: Identical flows

Test Number	40.30
Test Title	Flow Table Modification Messages / ADD / Identical flows
Test Purpose	Verify that adding an identical flow overwrites the existing flow and clears the counters
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *If a flow entry with identical header fields and priority already resides in any table, then that entry, including its counters, must be removed, and the new flow entry added."*
Profile Status	MANDATORY for ALL Profiles
Requirements	DUT implements adding identical flows while resetting counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add flow 1 into flow table. Send matching packets to data plane to increase counters. Add identical flow into flow table. Verify the new flow replaces the existing flow. Verify the counters are reset.
Results	Pass or Fail
Remarks	

Test case 40.40: No table to add

Test Number	40.40
Test Title	Flow Table Modification Messages / ADD / No table to add
Test Purpose	Verify that flow table full error messages are generated.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *If a switch cannot find any table in which to add the incoming flow entry, the switch should send an ofp_error_msg with OFPET_FLOW_MOD_FAILED type and OFPFMFC_ALL_TABLES_FULL code*
Profile Status	MANDATORY for ALL Profiles

Requirements	DUT returns correct error code when flow table is full.
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Create and add flows until flow table is full, verify OFPFMFC_ALL_TABLES_FULL error message is generated.
Results	Pass or Fail
Remarks	

Test case 40.50: Never valid output port

Test Number	40.50
Test Title	Flow Table Modification Messages / ADD / Never valid output port
Test Purpose	Verify that adding a flow with a never valid output port number triggers correct error
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *If the action list in a flow mod message references a port that will never be valid on a switch, the switch must return an ofp_error_msg with OFPET_BAD_ACTION type and OFPBAC_BAD_OUT_PORT code*
Profile Status	MANDATORY for ALL Profiles
Requirements	DUT returns correct error code when never valid port is referenced as output port
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Create and add flow pointing to a never existing port number. Verify OFPBAC_BAD_OUT_PORT error message is generated
Results	Pass or Fail
Remarks	

Test case 40.60: Currently non-existant output port

Test Number	40.60
Test Title	Flow Table Modification Messages / ADD / Currently non-existant output port Version A
Test Purpose	Verify that adding a flow with action OFPAT_OUTPUT to a currently not available port number (but possibly available later) generates one of the two possible responses from the switch
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *If the referenced port may be valid in the future, e.g. when a line card is added to a chassis switch, or a port is dynamically added to a software switch, the switch may either silently drop packets sent to the referenced port or immediately return an OFPBAC_BAD_OUT_PORT error and refuse the flow mod*
Profile Status	1 of 2 [40.60a | 40.60b] is MANDATORY for All Profiles
Requirements	DUT accepts flow pointing to a port that may be valid in the future.
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Create and add flow pointing to a currently non-existant port number.
Results	Pass or Fail

Remarks	

Test case 40.60a: Currently non-existant output port Version A

Test Number	40.60a
Test Title	Flow Table Modification Messages / ADD / Currently non-existant output port Version A
Test Purpose	Verify that adding a flow with action output to a currently not available port number (but possibly available later) gets added, and silently drops packets
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *If the referenced port may be valid in the future, e.g. when a line card is added to a chassis switch, or a port is dynamically added to a software switch, the switch may either silently drop packets sent to the referenced port*
Profile Status	OPTIONAL
Requirements	DUT accepts flow pointing to a port that may be in the future valid. Packets are dropped until then.
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Create and add flow pointing to a currently non-existant port number. Verify flow is added, but packets are dropped
Results	Pass or Fail or Not Tested
Remarks	Version A option

Test case 40.60b: Currently non-existent port Version B

Test Number	40.60b
Test Title	Flow Table Modification Messages / ADD / Currently non-existant port Version B
Test Purpose	Verify that adding a flow with action output to a currently not available port number triggers correct error message
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *or immediately return an OFPBAC_BAD_OUT_PORT error and refuse the flow mod*
Profile Status	OPTIONAL
Requirements	DUT does not accept flow pointing to a port that may be in the future valid, and generates correct error message
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Create and add flow pointing to a currently non-existant port number. Verify flow is not added, but error message generated
Results	Pass or Fail or Not Tested
Remarks	Version B option

Test case 40.70: No timeout for emergency flows

Test Number	40.70

Test Title	Flow Table Modification Messages / ADD / No timeout for emergency flows
Test Purpose	Verify that adding an emergency flow with a non-zero timeout value triggers correct error message.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *Emergency flow mod messages must have timeout values set to zero. Otherwise, the switch must refuse the addition and respond with an ofp_error_msg with OFPET_FLOW_MOD_FAILED type and OFPFMFC_BAD_EMERG_TIMEOUT code*
Profile Status	MANDATORY for Emergency Mode.
Requirements	DUT only accepts timeout = 0 for emergency flows, all other values MUST trigger correct error message
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Create and add emergency flow with timeout = 0, verify flow gets added. Create and add emergency flow with non-zero timeout, verify flow does not get added. Verify OFPFMFC_BAD_EMERG_TIMEOUT error message is returned.
Results	Pass or Fail or Not Tested
Remarks	

Test case 40.80: Modify non-existent flow

Test Number	40.80
Test Title	Flow Table Modification Messages / Modify / Modify non-existent flow
Test Purpose	Verify that modifying a non-existent flow adds the flow with zeroed counters.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *For MODIFY requests, if a flow entry with identical header fields does not current reside in any table, the MODIFY acts like an ADD, and the new flow entry must be inserted with zeroed counters*
Profile Status	MANDATORY for ALL Profiles
Requirements	DUT allows modifying non-existent flows and adds the respective flow to the flow table
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Send a modify request targeting a non-existent flow. Verify the flow gets added with zeroed counters
Results	Pass or Fail
Remarks	

Test case 40.90: Modify action preserves counters

Test Number	40.90
Test Title	Flow Table Modification Messages / Modify / Modify action preserves counters
Test Purpose	Verify that modifying the action of a flow does not reset counters

Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *Otherwise, the actions field is changed on the existing entry and its counters and idle time fields are left unchanged.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Modifying the action of an existing flow preserves the flow counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Insert a flow. Send data plane packets to increase counters. Modify the Flow Action with OFPFC_MODIFY. Get flow statistics, verify counters were preserved.
Results	Pass or Fail
Remarks	

Test case 40.100: Modify_strict of action preserves counters

Test Number	40.100
Test Title	Flow Table Modification Messages / Modify / Modify_strict action preserves counters
Test Purpose	Verify that modifying the action of a flow does not reset counters for modify_strict
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *Otherwise, the actions field is changed on the existing entry and its counters and idle time fields are left unchanged.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Modifying the action of an existing flow preserves counters for modify strict
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Insert a flow. Send data plane packets to increase counters. Modify the Flow Action with OFPFC_MODIFY_STRICT. Get flow statistics, verify counters are preserved.
Results	Pass or Fail
Remarks	

Test case 40.110: Delete non-existent flow

Test Number	40.110
Test Title	Flow Table Modification Messages / Modify / Delete non-existent flow
Test Purpose	Verify that deleting a non-existent flow does not generate an error
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *For DELETE requests, if no flow entry matches, no error is recorded, and no flow table modification occurs.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Deleting a non-existent flow does not generate an error

Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Send a delete request for a non-existent flow. Verify no error is returned.
Results	Pass or Fail
Remarks	

Test case 40.120: Delete flows with and without flow_removed flag set

Test Number	40.120
Test Title	Flow Table Modification Messages / Modify / Delete flows with and without flow_removed flag set
Test Purpose	Verify that deleting a flow with send flow removed flag set triggers a flow removed message, and deleting a flow without the send flow removed flag set does not trigger a flow removed message.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *If flow entries match, and must be deleted, then each normal entry with the OFPFF_SEND_FLOW_REM flag set should generate a flow removed message.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of the OFPFF_SEND_FLOW_REM flag
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate two flows, one of them with OFPFF_SEND_FLOW_REM set. Delete both entries, verify only the one with the OFPFF_SEND_FLOW_REM flag set, generates an OFPT_FLOW_REMOVED message.
Results	Pass or Fail
Remarks	

Test case 40.130: Delete emergency flow

Test Number	40.130
Test Title	Flow Table Modification Messages / Modify / Delete emergency flow
Test Purpose	Verify that deleting an emergency flow does not trigger a flow removed message.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *Deleted emergency flow entries generate no flow removed messages*
Profile Status	MANDATORY for Emergency Mode
Requirements	Correct implementation of emergency flow removal
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Insert a flow with the OFPFF_EMERG flag set. Send flow_mod message command DELETE with the OFPFF_EMERG flag set. Verify no OFPT_FLOW_REMOVED message is generated.
Results	Pass or Fail or Not Tested
Remarks	

Test case 40.140: Delete without wildcards

Test Number	40.140

Test Title	Flow Table Modification Messages / Modify / Delete without wildcards
Test Purpose	Verify that flow_mod delete and strict_delete map to the correct flows
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *MODIFY and DELETE flow mod commands have corresponding STRICT versions. Without STRICT appended, the wildcards are active and all flows that match the description are modified or removed. If STRICT is appended, all fields, including the wildcards and priority, are strictly matched against the entry, and only an identical flow is modified or removed*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of strict and non strict matching
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. In this case, strict_delete and delete are identical. Insert a flow F1 with all header fields set. Issue a strict_delete message matching F1, and verify F1 is deleted. Insert a flow F2 with all header fields set. Issue a delete message matching F2, and verify F2 is deleted.
Results	Pass or Fail
Remarks	

Test case 40.150: Delete with wildcards set

Test Number	40.150
Test Title	Flow Table Modification Messages / Modify / Delete with wildcards set
Test Purpose	Verify that delete maps to the correct flows
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *MODIFY and DELETE flow mod commands have corresponding STRICT versions. Without STRICT appended, the wildcards are active and all flows that match the description are modified or removed. If STRICT is appended, all fields, including the wildcards and priority, are strictly matched against the entry, and only an identical flow is modified or removed*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of non strict delete matching
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. In this case both flows are matched by the wildcards in the delete. Insert a flow T1 with all header fields set, except Ethernet source address, this is wildcarded. Insert a flow T2 with only ingress port set, all other fields are wilcarded. The ingress port of T2 is identical to the ingress port of T1. Issue a delete message matching on ingress port of both flows (T1,T2), all other fields are wild carded. Verify that both flows (T1, T2) are deleted.
Results	Pass or Fail
Remarks	

Test case 40.160: Strict_Delete with wildcards set

Test Number	40.160

Test Title	Flow Table Modification Messages / Modify / Strict_Delete with wildcards set
Test Purpose	Verify that strict_delete maps to the correct flows
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *MODIFY and DELETE flow mod commands have corresponding STRICT versions. Without STRICT appended, the wildcards are active and all flows that match the description are modified or removed. If STRICT is appended, all fields, including the wildcards and priority, are strictly matched against the entry, and only an identical flow is modified or removed*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of strict_delete matching
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. In this case, only T2 matches. Insert a flow T1 with all header fields set, except Ethernet source address, this is wildcarded. Insert a flow T2 with only ingress port set, all other fields are wilcarded. The ingress port of T2 is identical to the ingress port of T1. Issue a strict_delete message matching on ingress port of both flows (T1,T2), all other fields are wild carded. Verify that only flow T2 gets deleted.
Results	Pass or Fail
Remarks	

Test case 40.170: Testing that delete message ignores priorities

Test Number	40.170
Test Title	Flow Table Modification Messages / Modify / Testing that delete message ignores priorities
Test Purpose	Verify that delete maps to the correct flows
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *MODIFY and DELETE flow mod commands have corresponding STRICT versions. Without STRICT appended, the wildcards are active and all flows that match the description are modified or removed. If STRICT is appended, all fields, including the wildcards and priority, are strictly matched against the entry, and only an identical flow is modified or removed*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of non strict delete matching
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Insert a priority 100 flow T1 with all header fields set, except Ethernet source address, this is wildcarded. Insert a priority 200 flow T2 with only ingress port set, all other fields are wilcarded. The ingress port of T2 is identical to the ingress port of T1. Insert a flow T3, identical to T2, except the Priority is set to 300. Issue a delete message matching on ingress port of all flows (T1, T2, T3) with priority 200 as additional constraint. All other fields are wild carded. Verify all flows (T1, T2, T3) are deleted.
Results	Pass or Fail

Remarks	

Test case 40.180: Testing that strict_delete message does not ignore priorities

Test Number	40.180
Test Title	Flow Table Modification Messages / Modify Testing that strict_delete message does not ignore priorities
Test Purpose	Verify that delete maps to the correct flows
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *MODIFY and DELETE flow mod commands have corresponding STRICT versions. Without STRICT appended, the wildcards are active and all flows that match the description are modified or removed. If STRICT is appended, all fields, including the wildcards and priority, are strictly matched against the entry, and only an identical flow is modified or removed*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of strict and non strict matching
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Insert a priority 100 flow T1 with all header fields set, except Ethernet source address, this is wildcarded. Insert a priority 200 flow T2 with only ingress port set, all other fields are wilcarded. The ingress port of T2 is identical to the ingress port of T1. Insert a flow T3, identical to T2, except the Priority is set to 300. Issue a strict_delete message matching on ingress port of all flows (T1, T2, T3) with priority 200 as additional constraint. All other fields are wild carded. Verify only T2 gets deleted.
Results	Pass or Fail
Remarks	

Test case 40.190: Delete with constraint out_port

Test Number	40.190
Test Title	Flow Table Modification Messages / Modify / Delete with constraint out_port
Test Purpose	Verify that delete supports filtering on action out_port
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *DELETE and DELETE STRICT commands can be optionally filtered by output port. If the out_port field contains a value other than OFPP_NONE, it introduces a constraint when matching. This constraint is that the rule must contain an output action directed at that port.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of filtering delete commands based on out_port
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Insert two

	identical flows forwarding to two different out_ports. Send an exact match delete request for these flows, but specify only one of the two ports as out_port. Check that only the flow with the specified out_port is deleted.
Results	Pass or Fail
Remarks	

Test case 40.200: out_port ignored by Add and Modify requests

Test Number	40.200
Test Title	Flow Table Modification Messages / Modify / out_port ignored by Add and Modify requests
Test Purpose	Verify that out_port values in FLOW_MOD Add or Modify requests are ignored.
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification Messages, p. 13. *DELETE and DELETE STRICT commands can be optionally filtered by output port. If the out_port field contains a value other than OFPP_NONE, it introduces a constraint when matching.* *…* *This field is ignored by ADD, MODIFY, and MODIFY STRICT messages*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of FLOW_MOD Add and Modify requests
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Send command ofp_flow_mod add with out_port value set to some port. Verify flow is added but out_port field is ignored. Send command ofp_flow_mod modify with out_port value set to some port . Verify flow modification takes places but, out_port is ignored.
Results	Pass or Fail
Remarks	Flow Delete, Delete_Strict, Add, Modify and Modify Strict all share the same format.

Test case 40.210: Timeout with flow removed message

Test Number	40.210
Test Title	Flow Table Modification Messages / Flow removal / Timeout with flow removed message
Test Purpose	Verify flow removed message for timeout is implemented
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.7 Flow removal, p. 15. *When the OFPFF_SEND_FLOW_REM flag is set, the switch must send a flow removed message when the flow expires. The default is for the switch to not send flow removed messages for newly added flows*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of flow removed messages for timeout
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow

	with hard timeout = 1 sec and OFPFF_SEND_FLOW_REM flag set. Send for n seconds packets matching the flow to the data plane, then stop. Verify the OFPT_FLOW_REMOVED message is received with duration_sec field set to 1 sec
Results	Pass or Fail
Remarks	

Test case 40.220: Idle timeout

Test Number	40.220
Test Title	Flow Table Modification Messages / Flow removal / Idle timeout
Test Purpose	Verify that idle timeout is implemented
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.7 Flow removal, p. 15. *Each flow entry has an idle_timeout and a hard_timeout associated with it. If no packet has matched the rule in the last idle_timeout seconds, or it has been hard_timeout seconds since the flow was inserted, the switch removes the entry and sends a flow removed message.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of idle timeout
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with idle timeout = 1 sec. Send packets matching the flow to the data plane for n seconds, then stop. Verify the flow expiration message is received and duration_sec field is (n+1)sec.
Results	Pass or Fail
Remarks	

Test case 40.230: hard timeout

Test Number	40.230
Test Title	Flow Table Modification Messages / Flow removal / hard timeout
Test Purpose	Verify that hard timeout is implemented
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.7 Flow removal, p. 15. *Each flow entry has an idle_timeout and a hard_timeout associated with it. If no packet has matched the rule in the last idle_timeout seconds, or it has been hard_timeout seconds since the flow was inserted, the switch removes the entry and sends a flow removed message.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of hard timeout
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with hard timeout = 1 sec. Send packets matching the flow to the data plane for n >= 2 second, then stop. Verify the flow expiration message is received and duration_sec field is 1sec
Results	Pass or Fail
Remarks	

Test Suite 50: Flow Matching

Test suite 50 deals with all cases of matching the flow table. This is where the profiles defined for L2 and L3 are most relevant. In this group, each profile will have a different set of required versus OPTIONAL header fields to match on. Tests focusing on flow entry priority and fragmented IP packet matching are also part of this group.

Special cases:
Each profile defines a different set of MANDATORY and OPTIONAL test cases in this group. These are defined in the profiles section in this test suite. All matches MUST consider the constraints given in OpenFlow spec 1.0, Table2, pp. 3 and 4. So, for example, even the Single header field match for VLAN ID requires a non-wildcarded Ethertype. These constraints should be implemented in the test cases where needed, without being mentioned in the test list for every occurrence.

Profiles:
Full conformance:
MUST pass all tests from 50.10 to 50.190

L2 Profile:
MUST pass test-cases 50.10, 50.20, 50.30, 50.40, 50.50, 50.60, 50.140, 50.190

To satisfy the L2 profile requirements, the device MUST be able to match all of the fields listed in Table 1 both individually and simultaneously under the constraints given in the specification and outlined in table 1. When testing a device against this profile, all other match fields not listed in Table 1 will be wildcarded.

Additionally the DUT MUST be able to match with all fields wildcarded under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.

Table 1: L2 Profile Field lengths and the way they must be applied to flow entries (Excerpt from OpenFlow Switch Specification 1.0 Table 3 p. 4)

Field	Bits	When applicable	Notes
Ingress Port	(Implementation dependent)	All packets	Numerical representation of incoming port, starting at 1.
Ethernet source address	48	All packets on enabled ports	
Ethernet destination address	48	All packets on enabled ports	
Ethernet type	16	All packets on enabled ports	An OpenFlow switch is required to match the type in both standard Ethernet and 802.2 with a SNAP header and OUI of 0x000000. The special value of 0x05FF is used to match all 802.3 packets without SNAP headers.
VLAN id	12	All packets of Ethernet type 0x8100	

L3 Profile:
MUST pass 50.10,50.20, 50.50, 50.80, 50.90, 50.150, 50.190

To satisfy the L3 profile requirements, the device MUST be able to match all of the fields listed in Table2 both individually and simultaneously under the constraints given in the specification and outlined in Table 2. When testing a device against this profile, all other match fields not listed in Table 2 will be wildcarded.

Additionally it MUST be able to match with all fields wildcarded under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.

Table 2: L3 Profile Field lengths and the way they must be applied to flow entries (Excerpt from OpenFlow Switch Specification 1.0 Table 3 p. 4)

Field	Bits	When applicable	Notes
Ingress Port	(Implementation dependent)	All packets	Numerical representation of incoming port, starting at 1.
Ethernet type	16	All packets on enabled ports	An OpenFlow switch is required to match the type in both standard Ethernet and 802.2 with a SNAP header and OUI of 0x000000. The special value of 0x05FF is used to match all 802.3 packets without SNAP headers.
IP source address	32	All IP and ARP packets	Can be subnet masked
IP destination address	32	All IP and ARP packets	Can be subnet masked

Test case 50.10: All Wildcards

Test Number	50.10
Test Title	Data plane / Matching / All Wildcards
Test Purpose	Test matching a global (all wildcards) Flow
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Matching all wildcards
Topology	Control-plane connection between DUT and reference controller. At least three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with all header fields wildcarded and action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet is forwarded only to the second port.
Results	Pass or Fail
Remarks	

Test case 50.20: Ingress Port (uint16_t in_port)

Test Number	50.20
Test Title	Data plane / Matching Single Header Field/ Ingress Port (uint16_t in_port)
Test Purpose	Matching against a flow with Ingress Port set, all other fields are wildcarded
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field*

	has a value of ANY, it matches all possible values in the header
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Matching ingress port
Topology	Control-plane connection between DUT and reference controller. At least three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except Ingress Port and action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail
Remarks	

Test case 50.30: Ethernet source address (dl_src[OFP_ETH_ALEN])

Test Number	50.30
Test Title	Data plane / Matching Single Header Field/ Ethernet source address (dl_src[OFP_ETH_ALEN])
Test Purpose	Matching against a flow with Ethernet source address set, all other fields are wildcarded
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for Full and L2 Profile
Requirements	Correct implementation of Matching Ethernet source address
Topology	Control-plane connection between DUT and reference controller. At least three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except Ethernet source address and action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	

Test case 50.40: Ethernet destination address (dl_dst[OFP_ETH_ALEN])

Test Number	50.40
Test Title	Data plane / Matching Single Header Field/ Ethernet destination address (dl_dst[OFP_ETH_ALEN])
Test Purpose	Matching against a flow with Ethernet destination address set, all other fields are wildcarded
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*

Profile Status	MANDATORY for Full and L2 Profile
Requirements	Correct implementation of Matching Ethernet destination address
Topology	Control-plane connection between DUT and reference controller. At least three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except Ethernet destination address and action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	

Test case 50.50: Ethernet frame type (uint16_t dl_type)

Test Number	50.50
Test Title	Data plane / Matching Single Header Field/ Ethernet frame type (uint16_t dl_type)
Test Purpose	Matching against a flow with Ethernet frame type set, all other fields are wildcarded (under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Matching Ethernet frame type
Topology	Control-plane connection between DUT and reference controller. At least three data plane connections on the DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except Ethernet frame type and action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail
Remarks	Notes: An OpenFlow switch is required to match the type in both standard Ethernet and 802.2 with a SNAP header and OUI of 0x000000. The special value of 0x05FF is used to match all 802.3 packets without SNAP headers. To handle the various Ethernet framing types, matching the Ethernet type is handled in a slightly different manner. If the packet is an Ethernet II frame, the Ethernet type is handled in the expected way. If the packet is an 802.3 frame with a SNAP header and Organizationally Unique Identifier (OUI) of 0x000000, the SNAP protocol id is matched against the flows Ethernet type. A flow entry that specifies an Ethernet type of 0x05FF, matches all Ethernet 802.2 frames without a SNAP header and those with SNAP headers that do not have an OUI of

	0x000000."

Test case 50.60: Input VLAN id (uint16_t dl_vlan)

Test Number	50.60
Test Title	Data plane / Matching Single Header Field/ Input VLAN id (uint16_t dl_vlan)
Test Purpose	Matching against a flow with Input VLAN id set, all other fields are wildcarded (under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for Full and L2 Profile
Requirements	Correct implementation of Matching input VLAN id
Topology	Control-plane connection between DUT and reference controller. At least three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except Input VLAN id and action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	If the packet is a VLAN (Ethernet type 0x8100), the VLAN ID and PCP fields are used in the lookup.

Test case 50.70: Input VLAN priority (uint8_t dl_vlan_pcp)

Test Number	50.70
Test Title	Data plane / Matching Single Header Field/ Input VLAN priority (uint8_t dl_vlan_pcp)
Test Purpose	Matching against a flow with Input VLAN priority set, all other fields are wildcarded (under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for Full Profile
Requirements	Correct implementation of Matching VLAN priority
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except Input VLAN priority and action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send

	a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	If the packet is a VLAN (Ethernet type 0x8100), the VLAN ID and PCP fields are used in the lookup

Test case 50.80: IP source address (uint32_t nw_src)

Test Number	50.80
Test Title	Data plane / Matching Single Header Field/ IP source address (uint32_t nw_src)
Test Purpose	Matching against a flow with IP source address and netmask set, all other fields are wildcarded (under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for Full and L3 Profile
Requirements	Correct implementation of Matching IP source address
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except IP source address and netmask with action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	

Test case 50.90: IP destination address (uint32_t nw_dst)

Test Number	50.90
Test Title	Data plane / Matching Single Header Field/ IP destination address (uint32_t nw_dst)
Test Purpose	Matching against a flow with IP destination address and netmask set, all other fields are wildcarded (under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for Full and L3 Profile
Requirements	Correct implementation of Matching IP destination address
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.

Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except IP destination address and netmask with action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	

Test case 50.100: IP protocol (uint8_t nw_proto)

Test Number	50.100
Test Title	Data plane / Matching Single Header Field/ IP protocol (uint8_t nw_proto)
Test Purpose	Matching against a flow with IP protocol set, all other fields are wildcarded (under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for Full Profile
Requirements	Correct implementation of Matching IP protocol
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except IP protocol and action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	

Test case 50.110: IP TOS bits (uint8_t nw_tos)

Test Number	50.110
Test Title	Data plane / Matching Single Header Field/ IP TOS bits (uint8_t nw_tos)
Test Purpose	Matching against a flow with IP TOS bits set, all other fields are wildcarded (under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for Full Profile
Requirements	Correct implementation of Matching IP TOS bits
Topology	Control-plane connection between DUT and reference controller.

	At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except IP TOS and action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	

Test case 50.120: TCP/UDP source port (uint16_t tp_src)

Test Number	50.120
Test Title	Data plane / Matching Single Header Field/ TCP/UDP source port (uint16_t tp_src)
Test Purpose	Matching against a flow with TCP/UDP source port set, all other fields are wildcarded (under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for Full Profile
Requirements	Correct implementation of Matching TCP/UDP source port
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except TCP/UDP source port and action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	For IP packets that are TCP or UDP (IP protocol is equal to 6 or 17), the lookup includes the transport ports. For IP packets that are ICMP (IP protocol is equal to 1), the lookup includes the Type and Code fields

Test case 50.130: TCP/UDP destination port (uint16_t tp_dst)

Test Number	50.130
Test Title	Data plane / Matching Single Header Field/ TCP/UDP destination port (uint16_t tp_dst)
Test Purpose	Matching against a flow with TCP/UDP destination port set, all other fields are wildcarded (under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*

Profile Status	MANDATORY for Full Profile
Requirements	Correct implementation of Matching TCP/UDP destination port
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except TCP/UDP destination port and action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	For IP packets that are TCP or UDP (IP protocol is equal to 6 or 17), the lookup includes the transport ports. For IP packets that are ICMP (IP protocol is equal to 1), the lookup includes the Type and Code fields

Test case 50.140: L2

Test Number	50.140
Test Title	Data plane / Matching Multiple Header Fields / L2
Test Purpose	Matching against a flow with Ingress Port, Ethernet source address, Ethernet destination address, Ethernet type and VLAN id set, all other fields are wildcarded (under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for Full and L2 Profile
Requirements	Correct implementation of Matching all L2 profile fields
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except Ingress Port, Ethernet source address, Ethernet destination address, Ethernet type and VLAN id, action OFPAT_OUTPUT set to egress port. Send a packet matching the flow. Verify the packet is forwarded only to the second port. Send a non-matching packet. Verify a packet_in is generated and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	On hybrid switches VLAN id ranges may need to be preconfigured.

Test case 50.150: L3

Test Number	50.150
Test Title	Data plane / Matching Multiple Header Fields / L3
Test Purpose	Matching against a flow with Ingress Port, IP source address and netmask, IP destination address and netmask and Ethernet type set, all other fields are wildcarded (under the constraints given in the

	specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for Full and L3 Profile
Requirements	Correct implementation of Matching all L3 profile fields
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except Ingress Port, IP source address and netmask, IP destination address and netmask, and Ethernet type with action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	

Test case 50.160: L4

Test Number	50.160
Test Title	Data plane / Matching Multiple Header Fields / L4
Test Purpose	Matching against a flow with Ingress Port, IP protocol, TCP/UDP source port, TCP/UDP destination port and Ethernet type set, all other fields are wildcarded (under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for Full Profile
Requirements	Correct implementation of Matching all L4 fields
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except Ingress Port, IP protocol TCP/UDP source port, TCP/UDP destination port and Ethernet type with action OFPAT_OUTPUT set to second port. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	

Test case 50.170: Exact match

Test Number	50.170

Test Title	Data plane / Matching Multiple Header Fields / Exact match
Test Purpose	Matching against a flow with all Header fields set. (Under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	MANDATORY for Full Profile
Requirements	Correct implementation of Exact Matching
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields set. Send a packet matching the flow. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	

Test case 50.180: Exact match priority

Test Number	50.180
Test Title	Data plane / Matching / Exact match priority
Test Purpose	Verifying that a flow with all Header fields set has the highest priority. (Under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *Packets are matched against flow entries based on prioritization. An entry that specifies an exact match (i.e., it has no wildcards) is always the highest priority*
Profile Status	MANDATORY for Full Profile
Requirements	Correct implementation of Matching
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields set, action output port two. Add a second flow with at least one field not wildcarded and highest possible priority, action output port three. Send a packet matching both flows to the data plane. Verify the packet gets forwarded only to the second port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	

Test case 50.190: Match priorities

Test Number	50.190
Test Title	Data plane / Matching / Match priorities
Test Purpose	Verifying that flows with different priorities match in the correct order.

Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *Packets are matched against flow entries based on prioritization. An entry that specifies an exact match (i.e., it has no wildcards) is always the highest priority*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Matching priorities
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add two flows with different priorities. Send a packet matching both flows to the data plane. Verify the packet matches the higher priority flow.
Results	Pass or Fail
Remarks	

Test case 50.200: Fragments wildcard TCP port

Test Number	50.200
Test Title	Data plane / Matching / Fragments wildcard TCP port
Test Purpose	Verifying that when matching on fragments the TCP ports are ignored.
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *For IP packets with nonzero fragment offset or More Fragments bit set, the transport ports are set to zero for the lookup.*
Profile Status	OPTIONAL
Requirements	Correct implementation of Matching Fragments
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add two flows with different priorities and different TCP Ports set. Send a fragmented packet with TCP Ports matching the lower priority flow. Verify all packet fragments match the lower priority flow.
Results	Pass or Fail or Not Tested
Remarks	

Test case 50.210: IP source address of ARP packets(uint32_t nw_src)

Test Number	50.210
Test Title	Data plane / Matching Single Header Field/ IP source address of ARP packets(uint32_t nw_src)
Test Purpose	For ARP packets (Ethernet type equal to 0x0806), the lookup elds may also include the contained IP source and destination fields(under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	OPTIONAL
Requirements	Correct implementation of Matching IP source address of ARP packets.
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.

Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except Ether Type being 0x806 and IP source address and netmask with action OFPAT_OUTPUT set to egress port. Send a packet matching the flow. Verify the packet gets forwarded only to the egress port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	

Test case 50.220: IP destination address of ARP packets(uint32_t nw_src)

Test Number	50.220
Test Title	Data plane / Matching Single Header Field/ IP destination address of ARP packets(uint32_t nw_src)
Test Purpose	For ARP packets (Ethernet type equal to 0x0806), the lookup fields may also include the contained IP source and destination fields(under the constraints given in the specification as listed in OpenFlow Switch Specification 1.0.0 Table 3 on p. 4.).
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.4 Matching, p. 8. *A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header*
Profile Status	OPTIONAL
Requirements	Correct implementation of Matching IP destination address of ARP packets.
Topology	Control-plane connection between DUT and reference controller. At least Three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow with All header fields wildcarded except Ether Type being 0x806, IP destination address and netmask with action OFPAT_OUTPUT set to egress port. Send a packet matching the flow. Verify the packet gets forwarded only to the egress port. Send a packet not matching the flow, verify a packet_in is generated, and the packet is not forwarded on the data plane.
Results	Pass or Fail or Not Tested
Remarks	

Test Suite 60: Counters

Test suite 60, named Counters, checks for the correct implementation of counters in the Devices. Counters are checked per Flow, per Port, per Queue and per table. Table 4, p5 of the OpenFlow Switch Specification 1.0.0 lists required counters for use in statistics messages, but the existence of a capabilities reporting field for each of these types of counters in the features reply (p. 25 o the OpenFlow Switch Specification 1.0.0) would seem to indicate these are optional. Since queues are configured locally on the DUT and outside of the OpenFlow protocol, we have left support of the enqueue action and it's respective counters as OPTIONAL, but all other counters are MANDATORY except Packet Lookup & Matched Count counters.

60.10 to 60.40	per flow counters
60.50 to 60.160	per port counters
60.170 to 60.190	per queue counters
60.200	per table counters

Special cases:
Some counters may not be reliably triggered on every device (e.g. transmit overrun error). In these cases only the existence of the counter will be verified, but not correct counting.

Queue counters are OPTIONAL (60.170 to 60.190).
Packet Lookup & Matched Count counters are OPTIONAL (60.210).

Profiles:
All profiles MUST pass all tests except 60.170, 60.180, 60.190 and 60.210

Test case 60.10: Received Packets

Test Number	60.10
Test Title	Counters/ Per Flow / Received Packets
Test Purpose	Verify that the packet_count counter in the Flow-stats reply increments in accordance with packets received.
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per flow packet_count counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to the DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow. Send N matching packets. Send OFPST_FLOW request. Verify reply packet_count counter is incremented correctly.
Results	Pass or Fail
Remarks	

Test case 60.20: Received Bytes

Test Number	60.20
Test Title	Counters/ Per Flow / Received Bytes
Test Purpose	Verify that the byte_count counter in the Flow-stats reply increments in accordance with packets received.
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per flow byte_count counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.

Methodology	Configure and connect the Primary-controller on the DUT. Add a flow. Send matching packets with N total bytes. Send OFPST_FLOW request. Verify reply byte_count counter is incremented correctly
Results	Pass or Fail
Remarks	Some DUTs may count the FCS and others may not.

Test case 60.30: Duration (secs)

Test Number	60.30
Test Title	Counters/ Per Flow / Duration (secs)
Test Purpose	Verify that the duration_sec counter in the Flow_stats reply increments in accordance with the time the flow has been alive
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per flow duration_sec counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow. Send matching packets. Send multiple OFPST_FLOW requests within a certain time interval (e.g. 1 per 10sec for 60 sec). Verify duration_sec counter is incremented correctly
Results	Pass or Fail
Remarks	

Test case 60.40: Duration (nsecs)

Test Number	60.40
Test Title	Counters/ Per Flow / Duration (nsecs)
Test Purpose	Verify that the duration_nsec counter in the Flow_stats reply increments in accordance with the time the flow has been alive
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per flow duration_nsec counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow. Send matching packets. Send OFPST_FLOW request within certain time intervals (e.g. 1 per 10sec for 60 sec). Verify duration_nsec counter is incremented correctly.
Results	Pass or Fail
Remarks	

Test case 60.50: Received Packets

Test Number	60.50
Test Title	Counters/ Per Port / Received Packets
Test Purpose	Verify that the rx_packets counter in the Port_Stats reply increments in

	accordance with the packets received
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per port rx_packets counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Send OFPST_PORT request for ingress port . Note current rx_packets value. Add a flow. Send N matching packets to ingress port. Send OFPST_PORT request for ingress port, and verify the reply contains the correct rx_packets count (i.e. previous rx_packets count + N)
Results	Pass or Fail
Remarks	

Test case 60.60: Transmitted Packets

Test Number	60.60
Test Title	Counters/ Per Port / Transmitted Packets
Test Purpose	Verify that the tx_packets counter in the Port_Stats reply increments in accordance with the packets transmitted
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per port tx_packets counters
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Send OFPST_PORT stats request for egress port. Note current tx_packets value. Add a flow with output action to an egress port. Send N matching packets. Send OFPST_PORT request for the egress port, and verify the reply contains the correct tx_packets count (i.e previous tx_packets counter + N)
Results	Pass or Fail
Remarks	

Test case 60.70: Received bytes

Test Number	60.70
Test Title	Counters/ Per Port / Received bytes
Test Purpose	Verify that the rx_bytes counter in the Port_Stats reply increments in accordance with the bytes received
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per port rx_bytes counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.

Methodology	Configure and connect the Primary-controller on the DUT. Send OFPST_PORT request for ingress port. Note the current rx_bytes value. Add a flow. Send N matching packets to ingress port. Send OFPST_PORT request for the ingress port. Verify the reply contains the correct rx_bytes count (i.e previous rx_bytes + N *(No. of bytes in each packet))
Results	Pass or Fail
Remarks	

Test case 60.80: Transmitted bytes

Test Number	60.80
Test Title	Counters/ Per Port / Transmitted bytes
Test Purpose	Verify that the tx_bytes counter in the Port_Stats reply increments in accordance with the packets transmitted
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Counters
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections.
Methodology	Configure and connect the Primary-controller on the DUT. Send OFPST PORT stats request fot egress port. Add a flow with output action to egress port, send N matching packets. Send OFPST_PORT request for the egress port, and verify the reply contains the correct transmitted bytes count i.e previous tx_bytes counter + N*(No. of bytes in a packet)
Results	Pass or Fail
Remarks	

Test case 60.90: Receive drops

Test Number	60.90
Test Title	Counters/ Per Port / Receive drops
Test Purpose	Verify that the rx_dropped counter in the Port_Stats reply increments in accordance with the packets dropped
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per port rx_dropped counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow. Trigger rx_dropped counter. Send OFPST_PORT request for the ingress port, and verify the reply contains the correct rx_dropped count
Results	Pass or Fail
Remarks	rx_dropped counters may not be reliably triggered. If unable to trigger the rx_dropped counter, then the DUT will pass if rx_dropped counter

	exists.

Test case 60.100: Transmit drop

Test Number	60.100
Test Title	Counters/ Per Port / Transmit drops
Test Purpose	Verify that the tx_dropped counter in the Port_Stats reply increments in accordance with the packets dropped
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per port tx_dropped counters
Topology	Control-plane connection between DUT and reference controller. One egress data plane connection to DUT. At least one ingress data plane connection to DUT. Total ingress bandwidth must exceed total egress bandwidth.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow on each ingress port with the action set to output to the egress port. Send matching packets to each ingress port at a combined higher rate than the egress port supports. Send OFPST_PORT request for the egress port, and verify the reply contains the correct tx_dropped count.
Results	Pass or Fail
Remarks	

Test case 60.110: Receive Errors

Test Number	60.110
Test Title	Counters/ Per Port / Receive Errors
Test Purpose	Verify that the rx_errors counter in the Port_Stats reply increments in accordance with errors encountered while switch is receiving.
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per port rx_errors counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow. Send N matching packets containing errors (e.g. frame alignment or crc or overrun). Send OFPST_PORT request for the ingress port, and verify the reply contains the correct error count.
Results	Pass or Fail
Remarks	

Test case 60.120: Transmit Errors

Test Number	60.120
Test Title	Counters/ Per Port / Transmit Errors
Test Purpose	Verify that the tx_errors counter in the Port_Stats reply increments in accordance with errors encountered while switch is sending.

Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3; 5.3.5 Read State Messages, p. 30; struct ofp_port_stats p. 34 *uint64_t tx_errors;* */* Number of transmit errors. This is a super-set of more specific transmit errors and should be greater than or equal to the sum of all tx_*_err values (none currently defined.) */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per port tx_errors counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Send OFPST_PORT request for the egress port, and verify the reply contains the correct transmit error count.
Results	Pass or Fail
Remarks	Since no tx_*_err values are currently defined. Only the existence of the tx_error counter is verified.

Test case 60.130: Receive Frame Errors

Test Number	60.130
Test Title	Counters/ Per Port / Receive Frame Errors
Test Purpose	Verify that the rx_frame_err counter in the Port_Stats reply increments in accordance with errors the switch is receiving.
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per port rx_frame_err counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow. Send matching packets containing frame alignment errors. Send OFPST_PORT request for the ingress port, and verify the reply contains the correct rx_frame_err count.
Results	Pass or Fail
Remarks	

Test case 60.140: Receive Overrun Errors

Test Number	60.140
Test Title	Counters/ Per Port / Receive Overrun Errors
Test Purpose	Verify that the rx_over_err counter in the Port_Stats reply increments in accordance with errors the switch is receiving.
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per port rx_over_err counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger

	rx_over_err counter. Send OFPST_PORT request for the ingress port, and verify the reply contains the correct overrun count
Results	Pass or Fail
Remarks	rx_over_err counters may not be reliably triggered. If unable to trigger the rx_over_err counter, then the DUT will pass if rx_over_err counter exists.

Test case 60.150: CRC Errors

Test Number	60.150
Test Title	Counters/ Per Port / CRC Errors
Test Purpose	Verify that the rx_crc_err counter in the Port_Stats reply increments in accordance with crc errors the switch is receiving.
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per port rx_crc_err counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow. Send N matching packets containing crc errors. Send OFPST_PORT request for the ingress port, and verify the reply contains the correct rx_crc_err count
Results	Pass or Fail
Remarks	

Test case 60.160: collisions

Test Number	60.160
Test Title	Counters/ Per Port / Collisions
Test Purpose	Verify that the collisions counter in the Port_Stats reply increments in accordance with collisions errors the switch is receiving.
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per port collisions counters
Topology	Control-plane connection between DUT and reference controller. At least two data plane connection to DUT. Set one data plane connection to half duplex.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow in each direction. Send packets at a high rate to half-duplex ingress port. Generate collisions by sending packets at a high rate through DUT to half-duplex egress port. Send OFPST_PORT request for the half-duplex port, and verify the reply contains the correct collisions count
Results	Pass or Fail
Remarks	

Test case 60.170: Transmit Packets

Test Number	60.170
Test Title	Counters/ Per Queue / Transmit Packets
Test Purpose	Verify that the tx_packets counter in the Queue_Stats reply increments in accordance with packets transmitted from the queue.
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	OPTIONAL
Requirements	Correct implementation of per queue tx_packets counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Configure a queue and map it to a port. Create a flow with action OFPAT_ENQUEUE and mapped to the queue. Send N matching packets. Send queue_stats request for the ingress port, and verify the reply contains the correct tx_packets count
Results	Pass or Fail or Not Tested
Remarks	

Test case 60.180: Transmit bytes

Test Number	60.180
Test Title	Counters/ Per Queue / Transmit bytes
Test Purpose	Verify that the tx_bytes counter in the Queue_Stats reply increments in accordance with bytes transmitted from the queue.
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	OPTIONAL
Requirements	Correct implementation of per queue tx_bytes counters
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Configure a queue and map it to a port. Create a flow with action OFPAT_ENQUEUE and mapped to the queue. Send N matching packets. Send queue_stats request for the ingress port and verify the reply contains the correct tx_bytes count
Results	Pass or Fail or Not Tested
Remarks	

Test case 60.190: Transmit Overrun Errors

Test Number	60.190
Test Title	Counters/ Per Queue / Transmit Overrun Errors
Test Purpose	Verify that the tx_errors counter in the Queue_Stats reply increments in accordance with bytes transmitted from the queue.
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	OPTIONAL

Requirements	Correct implementation of per queue tx_errors counters
Topology	Control-plane connection between DUT and reference controller. One egress data plane connection to DUT. At least one ingress data plane connection to DUT. Total ingress bandwidth must exceed total egress bandwidth.
Methodology	Configure and connect the Primary-controller on the DUT. Add a flow on each ingress port with the action set to output to the egress port. Map each flow to an egress port queue. Send matching packets to each ingress port at a combined higher rate than the egress port supports. Send queue_stats request for the egress port, and verify the reply contains the correct tx_errors count.
Results	Pass or Fail or Not Tested
Remarks	

Test case 60.200: Active Entries

Test Number	60.200
Test Title	Counters/ Per Table / Active Entries
Test Purpose	Verify that the active_count counter in the Table_Stats reply increments in accordance with the number of active flow entries in the table of the switch.
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of per table active_count counters
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Create N flows with long idle timeout, Insert flows in the switch. Send Table_Stats request. Verify the reply contains the correct active_count value.
Results	Pass or Fail
Remarks	

Test case 60.210: Packet Lookup & Matched Count

Test Number	60.210
Test Title	Counters/ Per Table / Packet Lookup & Matched Count
Test Purpose	Verify that lookup_count & matched_count counter in the Table_Stats reply increments in accordance with the number of packets looked up and the packets matched.
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.2 Counters, p. 3, 5.3.5 Read State Messages, p. 30
Profile Status	OPTIONAL
Requirements	Correct implementation of per table lookup_count and matched_count counters
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Insert a

	flow. Send N packets matching the flow and N' non-matching packets. Send OFPST_TABLE request. Verify lookup_count = N'+ N and matched_count= N.
Results	Pass or Fail
Remarks	

Test Suite 70: Actions

Test suite 70 tests all the data plane actions a switch can support. The test suite contains tests for nine forwarding actions (70.30 to 70.110) and eleven header field write actions (70.120 to 70.230).

The OpenFlow Switch 1.0 Specification does not clarify what is considered an illegal action order. Due to this ambiguity, what is considered an illegal action order may vary based on the implementation. While it seems reasonable to consider a header modify action with no subsequent output or forward action as an illegal ordering, we have left tests 70.240 and 70.250 as OPTIONAL.

Special cases:

The following Forwarding actions are required:

All:	70.30
Controller:	70.40
Table:	70.60
In_port:	70.70

The following Forwarding actions are OPTIONAL:

Local:	70.50
Normal:	70.80
Flood:	70.90
Multiple Ports:	70.100
Enqueue:	70.110

All the write Actions are OPTIONAL:
70.120 to 70.230

Action ordering tests are OPTIONAL
70.240 and 70.250

Profiles:
All profiles MUST pass tests 70.10, 70.20, 70.30, 70.40, 70.60, 70.70

Test case 70.10: No action drops packet

Test Number	70.10
Test Title	Data plane / Actions / No action drops packet
Test Purpose	Verify that flows without a forward action drop matching packets

Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 3. *Required Action: Drop. A flow-entry with no specified action indicates that all matching packets should be dropped.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of drop action
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create flow without action and/or create flows with actions but no forward action. Send packets matching the flow. Verify packets are dropped and flow counters are incremented.
Results	Pass or Fail
Remarks	

Test case 70.20: Get supported actions

Test Number	70.20
Test Title	Data plane / Actions / Get supported actions
Test Purpose	Get the supported actions from switch and make sanity checks
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 3. *A switch is not required to support all action types — just those marked "Required Actions" below. When connecting to the controller, a switch indicates which of the "OPTIONAL Actions" it supports. OpenFlow enabled switches, routers, and access points may also support the NORMAL action*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of supported actions announcement
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST message. Parse the OFPT_FEATURES_REPLY and verify correct announcement of the supported actions.
Results	Pass or Fail
Remarks	

Test case 70.30: Forward: ALL

Test Number	70.30
Test Title	Data plane / Actions / Forward:ALL
Test Purpose	Verify implementation of the Forward: ALL function
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 3. *ALL: Send the packet out all interfaces, not including the incoming interface*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Action FORWARD:ALL
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action Forward:ALL .Send matching packet to ingress port. Verify all (or a meaningful subset of) ports receive the packet. Verify the ingress port does not receive the packet.
Results	Pass or Fail

Remarks	

Test case 70.40: Forward:CONTROLLER

Test Number	70.40
Test Title	Data plane / Actions / Forward:CONTROLLER
Test Purpose	Verify implementation of the Forward: CONTROLLER function
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 3. *CONTROLLER: Encapsulate and send the packet to the controller*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of action Forward:Controller
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action Forward:Controller. Send matching packet. Verify Controller receives OFPT_PACKET_IN message.
Results	Pass or Fail
Remarks	

Test case 70.50: Forward:Local

Test Number	70.50
Test Title	Data plane / Actions / Forward:Local
Test Purpose	Verify implementation of the Forward:LOCAL function
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 3. *LOCAL: Send the packet to the switch's local networking stack*
Profile Status	OPTIONAL
Requirements	Correct implementation of action Forward:LOCAL
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action Forward:LOCAL. Send matching packet. Send OFPST_TABLE request. Verify matched_count increases accordingly.
Results	Pass or Fail or Not Tested
Remarks	The behavior of the IP Stack is not defined enough to currently check it directly with a testcase.

Test case 70.60: Forward:TABLE

Test Number	70.60
Test Title	Data plane / Actions / Forward:TABLE
Test Purpose	Verify implementation of the Forward:TABLE function
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 3. *TABLE: Perform actions in flow table. Only for packet-out messages*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of action Forward:Table
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create an

	OFPT_PACKET_OUT message with action OFPP_TABLE. Create matching flow with an output action to an egress port. Send packet_out message. Verify packet hits flow in table and gets output at the egress port.
Results	Pass or Fail
Remarks	

Test case 70.70: Forward:INPORT

Test Number	70.70
Test Title	Data plane / Actions / Forward:INPORT
Test Purpose	Verify implementation of the Forward: INPORT function
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 3. *INPORT: Send the packet out the input port*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of action Forward:INPORT
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPP_IN_PORT. Send matching packet on ingress port. Verify packet is output on the ingress port.
Results	Pass or Fail
Remarks	

Test case 70.80: Forward:NORMAL

Test Number	70.80
Test Title	Data plane / Actions / Forward: NORMAL
Test Purpose	Verify implementation of the Forward: NORMAL function
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 6. *NORMAL: Process the packet using the traditional forwarding path supported by the switch (i.e. traditional L2, VLAN, and L3 processing).The switch may check the VLAN field to determine whether or not to forward the packet along the normal processing route. If the switch cannot forward entries for the OpenFlow-specific VLAN back to the normal processing route, it must indicate that it does not support this action.*
Profile Status	OPTIONAL
Requirements	Correct implementation of action Forward: NORMAL
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPP_NORMAL. Send matching packet. Send OFPST_TABLE request. Verify matched_count increases accordingly
Results	Pass or Fail or Not Tested
Remarks	Normal behavior is not specified, so we cannot check the behavior directly.

Test case 70.90: Forward:FLOOD

Test Number	70.90
Test Title	Data plane / Actions / Forward:FLOOD
Test Purpose	Verify implementation of the Forward:FLOOD function
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 3. *Flood the packet along the minimum spanning tree, not including the incoming interface.*
Profile Status	OPTIONAL
Requirements	Correct implementation of action Forward:FLOOD
Topology	Control-plane connection between DUT and reference controller. At least three data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Configure one data plane egress port as part of the flood group. Configure a second data plane egress port that is not a member of the flood group. Create a flow with action OFPP_FLOOD. Send matching packet to ingress port. Verify packet is output on the flood group member port, but not the non-member port or the ingress port.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.100: Forward:MULTIPLEPORTS

Test Number	70.100
Test Title	Data plane / Actions / Forward:MULTIPLEPORTS
Test Purpose	Verify implementation of the Forward:MULTIPLEPORTS function
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 3. *The controller will only ask the switch to send to multiple physical ports simultaneously if the switch indicates it supports this behavior in the initial handshake (see section 5.3.1).*
Profile Status	OPTIONAL
Requirements	Correct implementation of action Forward:MULTIPLEPORTS
Topology	Control-plane connection between DUT and reference controller. At least four data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPAT_OUTPUT to multiple egress ports. Leave at least one egress port out of the action list. Send matching packet to ingress port. Verify packet is output on all egress ports of the action list but not the ports left out of the action list.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.110: Forward:ENQUEUE

Test Number	70.110
Test Title	Data plane / Actions / Forward:ENQUEUE
Test Purpose	Verify implementation of the Forward: ENQUEUE
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 3. *OPTIONAL Action: Enqueue. The enqueue action forwards a packet through a queue attached to a port. Forwarding behavior is dictated by the configuration of the queue*

	and is used to provide basic Quality-of-Service (QoS) support (see section 5.2.2).
Profile Status	OPTIONAL
Requirements	Correct implementation of action Forward:ENQUEUE
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Configure a queue and map it to a port. Create a flow with action OFPAT_ENQUEUE and mapped to the queue. Send matching packet. Verify packet gets forwarded through the queue specified in the flow.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.120: Add VLAN ID

Test Number	70.120
Test Title	Data plane / Modify-Field Actions / Add VLAN ID
Test Purpose	Verify implementation of the Set VLAN ID action
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 6. *If no VLAN is present, a new header is added with the specified VLAN ID and priority of zero*
Profile Status	OPTIONAL
Requirements	Correct implementation of Action:Set VLAN ID
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPAT_SET_VLAN_VID and output to an egress port. Send matching untagged packet to the ingress port. Verify packet gets output to the egress port with correct VLAN Tag added.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.130: Set VLAN ID

Test Number	70.130
Test Title	Data plane / Modify-Field Actions / Set VLAN ID
Test Purpose	Verify implementation of the Set VLAN ID action
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 6. *If a VLAN header already exists, the VLAN ID is replaced with the specified value*
Profile Status	OPTIONAL
Requirements	Correct implementation of Action: Set VLAN ID
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPAT_SET_VLAN_VID and output to an egress port. Send matching packet tagged with random VLAN ID to ingress port. Verify packet gets output with correct VLAN Tag as specified in the flow.
Results	Pass or Fail or Not Tested

Remarks	

Test case 70.140: Add VLAN priority

Test Number	70.140
Test Title	Data plane / Modify-Field Actions / Add VLAN priority
Test Purpose	Verify implementation of the Set VLAN priority action
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 6. *If no VLAN is present, a new header is added with the specified priority and a VLAN ID of zero*
Profile Status	OPTIONAL
Requirements	Correct implementation of Action: Set VLAN priority
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPAT_SET_VLAN_PCP and output to an egress port. Send matching tagged packet to the ingress port. Verify packet is output to the egress port with the correct VLAN Tag and priority specified in the flow.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.150: Set VLAN priority

Test Number	70.150
Test Title	Data plane / Modify-Field Actions / Set VLAN priority
Test Purpose	Verify implementation of the Set VLAN priority action
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 6. *If a VLAN header already exists, the priority field is replaced with the specified value*
Profile Status	OPTIONAL
Requirements	Correct implementation of action Set VLAN priority
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPAT_SET_VLAN_PCP and output to an egress port. Send matching packet tagged with random VLAN priority to ingress port. Verify packet gets output with the correct VLAN priority set.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.160: Strip VLAN header

Test Number	70.160
Test Title	Data plane / Modify-Field Actions / Strip VLAN header
Test Purpose	Verify implementation of the Strip VLAN header action
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 6. *Strip VLAN header if it exists*
Profile Status	OPTIONAL

Requirements	Correct implementation of action Strip VLAN header
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPAT_STRIP_VLAN and output to an egress port. Send matching VLAN tagged packet to ingress port. Verify packet gets output to the egress port without a VLAN tag.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.170: Modify Ethernet source MAC address

Test Number	70.170
Test Title	Data plane / Modify-Field Actions / Modify Ethernet source MAC address
Test Purpose	Verify implementation of the Modify Ethernet source MAC address action
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 6.
Profile Status	OPTIONAL
Requirements	Correct implementation of action Modify Ethernet source MAC address
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPAT_SET_DL_SRC address and output to an egress port. Send matching packet to ingress port. Verify packet gets output to the egress port with the correct Ethernet source MAC address as specified in the flow.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.180: Modify Ethernet destination MAC address

Test Number	70.180
Test Title	Data plane / Modify-Field Actions / Modify Ethernet destination MAC address
Test Purpose	Verify implementation of the Modify Ethernet destination MAC address action
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 6.
Profile Status	OPTIONAL
Requirements	Correct implementation of action Modify Ethernet destination MAC address
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with OFPAT_SET_DL_DST and output to an egress port. Send matching packet to ingress port. Verify packet gets output to the egress

	port with the correct Ethernet destination MAC address as specified in the flow.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.190: Modify IPv4 source address

Test Number	70.190
Test Title	Data plane / Modify-Field Actions / Modify IPv4 source address
Test Purpose	Verify implementation of the Modify IPv4 source address action
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 6. *Replace the existing IP source address with new value and update the IP checksum (and TCP/UDP checksum if applicable). This action is only applicable to IPv4 packets*
Profile Status	OPTIONAL
Requirements	Correct implementation of action Modify IPv4 source address
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPAT_SET_NW_SRC and output to an egress port. Send matching packet to ingress port. Verify packet gets output to the egress port with the correct IPv4 source address as specified in the flow.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.200: Modify IPv4 destination address

Test Number	70.200
Test Title	Data plane / Modify-Field Actions / Modify IPv4 destination address
Test Purpose	Verify implementation of the Modify IPv4 destination address action
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 6. *Replace the existing IP destination address with new value and update the IP checksum (and TCP/UDP checksum if applicable). This action is only applicable to IPv4 packets*
Profile Status	OPTIONAL
Requirements	Correct implementation of action Modify IPv4 destination address
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPAT_SET_NW_DST and output to an egress port. Send matching packet to ingress port. Verify packet gets output to egress port with correct IPv4 destination address as specified in the flow.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.210: Modify IPv4 ToS bits

Test Number	70.210
Test Title	Data plane / Modify-Field Actions / Modify IPv4 ToS bits
Test Purpose	Verify implementation of the Modify IPv4 ToS bits action

Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 6. *Replace the existing IP ToS field. This action is only applied to IPv4 packets.*
Profile Status	OPTIONAL
Requirements	Correct implementation of action Modify IPv4 ToS bits
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPAT_SET_NW_TOS bits and output egress port. Send matching packet to ingress port. Verify packet gets output to the egress port with correct IPv4 ToS bits as specified in the flow.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.220: Modify TCP/UDP source port

Test Number	70.220
Test Title	Data plane / Modify-Field Actions / Modify TCP/UDP source port
Test Purpose	Verify implementation of the Modify TCP/UDP source port action
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 6. *Replace the existing TCP/UDP source port with new value and update the TCP/UDP checksum. This action is only applicable to TCP and UDP packets*
Profile Status	OPTIONAL
Requirements	Correct implementation of action Modify TCP/UDP source port
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPAT_SET_TP_SRC and output to an egress port. Send matching packet to ingress port. Verify packet gets output to egress port with correct TCP/UDP source port as specified in the flow.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.230: Modify TCP/UDP destination port

Test Number	70.230
Test Title	Data plane / Modify-Field Actions / Modify TCP/UDP destination port
Test Purpose	Verify implementation of the Modify TCP/UDP destination port action
Specification Reference	OpenFlow Switch Specification 1.0.0, 3.3 Actions, p. 6. *Replace the existing TCP/UDP destination port with new value and update the TCP/UDP checksum. This action is only applicable to TCP and UDP packets*
Profile Status	OPTIONAL
Requirements	Correct implementation of action Modify TCP/UDP destination port
Topology	Control-plane connection between DUT and reference controller. At least two data plane connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPAT_SET_TP_DST and output to an egress port. Send matching packet to ingress port. Verify packet gets output to the egress port with correct TCP/UDP destination port as specified in the flow.

Results	Pass or Fail or Not Tested
Remarks	

Test case 70.240: Ordering not possible

Test Number	70.240
Test Title	Data plane / Actions / Ordering not possible
Test Purpose	Verify implementation of action lists
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification messages, p. 13. *If a switch cannot process the action list for any flow mod message in the order specified, it MUST immediately return an OFPET_FLOW_MOD_FAILED : OFPFMFC_UNSUPPORTED error and reject the flow*
Profile Status	OPTIONAL
Requirements	Correct implementation of action ordering
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with action OFPAT_OUTPUT followed by action OFPAT_SET_TP_SRC. Verify the correct error is returned.
Results	Pass or Fail or Not Tested
Remarks	

Test case 70.250: Sequential execution

Test Number	70.250
Test Title	Data plane / Actions / Sequential execution
Test Purpose	Verify correct execution of sequential actions
Specification Reference	OpenFlow Switch Specification 1.0.0, 4.6 Flow Table Modification messages, p. 13. *Action lists for inserted flow entries MUST be processed in the order specified. However, there is no packet output ordering guaranteed within a port. For example, an action list may result in two packets sent to two different VLANs on a single port. These two packets may be arbitrarily re-ordered, but the packet bodies must match those generated from a sequential execution of the actions*
Profile Status	OPTIONAL
Requirements	Correct implementation of sequential execution
Topology	Control-plane connection between DUT and reference controller. At least two data plane ports connections to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create a flow with two OFPAT_SET_VLAN_VID actions (set to two different VLAN IDs) followed by two OFPAT_OUTPUT actions (set to two different egress ports). Send matching packet to ingress port. Verify the packets sent out both egress ports contain the second VLAN ID.
Results	Pass or Fail or Not Tested
Remarks	

Test Suite 80: Messages

Test suite 80 checks OpenFlow protocol messages and their correct implementation. In contrast to the basic checks, return values are checked for correctness, and configurations for functional implementation.

Special cases:
Fragmentation related test cases 80.270, 80.280, 80.290 & 80.300 are OPTIONAL

Profiles:
All profiles MUST pass all tests except fragmentation related tests 80.270, 80.280, 80.290 & 80.300.

Test case 80.10: OFPT_HELLO without body

Test Number	80.10
Test Title	Protocol Messages / Symmetric messages/ OFPT_HELLO without body
Test Purpose	Verify OFPT_HELLO without body is accepted by the device
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.5.1 Hello , p. 41. *The OFPT_HELLO message has no body; that is, it consists only of an OpenFlow header. Implementations must be prepared to receive a hello message that includes a body, ignoring its contents, to allow for later extensions*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Hello messages
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Send OFPT_HELLO message with empty body. Verify device accepts the message without generating an error.
Results	Pass or Fail Pass or Fail
Remarks	

Test case 80.20: OFPT_HELLO with body

Test Number	80.20
Test Title	Protocol Messages / Symmetric messages/ OFPT_HELLO with body
Test Purpose	Verify OFPT_HELLO with body is accepted by the device
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.5.1 Hello, p. 41. *The OFPT_HELLO message has no body; that is, it consists only of an OpenFlow header. Implementations must be prepared to receive a hello message that includes a body, ignoring its contents, to allow for later extensions.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Hello messages
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Send Hello message with body. Verify device accepts the message without generating an error.
Results	Pass or Fail
Remarks	

Test case 80.30: OFPT_ERROR

Test Number	80.30
Test Title	Protocol Messages / Symmetric messages/ OFPT_ERROR
Test Purpose	Verify basic error message type is implemented
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4, Error Message, p. 38.
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of error messages
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger a basic OFPT_ERROR message One way to trigger this error is to send an incompatible version in the OFPT_HELLO and verify OFPET_HELLO_FAILED error type is returned.
Results	Pass or Fail
Remarks	

Test case 80.40: OFPT_ECHO_REQUEST / Reply without body

Test Number	80.40
Test Title	Protocol Messages / Symmetric messages/ OFPT_ECHO_REQUEST / Reply without body
Test Purpose	Verify OFPT_ECHO_REQUEST / Reply
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.5.1 Echo, p. 41. *An Echo Request message consists of an OpenFlow header plus an arbitrary length data field. The data field might be a message timestamp to check latency, various lengths to measure bandwidth, or zero-size to verify liveness between the switch and controller*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Echo Request / Reply messages
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate OFPT_ECHO_REQUEST without body, and verify OFPT_ECHO_REPLY is returned.
Results	Pass or Fail
Remarks	

Test case 80.50: OFPT_ECHO_REQUEST / Reply with body

Test Number	80.50
Test Title	Protocol Messages / Symmetric messages/ OFPT_ECHO_REQUEST / Reply with body
Test Purpose	Verify OFPT_ECHO_REQUEST / Reply
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.5.2 Echo, p. 41. *An Echo Request message consists of an OpenFlow header plus an arbitrary length data field. The data field might be a message timestamp to check latency, various lengths to measure bandwidth, or zero-size to verify liveness between the switch and controller*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Echo Request / Reply messages

Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate OFPT_ECHO_REQUEST with arbitrary data field and verify the Reply has the identical data field.
Results	Pass or Fail
Remarks	

Test case 80.60: Features Request-Reply

Test Number	80.60
Test Title	Protocol Messages / Switch configuration messages/ Features Request-Reply
Test Purpose	Verify OFPT_FEATURES_REQUEST / Reply dialogue
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25. *Upon TLS session establishment, the controller sends an OFPT_FEATURES_REQUEST message. This message does not contain a body beyond the OpenFlow header. The switch responds with an OFPT_FEATURES_REPLY message*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Features Request and Reply messages
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID.
Results	Pass or Fail
Remarks	

Test case 80.70: Features Reply

Test Number	80.70
Test Title	Protocol Messages / Switch configuration messages/ Features Reply
Test Purpose	Verify OFPT_FEATURES_REPLY contains complete feature information
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25. *Upon TLS session establishment, the controller sends an OFPT_FEATURES_REQUEST message. This message does not contain a body beyond the OpenFlow header. The switch responds with an OFPT_FEATURES_REPLY message*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Features Reply message
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Verify reply has all the expected switch features information.
Results	Pass or Fail

Remarks	The returned values will be checked in the following test cases.

Test case 80.80: uint64_t datapath_id

Test Number	80.80
Test Title	Protocol Messages / Switch configuration messages/ uint64_t datapath_id
Test Purpose	Verify OFPT_FEATURES_REPLY contains valid datapath_id field
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25. *The datapath_id field uniquely identifies a datapath. The lower 48 bits are intended for the switch MAC address, while the top 16 bits are up to the implementer. An example use of the top 16 bits would be a VLAN ID to distinguish multiple virtual switch instances on a single physical switch. This eld should be treated as an opaque bit string by controllers*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of datapath_id field
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Verify reply has a valid datapath_id field
Results	Pass or Fail
Remarks	

Test case 80.90: uint32_t n_buffers

Test Number	80.90
Test Title	Protocol Messages / Switch configuration messages/ uint32_t n_buffers
Test Purpose	Verify OFPT_FEATURES_REPLY contains valid datapath_id field
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25. *Max packets buffered at once*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of uint32_t n_buffers
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Verify reply contains a valid uint32_t n_buffers value.
Results	Pass or Fail
Remarks	If possible verify buffer value against the information provided by the vendor.

Test case 80.100: uint8_t n_tables

Test Number	80.100
Test Title	Protocol Messages / Switch configuration messages/ uint8_t n_tables
Test Purpose	Verify OFPT_FEATURES_REPLY contains valid uint8_t n_tables field

Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25. *The n_tables field describes the number of tables supported by the switch,each of which can have a diferent set of supported wildcard bits and number of entries. When the controller and switch rst communicate, the controller will find out how many tables the switch supports from the Features Reply. If it wishes to understand the size, types, and order in which tables are consulted, the controller sends a OFPST_TABLE stats request. A switch must return these tables in the order the packets traverse the tables, with all exact-match tables listed before all tables with wildcards*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of uint8_t n_tables
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Verify reply contains a valid uint8_t n_tables value
Results	Pass or Fail
Remarks	

Test case 80.110: OFPC_FLOW_STATS

Test Number	80.110
Test Title	Protocol Messages / Switch configuration messages / OFPC_FLOW_STATS
Test Purpose	Verify OFPT_FEATURES_REPLY for Flow statistics support
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25.
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPC_FLOW_STATS
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Check whether the OFPC_FLOW_STATS bit is set. If yes, Flow statistics are supported.
Results	Pass or Fail
Remarks	

Test case 80.120: OFPC_TABLE_STATS

Test Number	80.120
Test Title	Protocol Messages / Switch configuration messages / OFPC_TABLE_STATS
Test Purpose	Verify OFPT_FEATURES_REPLY for Table statistics support
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25.
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPC_TABLE_STATS
Topology	Control-plane connection between DUT and reference controller.

Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Check whether the OFPC_TABLE_STATS bit is set. If yes, Table statistics are supported.
Results	Pass or Fail
Remarks	

Test case 80.130: OFPC_PORT_STATS

Test Number	80.130
Test Title	Protocol Messages / Switch configuration messages / OFPC_PORT_STATS
Test Purpose	Verify OFPT_FEATURES_REPLY for Port statistics support
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25.
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPC_PORT_STATS
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Check whether the OFPC_PORT_STATS bit is set. If yes, Port statistics are supported.
Results	Pass or Fail
Remarks	

Test case 80.140: OFPC_STP

Test Number	80.140
Test Title	Protocol Messages / Switch configuration messages / OFPC_STP
Test Purpose	Verify OFPT_FEATURES_REPLY for 802.1d spanning tree support
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25.
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPC_STP
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Check whether the OFPC_STP bit is set. If yes, 802.1d spanning tree is supported.
Results	Pass or Fail
Remarks	

Test case 80.150: OFPC_RESERVED

Test Number	80.150
Test Title	Protocol Messages / Switch configuration messages /

	OFPC_RESERVED
Test Purpose	Verify OFPT_FEATURES_REPLY for OFPC_RESERVED returns 0
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25. *OFPC_RESERVED = 1 <4, /* Reserved, must be zero.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPC_RESERVED
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Verify OFPC_RESERVED is 0.
Results	Pass or Fail
Remarks	

Test case 80.160: OFPC_IP_REASM

Test Number	80.160
Test Title	Protocol Messages / Switch configuration messages / OFPC_IP_REASM
Test Purpose	Verify OFPT_FEATURES_REPLY for IP packet reassembly
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25.
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPC_IP_REASM
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Check whether the OFPC_IP_REASM bit is set. If yes, Switch can reassemble IP fragments.
Results	Pass or Fail
Remarks	

Test case 80.170: OFPC_ARP_MATCH_IP

Test Number	80.170
Test Title	Protocol Messages / Switch configuration messages / OFPC_ARP_MATCH_IP
Test Purpose	Verify OFPT_FEATURES_REPLY for Match IP addresses in ARP packets.
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25.
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPC_ARP_MATCH_IP
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the

	OFPT_FEATURES_REPLY is received from the switch with the same XID. Check whether the OFPC_ARP_MATCH_IP bit is set. If yes, the Switch supports matching IP addresses in ARP packets.
Results	Pass or Fail
Remarks	

Test case 80.180: uint32_t actions

Test Number	80.180
Test Title	Protocol Messages / Switch configuration messages / uint32_t actions
Test Purpose	Verify OFPT_FEATURES_REPLY for Bitmap of supported ofp_action_types
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25.
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of uint32_t actions
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Check the bitmap of supported ofp_action_types, and create a list of supported actions. The switch MUST support the announced actions.
Results	Pass or Fail
Remarks	

Test case 80.190: struct ofp_phy_port ports[0]

Test Number	80.190
Test Title	Protocol Messages / Switch configuration messages / struct ofp_phy_port ports[0]
Test Purpose	Verify OFPT_FEATURES_REPLY for list of available ports
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3 Controller to switch messages, p. 25.
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of ofp_phy_port ports[0]
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Check the ofp_phy_port ports[0] and compare the returned list of available ports with the switch configuration to verify consistency.
Results	Pass or Fail
Remarks	

Test case 80.200: Get Config Request-Reply

Test Number	80.200
Test Title	Protocol Messages / Switch configuration messages / Get Config Request-Reply

Test Purpose	Verify implementation of Get Config Request-Reply
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3.2 Switch configuration messages, p. 26. *The controller is able to set and query configuration parameters in the switch with the OFPT_SET_CONFIG and OFPT_GET_CONFIG_REQUEST messages, respectively. The switch responds to a configuration request with an OFPT_GET_CONFIG_REPLY message; it does not reply to a request to set the configuration. There is no body for OFPT_GET_CONFIG_REQUEST beyond the OpenFlow header*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Get Config Request-Reply
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_GET_CONFIG_REQUEST, and verify OFPT_GET_CONFIG_REPLY is received. Verify reply has the expected fields (OFPC_FRAG_NORMAL, OFPC_FRAG_DROP, OFPC_FRAG_REASM, OFPC_FRAG_MASK).
Results	Pass or Fail
Remarks	Values of the fields will be checked in the following test cases

Test case 80.210: OFPC_FRAG_NORMAL

Test Number	80.210
Test Title	Protocol Messages / Switch configuration messages / OFPC_FRAG_NORMAL
Test Purpose	Check OFPT_GET_CONFIG_REPLY value for No special handling for fragments
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3.2 Switch configuration messages, p. 26
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPC_FRAG_NORMAL
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Verify OFPC_FRAG_NORMAL flag value and verify handling of fragments is consistent with the returned configuration.
Results	Pass or Fail
Remarks	

Test case 80.220: OFPC_FRAG_DROP

Test Number	80.220
Test Title	Protocol Messages / Switch configuration messages / OFPC_FRAG_DROP
Test Purpose	Check OFPT_GET_CONFIG_REPLY value for Drop fragments
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3.2 Switch configuration messages, p. 26
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPC_FRAG_DROP

Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Verify OFPC_FRAG_DROP flag value and verify handling of fragments is consistent with the returned configuration.
Results	Pass or Fail
Remarks	

Test case 80.230: OFPC_FRAG_REASM

Test Number	80.230
Test Title	Protocol Messages / Switch configuration messages / OFPC_FRAG_REASM
Test Purpose	Check OFPT_GET_CONFIG_REPLY value for Reassemble fragments
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3.2 Switch configuration messages, p. 26
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPC_FRAG_REASM
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Verify OFPC_FRAG_REASM flag value and verify handling of fragments is consistent with the returned configuration.
Results	Pass or Fail
Remarks	

Test case 80.240: OFPC_FRAG_MASK

Test Number	80.240
Test Title	Protocol Messages / Switch configuration messages / OFPC_FRAG_MASK
Test Purpose	Check OFPT_GET_CONFIG_REPLY value for OFPC_FRAG_MASK
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3.2 Switch configuration messages, p. 26
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPC_FRAG_MASK
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Verify OFPC_FRAG_MASK flag value and verify handling of fragments is consistent with the returned configuration.
Results	Pass or Fail
Remarks	

Test case 80.250: uint16_t miss_send_len

Test Number	80.250
Test Title	Protocol Messages / Switch configuration messages / uint16_t miss_send_len
Test Purpose	Check OFPT_GET_CONFIG_REPLY value for miss_send_len
Specification Reference	OpenFlow Switch Specification 11.0.0, 5.3.2 Switch configuration messages, p. 26. *The miss_send_len field defines the number of bytes of each packet sent to the controller as a result of both Flow table misses and Flow table hits with the controller as the destination. If this field equals 0, the switch must send a zero-size packet_in message.* 5.4.1 Packet-In Message p.36. *If the packet is sent because of a flow table miss, then at least miss_send_len bytes from the OFPT_SET_CONFIG message are sent. The default miss_send_len is 128 bytes. If the packet is not buffered,the entire packet is included in the data portion, and the buffer_id is -1*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of miss_send_len
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_FEATURES_REQUEST. Verify the OFPT_FEATURES_REPLY is received from the switch with the same XID. Document miss_send_len value. Send a packet .Verify data length in ofp_packet_in message is in accordance with miss_send_len value. If miss_send_len is 0 bytes, data length in ofp_packet_in is 0 bytes. If miss_send_len is x bytes (x>0) , then data length in ofp_packet_in is >= x bytes.
Results	Pass or Fail
Remarks	If the packet is not buffered the entire packet is included in the data portion, and the buffer_id is -1

Test case 80.260: OFPT_SET_CONFIG – miss_send_len

Test Number	80.260
Test Title	Protocol Messages / Switch configuration messages / OFPT_SET_CONFIG – miss_send_len
Test Purpose	Verify implementation of OFPT_SET_CONFIG – miss_send_len
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3.2 Switch configuration messages, p. 26. *The miss_send_len field defines the number of bytes of each packet sent to the controller as a result of both Flow table misses and Flow table hits with the controller as the destination. If this field equals 0, the switch must send a zero-size packet_in message.* 5.4.1 Packet-In Message p.36. *If the packet is sent because of a flow table miss, then at least miss_send_len bytes from the OFPT_SET_CONFIG message are sent. The default miss_send_len is 128 bytes. If the packet is not buffered,the entire packet is included in the data portion, and the buffer_id is -1*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of miss_send_len

Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_GET_CONFIG_REQUEST and verify reply is received. Verify the value in miss_send_len field (defines number of bytes of each packet sent to the controller). Generate OFPT_SET_CONFIG request. Overwrite the miss_send_len field. Again send an OFPT_GET_CONFIG_REQUEST and verify the change has taken effect.
Results	Pass or Fail
Remarks	

Test case 80.270: OFPT_SET_CONFIG – OFPC_FRAG_NORMAL = 0

Test Number	80.270
Test Title	Protocol Messages / Switch configuration messages / OFPT_SET_CONFIG – OFPC_FRAG_NORMAL
Test Purpose	Verify implementation of OFPT_SET_CONFIG – OFPC_FRAG_NORMAL
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3.2 Switch Configuration, p. 26. The OFPC_FRAG_* flags indicate whether IP fragments should be treated normally, dropped, or reassembled. “Normal" handling of fragments means that an attempt should be made to pass the fragments through the OpenFlow tables. If any field is not present (e.g., the TCP/UDP ports didn't fit), then the packet should not match any entry that has that field set.
Profile Status	OPTIONAL
Requirements	Correct implementation of OFPC_FRAG_NORMAL
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate a OFPT_GET_CONFIG_REQUEST and verify reply is received. Generate OFPT_SET_CONFIG_REQUEST. Set OFPC_FRAG_NORMAL = 0. Send an OFPT_GET_CONFIG request and verify the value is 0.
Results	Pass or Fail
Remarks	

Test case 80.280: OFPT_SET_CONFIG – OFPC_FRAG_DROP

Test Number	80.280
Test Title	Protocol Messages / Switch configuration messages / OFPT_SET_CONFIG – OFPC_FRAG_DROP
Test Purpose	Verify implementation of OFPT_SET_CONFIG – OFPC_FRAG_DROP
Specification Reference	OpenFlow 1.0.0, 5.3.2 Switch configuration messages, p. 26.
Profile Status	OPTIONAL
Requirements	Correct implementation of OFPC_FRAG_DROP
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an

	OFPT_GET_CONFIG_REQUEST and verify reply is received. Generate OFPT_SET_CONFIG request. Set OFPC_FRAG_DROP = 1 Send an OFPT_GET_CONFIG request and verify the change has taken effect.
Results	Pass or Fail
Remarks	Value changes from 0 (set in test case 80.270) to 1.

Test case 80.290: OFPT_SET_CONFIG – OFPC_FRAG_REASM

Test Number	80.290
Test Title	Protocol Messages / Switch configuration messages / OFPT_SET_CONFIG – OFPC_FRAG_REASM
Test Purpose	Verify implementation of OFPT_SET_CONFIG – OFPC_FRAG_REASM
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3.2 Switch configuration messages, p. 26
Profile Status	OPTIONAL
Requirements	Correct implementation of OFPC_FRAG_REASM
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_GET_CONFIG_REQUEST and verify OFPT_GET_CONFIG_REPLY is received. Generate OFPT_SET_CONFIG_REQUEST and set OFPC_FRAG_REASM = 2. Send the OFPT_GET_CONFIG_REQUEST and verify the value is 2.
Results	Pass or Fail or Not Tested
Remarks	

Test case 80.300: OFPT_SET_CONFIG – OFPC_FRAG_MASK = 3

Test Number	80.300
Test Title	Protocol Messages / Switch configuration messages / OFPT_SET_CONFIG – OFPC_FRAG_MASK
Test Purpose	Verify implementation of OFPT_SET_CONFIG – OFPC_FRAG_MASK = 3
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3.2 Switch configuration messages, p. 26
Profile Status	OPTIONAL
Requirements	Correct implementation of OFPC_FRAG_MASK
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Generate an OFPT_GET_CONFIG_REQUEST and verify reply is received. Generate OFPT_SET_CONFIG request and set OFPC_FRAG_MASK = 3. Send the OFPT_GET_CONFIG_REQUEST and verify the value is 3.
Results	Pass or Fail or Not Tested
Remarks	

Test Suite 90: Async Messages

Test suite 90 checks async OpenFlow protocol messages and their correct implementation. In contrast to the basic checks, return values are checked for correctness, and configurations for functional implementation.

Special cases:

90.120, 90.130, 90.140 are optional as these are port states for STP.
90.170 the get Queue config command is OPTIONAL.

Profiles:
All profiles have to pass all tests except 90.170.

Test case 90.10: OFPR_NO_MATCH uint8_t reason

Test Number	90.10
Test Title	Protocol Messages / Asynchronous messages - OFPT_PACKET_IN / OFPR_NO_MATCH uint8_t reason
Test Purpose	Verify packet_in specifies the right reason (no match or send to controller)
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4 Asynchronous Messages, p. 36.
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPR_NO_MATCH
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Send a packet to the data plane and trigger a packet_in. Verify Reason field is OFPR_NO_MATCH.
Results	Pass or Fail
Remarks	

Test case 90.20: OFPR_NO_MATCH unit8_t data[0] buffered

Test Number	90.20
Test Title	Protocol Messages / Asynchronous messages - OFPT_PACKET_IN / OFPR_NO_MATCH unit8_t data[0] buffered
Test Purpose	Verify packet_in OFPR_NO _MATCH implements buffer handling correct
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4 Asynchronous Messages, p. 36. *The buffer_id is an opaque value used by the datapath to identify a buffered packet. When a packet is buffered, some number of bytes from the message will be included in the data portion of the message. If the packet is sent because of a "send to controller" action, then max_len bytes from the action_output of the flow setup request are sent. If the packet is sent because of a flow table miss, then at least miss_send_len bytes from the OFPT_SET_CONFIG message are sent. The default miss_send_len is 128 bytes. If the packet is not buffered, the entire packet is included in the data portion, and the buffer_id is -1*
Profile Status	MANDATORY for ALL Profiles

Requirements	Correct implementation of OFPR_NO_MATCH unit8_t data[0]
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Verify the miss_send_len value is non-zero. Send a packet to the data plane and trigger a packet_in. Verify Reason field is OFPR_NO_MATCH. For buffered packets, verify the number of bytes transferred in the packet_in is in accordance to the miss_send_len configuration.
Results	Pass or Fail
Remarks	

Test case 90.30: OFPR_NO_MATCH unit8_t data[0] unbuffered

Test Number	90.30
Test Title	Protocol Messages / Asynchronous messages - OFPT_PACKET_IN / OFPR_NO_MATCH unit8_t data[0] unbuffered
Test Purpose	Verify packet_in OFPR_NO _MATCH implements buffer handling correct
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4 Asynchronous Messages, p. 36. The buffer_id is an opaque value used by the datapath to identify a buffered packet. When a packet is buffered, some number of bytes from the message will be included in the data portion of the message. If the packet is sent because of a "send to controller" action, then max_len bytes from the action_output of the flow setup request are sent. If the packet is sent because of a flow table miss, then at least miss_send_len bytes from the OFPT_SET_CONFIG message are sent. The default miss_send_len is 128 bytes. If the packet is not buffered, the entire packet is included in the data portion, and the buffer_id is -1. Switches that implement buffering are expected to expose, through documentation, both the amount of available buffering, and the length of time before buffers may be reused.
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPR_NO_MATCH unit8_t data[0]
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Set miss_send_len value to zero. Send a packet to the data plane and trigger a packet_in. Verify Reason field is OFPR_NO_MATCH. If the packet is buffered, verify no packet data is included in the packet_in. If it is possible to create unbuffered packet_ins, verify unbuffered packets are included completely in the packet_in, and the buffer-id is set to -1.
Results	Pass or Fail
Remarks	

Test case 90.40: OFPR_NO_MATCH uint16_t in_port

Test Number	90.40
Test Title	Protocol Messages / Asynchronous messages - OFPT_PACKET_IN / OFPR_NO_MATCH uint16_t in_port
Test Purpose	Verify packet_in OFPR_NO _MATCH reports correct inport
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4 Asynchronous Messages, p. 36. *uint16_t in_port; /* Port on which frame was received. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPR_NO_MATCH uint16_t in_port
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Send a packet to the data plane and trigger a packet_in. Verify Reason field is OFPR_NO_MATCH. Verify the correct in_port is reported
Results	Pass or Fail
Remarks	

Test case 90.50: OFPR_NO_MATCH int16_t total_len

Test Number	90.50
Test Title	Protocol Messages / Asynchronous messages - OFPT_PACKET_IN / OFPR_NO_MATCH int16_t total_len
Test Purpose	Verify packet_in OFPR_NO _MATCH reports correct value for full length of frame
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4 Asynchronous Messages, p. 36. *uint16_t total_len; /* Full length of frame. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPR_NO_MATCH uint16_t total_len
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Send packet to data plane and trigger packet_in. Verify Reason field is OFPR_NO_MATCH. Verify the correct total_len is reported
Results	Pass or Fail
Remarks	

Test case 90.60: OFPR_Action uint8_t reason

Test Number	90.60
Test Title	Protocol Messages / Asynchronous messages - OFPT_PACKET_IN / OFPR_Action uint8_t reason
Test Purpose	Verify packet_in specifies the correct reason for Action explicitly output to controller
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4 Asynchronous Messages, p. 36. *uint8_t reason; /* Reason packet is being sent (one of OFPR_*) */*

Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPR_Action uint8_t reason
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create flow with ACTION: output to controller. Send matching packet to data plane and trigger packet_in. Verify Reason field is OFPR_ACTION.
Results	Pass or Fail
Remarks	

Test case 90.70: OFPR_ACTION unit8_t data[0] buffered

Test Number	90.70
Test Title	Protocol Messages / Asynchronous messages - OFPT_PACKET_IN / OFPR_ACTION unit8_t data[0] buffered
Test Purpose	Verify packet_in OFPR_ACTION implements buffer handling correct
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4 Asynchronous Messages, p. 36. *The buffer_id is an opaque value used by the datapath to identify a buffered packet. When a packet is buffered, some number of bytes from the message will be included in the data portion of the message. If the packet is sent because of a "send to controller" action, then max_len bytes from the action_output of the flow setup request are sent. If the packet is sent because of a flow table miss, then at least miss_send_len bytes from the OFPT_SET_CONFIG message are sent. The default miss_send_len is 128 bytes. If the packet is not buffered, the entire packet is included in the data portion, and the buffer_id is -1*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPR_ACTION unit8_t data[0]
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Verify max_len value is non-zero. Create flow with ACTION: output to controller. Send matching packet to data plane and trigger packet_in. Verify Reason field is OFPR_ACTION. Verify that for buffered packets the amount of bytes transferred in the packet_in is in accordance to the max_len configuration.
Results	Pass or Fail
Remarks	

Test case 90.80: OFPR_ACTION unit8_t data[0] unbuffered

Test Number	90.80
Test Title	Protocol Messages / Asynchronous messages - OFPT_PACKET_IN / OFPR_ACTION unit8_t data[0] unbuffered
Test Purpose	Verify packet_in OFPR_ACTION implements buffer handling correct
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4 Asynchronous Messages, p. 36. *The buffer_id is an opaque value used by the datapath to identify a buffered packet. When a packet is buffered, some number of bytes from the message will be included in the data portion of the message. If the packet is sent because of a "send to controller"*

	action, then max_len bytes from the action_output of the flow setup request are sent. If the packet is sent because of a flow table miss, then at least miss_send_len bytes from the OFPT_SET_CONFIG message are sent. The default miss_send_len is 128 bytes. If the packet is not buffered, the entire packet is included in the data portion, and the buffer_id is -1
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPR_ACTION unit8_t data[0]
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create flow with ACTION: output to controller. Send matching packet size l to data plane and trigger packet_in. Verify Reason field is OFPR_ACTION. Verify that not buffered packets are included completely in the packet_in, and the buffer-id is set to -1.
Results	Pass or Fail
Remarks	

Test case 90.90: OFPR_ACTION uint16_t in_port

Test Number	90.90
Test Title	Protocol Messages / Asynchronous messages - OFPT_PACKET_IN / OFPR_ACTION uint16_t in_port
Test Purpose	Verify packet_in OFPR_ACTION reports correct inport
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4 Asynchronous Messages, p. 36. *uint16_t in_port; /* Port on which frame was received. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPR_ACTION uint16_t in_port
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create flow with ACTION: output to controller. Send matching packet to data plane and trigger packet_in. Verify Reason field is OFPR_ACTION . Verify the correct in_port is reported
Results	Pass or Fail
Remarks	

Test case 90.100: OFPR_ACTION int16_t total_len

Test Number	90.100
Test Title	Protocol Messages / Asynchronous messages - OFPT_PACKET_IN / OFPR_ACTION int16_t total_len
Test Purpose	Verify packet_in OFPR_ACTION reports correct value for full length of frame
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4 Asynchronous Messages, p. 36. *uint16_t total_len; /* Full length of frame. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPR_ACTION uint16_t total_len

Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Create flow with ACTION: output to controller. Send matching packet to data plane and trigger packet_in. Verify Reason field is OFPR_ACTION. Verify the correct total_len is reported.
Results	Pass or Fail
Remarks	

Test case 90.110: OFPT_PORT_STATUS

Test Number	90.110
Test Title	Protocol Messages / Asynchronous messages / OFPT_PORT_STATUS
Test Purpose	Verify packet_in OFPR_ACTION reports correct value for full length of frame
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.3 Port Status messages, p. 38. *As physical ports are added, modifed, and removed from the datapath, the controller needs to be informed with the OFPT_PORT_STATUS message.*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPT_PORT_STATUS
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Verify the data plane port has link. Bring port down; verify OFPT_PORT_STATUS reason DELETE message is send to the controller. Bring port up again; verify OFPT_PORT_STATUS reason ADD is received at the controller.
Results	Pass or Fail
Remarks	

Test case 90.120: OFPT_PORT_MOD - No_Flood

Test Number	90.120
Test Title	Protocol Messages / Controller to switch message/ OFPT_PORT_MOD - No_Flood
Test Purpose	Verify Controller is able to use the OFPT_PORT_MOD - No_Flood message to change port state on the DUT
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.2.1 Port Structures, Page 17. *OFPPC_NO_FLOOD = 1 << 4, /* Do not include this port when flooding. */ OFPPFL_NO_FLOOD is set to 0 when the STP port state is Forwarding, otherwise to 1.*
Profile Status	OPTIONAL
Requirements	Correct implementation of OFPT_PORT_MOD - No_Flood
Topology	Control-plane connection between DUT and reference controller. At least one data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Check port flood state, change port state with OFPT_PORT_MOD - No_Flood message. Verify state change took place. Change back to original

	port state with OFPT_PORT_MOD - No_Flood message. Verify port state change took place.
Results	Pass or Fail
Remarks	

Test case 90.130: OFPT_PORT_MOD - No_Forward

Test Number	90.130
Test Title	Protocol Messages / Controller to switch message/ OFPT_PORT_MOD - No_Forward
Test Purpose	Verify Controller is able to use the OFPT_PORT_MOD - OFPPFL_NO_FWD message to change port state on the DUT
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.2.1 Port Structures, Page 17. *OFPPC_NO_FWD = 1 << 5, /* Drop packets forwarded to port. */* *The OFPPFL_NO_RECV , OFPPFL_NO_RECV_STP , OFPPFL_NO_FWD , and OFPPFL_NO_PACKET_IN bits in the OpenFlow port flags may be useful for the controller to implement STP*
Profile Status	OPTIONAL
Requirements	Correct implementation of OFPT_PORT_MOD - OFPPFL_NO_FWD
Topology	Control-plane connection between DUT and reference controller.One data plane port.
Methodology	Configure and connect the Primary-controller on the DUT. Check port flood state, change port state with OFPT_PORT_MOD - No_Forward message, verify state change took place. Change back to original port state with OFPT_PORT_MOD - No_Forward message, verify port state change took place.
Results	Pass or Fail
Remarks	

Test case 90.140: OFPT_PORT_MOD - No_Packet_in

Test Number	90.140
Test Title	Protocol Messages / Controller to switch message/ OFPT_PORT_MOD - No_Packet_in
Test Purpose	Verify Controller is able to use the OFPT_PORT_MOD – OFPPC_NO_PACKET_IN message to change port state on the DUT
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.2.1 Port Structures, Page 17. *OFPPC_NO_PACKET_IN = 1 << 6 /* Do not send packet-in msgs for port. */* *The OFPPFL_NO_RECV , OFPPFL_NO_RECV_STP , OFPPFL_NO_FWD , and OFPPFL_NO_PACKET_IN bits in the OpenFlow port flags may be useful for the controller to implement STP*
Profile Status	OPTIONAL
Requirements	Correct implementation of OFPT_PORT_MOD – OFPPC_NO_PACKET_IN
Topology	Control-plane connection between DUT and reference controller. At least One data plane connection to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Check port

	flood state, change port state with OFPT_PORT_MOD - OFPPC_NO_PACKET_IN flag, verify state change took place. Change back to original port state. Verify port state change took place.
Results	Pass or Fail
Remarks	

Test case 90.150: OFPT_PACKET_OUT

Test Number	90.150
Test Title	Protocol Messages / Controller to switch message / OFPT_PACKET_OUT
Test Purpose	Verify Controller is able to use the OFPT_PACKET_OUT message to send a packet out of one of the DUT ports
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3.6 Send Packet Message, Page 35. *When the controller wishes to send a packet out through the datapath, it uses the OFPT_PACKET_OUT message*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPT_PACKET_OUT
Topology	Control-plane connection between DUT and reference controller. At least one data plane port connected to DUT.
Methodology	Configure and connect the Primary-controller on the DUT. Send a packet_out message targeting the data plane port. Verify the packet is sent out the switch port.
Results	Pass or Fail
Remarks	

Test case 90.160: OFPST_DESC

Test Number	90.160
Test Title	Protocol Messages / Controller to switch message / OFPST_DESC stats request / reply
Test Purpose	Verify Controller is able to respond to OFPST_DESC stats request, and returns valid field values
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3.5 Read State Messages, p. 31 *Information about the switch manufacturer, hardware revision, software revision, serial number, and a description field is available from the OFPST_DESC stats request type*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPST_DESC stats request / reply
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Send an OFPST_DESC stats request to the DUT, and verify a reply is received. Check valid return values for: char mfr_desc[DESC_STR_LEN; Manufacturer description char hw_desc[DESC_STR_LEN]; Hardware description. char sw_desc[DESC_STR_LEN]; Software description char serial_num[SERIAL_NUM_LEN]; Serial number.

	char dp_desc[DESC_STR_LEN]; Human readable description of datapath
Results	Pass or Fail
Remarks	

Test case 90.170: OFPT_QUEUE_GET_CONFIG_REPLY

Test Number	90.170
Test Title	Protocol Messages / Controller to switch message / OFPT_QUEUE_GET_CONFIG_REPLY
Test Purpose	Verify Controller is able to respond to OFPT_QUEUE_GET_CONFIG_REQUEST, and returns valid information
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.3.4 Queue Configuration Messages, p. 29. *The controller can query the switch for configured queues on a port*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of OFPT_QUEUE_GET_CONFIG request / reply
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Configure queues on the switch outside the OpenFlow protocol. Send an OFPT_QUEUE_GET_CONFIG_REQUEST for any port and verify reply is received. Verify reply has List of configured queues.
Results	Pass or Fail
Remarks	Queue configuration takes place outside the OpenFlow protocol, either through a command line tool or through an external dedicated configuration protocol.

Test Suite 100: Error Messages

Test group 100 checks for all possible error messages as mentioned in the spec.

Special cases:
Error messages pose some specific testing challenges. Some devices might never enter the state that triggers the error condition (e.g, soft switches might have unlimited tables, be able to simulate unlimited port numbers). Permission errors involve an entity outside the switch's OpenFlow implementation, and are optional test cases. All queue related error messages are optional. All emergency mode related error messages are optional.

Profiles:
All profiles have to pass all tests except (100.20, 100.50, 100.60, 100.70, 100.130, 100.140, 100.150, 100.170, 100.180, 100.190, 100.200, 100,220, 100.230, 100,250, 100.260, 100.270, 100.280, 100.290 and 100.300)

Test case 100.10: OFPHFC_INCOMPATIBLE

Test Number	100.10
Test Title	OFPT_ERROR / OFPET_HELLO_FAILED / OFPHFC_INCOMPATIBLE / No_Compatible_Version
Test Purpose	Verify DUT is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 38. *OFPET_HELLO_FAILED, /* Hello protocol failed. */* *OFPHFC_INCOMPATIBLE, /* No compatible version. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Error message OFPHFC_INCOMPATIBLE
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Send OFPT_HELLO message to DUT with an incompatible version. Verify correct error message is sent to the controller.
Results	Pass or Fail
Remarks	When the reason for a Hello failing is due to version incompatibility between switch and controller, then the switch generates OFPT_ERROR msg with Type Field OFPET_HELLO_FAILED and code field OFPHFC_INCOMPATIBLE

Test case 100.20: OFPHFC_EPERM

Test Number	100.20
Test Title	OFPT_ERROR / OFPET_HELLO_FAILED / OFPHFC_EPERM / Permission_Error
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 38. *OFPET_HELLO_FAILED, /* Hello protocol failed. */* *OFPHFC_EPERM /* Permissions error. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPHFC_EPERM
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPHFC_EPERM error condition. Verify correct error message is sent to the controller.
Results	Pass or Fail or Not Tested
Remarks	Permissions error generated by an entity between a controller and switch, such as an OpenFlow hypervisor This requires an intermediate device or emulation of an intermediate device to generate the permission error.

Test case 100.30: OFPBRC_BAD_VERSION

Test Number	100.30
Test Title	OFPT_ERROR / OFPET_BAD_REQUEST / OFPBRC_BAD_VERSION / Bad_Version
Test Purpose	Verify Controller is able to respond correctly to error condition

Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 38. *OFPET_BAD_REQUEST, /* Request was not understood. */* *OFPBRC_BAD_VERSION, /* ofp_header.version not supported. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Error message OFPBRC_BAD_VERSION
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBRC_BAD_VERSION Error condition. This can be done by sending a OFPT_STATS_REQUEST with a version field of 0. Verify that the correct error message is sent to the controller
Results	Pass or Fail
Remarks	When the header in the request msg contains a version field which is not supported by the switch, it generates OFPT_ERROR_msg with Type field OFPET_BAD_REQUEST and code field OFPBRC_BAD_VERSION

Test case 100.40: OFPBRC_BAD_TYPE

Test Number	100.40
Test Title	OFPT_ERROR / OFPET_BAD_REQUEST / OFPBRC_BAD_TYPE / Bad_Type
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 38. *OFPET_BAD_REQUEST, /* Request was not understood. */* *OFPBRC_BAD_TYPE, /* ofp_header.type not supported. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Error message OFPBRC_BAD_TYPE
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBRC_BAD_TYPE Error condition.This can be done by sending an unknown request to the switch. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	When the header in the request msg contains a type field which is not supported by the switch, it generates OFPT_ERROR_msg with Type field OFPET_BAD_REQUEST and code field OFPBRC_BAD_TYPE

Test case 100.50: OFPBRC_BAD_VENDOR

Test Number	100.50
Test Title	OFPT_ERROR / OFPET_BAD_REQUEST / OFPBRC_BAD_VENDOR/ Bad_Vendor
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 38. *OFPET_BAD_REQUEST, /* Request was not understood. */* *OFPBRC_BAD_VENDOR, /* Vendor not supported (in ofp_vendor_header * or ofp_stats_request or ofp_stats_reply). */*
Profile Status	OPTIONAL

Requirements	Correct implementation of Error message OFPBRC_BAD_VENDOR
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBRC_BAD_VENDOR Error condition. This can be done by specifying an unknown vendor-id in the OFPST_VENDOR request. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	

Test case 100.60: OFPBRC_BAD_SUBTYPE

Test Number	100.60
Test Title	OFPT_ERROR / OFPET_BAD_REQUEST / OFPBRC_BAD_SUBTYPE / Bad_Subtype
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 38. *OFPET_BAD_REQUEST, /* Request was not understood. */* *OFPBRC_BAD_SUBTYPE, /* Vendor subtype not supported.*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPBRC_BAD_SUBTYPE
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBRC_BAD_SUBTYPE Error condition. This can be done by specifying an unknown vendor subtype in the OFPST_VENDOR request. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	

Test case 100.70: OFPBRC_EPERM

Test Number	100.70
Test Title	OFPT_ERROR / OFPET_BAD_REQUEST / OFPBRC_EPERM / Permission_Error
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 38. *OFPET_BAD_REQUEST, /* Request was not understood. */* *OFPBRC_EPERM, /* Permissions error. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPBRC_EPERM
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPHFC_EPERM error condition. Verify correct error message is sent to the controller.
Results	Pass or Fail or Not Tested
Remarks	Permissions error generated by an entity between a controller and switch, such as an OpenFlow hypervisor This requires an intermediate device or emulation of an intermediate

	device to generate the permission error.

Test case 100.80: OFPBRC_BAD_LEN

Test Number	100.80
Test Title	OFPT_ERROR / OFPET_BAD_REQUEST / OFPBRC_BAD_LEN / Bad_Length
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 38. *OFPET_BAD_REQUEST, /* Request was not understood. */* *OFPBRC_BAD_LEN, /* Wrong request length for type. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Error message OFPBRC_BAD_LEN
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBRC_BAD_LEN Error condition. This can be done by sending a OFPT_STATS_REQUEST with incorrect header length. Verify correct error message is sent to the controller
Results	Pass or Fail
Remarks	

Test case 100.90: OFPBRC_BUFFER_EMPTY

Test Number	100.90
Test Title	OFPT_ERROR / OFPET_BAD_REQUEST / OFPBRC_BUFFER_EMPTY / Buffer_Empty
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 38. *OFPET_BAD_REQUEST, /* Request was not understood. */* *OFPBRC_BUFFER_EMPTY, /* Specified buffer has already been used. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Error message OFPBRC_BUFFER_EMPTY
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBRC_BUFFER_EMPTY Error condition. This can be done by sending two packet_out messages referencing the same buffer. The first packet_out should succeed and empty the buffer, the second packet_out should trigger the error. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	When the buffer specified by the controller has already been used , switch replies back with OFPT_ERROR msg with type field OFPET_BAD_REQUEST and code field OFPBRC_BUFFER_EMPTY" /* Specified buffer has already been used. */

Test case 100.100: OFPBRC_BUFFER_UNKNOWN

Test Number	100.100

Test Title	OFPT_ERROR / OFPET_BAD_REQUEST / OFPBRC_BUFFER_UNKNOWN / Buffer_Unknown
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 38. *OFPET_BAD_REQUEST, /* Request was not understood. */* *OFPBRC_BUFFER_UNKNOWN /* Specified buffer does not exist. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Error message OFPBRC_BUFFER_UNKNOWN
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBRC_BUFFER_UNKNOWN Error condition. This can be done by specifying a random or unknown buffer_id in the OFPT_PACKET_OUT message outside the scope reported by the switch. Verify correct error message is sent to the controller
Results	Pass or Fail
Remarks	When the buffer specified by the controller does not exist, the switch replies back with OFPT_ERROR msg with type field OFPET_BAD_REQUEST" /* Specified buffer does not exist. */

Test case 100.110: OFPBAC_BAD_TYPE

Test Number	100.110
Test Title	OFPT_ERROR / OFPT_BAD_ACTION / OFPBAC_BAD_TYPE / Bad_Type
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 39. *OFPET_BAD_ACTION, /* Error in action description. */* *OFPBAC_BAD_TYPE, * Unknown action type. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Error message OFPBAC_BAD_TYPE
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBAC_BAD_TYPE Error condition. This can be done by sending a flow with action OFPAT_OUTPUT such that type field in the action header is an unknown value. Verify correct error message is sent to the controller .
Results	Pass or Fail
Remarks	When the type field in the action header specified by the controller is unknown, the switch generates an OFPT_ERROR msg with type field OFPBET_BAD_ACTION and code field OFPBAC_BAD_TYPE" /* Unknown action type. */

Test case 100.120: OFPBAC_BAD_LEN

Test Number	100.120
Test Title	OFPT_ERROR / OFPT_BAD_ACTION / OFPBAC_BAD_LEN / Bad_Length2
Test Purpose	Verify Controller is able to respond correctly to error condition

Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 2. *OFPET_BAD_ACTION, /* Error in action description. */* *OFPBAC_BAD_LEN, /* Length problem in actions. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Error messages
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger Error condition. This can be done by sending a flow with action OFPAT_OUTPUT such that length field in the action_header is an incorrect value. Verify correct error message is sent to the controller
Results	Pass or Fail
Remarks	When the length field in the action header specified by the controller is wrong, the switch replies back with an OFPT_ERROR msg with Type Field OFPBAC_BAD_LEN" /* Length problem in actions. */

Test case 100.130: OFPBAC_BAD_VENDOR

Test Number	100.130
Test Title	OFPT_ERROR / OFPT_BAD_ACTION / OFPBAC_BAD_VENDOR / Bad_Vendor
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 2. *OFPET_BAD_ACTION, /* Error in action description. */* *OFPBAC_BAD_VENDOR, /* Unknown vendor id specified. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPBAC_BAD_VENDOR
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBAC_BAD_VENDOR Error condition.This can be done by sending a flow with action OFPAT_VENDOR such that vendor id specified in the ofp_action_vendor_header is an unknown value. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	Unknown vendor id specified.

Test case 100.140: OFPBAC_BAD_VENDOR_TYPE

Test Number	100.140
Test Title	OFPT_ERROR / OFPT_BAD_ACTION / OFPBAC_BAD_VENDOR_TYPE / Bad_Vendor_Type, /* Unknown action type for vendor id. */
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 2. *OFPET_BAD_ACTION, /* Error in action description. */* *OFPBAC_BAD_VENDOR_TYPE,* /* Unknown action type for vendor id. */
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message

	OFPBAC_BAD_VENDOR_TYPE
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBAC_BAD_VENDOR_TYPE Error condition. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	Unknown action type for vendor id

Test case 100.150: OFPBAC_BAD_OUT_PORT

Test Number	100.150
Test Title	OFPT_ERROR / OFPT_BAD_ACTION / OFPBAC_BAD_OUT_PORT / Bad_Out_Port, /* Problem validating output action. */
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 2. *OFPET_BAD_ACTION, /* Error in action description. */* *OFPBAC_BAD_OUT_PORT, /* Problem validating output action. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPBAC_BAD_OUT_PORT
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBAC_BAD_OUT_PORT Error condition. This can be done by sending a flow with action OFPAT_OUTPUT to egress_port OFPP_MAX. Verify correct error message is sent to the controller
Results	Pass or Fail
Remarks	/* When the output to switch port action refers to a port that does not exist, the switch generates an OFPT_ERROR msg , with type field OFPT_BAD_ACTION and code field OFPBAC_BAD_OUT_PORT" /* Problem validating output action. */

Test case 100.160: OFPBAC_BAD_ARGUMENT

Test Number	100.160
Test Title	OFPT_ERROR / OFPT_BAD_ACTION / OFPBAC_BAD_ARGUMENT / Bad_Argument
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 2. *OFPET_BAD_ACTION, /* Error in action description. */* *OFPBAC_BAD_ARGUMENT, /* Bad action argument. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Error message OFPBAC_BAD_ARGUMENT
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBAC_BAD_ARGUMENT Error condition. This can be done by sending a flow with action OFPAT_SET_VLAN_VID such that vlan_vid specified in the action is an incorrect value. Verify correct

	error message is sent to the controller
Results	Pass or Fail
Remarks	/* if the arguments specified in the action are wrong , then the switch reponds back with an OFPT_ERROR msg with type field OFPT_BAD_ACTION and code field OFPBAC_BAD_ARGUMENT" /* Bad action argument. */

Test case 100.170: OFPBAC_EPERM

Test Number	100.170
Test Title	OFPT_ERROR / OFPT_BAD_ACTION / OFPBAC_EPERM / Permission_Error3
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 2. *OFPET_BAD_ACTION, /* Error in action description. */* *OFPBAC_EPERM, /* Permissions error. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPBAC_EPERM
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBAC_EPERM Error condition. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	Permissions error generated by an entity between a controller and switch, such as an OpenFlow hypervisor" /* Permissions error. */ This requires an intermediate device or emulation of an intermediate device to generate the permission error.

Test case 100.180: OFPBAC_TOO_MANY

Test Number	100.180
Test Title	OFPT_ERROR / OFPT_BAD_ACTION / OFPBAC_TOO_MANY / Too_Many Actions
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 2. *OFPET_BAD_ACTION, /* Error in action description. */* *OFPBAC_TOO_MANY, /* Can't handle this many actions. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPBAC_TOO_MANY
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBAC_TOO_MANY Error condition. This can be done by sending a flow with lot of actions such that the switch is unable to support them. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	if the actions specified by the controller are more than that switch can support, the switch responds back with an OFPT_ERROR msg , with type field OFPT_BAD_ACTION and code field OFPBAC_TOO_MANY" /* Can't handle this many actions. */

	A software switch may not trigger such an error even on very large action_list.

Test case 100.190: OFPBAC_BAD_QUEUE

Test Number	100.190
Test Title	OFPT_ERROR / OFPT_BAD_ACTION / OFPBAC_BAD_QUEUE / Bad_Queue1
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 2. *OFPET_BAD_ACTION, /* Error in action description. */* *OFPBAC_BAD_QUEUE, /* Problem validating output queue. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPBAC_BAD_QUEUE
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPBAC_BAD_QUEUE Error condition. This can be done by sending a flow with action OFPAT_ENQUEUE such that queue_id specified in the action is an incorrect value. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	If the switch is not able to process the Enqueue action specified by the controller then the switch should generate an OFPT_ERROR msg , type field OFPT_BAD_ACTION and code field OFPBAC_BAD_QUEUE " /* Problem validating output queue. */

Test case 100.200: OFPFMFC_ALL_TABLES_FULL

Test Number	100.200
Test Title	OFPT_ERROR / OFPET_FLOW_MOD_FAILED / OFPFMFC_ALL_TABLES_FULL / All_Tables_Full
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 2. *OFPET_FLOW_MOD_FAILED, /* Problem modifying flow entry. */* *OFPFMFC_ALL_TABLES_FULL, /* Flow not added because of full tables. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPFMFC_ALL_TABLES_FULL
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPFMFC_ALL_TABLES_FULL Error condition. This can be done by inserting a lot of flows in the switch such that switch runs out of flow-tables. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	if the controller tries to insert a flow-entry when all the flow-tables are full , then the switch should respond back with an OFPT_ERROR msg , type field OFPET_FLOW_MOD_FAILED and code field OFPFMFC_ALL_TABLES_FULL" /* Flow not added because of full tables. */

Test case 100.210: OFPFMFC_OVERLAP

Test Number	100.210
Test Title	OFPT_ERROR / OFPET_FLOW_MOD_FAILED / OFPFMFC_OVERLAP / Overlap
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 40. *OFPET_FLOW_MOD_FAILED, /* Problem modifying flow entry. */* *OFPFMFC_OVERLAP, /* Attempted to add overlapping flow with * CHECK_OVERLAP flag set. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Error messages
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger Error condition OFPFMFC_OVERLAP .This can be done by sending a flow with OFPFF_CHECK_OVERLAP flag set. Then enter a second overlapping flow into the flow tableSend an overlapping flow .. Verify correct error message is sent to the controller
Results	Pass or Fail
Remarks	if the controller tries to insert an overlapping flow-entry with the Check overlap flag set , the switch responds back with an OFPT_ERROR msg, type field OFPET_FLOW_MOD_FAILED and code field OFPFMFC_OVERLAP" /* Attempted to add overlapping flow with CHECK_OVERLAP flag set. */

Test case 100.220: OFPFMFC_EPERM

Test Number	100.220
Test Title	OFPT_ERROR / OFPET_FLOW_MOD_FAILED / OFPFMFC_EPERM / Permission_Error4
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 40. *OFPET_FLOW_MOD_FAILED, /* Problem modifying flow entry. */* *OFPFMFC_EPERM, /* Permissions error. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPFMFC_EPERM
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPFMFC_EPERM Error condition. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	permissions error generated by an entity between a controller and switch, such as an OpenFlow hypervisor" /* Permissions error. */ This requires an intermediate device or emulation of an intermediate device to generate the permission error.

Test case 100.230: OFPFMFC_BAD_EMERG_TIMEOUT

Test Number	100.230
Test Title	OFPT_ERROR / OFPET_FLOW_MOD_FAILED /

	OFPFMFC_BAD_EMERG_TIMEOUT / Bad_Emergency_Timeout
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 40. *OFPET_FLOW_MOD_FAILED, /* Problem modifying flow entry. */ OFPFMFC_BAD_EMERG_TIMEOUT, /* Flow not added because of non-zero idle/hard * timeout. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPFMFC_BAD_EMERG_TIMEOUT
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPFMFC_BAD_EMERG_TIMEOUT Error condition.This can be done by sending an emergency flow i.e a flow with OFPFF_EMERG flag set and idle_timeout set to a non-zero value. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	When the emergency flows are added by the controller, (those flows which are marked with emergency bit set), they should have a zero idle/hard timeout. Otherwise, should switch should respond with an OFPT ERROR msg , type field OFPET_FLOW_MOD_FAILED, code field OFPFMFC_BAD_EMERG_TIMEOUT" /* Flow not added because of non-zero idle/hard timeout

Test case 100.240: OFPFMFC_BAD_COMMAND

Test Number	100.240
Test Title	OFPT_ERROR / OFPET_FLOW_MOD_FAILED / OFPFMFC_BAD_COMMAND / Bad_Command
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 40. *OFPET_FLOW_MOD_FAILED, /* Problem modifying flow entry. */ OFPFMFC_BAD_COMMAND, /* Unknown command. */*
Profile Status	MANDATORY for ALL Profiles
Requirements	Correct implementation of Error message OFPFMFC_BAD_COMMAND
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPFMFC_BAD_COMMAND Error condition. This can be done by sending a ofpt_flow_mod message with command field set to an incorrect value. Verify correct error message is sent to the controller
Results	Pass or Fail
Remarks	when the flow_mod msg request is sent by the controller with the some invalid command , the switch responds with an OFPT_ERROR msg , type field OFPET_FLOW_MOD_FAILED and code field OFPFMFC_BAD_COMMAND" /* Unknown command. */

Test case 100.250: OFPFMFC_UNSUPPORTED

Test Number	100.250
Test Title	OFPT_ERROR / OFPET_FLOW_MOD_FAILED / OFPFMFC_UNSUPPORTED / Unsupported_Actionlist
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 40. *OFPET_FLOW_MOD_FAILED, /* Problem modifying flow entry. */* *OFPFMFC_UNSUPPORTED /* Unsupported action list - cannot process in* ** the order specified. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPFMFC_UNSUPPORTED
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPFMFC_UNSUPPORTED Error condition.This can be done by sending a flow with an action list specified such that order of the actions is unsupported. E.g. first action : OFPAT_OUTPUT and second action OFPAT_SET_DL_SRC . Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	When the controller sends a flow_mod request with a action list which is not supported by the switch , the switch should respond back with an error msg OFPT_ERROR, type field OFPET_FLOW_MOD_FAILED and code field OFPFMFC_UNSUPPORTED " "/* Unsupported action list - cannot process in order specified /*

Test case 100.260: OFPPMFC_BAD_PORT

Test Number	100.260
Test Title	OFPT_ERROR / OFPET_PORT_MOD_FAILED / OFPPMFC_BAD_PORT / Bad_Port1
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 40. *OFPET_PORT_MOD_FAILED, /* Port mod request failed. */* *OFPPMFC_BAD_PORT, /* Specified port does not exist. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPPMFC_BAD_PORT
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPPMFC_BAD_PORT Error condition. This can be done by sending a OFPT_PORT_MOD message for an invalid port e.g OFPP_MAX .Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	if the controller sends a port_mod request for the port that is invalid , the switch will respond back with an OFPT_ERROR msg , type field OFPT_ERROR and code field OFPPMFC_BAD_PORT

Test case 100.270: OFPPMFC_BAD_HW_ADDR

Test Number	100.270
Test Title	OFPT_ERROR / OFPET_PORT_MOD_FAILED / OFPPMFC_BAD_HW_ADDR / Bad_HW_ADDR
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 40. *OFPET_PORT_MOD_FAILED, /* Port mod request failed. */* *OFPPMFC_BAD_HW_ADDR, /* Specified hardware address is wrong. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPPMFC_BAD_HW_ADDR
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPPMFC_BAD_HW_ADDR Error condition. This can be done by sending OFPT_PORT_MOD message for any port with hw_addr[OFP_ETH_ALEN] field set to an incorrect value. i.e a value different than what was returned in ofp_phy_port_stuct. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	If the controller sends a port_mod request for any port with a hardware address that is different from one returned in ofp_phy_port struct. , the switch will respond back with an OFPT_ERROR msg , type field OFPET_PORT_MOD_FAILED and code field OFPPMFC_BAD_PORT" /* Specified hardware address is wrong. */

Test case 100.280: OFPQOFC_BAD_PORT

Test Number	100.280
Test Title	OFPT_ERROR / OFPET_QUEUE_OP_FAILED / OFPQOFC_BAD_PORT / Bad_Port
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 40. *OFPET_QUEUE_OP_FAILED /* Queue operation failed. */* *OFPQOFC_BAD_PORT, /* Invalid port (or port does not exist). */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPQOFC_BAD_PORT
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPQOFC_BAD_PORT Error condition. This can be done by sending ofp_queue_stats_request for an invalid port e.g OFPP_MAX. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	If the port specifed for any queue operation (like enqeue --output to queue or retrieving queue stats) is an invalid port , then the switch responds back with an error msg OFPT_ERROR msg , type field OFPET_QUEUE_OP_FAILED , code field OFPQOFC_BAD_PORT" /* Invalid port (or port does not exist). */

Test case 100.290: OFPQFC_BAD_QUEUE

Test Number	100.290
Test Title	OFPT_ERROR / OFPET_QUEUE_OP_FAILED / OFPQFC_BAD_QUEUE / Bad_Queue
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 40. *OFPET_QUEUE_OP_FAILED /* Queue operation failed. */* *OFPQOFC_BAD_QUEUE, /* Queue does not exist. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPQFC_BAD_QUEUE
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPQFC_BAD_QUEUE Error condition. This can be done by sending ofp_queue_stats_request for a valid port but an invalid queue_id Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	if the queue_id specifed for any queue operation (like enqeue --output to queue or retrieving queue stats) is an invalid queue , then the switch responds back with an error msg OFPT_ERROR msg , type field OFPET_QUEUE_OP_FAILED , code field OFPQOFC_BAD_QUEUE" /* Queue does not exist. */

Test case 100.300: OFPET_QUEUE_OP_FAILED

Test Number	100.300
Test Title	OFPT_ERROR / OFPET_QUEUE_OP_FAILED / OFPQOFC_EPERM / Permission Error 5
Test Purpose	Verify Controller is able to respond correctly to error condition
Specification Reference	OpenFlow Switch Specification 1.0.0, 5.4.4 Error Messages, p. 40. *OFPET_QUEUE_OP_FAILED /* Queue operation failed. */* *OFPQOFC_EPERM /* Permissions error. */*
Profile Status	OPTIONAL
Requirements	Correct implementation of Error message OFPQOFC_EPERM
Topology	Control-plane connection between DUT and reference controller.
Methodology	Configure and connect the Primary-controller on the DUT. Trigger OFPQOFC_EPERM Error condition. Verify correct error message is sent to the controller
Results	Pass or Fail or Not Tested
Remarks	Permissions error generated by an entity between a controller and switch, such as an OpenFlow hypervisor" /* Permissions error. */ This requires an intermediate device or emulation of an intermediate device to generate the permission error.

7 Official Results Reporting for Conformance

Conformance testing should follow the guidelines of the Conformance Testing Program as outlined on the ONF Conformance Testing website. This document outlines specific reporting requirements of this test specification.

A single report SHOULD be submitted for each DUT tested. The report SHOULD include the ONF Conformance Test Application that was submitted by the vendor.

The report MUST clearly state all relevant test bed information. Including, but not limited to:
All testbed topology and configuration information.
For hardware based test tools, include:

- Vendor/Manufacturer
- Chassis Model
- Card Model(s)
- All Software and Firmware Versions

For Software based Test Tools, include:

- Test Framework
- Software Version
- Server hardware specifications
- Server OS and Configuration Information

The report MUST clearly state all DUT relevant information. Including, but not limited to:
For hardware based DUTs, include:

- Vendor/Manufacturer
- Chassis Model
- Card Modules
- All Software/Firmware Versions

For software based DUTs, include:

- Software Version
- Server hardware specifications
- Server OS and/or hypervisor version and Configuration Information

The report MUST clearly indicate each Conformance Profile for which testing was performed. Profiles are defined in this document under section 3.1 Conformance Profiles.

The report MUST clearly state the result and indicate all relevant profiles for each MANDATORY and OPTIONAL test case that was executed.

Test case numbers MUST be included and match the test case numbers as described in this document to avoid ambiguity in the results reporting.

The report MUST clearly indicate whether or not the DUT has passed conformance testing for each profile tested.

It is OPTIONAL to include additional caveat, recommendation or assessment information.

Bugs should be reported separately to the ONF Testing and Interop Working Group.

8 Appendix A: References

1. OpenFlow Switch Specification 1.0.0
2. OpenFlow Switch Specification Errata v1.0.1
3. RFC 2119, "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, http://www.ietf.org/rfc/rfc2119.txt
4. Reference Test Code - https://github.com/InCNTRE/oftest
5. ONF Certified Test Labs - <insert link>
6. ONF Member Trademark Terms and Conditions

9 Appendix B: Credits

Spec Contributions, in Alphabetical Order:

Uwe Dahlmann, Michael Haugh, Zoltan Lajos Kis, Ronald Milford, Shreya Pandita, Sibylle Schaller, Rob Sherwood, Mark Tassinari