OPEN NETWORKING FOUNDATION

CONFORMANCE TEST SPECIFICATION FOR OPENFLOW SWITCH SPECIFICATION V1.3.4 BASIC SINGLE TABLE CONFORMANCE TEST PROFILE

Version 1.0

April 15, 2015

ONF TS-026

OpenFlow

Disclaimer

Table of Contents

1. Introduction

This document defines the requirements and corresponding test procedures that determine the conformance of an OpenFlow 1.3.4 enabled switch to the Basic Single Table Profile. Requirements are derived from the OpenFlow Switch Specification 1.3.4 available on the ONF website at www.opennetworking.org.

Official conformance testing may only be performed by an ONF Approved Test Lab. A list of ONF Approved Test Labs is available on the ONF conformance certification website at https://www.opennetworking.org/openflow-conformance-certification. Current requirements and procedures for becoming an ONF Approved Test Lab are included in the Testing Lab Requirements on the ONF conformance certification website.

A certificate of conformance may only be issued by ONF after final validation and approval of the test results. This document covers reporting requirements but does not cover the administrative process for submitting results or applying to ONF for a certificate of conformance. Current process and requirements can be found on the ONF conformance certification website.

Vendors may refer to these requirements and test procedures during development of their product. Detailed information on the conformance testing program and the procedures for applying can be found on the ONF conformance certification website.

Test tool manufacturers may use these requirements and test procedures in development of their testing products. All official conformance tests MUST be performed using an official certified test tool. Policies and procedures for certifying test tools can be found on the ONF conformance certification website.

Consumers may use these requirements and test results to determine the viability of products for inclusion within their network infrastructure.

This document does not cover requirement and test procedures for extensions outside of the main specification. Additionally, this document does not cover requirements for devices supporting multiple tables. Devices that support multiple tables may be tested and acquire conformance for the Basic Single Table Profile under certain restrictions as described in the section Basic Conformance Requirements.

Requirements and test procedures to determine conformance for any changes, clarifications or additions to the areas of the OpenFlow Switch Specification 1.3.4 covered by this test specification will be included in addendums to this document.

Requirements and test procedures to determine conformance for any major specification

release beyond the areas that are covered of the OpenFlow Switch Specification 1.3.4 and other versions (1.4, 1.5, etc.…) will be covered in a separate document. Changes will be considered and updates made according to the test specification maintenance and release process on the ONF conformance certification website.

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

2. Glossary

This glossary defines words used in this document within the context of this document only. These definitions are not intended to be the official definition as outlined by ONF or any other standards organization.

- Action: An operation that forwards the packet to a port or modifies the packet or its metadata.
- Action Bucket: A set of Actions that are applied to a packet.
- Action Set: A set of Action types that are applied to a packet when matching a flow with no specified GOTO instructions.
- Byte: An 8-bit octet.
- Controller: Test Framework, Controller software and supporting hardware that interacts with DUT using the OpenFlow protocol.
- Control Plane: Includes all elements responsible for controlling the Data Plane (Controller, Control Plane Connection and OpenFlow Agent on the DUT).
- Control Plane Connection: The TCP connection between the DUT and the Controller Software.
- Controller Software: Software residing on the Controller which implements the OpenFlow protocol to exchange OpenFlow messages over the Control Plane Connection with the OpenFlow Agent on the DUT.
- Data Plane: The Hardware or Software within a Network Device that applies instructions and actions to Packets.
- 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 an n-tuple consisting of Layer 1-4 header information and metadata.
- 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: A Forwarding Table in a Networking Device that defines how the device should process the flow.
- Group: A List of Action Buckets, which may be applied to a matching packet.
- Hybrid: Data Plane that simultaneously supports OpenFlow and Non-OpenFlow control.
- Ingress Port: Data Plane port on which the data packets enters the DUT.
- Instruction: An operation, which applies actions, adds a packet header, or modifies a packet header.
- Layer 2: Functionality and protocols associated with network switching.
- Layer 3: Functionality and protocols associated with network routing.
- Local: Represents the DUT's internal networking stack and management stack.
- 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.
- Meter: A switch element that can measure and control the rate of packets.
- Meter Band: Specifies the rate at which the band applies and the way packets should be processed.
- OpenFlow: ONF standard protocol that enables OpenFlow Controllers to control Networking Devices.
- OpenFlow Agent: Software resident on a switch to allow for OpenFlow support.
- OpenFlow Controller: See Controller
- 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 and implements at least one OpenFlow pipeline.
- 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.
- Product Family: Multiple Vendor/Manufacturer device models that meet a certain standard of "Similarity" as Defined by the Similarity Policy in the Conformance Test Program Policies and Procedures Manual.
- Test Framework: Application that implements NW test functions. Interfaces with the Controller to send and receive OpenFlow messages over the Control Plane. Interfaces with a traffic generator to send and receive packets over the Data Plane.
- TUT: Table Under Test
- 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. Basic Single Table Conformance Test Profile Requirements

Official conformance testing will be performed as outlined by the ONF Conformance Testing Program https://www.opennetworking.org/openflow-conformance-certification.

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

All test cases included in this test specification are mandatory to achieve certification for the Basic Single Table Conformance Test Profile unless specifically identified otherwise.

The Basic Single Table Conformance Test Profile is primarily based upon features listed as mandatory in the OpenFlow Switch Specification 1.3.4 available on the ONF website at www.opennetworking.org. However, due to the restricted Single Table focus, difficulties in implementing specific features, and the realities of current market demand for some features, this test specification may not strictly adhere to the OpenFlow Switch Specification when determining which features are mandatory or optional for the Basic Single Table Conformance Test Profile. Not all test cases developed for OpenFlow 1.3.4 conformance testing apply to this profile. Therefore, test case numbers may not be sequential.

This document excludes test definitions for the following OpenFlow features; multiple controllers, auxiliary connections, groups, meters, queues, and multiple table pipeline features. Additionally this document excludes certain Match fields, Actions, Counters, and other Optional features. Testing of these advanced optional features may be included in additional profiles and covered under subsequent test specifications.

This OpenFlow specification does not indicate any ordering of match fields except where a pre-requisite is required, therefore this document only tests ordering of match fields for pre-requisites.

This document does not cover pipeline validation. Testing of multi-table pipelines and multi-table features may be included in additional profiles and covered under subsequent test specifications. Because multi-table pipeline features are not tested in these documents, Basic Single Table conformance is only applicable to single table devices, multi-table devices that support a single table configuration, or multi-table devices that emulate a single table device.

In a Single Table Implementation, the normally mandatory Instruction type OFPIT_GOTO_TABLE will be left as optional, however a conformant switch MUST respond with correct error messages when a controller attempts to issue an OFPIT_GOTO_TABLE instruction if it is unsupported or the table does not exist.

Multi-table devices that wish to obtain Basic Single Table Profile conformance MUST support all mandatory match fields, Actions and Instructions in a single Table Under Test (TUT). If the TUT is not table zero, the TUT MUST be specified prior to testing. The chosen test tool MUST be able to perform all testing in the specified TUT. All test traffic on the data plane MUST pass through the TUT. Data Plane Test traffic may pass through additional tables prior to reaching the TUT as long as no modifications are made to the test traffic in any table other than the Table Under Test. If the TUT is not Table 0, then all pipeline configurations required to direct traffic to the specific TUT MUST be transparent to the test tool or table 0 MUST support the GOTO_TABLE Instruction and allow the test tool to forward all traffic to the TUT (i.e. GOTO_TABLE <table_id = TUT>). All mandatory Instructions and Actions MUST be supported in the TUT. Data plane test traffic MUST exit the table pipeline after Instructions and Actions have been applied in the TUT. Any modifications to the test traffic outside of the Table Under Test may not be considered conformant and MUST be described in full detail as a caveat in the final test report and reviewed by ONF to determine if the behavior is allowed.

In some cases, test cases described in this document are mutually exclusive. For example, a device can only be running in either fail standalone or fail secure mode. It is sufficient for Basic Single Table conformance to pass one of the two. These test cases will be identified in each chapter introduction. Some test cases are only relevant for specific implementations. Each test case will state the conditions making it mandatory or not applicable for a specific implementation. For example a device which does not implement prerequisite match checking will be unable to trigger an OFPET_BAD_MATCH error message with an OFPBMC_BAD_PREREQ code. The test result of test cases like this may be marked ‘Not Applicable’ by the certified test tool.

In some cases, several methods can be used to verify behavior. For example, whether the switch inserted a flow into the flow table can be verified through data plane traffic, or by parsing flow statistics or table statistics messages. In these cases it is up to the test tool vendor to decide which method to use when. These methods should be based on the most commonly implemented features, and should rely on as few secondary features as possible.

For the sake of brevity, duplicate test cases have been excluded from this document. Each of these test cases includes specification excerpts, which have been either implicitly or explicitly tested by another test case.

In most ways OpenFlow does not distinguish between physical and logical ports. The results of most test cases will be independent of the underlying OpenFlow port type. Exceptions to this statement include port state, in_phy_port, port counters, and port descriptions. In these areas logical ports may have a unique behavior, and should be handled appropriately during testing as described below.

1. Port state MUST be meaningfully mapped to the underlying transport. If for example, the logical port is a tunnel, the port state should be up as long as the tunnel remains in an up state.

If the logical port is a LAG, the state should be up as long as one of the physical ports of the LAG is up.
2. In packet_in messages, the correct in_phy_port MUST be reported along with the in_port of the logical OpenFlow interface.
3. Port counters MUST be consistent with vendor-defined behavior. For example, counters of a LAG may be the sum of the physical port statistics.
4. Port descriptions should be described as closely as possible using the defined port description flags. For example, a tunnel may be neither copper nor fiber. The speed may be other.

Whenever possible, we recommend physical ports be used for running the complete test suite. As logical ports may introduce a wide array of unexpected behaviors, it is not required that logical ports be tested, or used at all if enough physical ports are available. In the case of limited physical ports, logical ports may be used to complete this test suite. In this case however, the above four sections (port state, in_phy_port, port counters, and port description) MUST be verified for both logical and physical port types. All relating tests MUST be passed.

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.

4. Test Bed Configuration

The primary testbed will consist of

- A test controller with a single control channel connection to the DUT.
- The test controller should have the ability to perform a packet trace and decode OpenFlow 1.3 packets.
- A traffic generator/analyzer with a minimum of 4 ports compatible with the DUT for data plane connections to allow completion of all test cases. If applicable, it is allowable to use IN_PORT or CONTROLLER as one of the output ports.
- A backup test controller MAY be used for some tests, but is not required for Basic Single Table conformance.

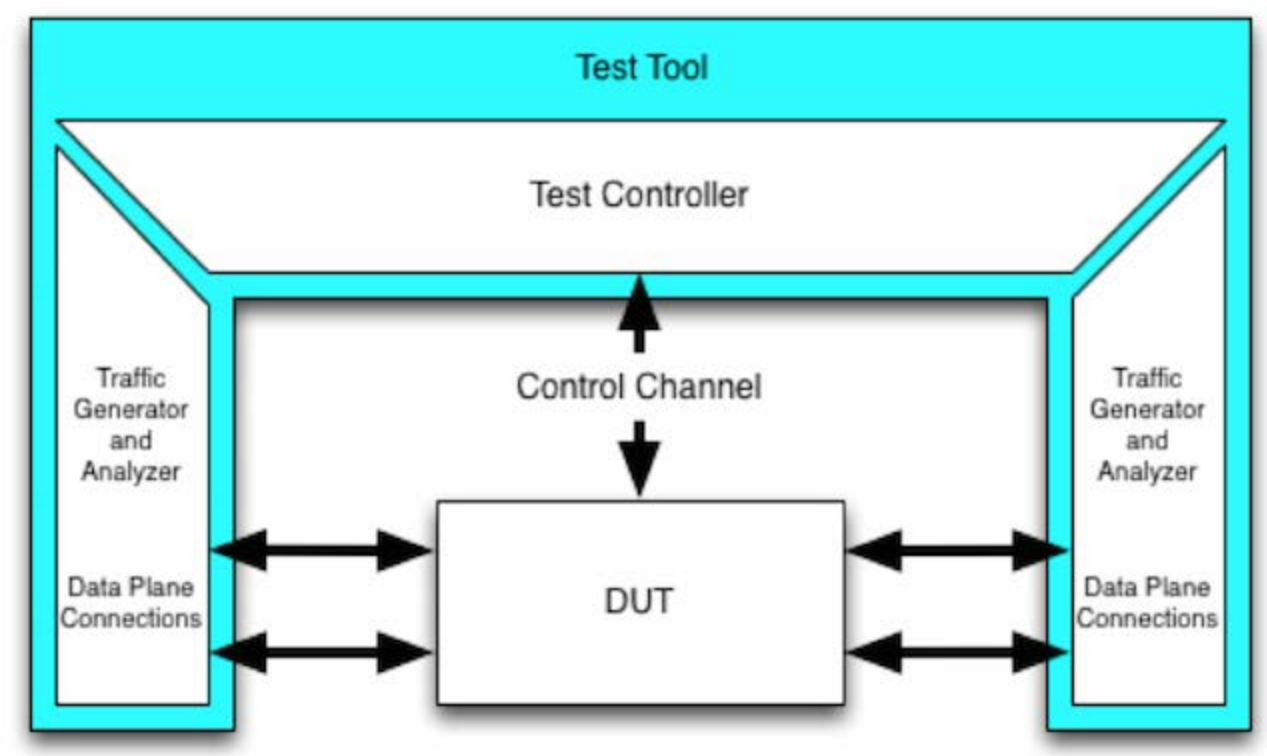

Figure 1: Test Bed Diagram OpenFlow Switch Test Suite

5. Test Case Template

<SuiteNumber> - <ChapterTitle>

This is a chapter overview.

Remarks
A section clarifying problems explicitly tied to this test chapter.

Basic Single Table Conformance Test Profile Requirements
A list of all test cases in this chapter that MUST be passed for Basic Single Table Conformance.

<SuiteNumber.TestNumber>. - <TestCaseTitle>

<TestSuite> / <TestGroup> / <SubGroup1> / <SubGroup2>

Purpose
Purpose and goal of test case.

Methodology
Specified test plan.

Specification text
OpenFlow Specification text relating to test case.
- OpenFlow Switch Specification reference

Topology
Topology required for this test case.

Results
Possible results of running this test case.

Test recommendations (Optional)
Information that may be relevant to test case implementation, including suggested test values.

Additional remarks (Optional)
Information which may be relevant to the test case, such as ambiguities in the specification or indications of when a test may not be applicable.

6. Official Results Reporting

This document does not cover policies and procedures outside of the specific testing methodology. Conformance Testing Program guidelines are outlined in the Conformance Test Program Policies and Procedures Manual. This document outlines specific reporting requirements of this test specification only.

A single report SHOULD be submitted for each DUT and each Profile tested. The report MUST include the following:

- Name of Approved Test Lab conducting the test
- Date(s) Test Performed
- Report Version
- Unique report ID
- Specification and Version tested (OpenFlow 1.3)
- Conformance Profile tested

The report MUST clearly indicate whether or not the DUT has passed all mandatory tests for the profile tested.

The report MUST clearly state all DUT relevant information. Including, but not limited to:

- Name of Vendor/Manufacturer of DUT
- Product Family if Applicable
- Chassis Model & Serial Number
- Line Card Model & Serial Numbers(s)
- All Software/Firmware Revision Information
- Unique Chassis MAC Address
- Brief Product Description
- All Configuration Information including any modifications made for specific test cases

For Software based DUTs, the report MUST also include:

- Server Hardware Specification
- Server OS and/or hypervisor version and configuration information
- All Software Version Information

The report MUST clearly state all relevant test bed information. Including, but not limited to:

- Testbed topology description or diagram
- All Testbed configuration information
- All Test Tool Information
    - For Hardware-based test tools
        - Vendor/Manufacturer
        - Chassis Model Number
        - Line Card Model Number(s)

        - All Software/Firmware Revision Information
    - For Software-based test tools
        - Framework Name
        - Framework Software Version
        - Server Hardware Specification
        - Server OS and/or hypervisor version and configuration information
        - All Software Version information
    - Wireshark Version if applicable

DUT Software or firmware changes are not allowed and MUST remain consistent throughout the entire test for a single profile test.

Test Framework Software or firmware changes are not allowed and MUST remain consistent throughout the entire test for a single profile test.

The report MUST clearly state any other changes from the initial state made during the test and identify the test case(s) for which the change was made, including, but not limited to:

- Topology changes
- DUT configuration changes
- DUT hardware changes
- Test tool hardware changes
- Test tool software version changes
- Test tool configuration changes

The report MUST clearly describe any manual testing that was performed outside of an authorized test tool and identify the test case(s) for which the manual testing was performed.

The report MUST clearly state the result 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.

All test tool logs, control plane traffic and data plane traffic MUST be captured and saved. All logs and traces MUST be made available upon request to authorized parties for validation purposes as outlined in the Conformance Test Program Policies and Procedures Manual.

Bugs in this document or approved test tools SHOULD be reported separately to the ONF Testing and Interop Working Group.

10 - Control Channel

The Control Channel test suite verifies establishment of a control channel, version negotiation, and device behavior when the control channel is lost.

Remarks

Control channel TCP port and encryption

Four methods of control channel establishment exist on encrypted/unencrypted channel ports; these are tested in test cases 10.30 through 10.60. To be considered conformant, a device MUST support at least one of two subsets of test cases. Results of test cases exclusive to the other subset MAY be recorded as 'Not Applicable', based on conditions as outlined in test case Remarks.

Control channel failure

After a loss of the control channel, a conformant device MUST enter either **Fail Secure** mode or **Fail Standalone** mode.

Basic Single Table Conformance Test Profile Requirements

All devices MUST pass Test cases 10.10, and 10.70 - 10.100.

An OpenFlow-enabled device MUST support a TLS-encrypted control channel on the default port (test case 10.50) and allow for user configurable certificates (test case 10.20). Additionally, a device *MUST* support a control channel connection on alternative ports.

> Either encrypted control channel with the default port and unencrypted control channel with default and non-default ports (pass all 4 test cases 10.20, 10.30, 10.40, and 10.50)
>
> OR
>
> Encrypted control channel with both default and non-default port (pass all 3 test cases 10.20, 10.50, and 10.60).

An OpenFlow-enabled device MUST support one of the required failure modes.

> Either Fail Secure (pass test cases 10.110 and 10.130).
>
> OR
>
> Fail Standalone mode (pass test case 10.120).

10.10 - Startup behavior without established control channel

Control Channel / Basic / Switch startup / Switch startup

Purpose
Startup from factory default mode. Expected behavior should be as defined in the switch documentation.

Methodology
One OpenFlow instance is configured, controller is not reachable. The switch starts up, and no control channel is established. Packets are sent to the OpenFlow data plane ports. They are either dropped, or switched by a learning switch. Behavior is verified by data plane packet traces. The vendor must provide the expected default behavior, which will be either "fail secure" or "fail standalone" mode. If the default behavior is "fail secure", verify that all data plane traffic is dropped. If "fail standalone" mode is specified, verify traffic is forwarded as per the vendor's defined behavior.

Specification text
The first time a switch starts up, it will operate in either "fail secure mode" or "fail standalone mode" mode, until it successfully connects to a controller. Configuration of the default set of flow entries to be used at startup is outside the scope of the OpenFlow protocol.
- *OpenFlow Switch Specification 1.3.4 (ch. 6.3.3; pg. 34)*

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

Test recommendations
Some devices might have an in-band control plane. These devices might implement some default behavior for allowing the local ip-stack to communicate with an openflow controller through the data plane. These devices might do the following: 1. Try to get a dhcp lease for their local interface through the data plane. These behaviors should be accepted for in-band control planes. 2. ARP: The device might arp for a default gateway if provided by the dhcp server, or a controller if the controller is configured and on the local subnet, and might answer to ARP request for their own interface. This behavior should be accepted for in-band control planes. 3. The device might forward traffic for the local ethernet MAC address and/or local IP address to the local IP stack, and perhaps generate answer packets. This behavior should be accepted for devices with in-band behavior. For devices with in-band control planes, all packets not hitting these special edge cases need to be processed in either fail-secure, or fail-standalone mode. If the T&I working group publishes a best practice white paper regarding in-band control, this white paper will define these requirements in more detail.

Additional remarks

This test is meant to verify the default behavior without any previous additional configuration by an operator.

10.20 - Certificate configuration for TLS

Control Channel / Basic / Encryption / Certificates

Purpose
Check the configuration for TLS encrypted control plane connections.

Methodology
Configure test framework and switch for a TLS encrypted control channel. Prepare necessary management plane if necessary (pki).

Specification text
The switch and controller mutually authenticate by exchanging certificates signed by a site-specific private key. Each switch must be user-configurable with one certificate for authenticating the controller (controller certificate) and the other for authenticating to the controller (switch certificate).
- *OpenFlow Switch Specification 1.3.4 (ch. 6.3.4; pg. 35)*

Topology
One control plane connection

Results
Pass / Fail

10.30 - TCP default Port

Control Channel / Basic / Establishment / Establishment

Purpose
Test unencrypted control channel establishment on default port.

Methodology
Reference controller must be running and reachable at configured IP and Port 6653. Configure DUT to connect with reference controller using unencrypted TCP. If required, manually configure switch to connect to controller using TCP port 6653.

Specification text
The switch must be able to establish communication with a controller at a user-configurable (but otherwise fixed) IP address, using either a user-specified transport port or the default transport port. If the switch is configured with the IP address of the controller to connect to, the switch initiates a standard TLS or TCP connection to the controller. Traffic to and from the OpenFlow channel is not run through the OpenFlow pipeline. Therefore, the switch must identify incoming traffic as local before checking it against the flow tables.
- OpenFlow Switch Specification 1.3.4 (ch. 6.3.1; pg. 33)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If control channel encryption test cases are passed for both default and non-default ports, then unencrypted control channels need not be supported and the result may be 'Not Applicable'.

10.40 - TCP non default port

Control Channel / Basic / Establishment / Establishment

Purpose
Test unencrypted control channel establishment on non-default port.

Methodology
Reference controller must be running and reachable at configured IP and Port unequal to 6653. Configure DUT to connect with reference controller using unencrypted TCP. Manually configure switch to connect to controller using configured TCP port.

Specification text
The switch must be able to establish communication with a controller at a user-configurable (but otherwise fixed) IP address, using either a user-specified transport port or the default transport port. If the switch is configured with the IP address of the controller to connect to, the switch initiates a standard TLS or TCP connection to the controller. Traffic to and from the OpenFlow channel is not run through the OpenFlow pipeline. Therefore, the switch must identify incoming traffic as local before checking it against the flow tables.
- OpenFlow Switch Specification 1.3.4 (ch. 6.3.1; pg. 33)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If control channel encryption test cases are passed for both default and non-default ports, then unencrypted control channels need not be supported and the result may be 'Not Applicable'.

10.50 - TLS with default TCP port

Control Channel / Basic / Establishment / Establishment

Purpose
Test encrypted control channel establishment on default port.

Methodology
Reference controller must be running and reachable at configured IP and Port 6653. Configure DUT to connect with reference controller using encrypted TLS. If required, manually configure switch to connect to controller using TCP port 6653.

Specification text
The switch must be able to establish communication with a controller at a user-configurable (but otherwise fixed) IP address, using either a user-specified transport port or the default transport port. If the switch is configured with the IP address of the controller to connect to, the switch initiates a standard TLS or TCP connection to the controller. Traffic to and from the OpenFlow channel is not run through the OpenFlow pipeline. Therefore, the switch must identify incoming traffic as local before checking it against the flow tables.
- OpenFlow Switch Specification 1.3.4 (ch. 6.3.1; pg. 33)

Topology
One control plane connection

Results
Pass / Fail

10.60 - TLS non default port

Control Channel / Basic / Establishment / Establishment

Purpose
Test encrypted control channel establishment on non-default port.

Methodology
Reference controller must be running and reachable at configured IP and Port unequal 6653. Configure DUT to connect with reference controller using encrypted TLS. Manually configure switch to connect to controller using configured TCP port.

Specification text
The switch must be able to establish communication with a controller at a user-configurable (but otherwise fixed) IP address, using either a user-specified transport port or the default transport port. If the switch is configured with the IP address of the controller to connect to, the switch initiates a standard TLS or TCP connection to the controller. Traffic to and from the OpenFlow channel is not run through the OpenFlow pipeline. Therefore, the switch must identify incoming traffic as local before checking it against the flow tables.
- OpenFlow Switch Specification 1.3.4 (ch. 6.3.1; pg. 33)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If unencrypted control channel test cases are passed for both default and non-default ports, then encrypted control channels need not support non-standard connection ports and the result may be 'Not Applicable'.

10.70 - Version negotiation on version field success

Control Channel / Basic / Version negotiation / Field success

Purpose
Check that the switch negotiates the correct version with the controller, based on the version field.

Methodology
Configure and connect the Primary-controller on the DUT. Verify that the switch negotiates the correct version number (4) with the reference controller. For negotiation, the controller must only use the version field. It does not send an OFPHET_VERSIONBITMAP hello element in its hello message.

Specification text
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 (see 7.1). This Hello message may optionally include some OpenFlow elements to help connection setup (see 7.5.1). Upon receipt of this message, the recipient must calculate the OpenFlow protocol version to be used. If both the Hello message sent and the Hello message received contain an OFPHET_VERSIONBITMAP hello element, and if those bitmaps have some common bits set, the negotiated version must be the highest version set in both bitmaps. Otherwise, the negotiated version must be the smaller of the version number that was sent and the one that was received in the version fields. If the negotiated version is supported by the recipient, then the connection proceeds.
- OpenFlow Switch Specification 1.3.4 (ch. 6.3.1; pg. 33)

Topology
One control plane connection

Results
Pass / Fail

10.80 - Version negotiation failure

Control Channel / Basic / Version negotiation / Failure

Purpose
Verify the correct behavior in the case of version negotiation failure.

Methodology
If existing, send a version that is not supported by switch, and lower than the switch side provided version number. Verify that the switch generates an OFPT_ERROR message with a type field of OFPET_HELLO_FAILED, a code field of OFPHFC_INCOMPATIBLE, and then terminates the connection.

Specification text
Otherwise, the recipient must reply with an OFPT_ERROR message with a type field of OFPET_HELLO_FAILED, a code field of OFPHFC_INCOMPATIBLE, and optionally an ASCII string explaining the situation in data, and then terminate the connection.
- OpenFlow Switch Specification 1.3.4 (ch. 6.3.1; pg. 33)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
If a switch supports all possible version numbers below the current version 1.3, this error might not be triggered.

Additional remarks
If a switch supports all possible version numbers below the current version 1.3, this error might not be triggered. Instead attempt to use a non-published OpenFlow version number.

10.90 - Version negotiation based on bitmap

Control channel / Basic / Version negotiation / bitmap

Purpose
Verify that version negotiation based on bitmap is successful.

Methodology
Send a Version field that is lower than the current version. Send an OFPHET_VERSIONBITMAP field with the first Hello message that sets the current version as valid. Verify that the switch accepts the current version by checking the version number in subsequent messages.

Specification text
If both the Hello message sent and the Hello message received contain an OFPHET_VERSIONBITMAP hello element, and if those bitmaps have some common bits set, the negotiated version must be the highest version set in both bitmaps.
- OpenFlow Switch Specification 1.3.4 (ch. 6.3.1; pg. 33)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Test recommendations
We want to hit a corner case where the traditional negotiation does not provide the best result (i.e. switch replies with a version not supported error.) Then the bitmap is used to find a common version between switch and controller.

Additional remarks
Sending a version field that is lower than the current version, and in contradiction to the bitmap field information, might be problematic. It is done to ensure the results of version field and bitmap are different in order to determine the correct information source was used.

If Device does not support version bitmaps, this test case shall be considered 'Not Applicable'.

10.100 - Control channel failure mode

Control channel / Basic / Connection Interruption / Failure mode

Purpose
Verify that the switch enters the correct state after loss of the controller connection.

Methodology
Configure and connect the Primary-controller on the DUT. After control channel establishment, create and install a flow entry with no timeout with an output action to a specific port. Fail the control channel by one of the methods mentioned in the specification text (echo request timeout, TLS session timeout, or other disconnection), and verify the switch falls into the configured failure mode by sending matching data plane packets.

Specification text
In the case that a switch loses contact with all controllers, as a result of echo request timeouts, TLS session timeouts, or other disconnections, the switch must immediately enter either “fail secure mode” or “fail standalone mode”, depending upon the switch implementation and configuration.
- OpenFlow Switch Specification 1.3.4 (ch. 6.3.2; pg. 34)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

Additional remarks
If DUT falls back to fail-secure, data plane traffic should only be seen on specific port configured in the output action. If DUT falls back to fail-standalone, then expect to see no traffic forwarded or traffic broadcast out all port.

10.110 - Fail secure mode behavior

Control Channel / Basic / Connection Interruption / Fail secure mode

Purpose
Verify that in fail-secure mode the switch does not try to send packets to the controller, and that flow entries stay active and expire as expected.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install two flow entries, one with a long timeout and one with a short timeout. Fail the control channel by one of the previously mentioned methods. Verify with a network capture that no packets are sent on the control plane. Verify with data plane traffic that flows stay active, and are timed out as expected.

Specification text
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”.
- OpenFlow Switch Specification 1.3.4 (ch. 6.3.2; pg. 34)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail / Not Applicable

Test recommendations
If "fail standalone" mode is specified, verify traffic is forwarded as per the vendor’s defined behavior.

Additional remarks
If DUT supports fail-standalone mode only, then this test may be considered 'Not Applicable'.

10.120 - Fail standalone mode - OFPP_Normal - Hybrids

Control Channel / Basic / Connection interruption / Fail standalone mode

Purpose
Verify the correct operation of fail-standalone mode.

Methodology
Configure and connect the Primary-controller on the DUT. After control channel establishment, create and install flow entries with no timeout. Fail the control channel by one of the previously mentioned methods. Verify that the traffic is forwarded as per the vendor's defined behavior.

Specification text
In "fail standalone mode", the switch processes all packets using the OFPP_NORMAL reserved port; in other words, the switch acts as a legacy Ethernet switch or router. The "fail standalone mode" is usually only available on Hybrid switches (see 5.1.1).
- OpenFlow Switch Specification 1.3.4 (ch. 6.3.2; pg. 34)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail / Not Applicable

Test recommendations
We currently expect most implementations to support L2 learning switch behavior, however the OpenFlow spec allows for other default behaviors. If a switch supports another default behavior, the test must be adapted accordingly.

Either 10.120 or 10.130 must be passed by a device to be considered conformant.

Additional remarks
If DUT supports fail-secure mode only, then this test may be considered 'Not Applicable'.

10.130 - Existing flow entries stay active

Control Channel / Basic / Fail-secure mode / Reconnection

Purpose
Verify that flows stay active and timeout as configured after control channel re-establishment.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install two flow entries, one with a long timeout and one with a short timeout. Disconnect control channel, reconnect control channel after short timeout is reached, verify that the long flow entries are still in place after reconnection. Verify that the long flow entries time-out as expected by probing with data plane packets.

Specification text
Upon connecting to a controller again, the existing flow entries remain. The controller then has the option of deleting all flow entries, if desired.
- OpenFlow Switch Specification 1.3.4 (ch. 6.3.2; pg. 34)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail / Not Applicable

Test recommendations
If a device does not support the fail-secure mode this test case doesn't need to be executed; in this case the result may be recorded as "Pass" or "Not Tested". Either 10.120 or 10.130 must be passed by a device to be considered conformant.

Additional remarks
If DUT supports fail-standalone mode only, then this test may be considered 'Not Applicable'.

40 - Controller to Switch Messages

Test suite 40 verifies that all basic information is correctly reported by a device.

Remarks

Correct values

Test cases 40.10 through 40.40, 40.120, and 40.170 - 40.210 verify values reported by the device are correct. These tests require information to be provided by the vendor for correct verification.

Information gathering

Test cases 40.50 through 40.110, and 40.130 through 40.160 only verify that the messages do not generate an error message. We do not check for correct response values or implementation; these are verified in later test suites.

Logical ports

The behavior of logical ports should be identical to physical ports. When testing we recommend testing on physical ports. If testing on physical ports is not supported, or if there is an inadequate number of test ports, then logical ports MUST be used.

In general for logical ports, when verifying an ofp_packet_in message ensure that either an OFPXMT_IN_PORT and OFPXMT_PHY_PORT, or an OFPXMT_IN_PORT and OFPXMT_TUNNEL_ID, or OFPXMT_IN_PORT OFPXMT_PHY_PORT and OFPXMT_TUNNEL_ID are included in the ofp_packet_in match field. Additional information has been included in the "implementation recommendations" for test cases, which require special care when testing with logical ports.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

40.10 - Features reply - Datapath ID

Controller-to-Switch messages / Information gathering / Features / Datapath ID

Purpose
Verify that an OFPT_FEATURES_REQUEST message generates an OFPT_FEATURES_REPLY from the switch containing a valid datapath ID.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_FEATURES_REQUEST message. Verify that the switch responds with an OFPT_FEATURES_REPLY message containing a valid datapath ID matching configured value or the lower 48-bits match the MAC Address of the Control Channel interface.

Specification text
The OFPT_FEATURES_REQUEST message is used by the controller to identify the switch and read its basic capabilities. Upon session establishment (see 6.3.1), the controller should send an OFPT_FEATURES_REQUEST message. This message does not contain a body beyond the OpenFlow header. The switch must respond with an OFPT_FEATURES_REPLY message:
/* Switch features. */
struct ofp_switch_features {
    struct ofp_header header;
    uint64_t datapath_id; /* Datapath unique ID. The lower 48-bits are for a MAC address, while the upper 16-bits are implementer-defined. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.1; pg. 71)*

Topology
One control plane connection

Results
Pass / Fail

40.20 - Features reply - max buffers

Controller-to-Switch messages / Information gathering / Features / Buffers

Purpose
Verify that the OFPT_FEATURES_REQUEST message generates an OFPT_FEATURES_REPLY from the switch containing the correct number of buffers.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_FEATURES_REQUEST message. Verify that the switch responds with an OFPT_FEATURES_REPLY message containing the correct number of buffers available.

Specification text
uint32_t n_buffers; /* Max packets buffered at once. */ The n_buffers field specifies the maximum number of packets the switch can buffer when sending packets to the controller using packet-in messages (see 6.1.2).
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.1; pg. 71)*

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
The correct number of buffers should be documented by the vendor.

40.30 - Features reply - Number of tables supported

Controller-to-Switch messages / Information gathering / Features / Number of tables

Purpose
Verify that the OFPT_FEATURES_REQUEST message generates an OFPT_FEATURES_REPLY from the switch containing the correct number of tables supported.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_FEATURES_REQUEST message. Verify that the switch responds with an OFPT_FEATURES_REPLY message containing the correct number of tables supported.

Specification text
uint8_t n_tables; /* Number of tables supported by datapath. */ The n_tables field describes the number of tables supported by the switch, each of which can have a different set of supported match fields, actions and number of entries. When the controller and switch first 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 an OFPMP_TABLE_FEATURES multipart request (see 7.3.5.5). A switch must return these tables in the order the packets traverse the tables.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.1; pg. 71)*

Topology
One control plane connection

Results
Pass / Fail

Additional remarks
The correct number of tables supported should be documented by the vendor.

40.40 - Features reply - Auxiliary ID

Controller-to-Switch messages / Information gathering / Features / Aux identifier

Purpose
Verify that an OFPT_FEATURES_REQUEST message generates an OFPT_FEATURES_REPLY from the switch containing a valid Auxiliary ID.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_FEATURES_REQUEST message. Verify that the switch responds with an OFPT_FEATURES_REPLY message containing a valid Auxiliary ID.

Specification text
uint8_t auxiliary_id; /* Identify auxiliary connections */

The auxiliary_id field identify the type of connection from the switch to the controller, the main connection has this field set to zero, an auxiliary connection has this field set to a non-zero value (see 6.3.6).
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.1; pg. 71)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
This test is normally run on the main connection and the Auxiliary ID should be 0.

Additional remarks
This check is redundant with the main aux tests in testgroup 30. It is checked here again to have all feature reply fields tested in one place.

40.50 - Features reply - Flow statistics

Controller-to-Switch messages / Information gathering / Capabilities / Flow statistics

Purpose
Check whether the switch supports flow statistics.

Methodology
Configure and connect DUT to controller. Following a control channel establishment, send an OFPT_FEATURES_REQUEST message. Verify that the switch responds with an OFPT_FEATURES_REPLY message containing a valid capabilities bitmap.

Verify whether the switch announces support of flow statistics.

Specification text
OFPC_FLOW_STATS = 1 << 0 ,/* Flow statistics. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.1; pg. 71)

Topology
One control plane connection

Results
Pass / Fail

Additional remarks
This test checks an ofp_capabilities enum. The result of this test may influence which other tests need to be run later. It may be checked against the vendor-supplied documentation.

40.60 - Features reply - Table statistics

Controller-to-Switch messages / Information gathering / Capabilities / Table statistics

Purpose
Check whether the switch supports table statistics.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_FEATURES_REQUEST message. Verify that the switch responds with an OFPT_FEATURES_REPLY message containing a valid capabilities bitmap.

Verify whether the switch announces support of table statistics.

Specification text
OFPC_TABLE_STATS = 1 << 1, /* Table statistics. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.1; pg. 71)

Topology
One control plane connection

Results
Pass / Fail

Additional remarks
The result of this test may influence which other tests need to be run later. It may also be checked against the vendor-supplied documentation.

40.70 - Features reply - Port statistics

Controller-to-Switch messages / Information gathering / Capabilities / Port statistics

Purpose
Check whether the switch supports port statistics.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_FEATURES_REQUEST message. Verify that the switch responds with an OFPT_FEATURES_REPLY message containing a valid capabilities bitmap.

Verify whether the switch announces support of port statistics. If statistics for logical ports are excluded, the result of this test shall be "fail."

Specification text
OFPC_PORT_STATS = 1 << 2, /* Port statistics. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.1; pg. 71)

Topology
One control plane connection

Results
Pass / Fail

Additional remarks
The result of this test may influence which other tests need to be run later. It may also be checked against the vendor-supplied documentation.

40.80 - Features reply - Group statistics

Controller-to-Switch messages / Information gathering / Capabilities / Group statistics

Purpose
Check whether the switch supports group statistics.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_FEATURES_REQUEST message. Verify that the switch responds with an OFPT_FEATURES_REPLY message containing a valid capabilities bitmap.

Verify whether the switch announces support of group statistics.

Specification text
OFPC_GROUP_STATS = 1 << 3, /* Group statistics. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.1; pg. 71)

Topology
One control plane connection

Results
Pass / Fail

Additional remarks
The result of this test may influence which other tests need to be run later. It may also be checked against the vendor-supplied documentation.

40.90 - Features reply - reassemble IP fragments

Controller-to-Switch messages / Information gathering / Capabilities / Can reassemble IP fragments

Purpose
Check whether the switch supports reassembling IP fragments.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_FEATURES_REQUEST message. Verify that the switch responds with an OFPT_FEATURES_REPLY message containing a valid capabilities bitmap.

Verify whether the switch announces support of IP fragments reassembly.

Specification text
OFPC_IP_REASM = 1 << 5, /* Can reassemble IP fragments. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.1; pg. 71)

Topology
One control plane connection

Results
Pass / Fail

Additional remarks
The result of this test may influence which other tests need to be run later. It may also be checked against the vendor-supplied documentation.

40.100 - Features reply - Queue statistics

Controller-to-Switch messages / Information gathering / Capabilities / Queue statistics

Purpose
Check whether the switch supports queue statistics.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_FEATURES_REQUEST message. Verify that the switch responds with an OFPT_FEATURES_REPLY message containing a valid capabilities bitmap.

Verify whether the switch announces support of queue statistics.

Specification text
OFPC_QUEUE_STATS = 1 << 6, /* Queue statistics. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.1; pg. 71)

Topology
One control plane connection

Results
Pass / Fail

Additional remarks
The result of this test may influence which other tests need to be run later. It may also be checked against the vendor-supplied documentation.

40.110 - Features reply - Block looping ports

Controller-to-Switch messages / Information gathering / Capabilities / Switch will block looping ports

Purpose
Check whether the switch supports blocking of looping ports.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_FEATURES_REQUEST message. Verify that the switch responds with an OFPT_FEATURES_REPLY message containing a valid capabilities bitmap.

Verify whether the switch announces support of blocking looping ports.

Specification text
OFPC_PORT_BLOCKED = 1 << 8 /* Switch will block looping ports. */ The OFPC_PORT_BLOCKED bit indicates that a switch protocol outside of OpenFlow, such as 802.1D Spanning Tree, will detect topology loops and block ports to prevent packet loops. If this bit is not set, in most cases the controller should implement a mechanism to prevent packet loops.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.1; pg. 71)

Topology
One control plane connection

Results
Pass / Fail

Additional remarks
The result of this test may influence which other tests need to be run later. It may also be checked against the vendor-supplied documentation.

40.120 - Get switch config - Miss send len

Controller-to-Switch messages / Information gathering / Get switch configuration / Miss send len

Purpose
Check the miss_send_len value returned by the switch.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_GET_CONFIG_REQUEST message. Verify that the switch responds with an OFPT_GET_CONFIG_REPLY message.

Verify that the returned miss_send_len value is valid.

Specification text
The miss_send_len field defines the number of bytes of each packet sent to the controller by the OpenFlow pipeline when not using an output action to the OFPP_CONTROLLER logical port, for example sending packets with invalid TTL if this message reason is enabled. If this field equals 0, the switch must send zero bytes of the packet in the ofp_packet_in message. If the value is set to OFPCML_NO_BUFFER the complete packet must be included in the message, and should not be buffered
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.2; pg. 72)*

Topology
One control plane connection

Results
Pass / Fail

Additional remarks
The result of this test may influence which other tests need to be run later. It may also be checked against the vendor-supplied documentation.

40.130 - Get switch config - Frag normal

Controller-to-Switch messages / Information gathering / Get switch configuration / Frag normal

Purpose
Check whether the switch is configured for "No special handling for fragments".

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_SET_CONFIG_REQUEST message setting the OFPC_FRAG_NORMAL bit. Verify that the device does not generate an error message. Send an OFPT_GET_CONFIG_REQUEST message. Verify that the switch responds with an OFPT_GET_CONFIG_REPLY message.

Check the ofp_config_flags for OFPC_FRAG_NORMAL. If set, there should be no special handling for fragments. No other config bits must be set.

Specification text
enum ofp_config_flags { /* Handling of IP fragments. */ OFPC_FRAG_NORMAL = 0, /* No special handling for fragments. */ 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.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.2; pg. 72)

Topology
One control plane connection

Results
Pass / Fail

Additional remarks
The result of this test may influence which other tests need to be run later. It may also be checked against the vendor-supplied documentation. If this bit is set, no other config bits must be set.

40.140 - Get switch config - Frag drop

Controller-to-Switch messages / Information gathering / Get switch configuration / Frag drop

Purpose
Check whether the switch is configured for drop fragments.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_SET_CONFIG_REQUEST message setting the OFPC_FRAG_DROP bit. If the device does not generate an error message, send an OFPT_GET_CONFIG_REQUEST message. Verify that the switch responds with an OFPT_GET_CONFIG_REPLY message. Check the ofp_config_flags for OFPC_FRAG_DROP. If the device generated an error message, OFPC_FRAG_DROP is not supported.

Specification text
OFPC_FRAG_DROP = 1 << 0, /* Drop fragments. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.2; pg. 72)

Topology
One control plane connection

Results
Pass / Fail

Additional remarks
The result of this test may influence which other tests need to be run later. It may also be checked against the vendor-supplied documentation.

40.150 - Get switch config - Frag reasm

Controller-to-Switch messages / Information gathering / Get switch configuration / Frag reasm

Purpose
Check whether the switch is configured for reassembling fragments.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_SET_CONFIG_REQUEST message setting the OFPC_FRAG_REASM bit. If the device does not generate an error message, send an OFPT_GET_CONFIG_REQUEST message. Verify that the switch responds with an OFPT_GET_CONFIG_REPLY message. Check the ofp_config_flags for OFPC_FRAG_REASM. If set, fragments should be reassembled. Verify that OFPC_IP_REASM is also set. If the device generated an error message, OFPC_FRAG_REASM is not supported.

Specification text
OFPC_FRAG_REASM = 1 << 1, /* Reassemble (only if OFPC_IP_REASM set). */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.2; pg. 72)*

Topology
One control plane connection

Results
Pass / Fail

Additional remarks
The result of this test may influence which other tests need to be run later. It may also be checked against the vendor-supplied documentation.

40.160 - Get switch config - Frag mask

Controller-to-Switch messages / Information gathering / Get switch configuration / Frag mask

Purpose
Check whether the switch is configured for Frag Mask.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPT_GET_CONFIG_REQUEST message. Verify that the switch responds with an OFPT_GET_CONFIG_REPLY message.

Check the ofp_config_flags for OFPC_FRAG_MASK (which represent the valid port config flags). The FRAG_MASK does not represent a flag field. Indeed, the ONF Extensibillity Working Group used this as a short way to state that only the two lower bits of this field have a valid meaning. All other bits should be ignored or set to zero. If the value of this field is OFPC_FRAG_MASK, the result of this test case shall be a failure.

Specification text
OFPC_FRAG_MASK = 3,
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.2; pg. 72)

Topology
One control plane connection

Results
Pass / Fail

40.170 - Manufacturer description

Controller-to-Switch messages / Information gathering / Description / Manufacturer description

Purpose
Verify that the switch reports the Manufacturer description.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPMP_DESC request message. Verify that the switch responds with an OFPMP_DESC reply message.

Check the mfr_desc field for a correct return value.

Specification text
Information about the switch manufacturer, hardware revision, software revision, serial number, and a description field is available from the OFPMP_DESC multipart request type: /* Body of reply to OFPMP_DESC request. Each entry is a NULL-terminated * ASCII string. */ struct ofp_desc { char mfr_desc[DESC_STR_LEN]; /* Manufacturer description. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.1; pg. 85)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
The return value should be mentioned by the vendor, for example in the documentation of the device.

40.180 - Hardware description

Controller-to-Switch messages / Information gathering / Description / Hardware description

Purpose
Verify that the switch reports the Hardware description.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPMP_DESC request message. Verify that the switch responds with an OFPMP_DESC reply message.

Check the hw_desc field for a correct return value.

Specification text
char hw_desc[DESC_STR_LEN]; /* Hardware description. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.1; pg. 85)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
The return value should be mentioned by the vendor, for example in the documentation of the device.

40.190 - Software description

Controller-to-Switch messages / Information gathering / Description / Software description

Purpose
Verify that the switch reports the Software description.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPMP_DESC request message. Verify that the switch responds with an OFPMP_DESC reply message.

Check the sw_desc field for a correct return value.

Specification text
char sw_desc[DESC_STR_LEN]; /* Software description. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.1; pg. 85)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
The return value should be mentioned by the vendor, for example in the documentation of the device.

40.200 - Serial Number

Controller-to-Switch messages / Information gathering / Description / Serial Number

Purpose
Verify that the switch reports the Serial Number.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPMP_DESC request message. Verify that the switch responds with an OFPMP_DESC reply message.

Check the serial_num field for a correct return value.

Specification text
char serial_num[SERIAL_NUM_LEN]; /* Serial number. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.1; pg. 85)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
The return value should be mentioned by the vendor, for example in the documentation of the device.

40.210 - Human readable datapath description of datapath

Controller-to-Switch messages / Information gathering / Description / Human readable datapath description of datapath

Purpose
Verify that the switch reports the Human readable datapath description of datapath.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an OFPMP_DESC request message. Verify that the switch responds with an OFPMP_DESC reply message.

Check the dp_desc field for a correct return value.

Specification text
char dp_desc[DESC_STR_LEN]; /* Human readable description of datapath. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.1; pg. 85)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
The description in the spec indicates this field might be configurable by the switch administrator. In this case it might make sense to set it to some defined value (max length) if the field is empty by default.

50 - Flow Table Miss

Test suite 50 verifies the basic behavior of packets, which fail to match against any standard flow table entry (non table miss entry).

Remarks

A table miss flow entry is defined as a fully wildcarded flow with priority zero.

Specification contradiction

The OpenFlow v1.3.4 specification states, “The table-miss flow entry MUST support at least sending packets to the controller using the CONTROLLER reserved port (see 4.5) and dropping packets using the Clear-Actions instruction (see 5.9).” (OpenFlow Switch Specification 1.3.4, ch. 5.4; pg. 19). However in chapter 5.9, the specification goes on to list Clear-Actions as an optional instruction. The seeming contradiction relates to packet pipeline termination.

Normally when a packet matches against a flow with no instructions, any actions that have been written to the packet’s action set are executed. If (when matching against a flow) a packet should be dropped, the clear actions instruction should be used to discard all actions that may have been written during pipeline processing. This ensures that no output or group actions are applied to the packet. We require this behavior of dropping a packet on table miss as it is an essential function for production environments.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

50.10 - Default behavior

Flow table miss / Table miss / Table miss / Table miss

Purpose
Verify that the switch drops unmatched packets if no table miss flow entry exists.

Methodology
Configure and connect DUT to controller. After control channel establishment make sure the switch has no flow entries. Send a packet to one data plane port, and verify that the packet gets dropped.

Specification text
If the table-miss flow entry does not exist, by default packets unmatched by flow entries are dropped (discarded).
- OpenFlow Switch Specification 1.3.4 (ch. 5.4; pg. 19)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

Test recommendations
We use two data plane connections to make sure there is no learning switch behavior.

50.20 - Packet in

Flow table miss / Table miss / Table miss / Table miss

Purpose
Verify that an entry with all wildcards, priority 0 and action send to the controller, can be created in all tables reported in the ofp_table_features reply.

Methodology
Configure and connect DUT to controller. After control channel establishment, add one flow entry with wildcard all, priority 0 and action output to the table under test. Verify that the flow can be added to the table under test. Send a packet to one data plane port, and verify that it gets sent to the controller.

Specification text
The table-miss flow entry is identified by its match and its priority (see 5.2), it wildcards all match fields (all fields omitted) and has the lowest priority (0). The match of the table-miss flow entry may fall outside the normal range of matches supported by a flow table, for example an exact match table would not support wildcards for other flow entries but must support the table-miss flow entry wildcarding all fields. The table-miss flow entry must support at least sending packets to the controller using the CONTROLLER reserved port (see 4.5) and dropping packets using the Clear-Actions instruction (see 5.9).
- OpenFlow Switch Specification 1.3.4 (ch. 5.4; pg. 19)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

50.30 - Packet in reason

Flow table miss / Table miss / Table miss / Table miss

Purpose
Verify that action output:CONTROLLER sets the reason field to table-miss.

Methodology
Configure and connect DUT to controller. After control channel establishment, add one flow entry with wildcard all, priority 0 and action output to the controller. Verify that the flow has been added. Send a packet to one data plane port, and verify that it gets sent to the controller. Verify that the reason field in the packet_in is OFPR_NO_MATCH table miss.

Specification text
If the table-miss flow entry directly sends packets to the controller using the CONTROLLER reserved port (see 4.5), the packet-in reason must identify a table-miss (see 7.4.1).
- OpenFlow Switch Specification 1.3.4 (ch. 5.4; pg. 19)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

Test recommendations
To verify that every table implements the table-miss correct, you may start entering flows that pass the packet to subsequent table numbers.

50.40 - Drop by clear actions

Flow table miss / Table miss / Table miss / Table miss

Purpose
Verify that using the Clear-Actions instruction drops the packet.

Methodology
Configure and connect DUT to controller. After control channel establishment, add one flow entry with wildcard all, priority 0 and action Clear Actions. Verify that the flow has been added. Send a packet to one data plane port, and verify that it gets dropped.

Specification text
The table-miss flow entry must support at least sending packets to the controller using the CONTROLLER reserved port (see 4.5) and dropping packets using the Clear-Actions instruction (see 5.9).
- OpenFlow Switch Specification 1.3.4 (ch. 5.4; pg. 19)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

Additional remarks
The Clear-actions instruction is now required based on Ext ticket https://rs.opennetworking.org/bugs/browse/EXT-509 for table-miss flow entries to allow for dropping of packets when using write_actions instructions.

50.60 - Entry timeout

Flow table miss / Table miss / Table miss / Table miss

Purpose
Verify that the table-miss entries timeout accordingly to their hard and idle timeouts.

Methodology
Configure and connect DUT to controller. After control channel establishment, enter a table miss flow with idle and hard timeout, and verify that the switch times this flow entry out as expected.

Specification text
The table-miss flow entry behaves in most ways like any other flow entry: it does not exist by default in a flow table, the controller may add it or remove it at any time (see 6.4), and it may expire (see 5.5).
- OpenFlow Switch Specification 1.3.4 (ch. 5.4; pg. 19)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

Test recommendations
One example setup:

First run, use hard timeout 15 sec, idle timeout 5 sec. Send packets to the data plane every second for 20 seconds. Verify that the flow times out after 15 seconds.

Second run, use hard timeout 15 sec, idle timeout 5 sec. Immediately send a single packet, wait 6 seconds, send a second packet and verify that the flow has timed out and the packet is no longer forwarded.

60 - Flow Table Matching

Test suite 60 verifies the device under test is able to match on the thirteen OXM types marked as required in table 11 of the OpenFlow v1.3 Specification.

Remarks

Masked OXM types

Masking for OXM types is not tested in test suite 60. Masking for IPv4 and IPv6 source and destination OXM types is tested in test suite 80.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

60.10 - Request the list of supported tables and matches per table.

Flow table matching / Matching / Query matches and tables / Query matches and tables

Purpose

Different tables may support different match fields. Here, we check that the 13 required match fields are each supported in at least one table. We also gather the information about which match groups are supported per table.

For the single-table test profile, all required match fields must be supported by the table under test.

Methodology

Configure and connect DUT to controller. If the device is not properly configured upon connection, use table configuration messages to set up the device for the correct test profile. Verify that the device reports support for all required match fields defined in table 11 of the v1.3.4 OpenFlow Specification, as required by the correct test profile.

Specification text

A switch must support the required match fields listed in Table 11 in its pipeline. Each required match field must be supported in at least one flow table of the switch: that flow table must enable matching that field and the match field prerequisites must be met in that table (see 7.2.3.6). The required fields don't need to be implemented in all flow tables, and don't need to be implemented in the same flow table. Flow tables can support non-required and experimenter match fields. The controller can query the switch about which match fields are supported in each flow table (see 7.3.5.5).
- OpenFlow Switch Specification (ch. 7.2.3.7; pg. 59)

Topology

One control plane connection

Results

Pass / Fail

60.20 - OXM_OF_IN_PORT: Ingress port. This may be a physical or switch-defined logical port.

Flow table matching / Matching / Single Header Field / OXM_OF_IN_PORT

Purpose
Verify that the switch is able to match on the OXM type named in this test case's title as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it and that a table-miss is triggered.

Specification text
OXM_OF_IN_PORT
Bits: 32 Mask: No
Prerequisite: None

Description: Ingress port. Numerical representation of incoming port, starting at 1. This may be a physical or switch-defined logical port.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.9; pg. 62)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

60.30 - OXM_OF_ETH_DST: Ethernet destination address. Can use arbitrary bitmask

Flow table matching / Matching / Single Header Field / OXM_OF_ETH_DST

Purpose
Verify that the switch is able to match on the OXM type named in this test case's title as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it and that a table-miss is triggered.

Specification text
OXM_OF_ETH_DST

Bits: 48

Mask: Yes

Prerequisite: None

Description: Ethernet destination MAC address.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.8; pg. 59)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

Test recommendations
This is testing the exact match on an Eth Dst address, masking is not tested here.

60.40 - OXM_OF_ETH_SRC: Ethernet source address. Can use arbitrary bitmask

Flow table matching / Matching / Single Header Field / OXM_OF_ETH_SRC

Purpose
Verify that the switch is able to match on the OXM type named in this test case's title as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it and that a table-miss is triggered.

Specification text
OXM_OF_ETH_SRC

Bits: 48

Mask: Yes

Prerequisite: None

Description: Ethernet source MAC address.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.8; pg. 59)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

Test recommendations
This is testing the exact match on an Eth Src address, masking is not tested here.

60.50 - OXM_OF_ETH_TYPE: Ethernet type of the OpenFlow packet payload, after VLAN tags.

Flow table matching / Matching / Single Header Field / OXM_OF_ETH_TYPE

Purpose
Verify that the switch is able to match on the OXM type named in this test case's title as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered.

Specification text
OXM_OF_ETH_TYPE

Bits: 16

Mask: No

Prerequisite: None

Description: Ethernet type of the OpenFlow packet payload, after VLAN tags.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.8; pg. 59)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

Test recommendations
We recommend testing this testcase with untagged, VLAN tagged and QinQ packets,and logging the results for reporting purposes.

60.60 - OXM_OF_IP_PROTO: IPv4 or IPv6 protocol number

Flow table matching / Matching / Single Header Field / OXM_OF_IP_PROTO

Purpose
Verify that the switch is able to match on the OXM type named in this test case's title as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered.

Specification text
OXM_OF_IP_PROTO

Bits: 8

Mask: No

Prerequisite: ETH TYPE=0x0800 or ETH TYPE=0x86dd

Description: IPv4 or IPv6 protocol number.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.8; pg. 59)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

60.70 - OXM_OF_IPV4_SRC: IPv4 source address.

Flow table matching / Matching / Single Header Field / OXM_OF_IPV4_SRC

Purpose
Verify that the switch is able to match on the OXM type named in this test case's title as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered.

Specification text
OXM_OF_IPV4_SRC

Bits: 32

Mask: Yes

Prerequisite: ETH TYPE=0x0800

Description: IPv4 source address. Can use subnet mask or arbitrary bitmask
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.8; pg. 59)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

Test recommendations
This is testing the exact match on an IPv4 address, masking is not tested here.

60.80 - OXM_OF_IPV4_DST: IPv4 destination address.

Flow table matching / Matching / Single Header Field / OXM_OF_IPV4_DST

Purpose
Verify that the switch is able to match on the OXM type named in this test case's title as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered.

Specification text
OXM_OF_IPV4_DST

Bits: 32

Mask: Yes

Prerequisite: ETH TYPE=0x0800

Description: IPv4 destination address. Can use subnet mask or arbitrary bitmask.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.8; pg. 59)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

Test recommendations
This is testing the exact match on an IPv4 address, masking is not tested here.

60.90 - OXM_OF_IPV6_SRC: IPv6 source address.

Flow table matching / Matching / Single Header Field / OXM_OF_IPV6_SRC

Purpose
Verify that the switch is able to match on the OXM type named in this test case's title as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered.

Specification text
OXM_OF_IPV6_SRC

Bits: 128

Mask: Yes

Prerequisite: ETH TYPE=0x86dd

Description: IPv6 source address. Can use subnet mask or arbitrary bitmask
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.8; pg. 59)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

Test recommendations
This is testing the exact match on an IPv6 address, masking is not tested here.

60.100 - OXM_OF_IPV6_DST: IPv6 destination address.

Flow table matching / Matching / Single Header Field / OXM_OF_IPV6_DST

Purpose
Verify that the switch is able to match on the OXM type named in this test case's title as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered.

Specification text
OXM_OF_IPV6_DST

Bits: 128

Mask: Yes

Prerequisite: ETH TYPE=0x86dd

Description: IPv6 destination address. Can use subnet mask or arbitrary bitmask
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.8; pg. 59)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

Test recommendations
This is testing the exact match on an IPv6 address, masking is not tested here.

60.110 - OXM_OF_TCP_SRC: TCP source port

Flow table matching / Matching / Single Header Field / OXM_OF_TCP_SRC

Purpose
Verify that the switch is able to match on the OXM type named in this test case's title as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered.

Specification text
OXM_OF_TCP_SRC

Bits: 16

Mask: No

Prerequisite: IP PROTO=6

Description: TCP source port
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.8; pg. 59)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

60.120 - OXM_OF_TCP_DST: TCP destination port

Flow table matching / Matching / Single Header Field / OXM_OF_TCP_DST

Purpose
Verify that the switch is able to match on the OXM type named in this test case's title as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered.

Specification text
OXM_OF_TCP_DST

Bits: 16

Mask: No

Prerequisite: IP PROTO=6

Description: TCP destination port
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.8; pg. 59)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

60.130 - OXM_OF_UDP_SRC: UDP source port

Flow table matching / Matching / Single Header Field / OXM_OF_UDP_SRC

Purpose
Verify that the switch is able to match on the OXM type named in this test case's title as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered.

Specification text
OXM_OF_UDP_SRC

Bits: 16

Mask: No

Prerequisite: IP PROTO=17

Description: UDP source port
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.8; pg. 59)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

60.140 - OXM_OF_UDP_DST: UDP destination port

Flow table matching / Matching / Single Header Field / OXM_OF_UDP_DST

Purpose
Verify that the switch is able to match on the OXM type named in this test case's title as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered.

Specification text
OXM_OF_UDP_DST

Bits: 16

Mask: No

Prerequisite: IP PROTO=17

Description: UDP destination port
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.8; pg. 59)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

80 - Flow Table Match Prerequisites

Test suite 80 verifies the behavior of OXM types and their corresponding prerequisites.

Remarks

Test case results will be based upon the prerequisite behavior presented by the device vendor. Devices that do not have prerequisite checking MUST successfully install a flow. Devices with prerequisite checking MUST throw errors for inconsistent OXM match types. Test suite 80 only verifies masking for IPv4 and IPv6 source and destination OXM types in CIDR format.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite except where result is ‘Not Applicable’, based on conditions as outlined in test case Remarks.

80.50 - Mask: OXM_OF_IPV4_SRC: IPv4 source address.

Flow table / Masking / Single Header Field / OXM_OF_IPV4_SRC

Purpose
Verify correct matching on masked match fields.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named masked field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a packet matching the masked flow range on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that it does not get forwarded by the flow, and that a table-miss is triggered.

Specification text
When oxm_hasmask is 1, the OXM TLV contains a bitmask and its length is effectively doubled, so oxm_length is always even. The bitmask follows the field value and is encoded in the same way. The masks are defined such that a 0 in a given bit position indicates a "don't care" match for the same bit in the corresponding field, whereas a 1 means match the bit exactly.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.5; pg. 56)*

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

Test recommendations
We recommend testing the following edge cases: full match (all 1's), full wildcard (all 0's) and a real masked value. In the basic profile only masks in CIDR notation are required to be supported.

80.60 - Mask: OXM_OF_IPV4_DST: IPv4 destination address

Flow table / Masking / Single Header Field / OXM_OF_IPV4_DST

Purpose
Verify correct matching on masked match fields.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named masked field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a packet matching the masked flow range on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that it does not get forwarded by the flow, and that a table-miss is triggered.

Specification text
When oxm_hasmask is 1, the OXM TLV contains a bitmask and its length is effectively doubled, so oxm_length is always even. The bitmask follows the field value and is encoded in the same way. The masks are defined such that a 0 in a given bit position indicates a "don't care" match for the same bit in the corresponding field, whereas a 1 means match the bit exactly.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.5; pg. 56)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

Test recommendations
We recommend testing the following edge cases: full match (all 1's), full wildcard (all 0's) and a real masked value. In the basic profile only masks in CIDR notation are required to be supported.

80.70 - Mask: OXM_OF_IPV6_SRC: IPv6 source address.

Flow table / Masking / Single Header Field / OXM_OF_IPV6_SRC

Purpose
Verify correct matching on masked match fields.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named masked field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a packet matching the masked flow range on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that it does not get forwarded by the flow, and that a table-miss is triggered.

Specification text
When oxm_hasmask is 1, the OXM TLV contains a bitmask and its length is effectively doubled, so oxm_length is always even. The bitmask follows the field value and is encoded in the same way. The masks are defined such that a 0 in a given bit position indicates a "don't care" match for the same bit in the corresponding field, whereas a 1 means match the bit exactly.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.5; pg. 49)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

Test recommendations
We recommend testing the following edge cases: full match (all 1's), full wildcard (all 0's) and a real masked value. In the basic profile only masks in CIDR notation are required to be supported.

80.80 - Mask: OXM_OF_IPV6_DST: IPv6 destination address

Flow table / Masking / Single Header Field / OXM_OF_IPV6_DST

Purpose
Verify correct matching on masked match fields.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named masked field (under the given Pre-requisites for the match), action is forwarding to an output port. Send a packet matching the masked flow range on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that it does not get forwarded by the flow, and that a table-miss is triggered.

Specification text
When oxm_hasmask is 1, the OXM TLV contains a bitmask and its length is effectively doubled, so oxm_length is always even. The bitmask follows the field value and is encoded in the same way. The masks are defined such that a 0 in a given bit position indicates a "don't care" match for the same bit in the corresponding field, whereas a 1 means match the bit exactly.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.5; pg. 56)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

Test recommendations
We recommend testing the following edge cases: full match (all 1's), full wildcard (all 0's) and a real masked value. In the basic profile only masks in CIDR notation are required to be supported.

80.180 - Missing prerequisite on Single Header field

Flow table match prerequisites / Prerequisites / Error / Prerequisites not met

Purpose
Verify device behavior with missing required pre-requisite field.

Methodology
Configure and connect DUT to controller. After control channel establishment, add flows with missing prerequisites according to table 7.2.3.8. Verify that the switch does not accept the flows, and returns the correct error message OFPET_BAD_MATCH and code OFPBMC_BAD_PREREQ for each instance.

Specification text
The presence of an OXM TLV with a given oxm_type may be restricted based on the presence or values of other OXM TLVs. In general, matching header fields of a protocol can only be done if the OpenFlow match explicitly matches the corresponding protocol. These restrictions are noted in specifications for individual fields (see 7.2.3.7). A switch may implement relaxed versions of these restrictions. For example, a switch may accept no prerequisite at all. A switch must reject an attempt to set up a flow entry that violates its restrictions (see 6.4), and must deal with inconsistent matches created by the lack of prerequisite (for example matching both a TCP source port and a UDP destination port).
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.6; pg. 57)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Test recommendations
If a switch implements some proprietary restrictions there might not be any existing default code for testing. In such a case, the test tool may manually configure the field set in this test.

Additional remarks
This tests for the default prerequisites as outlined in the specification. If the switch has fewer restrictions, only returning the error on its restrictions is considered a pass. If there aren’t any restrictions, this error might never be triggered and the result may be 'Not Applicable'.

80.190 - Pre-requisite field on wrong position in flow entry

Flow table match prerequisites / Prerequisites / Error / Bad order

Purpose
Verify device behavior when pre-requisite field is entered too late in a flow entry.

Methodology
Configure and connect DUT to controller. After control channel establishment, add flows with prerequisites according to table 7.2.3.8, but with the needed pre-requisite field entered after the respective match field. The match fields should make sense (ie 0x800 with ip_src). Verify that the switch does not accept the flows, and returns the correct error message OFPET_BAD_MATCH and code OFPBMC_BAD_PREREQ for each instance.

Specification text
An OXM TLV that has prerequisite restrictions must appear after the OXM TLVs for its prerequisites. Ordering of OXM TLVs within an OpenFlow match is not otherwise constrained.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.6; pg. 57)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
This tests for the default prerequisites as outlined in the specification. If the switch has fewer restrictions only returning the error on its restrictions is considered a pass. If there aren’t any restrictions, this error might never be triggered and the result may be 'Not Applicable'.

80.200 - Multiple instances of the same OXM_TYPE in a flow entry

Flow table match prerequisites / Prerequisites / Error / OXM_TYPE repeated

Purpose
Verify behavior when a flow entry repeats an OXM_TYPE.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow with a duplicated OXM_TYPE field. Verify that the switch does not accept the flow, and returns the correct error message OFPET_BAD_MATCH and code OFPBMC_DUP_FIELD.

Specification text
Any given oxm_type may appear in an OpenFlow match at most once, otherwise the switch must generate an error (see 6.4). A switch may implement a relaxed version of this rule and may allow in some cases a oxm_type to appear multiple time in an OpenFlow match, however the behavior of matching is then implementation-defined.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.6; pg. 57)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
This tests for the default prerequisites as outlined in the specification. If the switch has fewer restrictions only returning the error on its restrictions is considered a pass. If there aren’t any restrictions, this error might never be triggered and the result may be 'Not Applicable'.

90 - Flow Table Match Combinations

Test suite 90 verifies the behavior of various combinations of OXM types used in a single flow entry.

Remarks

Notes

All supported matches in an individual table (as reported by a device) MUST be available in parallel as long as they are not mutually exclusive. This test suite is responsible for testing each of these match combinations with as many OXM types as possible.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

90.60 - All supported

Flow table matching combinations / Matching / Multiple Header Fields / Mandatory

Purpose

This needs several subtests, as not all mandatory header match fields can exist in one flow -- Match on: OXM_OF_IN_PORT; OXM_OF_ETH_DST; OXM_OF_ETH_SRC; OXM_OF_ETH_TYPE==IPv4; OXM_OF_IP_PROTO; OXM_OF_IPV4_SRC; OXM_OF_IPV4_DST; OXM_OF_TCP_SRC; OXM_OF_TCP_DST

Methodology

Configure and connect DUT to controller. After control channel establishment, install four separate flows, each with an output action to a data plane port.

Flow1 matches on	Flow2 matches on	Flow3 matches on	Flow4 matches on
OXM_OF_IN_PORT	OXM_OF_IN_PORT	OXM_OF_IN_PORT	OXM_OF_IN_PORT
OXM_OF_ETH_DST	OXM_OF_ETH_DST	OXM_OF_ETH_DST	OXM_OF_ETH_DST
OXM_OF_ETH_SRC	OXM_OF_ETH_SRC	OXM_OF_ETH_SRC	OXM_OF_ETH_SRC
OXM_OF_ETH_TYPE	OXM_OF_ETH_TYPE	OXM_OF_ETH_TYPE	OXM_OF_ETH_TYPE
OXM_OF_IP_PROTO	OXM_OF_IP_PROTO	OXM_OF_IP_PROTO	OXM_OF_IP_PROTO
OXM_OF_IPV4_SRC	OXM_OF_IPV4_SRC	OXM_OF_**IPV6**_SRC	OXM_OF_IPV6_SRC
OXM_OF_IPV4_DST	OXM_OF_IPV4_DST	OXM_OF_**IPV6**_DST	OXM_OF_IPV6_DST
OXM_OF_TCP_SRC	OXM_OF_**UDP**_SRC	OXM_OF_TCP_SRC	OXM_**UDP**_SRC
OXM_OF_TCP_DST	OXM_OF_**UDP**_DST	OXM_OF_TCP_DST	OXM_**UDP**_DST

Generate matching packets for each flow, and verify that packets are forwarded to the correct data plane ports.

Specification text

Flow tables can support non-required and experimenter match fields.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.7; pg. 58)

Topology

One control plane connection and two dataplane connections

Results

Pass / Fail

Test recommendations

In the one table case, all mandatory and all supported optional matches must be available in parallel, as long as they are not mutually exclusive. In this test case, we create and verify several match combinations, trying to exercise every field in combination with as many other fields as possible. For the mandatory case, the flows might be the following: IPv4-UDP, IPv4-TCP, IPv6-UDP, IPv6-TCP.

100 - Flow Table Actions

Test suite 100 verifies that all actions a device MUST support are correctly implemented. The output action is the only action type required for Basic Single Table conformance. Basic Single Table conformance requires the output action to work with the following reserved ports; OFPP_IN_PORT, OFPP_ALL, OFPP_TABLE (for packet out messages only), and OFPP_CONTROLLER.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

100.10 - Drop

Flow table actions / Actions / Mandatory / Output

Purpose
Verify that a packet matching a flow with no associated output action gets dropped.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match), make sure there is no associated-action in this flow. Send a matching packet on the data plane. Verify that the packet is dropped.

Specification text
Required Action: Drop. There is no explicit action to represent drops. Instead, packets whose action sets have no output actions should be dropped. This result could come from empty instruction sets or empty action buckets in the processing pipeline, or after executing a Clear-Actions instruction.
- OpenFlow Switch Specification 1.3.4 (ch. 5.12; pg. 27)

Topology
One control plane connection and one dataplane connection

Results
Pass / Fail

Test recommendations
It might make sense to verify this behavior in the action set and in groups.

100.20 - Single Port

Flow table actions / Actions / Mandatory / Output

Purpose
Verify that a packet matching a flow with an associated single port output action gets forwarded.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match), action is output to a single specific port. Send a matching packet on the data plane. Verify that the packet is forwarded only to this specific port.

Specification text
Required Action: Output. The Output action forwards a packet to a specified OpenFlow port (see 4.1). OpenFlow switches must support forwarding to physical ports, switch-defined logical ports and the required reserved ports (see 4.5).
- OpenFlow Switch Specification 1.3.4 (ch. 5.12; pg. 26)

Topology
One control plane connection and two dataplane connections

Results
Pass / Fail

100.30 - Multiple Action with Output to multiple ports

Flow table actions / Actions / Mandatory / Output

Purpose
Verify that a packet matching a flow with multiple associated single port output actions gets forwarded.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match), action is output to three specific ports. Send a matching packet on the data plane. Verify that the packet is forwarded only to the specified ports.

Specification text
Required Action: Output. The Output action forwards a packet to a specified OpenFlow port (see 4.1). OpenFlow switches must support forwarding to physical ports, switch-defined logical ports and the required reserved ports (see 4.5).
- OpenFlow Switch Specification 1.3.4 (ch. 5.12; pg. 26)

Topology
One control plane connection and five data plane connections

Results
Pass / Fail

Test recommendations
One ingress port, three output ports, one additional port to verify that the packet is not broadcast to all ports. If 5 data plane ports are not available, the in_port may be used for one of the three output ports.

100.40 - Single Action with Output to multiple ports

Flow table actions / Actions / Mandatory / Output

Purpose

Verify that a packet matching a flow with multiple associated output actions using reserved ports is forwarded to all listed ports.

Methodology

Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match), with action output to OFPP_ALL, action output to OFPP_IN_PORT, and action output to OFPP_CONTROLLER. Send a matching packet on the data plane. Verify that the packet is forwarded to all ports including the original ingress port.

Specification text

Required Action: Output. The Output action forwards a packet to a specified OpenFlow port (see 4.1). OpenFlow switches must support forwarding to physical ports, switch-defined logical ports and the required reserved ports (see 4.5).
- OpenFlow Switch Specification 1.3.4 (ch. 5.12; pg. 26)

Topology

One control plane connection and three data plane connections

Results

Pass / Fail

100.50 - ALL

Flow table actions / Actions / Mandatory / Output

Purpose
Verify that a packet matching a flow with an associated output:ALL action gets forwarded to all ports except the ingress port.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is output to port ALL. Send a matching packet on the data plane. Verify that the packet is forwarded to all ports except the ingress port.

Specification text
Required: ALL: Represents all ports the switch can use for forwarding a specific packet. Can be used only as an output port. In that case a copy of the packet is sent to all standard ports, excluding the packet ingress port and ports that are configured OFPPC_NO_FWD.
- *OpenFlow Switch Specification 1.3.4 (ch. 4.5; pg. 13)*

Topology
One control plane connection and four data plane connections

Results
Pass / Fail

Test recommendations
This test case is not testing the OFPPC_NO_FWD. Do not configure ports as OFPPC_NO_FWD.

100.60 - All excludes OFPPC_NO_FWD

Flow table actions / Actions / Mandatory / Output

Purpose
Verify that a packet matching a flow with an associated output:ALL action gets forwarded to all ports except the ingress port and except ports configured for OFPPC_NO_FWD.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is output to port ALL. Send OFPT_PORT_MOD message to make certain port(s) configured for OFPPC_NO_FWD. Send a matching packet on the data plane. Verify that the packet is forwarded to all ports except the ingress port and ports configured for OFPPC_NO_FWD.

Specification text
Required: ALL: Represents all ports the switch can use for forwarding a specific packet. Can be used only as an output port. In that case a copy of the packet is sent to all standard ports, excluding the packet ingress port and ports that are configured OFPPC_NO_FWD.
- *OpenFlow Switch Specification 1.3.4 (ch. 4.5; pg. 13)*

Topology
One control plane connection and four data plane connections

Results
Pass / Fail

100.70 - Output to Controller

Flow table actions / Actions / Mandatory / Output

Purpose
Verify that a packet matching a flow with an associated output:controller action generates a packet_in to the controller.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is output to port OFPP_CONTROLLER. Send a matching packet on the data plane. Verify that a packet_in message encapsulates the matching packet that is triggered.

Specification text
Required: CONTROLLER: Represents the control channel with the OpenFlow controller. Can be used as an ingress port or as an output port. When used as an output port, encapsulate the packet in a packet-in message and send it using the OpenFlow protocol (see 7.4.1). When used as an ingress port, this identifies a packet originating from the controller.
- *OpenFlow Switch Specification 1.3.4 (ch. 4.5; pg. 13)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

100.80 - Table

Flow table actions / Actions / Mandatory / Output

Purpose
Verify that a packet_out with output:table gets submitted to the flow table.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is output to a specific port. Generate a matching packet and send it via packet_out message with output action to port TABLE in its action list. Verify that the packet gets forwarded to the specific port.

Specification text
Required: TABLE: Represents the start of the OpenFlow pipeline (see 5.1). This port is only valid in an output action in the action list of a packet-out message (see 7.3.7), and submits the packet to the first flow table so that the packet can be processed through the regular OpenFlow pipeline.
- OpenFlow Switch Specification 1.3.4 (ch. 4.5; pg. 13)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

100.90 - IN_PORT

Flow table actions / Actions / Mandatory / Output

Purpose
Verify that a packet with output:IN_PORT action is sent out through its ingress port.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is output to port IN_PORT. Send a matching packet on the data plane via a certain ingress port. Verify that the packet gets forwarded to the ingress port.

Specification text
Required: IN PORT: Represents the packet ingress port. Can be used only as an output port, send the packet out through its ingress port.
- OpenFlow Switch Specification 1.3.4 (ch. 4.5; pg. 13)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

130 - Flow Table Action Set

Test suite 130 verifies the correct behavior of the action set. Because most action set tests require more than a single table to properly execute, a majority of test suite 130 is excluded from the Basic Single Table conformance requirements.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass 130.220. If a device only supports the minimal output action type, then test case 130.250 cannot be tested, and its result MAY be *'Not Applicable'*.

130.220 - Action Set Output

Flow table action set / Action Set / Output

Purpose
Default behavior in case no group action is specified.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match), Write-Action instruction with output action to a data port. Send a matching packet on the data plane. Verify that the packet is forwarded only to this specific port.

Specification text
if no group action is specified, forward the packet on the port specified by the output action
- OpenFlow Switch Specification 1.3.4 (ch. 5.10; pg. 26)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

130.250 - Action Set Order

Flow table action set / Action Set / Action Order

Purpose
Order in which action is applied for action sets.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match), the flow populates the action set with at least two supported actions in inverse order of table 5.10. Verify that the switch executes actions according to table 5.10 regardless of the order they were added. If the device only supports the output action, the result of this test may be considered a pass.

Specification text
The actions in an action set are applied in the order specified below, regardless of the order that they were added to the set.
- OpenFlow Switch Specification 1.3.4 (ch. 5.10; pg. 25)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail / Not Applicable

Additional remarks
If a device only supports the output action type, this case is unable to yield a positive or negative result and the result is 'Not Applicable'.

140 - Flow Table Modifications

Test suite 140 verifies the behavior of flow modification messages with a focus on testing overlapping flows, flow flags, and flow commands.

Basic Single Table Conformance Test Profile Requirements

To satisfy Basic Single Table conformance, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

140.10 - Add with overlap check - overlapping

Flow table modifications / Modification messages / Add / Check_Overlap

Purpose
Verify how the "OFPFC_ADD" is processed while "OFPFF_CHECK_OVERLAP" flag is set and how error reporting is handled.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match). Add a second flow with an overlapping match, OFPFF_CHECK_OVERLAP flag set, and a different priority. Verify that the flow gets installed in the flow table. Install a third flow with the OFPFF_CHECK_OVERLAP flag set, with a match overlapping that of flow 1, and a priority set equal to flow 1. Verify the corresponding error type and code. (OFPET_FLOW_MOD_FAILED type OFPFMFC_OVERLAP code) message is triggered and the flow is not added to the flow table.

Specification text
For add requests (OFPFC_ADD) with the OFPFF_CHECK_OVERLAP Flag set, the switch must first check for any overlapping flow entries in the requested table. 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.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)

Topology
One control plane connection

Results
Pass / Fail

140.20 - Add with no overlap

Flow table modifications / Modification messages / Add / No Overlap

Purpose
Verify how the "OFPFC_ADD" is processed while "OFPFF_CHECK_OVERLAP" flag is NOT set.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match). Add a second flow with a non-overlapping match, the OFPFF_CHECK_OVERLAP flag set, and the same priority as flow one. Verify that the flow gets installed in the flow table. Install a third flow with the OFPFF_CHECK_OVERLAP flag not set, with a match overlapping that of flow 1, and a priority set equal to flow 1. Verify that all three flows are installed in the flow table.

Specification text
For non-overlapping add requests, or those with no overlap checking, the switch must insert the flow entry in the requested table.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
This test case uses overlapping flows, not identical flows.

Additional remarks
The behavior of overlapping flows is undefined, and may be unique to each OpenFlow implementation.

140.30 - Add - Identical flows

Flow table modifications / Modification messages / Add / Identical flows

Purpose

Verify how the "OFPFC_ADD" is processed while "OFPFF_CHECK_OVERLAP" flag is set for identical flow entries in the table.

Methodology

Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match). Wait a set period of time. Add a second flow with an identical match, the OFPFF_CHECK_OVERLAP flag not set, the same priority as flow one, but a different cookie value from flow one. Verify that flow one has been removed, that flow two is installed, the second flow's cookie field is set correctly, and the flow's duration counter has reset.

Specification text

If a flow entry with identical match fields and priority already resides in the requested table, then that entry, including its duration, must be cleared from the table, and the new flow entry added.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)

Topology

One control plane connection and two data plane connections

Results

Pass / Fail

140.40 - Add with Reset Counters Flag Set

Flow table modifications / Modification messages / Add / OFPFF_RESET_COUNTS

Purpose
Verify that the counters are cleared when "OFPFC_ADD" is processed and "OFPFF_CHECK_OVERLAP" and "OFPFF_RESET_COUNTS" flags are set.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match). Wait a set period of time. Create some data plane traffic matching the flow so the counters are increased. Add a second flow with an identical match, the OFPFF_CHECK_OVERLAP flag not set, the OFPFF_RESET_COUNTS flag set, and the same priority as flow one, but a different cookie value from flow one. Verify that flow one has been removed, that flow two is installed, the second flow's cookie field is set correctly, and all counters have been reset.

Specification text
If the OFPFF_RESET_COUNTS flag is set, the flow entry counters must be cleared,
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

140.60 - Add generates no flow removed message

Flow table modifications / Modification messages / Add / No Flow removed message

Purpose
Verify the flow-removed msg status when "OFPFC_ADD" is processed and "OFPFF_CHECK_OVERLAP" flag is set.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match), with the flow-removed flag set. Add a second flow with an identical match, the OFPFF_CHECK_OVERLAP flag not set, and the same priority as flow one, but a different cookie value from flow one. Verify that flow one has been removed, that flow two is installed, and the second flow's cookie field is set correctly. Verify that no flow removed message was generated.

Specification text
No flow-removed message is generated for the flow entry eliminated as part of an add request; if the controller wants a flow-removed message it should explicitly send a delete request for the old flow entry prior to adding the new one.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)

Topology
One control plane connection

Results
Pass / Fail

140.70 - Modify - Preserved fields

Flow table modifications / Modification messages / Modify / preserved fields

Purpose

For ofp_flow_mod messages using a modify or modify_strict command, verify that the instruction field of all matching flows is updated. In addition, verify that cookies, timeouts, flags, counters, and durations are not modified.

Methodology

Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match). Modify this flow's instructions. Verify that its cookie, idle_timeout, hard_timeout, flags, counters and duration fields are left unchanged.

Specification text

For modify requests (OFPFC_MODIFY or OFPFC_MODIFY_STRICT), if a matching entry exists in the table, the instructions field of this entry is updated with the value from the request, whereas its cookie, idle_timeout, hard_timeout, flags, counters and duration fields are left unchanged.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)

Topology

One control plane connection

Results

Pass / Fail

140.80 - Modify with Reset Counters Flag Set

Flow table modifications / Modification messages / Modify / OFPFF_RESET_COUNTS

Purpose
Verify that the counters are cleared for "OFP_FLOW_MOD" with "OFPFF_RESET_COUNTS" flag set.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match). Send data plane traffic matching the installed flow. Modify this flow's instructions with the OFPFF_RESET_COUNTS flag set in the flow mod command. Verify that its cookie, idle_timeout, hard_timeout, flags, and duration fields are left unchanged. Verify that its counter fields are cleared.

Specification text
If the OFPFF_RESET_COUNTS flag is set, the flow entry counters must be cleared.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

140.90 - Modify non existent flow

Flow table modifications / Modification messages / Modify / Modify non existent flow

Purpose
Check error handling and table modification for "OFP_FLOW_MOD" with no table match.

Methodology
Configure and connect DUT to controller. After control channel establishment, send OFPFC_MODIFY request for a flow not present in the table with matching field (under given Pre-requisites for match) and action as output port X. Verify that no errors are received. Send a packet matching this flow, verify that the packet is dropped by the switch. The packet should be dropped since neither table-miss flow nor modify request was configured on switch.

Specification text
For modify requests, if no flow entry currently residing in the requested table matches the request, no error is recorded, and no flow table modification occurs.
- *OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)*

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

140.100 - Default delete

Flow table modifications / Modification messages / Delete / Default delete

Purpose
Check that a flow is deleted for "OFPFC_DELETE" or "_STRICT".

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the Pre-requisites for the match) with action as output port X. Send a packet for matching field and verify that the packet is received on port X. Send an OFPFC_DELETE request for previous flow. Send the same packet as earlier and verify that the packet is not received by output port and dropped by the switch.

Specification text
For delete requests (OFPFC_DELETE or OFPFC_DELETE_STRICT), if a matching entry exists in the table, it must be deleted,
- *OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)*

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

140.110 - Delete with flow removed flag set

Flow table modifications / Modification messages / Delete / Delete with flow removed flag set

Purpose

Check that a flow is deleted while "OFPFF_SEND_FLOW_REM" flag is set, and verify that a flow removed message is generated.

Methodology

Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the Pre-requisites for the match) with OFPFF_SEND_FLOW_REM flag set and with action as output port X. Send a packet for matching field and verify that the packet is received on port X. Send an OFPFC_DELETE request for previous flow. Send the same packet as earlier and verify that the packet is not received by output port and dropped by the switch. Verify that the controller receives the flow removed message.

Specification text

and if the entry has the OFPFF_SEND_flow_REM flag set, it should generate a flow removed message.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)

Topology

One control plane connection and two data plane connections

Results

Pass / Fail

140.120 - Delete non existing entry

Flow table modifications / Modification messages / Delete / Delete non existing entry

Purpose
Check error handling and table modification when "OFPFC_DELETE" sent for no matching flow.

Methodology
Configure and connect DUT to controller. After control channel establishment, send OFPMP_TABLE request. Calculate the active entries in the table. Now send OFPFC_DELETE request for a flow not present in the table with matching field (under given Pre-requisites for match). Verify that no errors are received. Send OFPMP_TABLE request. Verify that the active entries are same as before the delete request.

Specification text
For delete requests, if no flow entry currently residing in the requested table matches the request, no error is recorded, and no flow table modification occurs.
- *OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)*

Topology
One control plane connection

Results
Pass / Fail

140.130 - Priority strict / non-strict

Flow table modifications / Modification messages / Strict / Priority strict / non-strict

Purpose
Verify how modify and delete strict vs non-strict command is processed.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the Pre-requisites for the match) with action as output port X and priority p1. Add another flow with same match as previous flow with action as output port Y and priority p2. Verify that flows are installed. Send an OFPFC_DELETE_STRICT request for flow 1 with appropriate match and priority p1. Verify that only flow 1 is deleted and that flow 2 remains in the switch table.

Specification text
Modify and delete flow mod commands have non-strict versions (OFPFC_MODIFY and OFPFC_DELETE) and strict versions (OFPFC_MODIFY_STRICT or OFPFC_DELETE_STRICT). In the strict versions, the set of match fields, all match fields, including their masks, and the priority, are strictly matched against the entry, and only an identical flow entry is modified or removed.
- *OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)*

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

140.140 - Strict / non-strict delete checks

Flow table modifications / Modification messages / Strict / Strict / non-strict delete checks

Purpose

For ofp_flow_mod messages using a delete or delete_strict command, verify that the DUT removes all matching flows according to the behaviors defined for strict and non-strict.

Methodology

Configure and connect DUT to controller. After control channel establishment, add first flow flow1 matching on a named field (under the Pre-requisites for the match) with action as output port X with priority 100. Add second flow flow2 matching on a named field same as flow1 with action as output port Y with priority 200. Add third flow flow3 matching on a named field same as flow1 with action as output port Z with priority 300. Send a packet matching to above flow and verify it is received out of port Z. Send an OFPFC_DELETE_STRICT flow_mod request with the same match fields as before and priority 300. Send a packet matching above flow1 and verify that it is now received on port Y instead of Z. Send OFPFC_DELETE flow_mod request with no match fields. Send a packet matching above flow1 and verify that the packet is dropped (table-miss) at switch.

Specification text

For example, if a message to remove entries is sent that has no match fields included, the OFPFC_DELETE command would delete all flow entries from the tables, while the OFPFC_DELETE_STRICT command would only delete a flow entry that applies to all packets at the specified priority. For non-strict modify and delete commands, all flow entries that match the flow mod description are modified or removed.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)

Topology

One control plane connection and two data plane connections

Results

Pass / Fail

Test recommendations

At least one field should be wildcarded

140.150 - non-strict delete multiple matches

Flow table modifications / Modification messages / Strict / non-strict delete multiple matches

Purpose
Verify the behavior of "non-strict" version for exact or more specific match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add first flow flow1 matching a named field A (under the Pre-requisites for the match) with specific value, dst_port = 80 and with action as output port X with priority 100. Add second flow with same matching field as above but with ANY (wildcard) value, dst_port=80 and with action as output port Y with priority 200. Send a packet that matches above flow and verify that the packet is received on port Y. Add a third flow with matching src_port=80, and with action as output port Z. Now, send an OFPFC_DELETE flow_mod request with match fields as ANY for field A and 80 for dst_port. Verify that the first two flows are removed from flow table. Send a packet matching first flow and verify that the packet is dropped on switch. Verify that the third flow is still installed.

Specification text
In the non-strict versions, a match will occur when a flow entry exactly matches or is more specific than the description in the flow mod command; in the flow mod the missing match fields are wildcarded, field masks are active, and other flow mod fields such as priority are ignored.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 40)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

140.170 - Delete - match syntax

Flow table modifications / Modification messages / Flow stats / Match syntax

Purpose
Verify the behavior for modified tables for AGGREGATE FLOWS.

Methodology
Configure and connect DUT to controller. After control channel establishment, install three flows. Flow1 matching on hw_src=a with a priority of 100. Flow2 matching on hw_src=a,hw_dst=b with a priority of 200. Flow3 matching on ANY with a priority of 300. Send an ofp_multipart_request with type OFPMP_AGGREGATE that matches on hw_src=a. Verify that aggregate statistics are received for Flows 1 and 2. Send ofp_multipart_request with type OFPMP_AGGREGATE that matches on priority=300. Verify that aggregate statistics are received for Flow3.

Specification text
This same interpretation of mixed wildcard and exact match fields also applies to individual and aggregate flows stats requests.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)

Topology
One control plane connection

Results
Pass / Fail

140.180 - Delete - Filters-1

Flow table modifications / Modification messages / Delete / Filters

Purpose
Verify the OFPFC_DELETE command functionality when filtered by destination group or OUT_PORT.

Methodology
Configure and connect DUT to controller. After control channel establishment, add first flow flow1 matching a named field (under the Pre-requisites for the match) with priority 100 and actions as output port X. Add second flow flow2 with same matching fields but with priority 200 and actions as output port Y. Send a packet matching above flows and verify that the packet is received from port Y. Send an OFPFC_DELETE flow_mod request with above match fields and out_port field as Y. Verify that flow2 is removed from the flow table. Send a packet again and now verify that the packet is received from port X.

Specification text
Delete commands can be optionally filtered by destination group or output port. If the out_port field contains a value other than OFPP_ANY, it introduces a constraint when matching. This constraint is that each matching flow entry must contain an output action directed at the specified port in the actions associated with that flow entry. This constraint is limited to only the actions directly associated with the flow entry.
- *OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)*

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

140.200 - Add and modify ignore filters

Flow table modifications / Modification messages / Non Delete / Filters ignored

Purpose
Verify that the OUT_PORT of OFPT_FLOW_MOD is ignored by OFPFC_ADD, OFPFC_MODIFY and OFPFC_MODIFY_STRICT.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match) with actions as output port X. Using a flow mod with a modify command, modify this flow's instructions with ofp_flow_mod.out_port=Y and instructions as output set to Z. Verify that flow is modified by sending a packet matching the flow and packet received from port Z.

Specification text
These fields are ignored by OFPFC_ADD, OFPFC_MODIFY and OFPFC_MODIFY_STRICT messages.
- *OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)*

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

140.210 - Delete Cookie

Flow table modifications / Modification messages / Delete / Cookie

Purpose
OFPFC_DELETE and OFPFC_MODIFY commands can use cookie for filtering.

Methodology
Configure and connect DUT to controller. After control channel establishment, add first flow flow1 matching on a named field (under the Pre-requisites for the match) with action as output port X, priority 100, and cookie 0x1. Add second flow flow2 matching on a named field same as flow1 with action as output port Y, priority 200 and cookie 0x5. Add third flow flow3 matching on a named field same as flow1 with action as output port Z, priority 300, and cookie 0x6. Send a packet matching the above flow and verify that the packet is received out of port Z. Send an OFPFC_DELETE flow_mod request with cookie of 0x4 and cookie_mask of 0x4. Verify that flow2 and flow3 are removed. Send a packet matching above flow1 and verify that the packet is received from port X.

Specification text
Modify and delete commands can also be filtered by cookie value, if the cookie_mask field contains a value other than 0. This constraint is that the bits specified by the cookie_mask in both the cookie field of the flow mod and a flow entry's cookie value must be equal. In other words, (flow entry:cookie & flow mod:cookie mask) == (flow mod:cookie & flow mod:cookie mask).
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

140.220 - Delete in all tables

Flow table modifications / Modification messages / Delete / All tables

Purpose
OFPFC_DELETE command for "OFPTT_ALL" addresses all tables.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow into the table under test, matching a named field (under the Pre-requisites for the match) with priority 100 and actions as output port X. Send an OFPFC_DELETE flow_mod request matching all the flows and OFPTT_ALL as table_id. Verify that the flow is removed from all tables under test.

Specification text
Delete commands can use the OFPTT_ALL value for table-id to indicate that matching flow entries are to be deleted from all flow tables.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
Code should be prepared to deal with single and multiple table pipelines.

150 - Flow Table Errors

Test suite 150 verifies the correct implementation of error messages associated with flow modification messages.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite except where result is 'Not Applicable', based on conditions as outlined in test case Remarks.

Exceptions

If a device supports all instructions types defined in chapter 5.9 of the OpenFlow v1.3 Specification, test case 150.50 cannot be tested, and the recorded test result MAY be *'Not Applicable'*.

If a device only exposes a single table the error message in test case 150.60 cannot be triggered, and the recorded test result MAY be *'Not Applicable'*.

If a device supports all possible table ids the test result of 150.10, and 150.60 MAY be *'Not Applicable'*.

If a device supports arbitrary bitmasks then the error message in test cases 150.110, 150.120 and 150. 130 cannot be triggered. In this case the result is considered and the recorded test result shall be *'Not Applicable'*.

If the specified error for test cases 150.70, 150.110, 150.120, 150.130, 150.150, 150.160, 150.170, 150.180 cannot be reliably triggered, the result of the test case MAY be 'Not Applicable'.

150.10 - Error: Invalid table

Flow table / Modification messages / Errors / Invalid Table

Purpose
Verify how "FLOW_MOD" with invalid TABLE-ID is handled.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow with matching on named field. Send OFPFC_DELETE flow_mod message for this flow with invalid table-id. Verify that the switch sends the ofp_error_msg with OFPET_FLOW_MOD_FAILED type and OFPFMFC_BAD_TABLE_ID code. Verify that the flow remains installed.

Specification text
If the flow modification message specifies an invalid table-id, the switch must send an ofp_error_msg with OFPET_flow_MOD_FAILED type and OFPFMFC_BAD_TABLE_ID code.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If a device supports all possible table ids the test result of this test shall be 'Not Applicable'.

150.20 - Error: modify with table-id OFPT-ALL

Flow table / Modification messages / Errors / Modify all tables

Purpose
Verify how "FLOW_MOD" with "OFPTT_ALL" in add or modify request is handled.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow with table-id OFPTT_ALL. Verify that the switch sends the ofp_error_msg with OFPET_flow_MOD_FAILED type and OFPFMFC_BAD_TABLE_ID code. Verify that the flow was not added to the flow tables. Add at least one flow to the flow table, and send a matching modify command with table-id OFPTT_ALL changing the action. Verify that the switch sends the ofp_error_msg with OFPET_flow_MOD_FAILED type and OFPFMFC_BAD_TABLE_ID code. Verify that the flow did not get modified.

Specification text
If the flow modification message specifies OFPTT_ALL for table-id in a add or modify request, the switch must send the same error message.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)

Topology
One control plane connection

Results
Pass / Fail

150.30 - Error: Table full

Flow table / Modification messages / Errors / Table full

Purpose
Verify how "OFPFC_ADD" is handled if table has no space.

Methodology
Configure and connect DUT to controller. After control channel establishment, keep sending OFPFC_ADD flow_mod messages until no further flow can be added due to lack of space. Now, send another OFPFC_ADD request, verify that the switch sends ofp_error_msg with OFPET_flow_MOD_FAILED type and OFPFMFC_TABLE_FULL code. Verify that the flow was not added to the flow tables.

Specification text
If a switch cannot find any space in the requested table in which to add the incoming flow entry, the switch must send an ofp_error_msg with OFPET_flow_MOD_FAILED type and OFPFMFC_TABLE_FULL code.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
On some devices, this error may take an extremely long time (greater than half an hour) to be triggered. It is acceptable for a device to impose a configured hard limit in order to verify this error message is properly triggered. If a hard limit is not configurable on such a device, the result of this test MAY be recorded as 'Not Applicable'.

150.40 - Error: unknown instruction

Flow table / Modification messages / Errors / Unknown instruction

Purpose
Verify how unknown instructions in "FLOW_MOD" are handled.

Methodology
Configure and connect DUT to controller. After control channel establishment, send OFPFC_ADD flow_mod message with unknown instruction. Verify that the switch sends ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_UNKNOWN_INST code. Verify that the flow was not added to the flow tables.

Specification text
If the instructions requested in a flow mod message are unknown, the switch must return an ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_UNKNOWN_INST code.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)

Topology
One control plane connection

Results
Pass / Fail

150.50 - Error: unsupported instructions

Flow table / Modification messages / Errors / Unsupported instructions

Purpose
Verify how unsupported instructions in "FLOW_MOD" are handled.

Methodology
Configure and connect DUT to controller. Refer to switch documentation for switch capabilities for supported instructions. After control channel establishment, send OFPFC_ADD flow_mod message with unsupported instruction. Verify that the switch sends ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_UNSUP_INST code.

Specification text
If the instructions requested in a flow mod message are unsupported, the switch must return an ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_UNSUP_INST code.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If a device supports all instructions types defined in chapter 5.9 of the OpenFlow v1.3 Specification, this test case cannot be tested, and the recorded test result shall be 'Not Applicable'.

150.60 - Error: Goto Invalid Table

Flow table / Modification messages / Errors / Goto Invalid Table

Purpose
Verify how invalid table is handled in Goto-Table and next-table-id.

Methodology
Configure and connect DUT to controller. After control channel establishment, send OFPFC_ADD flow_mod message with actions as Goto-Table with invalid value. Verify that the switch sends ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_BAD_TABLE_ID code.

Specification text
If the instructions requested contain a Goto-Table and the next-table-id refers to an invalid table the switch must return an ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_BAD_TABLE_ID code.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If a device only exposes a single table, the error message in this test case cannot be triggered, and the recorded test result shall be not applicable.

If a device supports all possible table ids the test result of this test shall be 'Not Applicable'.

150.70 - Error: Unsupported Meta Data

Flow table / Modification messages / Errors / Unsupported metadata

Purpose
Verify how unsupported metadata value or mask values are handled in Write-Metadata.

Methodology
Configure and connect DUT to controller. After control channel establishment, send OFPFC_ADD flow_mod message with a write-metadata instruction with unsupported write-metadata or metadata mask. If the device does not support the write-metadata instruction verify that the switch sends an OFPET_BAD_INSTRUCTION error with an OFPBIC_UNSUP_INST code. Otherwise verify that the switch sends ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_UNSUP_METADATA or OFPBIC_UNSUP_METADATA_MASK code.

Specification text
If the instructions requested contain a Write-Metadata and the metadata value or metadata mask value is unsupported, then the switch must return an ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_UNSUP_METADATA or OFPBIC_UNSUP_METADATA_MASK code.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If a device only exposes a single table, the error message in this test case may not be relevant, and the recorded test result shall be Not Applicable'.

150.80 - Error: Bad Match Field

Flow table / Modification messages / Errors / Bad Match Field

Purpose
Verify how OXM_TVL with unsupported value in FLOW_MOD is handled.

Methodology
Configure and connect DUT to controller. After control channel establishment, define an oxm TLV header including the basic openflow match class, an undefined oxm_field value (x: x > 39), the oxm_mask bit unset, a match value length of 1 byte, and a data payload of size 1. Create and install an ofp_flow_mod matching on the previously defined OXM type with an output action to controller. Verify that an error is returned with error type OFPET_BAD_MATCH and error code OFPBMC_BAD_FIELD.

Specification text
If the match in a flow mod message specifies a field or a class that is unsupported in the table, the switch must return an ofp_error_msg with OFPET_BAD_MATCH type and OFPBMC_BAD_FIELD code.
- *OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)*

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
Suggest testing an additional predefined oxm_field value (x: x <39) that is not supported by the DUT.

150.85 - Error: Bad Match Class

Flow table / Modification messages / Errors / Bad Match Class

Purpose
Verify how OXM_TVL with unsupported class in FLOW_MOD is handled.

Methodology
Configure and connect DUT to controller. After control channel establishment, define an oxm TLV header, set the openflow match class to an undefined value, the oxm_field to a defined value, the oxm_mask bit unset, a match value length of 1 byte, and a data payload of size 1. Create and install an ofp_flow_mod matching on the previously defined OXM type with an output action to controller. Verify that an error is returned with error type OFPET_BAD_MATCH and error code OFPBMC_BAD_FIELD.

Specification text
If the match in a flow mod message specifies a field or a class that is unsupported in the table, the switch must return an ofp_error_msg with OFPET_BAD_MATCH type and OFPBMC_BAD_FIELD code.
- *OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 41)*

Topology
One control plane connection

Results
Pass / Fail

150.110 - Error: Bad Network Mask

Flow table / Modification messages / Errors / Bad Network or Datalink Mask

Purpose
Verify how OFP_FLOW_MOD handles an arbitrary not supported mask in Layer 2 OR 3.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install an ofp_flow_mod matching on a masked eth_src address with a mask value of "00:00:00:ff:ff:ff". Verify that an error is returned with error type OFPET_BAD_MATCH and error code OFPBMC_BAD_DL_ADDR_MASK. Create and install a second ofp_flow_mod matching on a masked ipv4_src address with a mask value of "0.0.255.255". Verify that an error is returned with error type OFPET_BAD_MATCH and error code OFPBMC_BAD_NW_ADDR_MASK.

Specification text
If the match in a flow mod specifies an arbitrary bitmask for either the data link or network addresses which the switch cannot support, the switch must return an ofp_error_msg with OFPET_BAD_MATCH type and either OFPBMC_BAD_DL_ADDR_MASK or OFPBMC_BAD_NW_ADDR_MASK.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 42)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If the DUT supports arbitrary bitmasks then the error is not triggerable. In this case the result is considered as 'Not Applicable'.

150.120 - Error: ND and DL Mask Wrong

Flow table / Modification messages / Errors / Mask for NW and DL wrong

Purpose
Verify how OFP_FLOW_MOD handles an arbitrary not supported mask in Layer 2 AND 3.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install an ofp_flow_mod matching on a masked eth_src address with a mask value of "00:00:00:ff:ff:ff", and matching on a masked ipv4_src address with a mask value of "0.0.255.255". Verify that an error is returned with error type OFPET_BAD_MATCH and error code OFPBMC_BAD_DL_ADDR_MASK.

Specification text
If the bit masks specified in both the data link and network addresses are not supported then OFPBMC_BAD_DL_ADDR_MASK should be used.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 42)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If the DUT supports arbitrary bitmasks then the error is not triggerable. In this case the result is considered as 'Not Applicable'.

150.130 - Error: Unsupported Mask

Flow table / Modification messages / Errors / Unsupported Mask

Purpose
Verify how OFP_FLOW_MOD handles an arbitrary mask for the fields that don't support it.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install an ofp_flow_mod matching on a masked match type that is not supported. Verify that an error is returned with error type OFPET_BAD_MATCH and error code OFPBMC_BAD_MASK.

Specification text
If the match in a flow mod specifies an arbitrary bitmask for another field which the switch cannot support, the switch must return an ofp_error_msg with OFPET_BAD_MATCH type and OFPBMC_BAD_MASK code.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 42)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Test recommendations
This testcase is applicable to all devices that do not support arbitrary bitmask on every supported field except eth_src, eth_dst or ipv4_src_ipv4_dst. The testtool needs to be set specifically to a combination of field / mask not supported by the DUT under tests. If a device allows arbitrary bitmasks on all fields it supports matching on except eth_src, eth_dst or ipv4_src_ipv4_dst, this testcase should be considered passed / NA.

Additional remarks
If the DUT supports arbitrary bitmasks then the error is not triggerable. In this case the result is considered as 'Not Applicable'.

150.140 - Error: Illegal Value

Flow table / Modification messages / Errors / Illegal Value

Purpose
Verify that the DUT handles OFP_FLOW_MOD with values that can't be matched correctly.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install an ofp_flow_mod matching on VLAN ID 0. Check if an error is received, verify that an error is returned with error type OFPET_BAD_MATCH and error code OFPBMC_BAD_VALUE. If not, create and install a second ofp_flow_mod matching on VLAN ID 4095. Verify that an error is returned with error type OFPET_BAD_MATCH and error code OFPBMC_BAD_VALUE.

Specification text
If the match in a flow mod specifies values that cannot be matched, for example, a VLAN ID greater than 4095 and not one of the reserved values, or a DSCP value using more than 6 bits, the switch must return an ofp_error_msg with OFPET_BAD_MATCH type and OFPBMC_BAD_VALUE code.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 42)

Topology
One control plane connection

Results
Pass / Fail

150.145 - Error: Bad Type

Flow table / Modification messages / Errors / Unsupported Action

Purpose
Verify how OFP_FLOW_MOD handles unsupported actions.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match) with an output action that is not supported in the table. Verify that an error is returned with error type OFPET_BAD_ACTION type and OFPBAC_BAD_TYPE code.

Specification text
If an instruction in a flow mod message specifies an action that is not supported in the table, the switch must return an ofp_error_msg with OFPET_BAD_ACTION type and OFPBAC_BAD_TYPE code
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 42)

Topology
One control plane connection

Results
Pass / Fail

150.150 - Error: Never Valid Port

Flow table / Modification messages / Errors / Never Valid Port

Purpose
Verify how OFP_FLOW_MOD handles invalid port in output action.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match) with an output action to OFPP_ANY. Verify that an error is returned with error type OFPET_BAD_ACTION and error code OFPBAC_BAD_OUT_PORT.

Specification text
If any action 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.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 42)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If the DUT supports all possible port numbers that can be generated by the test tool, then this error message may not be triggerable and the result may be 'Not Applicable'.

150.160 - Error: Currently Invalid Port

Flow table / Modification messages / Errors / Currently Invalid Port

Purpose
Verify how OFP_FLOW_MOD handles ports that might be valid in the future.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match) with an output action to a port number that could potentially be supported in the future (this information may be provided by the device vendor). If an error is returned, verify that the received error type is OFPET_BAD_ACTION and error code is OFPBAC_BAD_OUT_PORT. Otherwise verify that the flow has been installed, and that any matching traffic is not forwarded.

Specification text
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 must either silently drop packets sent to the referenced port, or immediately return an OFPBAC_BAD_OUT_PORT error and refuse the flow mod.
- *OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 42)*

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If the DUT supports all possible port numbers that can be generated by the test tool, then this error message may not be triggerable and the result may be 'Not Applicable'.

150.170 - Error: Undefined Group

Flow table / Modification messages / Errors / Undefined Group

Purpose
Verify that how OFP_FLOW_MOD handles a non existent group.

Methodology
Configure and connect DUT to controller. After control channel establishment, verify DUT supports groups, if so, add a flow matching on a named field (under the given Pre-requisites for the match) with a group action to group_id OFPG_ALL. Verify that an error is returned, and that the received error type is OFPET_BAD_ACTION and error code is OFPBAC_BAD_OUT_GROUP.

Specification text
If an action in a flow mod message references a group that is not currently defined on the switch, or is a reserved group, such as OFPG_ALL, the switch must return an ofp_error_msg with OFPET_BAD_ACTION type and OFPBAC_BAD_OUT_GROUP code.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 42)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If a device does not support groups, then the result of this test case may be 'Not Applicable'.

150.175 - Error: Undefined Meter

Flow table / Modification messages / Errors / Undefined Meter

Purpose
Verify how OFP_FLOW_MOD handles an undefined meter.

Methodology
Configure and connect DUT to controller. After control channel establishment, check if the OFPIT_METER is supported by the table under test. Send a flow mod message for a meter that has not been defined. If the OFPIT_METER is supported, verify that an error is returned, and that the received error type is OFPET_METER_MOD_FAILED type and OFPMMFC_UNKNOWN_METER code. Otherwise verify that an error is returned, and that the received error type is OFPET_BAD_INSTRUCTION type and OFPBIC_UNSUP_INST code.

Specification text
If an action in a flow mod message references a meter that is not currently defined on the switch, the switch must return an ofp_error_msg with OFPET_METER_MOD_FAILED type and OFPMMFC_UNKNOWN_METER code.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 42)

Topology
One control plane connection

Results
Pass / Fail

150.180 - Error: Bad Set Argument

Flow table / Modification messages / Errors / Bad Action

Purpose
Verify how OFP_FLOW_MOD handles an invalid set-field.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a VLAN. Send a flow mod with a value (x: x> 4095) or a DSCP value using more than 6 bits. Verify that an error is returned, and that the received error type is OFPET_BAD_ACTION type and OFPBAC_BAD_SET_ARGUMENT code.

Specification text
If a set-field action in a flow mod message has a value that is invalid, for example a set-field action for OXM VLAN_VID with value greater than 4095, or a DSCP value using more than 6 bits, the switch must return an ofp_error_msg with OFPET_BAD_ACTION type and OFPBAC_BAD_SET_ARGUMENT code.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 42)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If a device does not support the set-field-action, then the result of this test case may be 'Not Applicable'.

150.190 - Error: Bad Argument

Flow table / Modification messages / Errors / Bad Action

Purpose
Verify how OFP_FLOW_MOD handles an invalid value.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match) with a push_vlan action with an invalid ethertype. Verify that an error is returned, and that the received error type is OFPET_BAD_ACTION and error code is OFPBAC_BAD_ARGUMENT.

Specification text
If an action in a flow mod message has a value that is invalid, for example a Push action with an invalid Ethertype, and this situation is not covered by other error codes (such as bad output port, bad group id or bad set argument), the switch must return an ofp_error_msg with OFPET_BAD_ACTION type and OFPBAC_BAD_ARGUMENT code.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 42)

Topology
One control plane connection

Results
Pass / Fail

150.260 - Error: Bad instruction

Flow table / Modification messages / Errors / Clear Action with Actions

Purpose
Verify how OFP_FLOW_MOD handles Clear action with some actions.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a table miss entry with a clear actions instruction with an output action. Verify that an error of type OFPET_BAD_ISTRUCTION is generated by the device with an error code of OFPBIC_BAD_LEN.

Specification text
If a Clear-Actions instruction contains some actions, the switch must return an ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_BAD_LEN code.
- OpenFlow Switch Specification 1.3.4 (ch. 6.4; pg. 43)

Topology
One control plane connection

Results
Pass / Fail

180 - Counters

Test suite 180 verifies the behavior of counters marked as required in table 5 of the v1.3 OpenFlow Specification.

Remarks

Logical ports

As with physical ports, it is required that unsupported counters have a reported value of -1. For mandatory port counters, documentation describing the counter mapping from physical ports to logical ports MUST be provided by the device vendor. Logical port counters and their correct values MUST be manually verified by the tester if the test tool is unable to perform the required operations.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite except where result is ‘Not Applicable’ based on conditions as outlined in test case Remarks.

180.10 - Reference Count (active entries)

Counters / Per Flow Table / Reference Count (active entries) 32 Required

Purpose
Test Reference Count of active entries counter.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given pre-requisites for the match) with an output action to a data plane test port. Generate N matching and M non-matching packets on the data plane. Send an ofp_multipart_request of type OFPMP_TABLE. From the response verify active_count==1, lookup_count==N+M, and matched_count==N.

Specification text
Reference Count (active entries) 32 Required
- OpenFlow Switch Specification 1.3.4 (ch. 5.8; pg. 23)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

180.60 - Per Flow Duration (seconds) Counter

Counters / Per Flow Entry / Duration (seconds) 32 Required

Purpose
Test Flow Duration counter.

Methodology
Configure and connect DUT to controller. After control channel establishment, send a multipart_request for flow_stats. Verify that the switch sends a multipart_reply for flow_stats. Record the value in the duration field. Wait 5 seconds. Send another multipart_request for flow_stats. Verify that the switch sends a multipart_reply for flow_stats. Record the value in the duration field. Verify that the duration field has increased by 5.

Specification text
Duration (seconds) 32 Required
- OpenFlow Switch Specification 1.3.4 (ch. 5.8; pg. 23)

Topology
One control plane connection

Results
Pass / Fail

180.100 - Per Port Duration (seconds)

Counters / Per Port / Duration (seconds) 32 Required

Purpose
Test Port Duration counter.

Methodology
340,170

Specification text
Duration (seconds) 32 Required
- OpenFlow Switch Specification 1.3.4 (ch. 5.8; pg. 23)

Topology
One control plane connection

Results
Pass / Fail

180.230 - Correct packet drop counters

Counters / Packet counters / Packet counters

Purpose
Counters_packet dropped_Port_down.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given pre-requisites for the match) with an output action to a data plane test port which is administratively down. Send N matching data plane packets to the switch. Verify that the flow's packet counter is incremented correctly.

Specification text
Packet related counters for an OpenFlow object must count every packet using that object, even if the object is having no effect on the packet or if the packet is ultimately dropped or sent to the controller. For example, the switch should maintain the packet related counters of the following: a port, which is down - *OpenFlow Switch Specification 1.3.4 (ch. 5.8; pg. 23)*

Topology
One control plane connection

Results
Pass / Fail

180.410 - Duration Precision

Counters / Global

Purpose
Test Duration is always metered in second precision.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match). Request the duration counter every 5 seconds for 120 seconds, and verify that the returned value is correct with second precision (expected duration +-.5 sec).

Specification text
Duration refers to the amount of time the flow entry, a port, a group, a queue or a meter has been installed in the switch, and must be tracked with second precision.
- OpenFlow Switch Specification 1.3.4 (ch. 5.8; pg. 23)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
The test tool should take into account the approximate flow installation time.

180.430 - Counter Wrap Around

Counters / Global

Purpose
Test Duration counters wrap around and have no overflow indicator.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match). Request the duration_ns counter every .5 seconds for 6 seconds. Verify that the counter wraps around after the expected time has passed (around 4 seconds).

Specification text
Counters are unsigned and wrap around with no overflow indicator.
- OpenFlow Switch Specification 1.3.4 (ch. 5.8; pg. 23)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
This can only be checked reliably using the duration_ns counter, as getting 64 byte counters to wrap takes too long. If the switch does not support this optional counter, the test may be considered Not Applicable.

200 - Protocol Messages

Test suite 200 verifies the basic behavior of each OpenFlow message type.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass all test cases in this test suite.

200.30 - Basic OFPT_ECHO_REQUEST / OFPT_ECHO_REPLY

Protocol Messages / Symmetric message / Immutable messages / OFPT_ECHO_REQUEST / OFPT_ECHO_REPLY

Purpose
Verify the correct response to ECHO.

Methodology
Send an echo request and verify the echo reply.

Specification text
OFPT_ECHO_REQUEST = 2, and OFPT_ECHO_REPLY = 3, /* Symmetric message */
- OpenFlow Switch Specification 1.3.4 (ch. 7.1; pg. 47)

Topology
One control plane connection

Results
Pass / Fail

200.70 - Basic OFPT_PORT_STATUS

Protocol Messages / Async message / Asynchronous messages / OFPT_PORT_STATUS

Purpose
Verify that the Port Status is forwarded correctly by the switch.

Methodology
Change the port status through a method outside the openflow protocol (cli, disconnect cable), and verify that the switch sends the correct port status message to the controller, informing about the state change.

Specification text
OFPT_PORT_STATUS = 12, /* Async message */
- OpenFlow Switch Specification 1.3.4 (ch. 7.1; pg. 47)

Topology
One control plane connection

Results
Pass / Fail

200.100 - Basic OFPT_SET_CONFIG

Protocol Messages / Controller/switch message / Switch configuration messages / OFPT_SET_CONFIG

Purpose
Verify that MISS_SEND_LEN is set correctly.

Methodology
Set miss send length to value x, and verify that the value was set using a get configuration request.

Specification text
OFPT_SET_CONFIG = 9, /* Controller/switch message */
- OpenFlow Switch Specification 1.3.4 (ch. 7.1; pg. 47)

Topology
One control plane connection

Results
Pass / Fail

200.110 - Basic OFPT_PACKET_OUT

Protocol Messages / Controller/switch message / Controller command messages / OFPT_PACKET_OUT

Purpose
Verify that packets sent via packet_out are received.

Methodology
Send packet via packet_out, verify received packet.

Specification text
OFPT_PACKET_OUT = 13, /* Controller/switch message */
- OpenFlow Switch Specification 1.3.4 (ch. 7.1; pg. 47)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

Test recommendations
Important features to check for are: in_port values (controller, …), encapsulated packets (vlan, mpls), buffered packet, invalid buffer ID. To table is tested in 100.80.

200.130 - Basic OFPT_GROUP_MOD

Protocol Messages / Controller/switch message / Controller command messages / OFPT_GROUP_MOD

Purpose
Verify that group processes message correctly.

Methodology
Try to create a group, verify that the message gets processed as expected. If device does not support groups, verify that OFPET_BAD_REQUEST error is received with code OFPBRC_BAD_TYPE.

Specification text
OFPT_GROUP_MOD = 15, /* Controller/switch message */
- OpenFlow Switch Specification 1.3.4 (ch. 7.1; pg. 47)

Topology
One control plane connection

Results
Pass / Fail

200.140 - Basic OFPT_PORT_MOD

Protocol Messages / Controller/switch message / Controller command messages / OFPT_PORT_MOD

Purpose
Verify that port status msg is received in response to OFPT_PORT_MOD.

Methodology
Bring one of the data plane ports down.

Specification text
OFPT_PORT_MOD = 16, /* Controller/switch message */
- OpenFlow Switch Specification 1.3.4 (ch. 7.1; pg. 47)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

200.150 - Basic OFPT_TABLE_MOD

Protocol Messages / Controller/switch message / Controller command messages / OFPT_TABLE_MOD

Purpose
Verify that table modification can recognize lower 2 bits and returns error for others.

Methodology
Send an ofp_table_mod with a config bitmap not equal to ofptc_deprecated_mask (3). Verify that OFPET_TABLE_MOD_FAILED error is received with error code OFPTMFC_BAD_CONFIG.

Specification text
OFPT_TABLE_MOD = 17, /* Controller/switch message */
- OpenFlow Switch Specification 1.3.4 (ch. 7.1; pg. 47)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
Because the ofp_table_mod message is deprecated, it is acceptable to return an ofp_error with OFPET_BAD_REQUEST and OFPBRC_BAD_CONFIG, OFPET_BAD_REQUEST and OFPBRC_BAD_TYPE, or OFPET_TABLE_MOD_FAILED and OFPTFMFC_EPERM when the message is not supported. Otherwise the error message defined in the methodology is expected.

200.170 - Basic OFPT_BARRIER_REQUEST / OFPT_BARRIER_REPLY

Protocol Messages / Controller/switch message / Barrier messages / OFPT_BARRIER_REQUEST / OFPT_BARRIER_REPLY

Purpose

Check switch replies to controller with Barrier Reply following all commands execution after Barrier Request received.

Methodology

Add the x flows to flow table (where x=10,000 or maximum number of flows supported by switch) with flag set for OFPFF_SEND_FLOW_REM. Send a flow mod message to delete all the flows and then immediately send a barrier request. The controller should receive the barrier response after receiving flow_removed message for all the flows.

Specification text

OFPT_BARRIER_REQUEST = 20, OFPT_BARRIER_REPLY = 21, /* Controller/switch message */
- OpenFlow Switch Specification 1.3.4 (ch. 7.1; pg. 47)

Topology

One control plane connection

Results

Pass / Fail

Test recommendations

It would be better to also include messages that generate replies, for example flow removed, echo, or error messages. It is recommended to define some command groups to test, and use the pilot program to verify their effectiveness during the pilot program. Please use this field to write down suggested command sets.

200.210 - Basic OFPT_GET_ASYNC_REQUEST / REPLY

Protocol Messages / Controller/switch message / Asynchronous message configuration / OFPT_GET_ASYNC_REQUEST / REPLY

Purpose

Verify that the DUT responds with an OFPT_GET_ASYNC_REPLY when it receives an OFPT_GET_ASYNC_REQUEST.

Methodology

Send an OFPT_GET_ASYNC_REQUEST request and verify that the switch either replies with OFPT_GET_ASYNC_REPLY or OFPT_ERROR of type OFPET_BAD_REQUEST and code OFPBRC_BAD_TYPE.

Specification text

OFPT_GET_ASYNC_REQUEST = 26, OFPT_GET_ASYNC_REPLY = 27,/* Controller/switch message */
- OpenFlow Switch Specification 1.3.4 (ch. 7.1; pg. 47)

Topology

One control plane connection

Results

Pass / Fail

200.230 - Basic OFPT_SET_ASYNC

Protocol Messages / Controller/switch message / Asynchronous message configuration / OFPT_SET_ASYNC

Purpose
Verify that the controller is able to receive asynchronous messages from switch.

Methodology
Send set async config message, verify processing or error. See also 20.150.

Specification text
OFPT_SET_ASYNC = 28, /* Controller/switch message */
- OpenFlow Switch Specification 1.3.4 (ch. 7.1; pg. 47)

Topology
One control plane connection

Results
Pass / Fail

200.250 - Reserved_Value_error

Protocol Messages / Controller/switch message / Reserved values messages

Purpose
Verify that the switch rejects requests containing a reserved value.

Methodology
Send a request to the DUT containing a reserved value or optional value it does not support, verify that the request is rejected and an appropriate error message is returned.

Specification text
If an OpenFlow implementation receives a request containing a reserved value or an optional value it does not support, it must reject the request and return an appropriate error message.
- OpenFlow Switch Specification 1.3.4 (ch. 7.1.3; pg. 48)

Topology
One control plane connection

Results
Pass / Fail

200.270 - Reserved_bit_position_error

Protocol Messages / Controller/switch message / Reserved bit position

Purpose
Verify that the switch rejects requests containing a reserved bit position.

Methodology
Send a request to the DUT containing a reserved bit position or an optional bit position it does not support set to 1, verify that the request is rejected and an appropriate error message is returned.

Specification text
If an OpenFlow implementation receives a request containing a reserved bit position or an optional bit position it does not support set to 1, it must reject the request and return an appropriate error message.
- OpenFlow Switch Specification 1.3.4 (ch. 7.1.3; pg. 48)

Topology
One control plane connection

Results
Pass / Fail

200.290 - Reserved_TLV_error

Protocol Messages / Controller/switch message / Reserved values messages

Purpose
Verify that the switch rejects requests containing a reserved TLV.

Methodology
Send a request to the DUT containing a reserved TLV type or an optional TLV type it does not support, verify that the request is rejected and an appropriate error message is returned.

Specification text
If an OpenFlow implementation receives a request containing a reserved TLV type or an optional TLV type it does not support, it must reject the request and return an appropriate error message.
- OpenFlow Switch Specification 1.3.4 (ch. 7.1.3; pg. 49)

Topology
One control plane connection

Results
Pass / Fail

210 - Port Structure Protocol Message

Test suite 210 verifies the ofp_port structure's various fields.

Remarks

Purpose of configuration fields

There are four distinct port configuration fields contained in the ofp_port structure. "Curr" represents the currently active configuration between the DUT and the other end of the active link. "Advertised" represents the configuration options configured to be advertised during auto negotiation. "Supported" represents all possible configurations the DUT may be configured to advertise. And "Peer" represents the advertised configuration options of the other end of the active link.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass 210.50. All other test cases are duplicates, which have been implicitly tested in other test suites and thus removed.

The report MUST indicate all port types and configurations tested for all tests where a specific port configuration or state is tested or verified.

210.50 - Port administratively down

Protocol Messages / Port Structures struct ofp_port / uint32_t config / OFPPC_PORT_DOWN

Purpose
Verify that a port status change message is received, and that the bitmap reflects the change in the port config.

Methodology
Configure and connect DUT to controller. After control channel establishment, install a table_miss flow entry to generate ofp_packet_in messages. Send an ofp_port_mod message that sets the all configuration bits to zero except OFPPC_PORT_DOWN, for a data plane port X. Verify that the port config bits are correctly set. Send traffic on data plane port X. Verify that no ofp_packet_in message is received. Send an ofp_packet_out message with an output action to port X. Verify that no traffic is forwarded.

Specification text
The OFPPC_PORT_DOWN bit indicates that the port has been administratively brought down and should not be used by OpenFlow.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.2.1; pg. 50)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

230 - Action Header Protocol Message

Test suite 230 verifies the ofp_action_header structure's various fields.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

230.40 - MAX_LEN of 0 is empty packet

Protocol Messages / Action header ofp_action_header / ofp_action_output / OFPP_CONTROLLER

Purpose
Verify that packet "send to controller" action with MAX_LEN set to 0 sends 0 bytes of the packet.

Methodology
Insert flow with output controller, max_len set to 0. Verify that 0 bytes of the packet are included in the packet_in message. If device does not support buffers, verify that the whole data plane message is received and the buffer id is -1.

Specification text
A ’max_len’ of zero means no bytes of the packet should be sent.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.5; pg. 67)

Topology
One control plane connection

Results
Pass / Fail

230.50 - OFPCML_MAX = 0xffe5 smaller packets sent entirely

Protocol Messages / Action header ofp_action_header / ofp_action_output / OFPP_CONTROLLER

Purpose

For packets that are smaller than OFPCML_MAX = 0xffe5, verify that they are sent to the controller in their entirety using the "send to controller" action.

Methodology

Insert flow with output controller, max_len set to OFPCML_MAX = 0xffe5. Verify that packets less than that length are contained fully in the packet_in message. Repeat the above methodology with a max_len value of 1000, 100, and 10. Verify the correctness of each. If the device does not support buffers, verify that the entire data plane message is received and the buffer id is -1.

Specification text

OFPCML_MAX = 0xffe5, maximum max_len value which can be used to request a specific byte length
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.5; pg. 67)

Topology

One control plane connection

Results

Pass / Fail

230.60 - OFPCML_NO_BUFFER = 0xffff packets are sent entirely

Protocol Messages / Action header ofp_action_header / ofp_action_output / OFPP_CONTROLLER

Purpose

With MAX_LEN of OFPCML_NO_BUFFER set to 0xffff, verify that packets are sent to the controller in their entirety using the "send to controller" action.

Methodology

Insert flow with output controller, max_len set to OFPCML_NO_BUFFER = 0xffff. Verify that packets are not buffered, and that the full packet is included in the packet_in.

Specification text

A ’max_len’ of OFPCML_NO_BUFFER = 0xffff means that the packet is not buffered and the complete packet is to be sent to the controller.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.5; pg. 67)

Topology

One control plane connection

Results

Pass / Fail

Test recommendations

This means to also check for a correct buffer_id of 0xFFFFFFFF

240 - Switch Features Protocol Message

Test suite 240 verifies the ofp_switch_features structure's various fields.

Remarks

Correct values

Test cases 40.10 through 40.40, 40.120, and 40.170 through 40.210 verify values reported by the device are correct. These tests require information to be provided by the vendor for correct verification.

Basic Single Table Conformance Test Profile Requirements

All test cases in Suite 240 are duplicates which have been implicitly tested in other test suites, and thus removed.

250 - Switch Config Protocol Message

Test suite 250 verifies the ofp_switch_config structure's various fields.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass test case 250.70 and 250.140 in this test suite except where result is 'Not Applicable' based on conditions as outlined in test case Remarks. All other test cases are duplicates which have been implicitly tested in other test suites, and thus removed.

250.70 - Max bytes of packet that data path should send to the controller. See ofp_controller_max_len for valid values.

Protocol Messages / GET ofp_switch_config / uint16_t miss_send_len

Purpose
Verify that OFP_CONTROLLER_MAX_LEN value is the max bytes for packets sent to the controller.

Methodology
Configure the DUT's default table-miss behavior to trigger an ofp_packet_in message.

Connect DUT to controller. After control channel establishment, send an OFPT_GET_CONFIG_REQUEST and verify that the correct default value of 128 bytes is returned. Send a packet to the data plane that is longer than 128 bytes and verify that an ofp_packet_in message is triggered and that it only contains the first 128 bytes of the packet.

Specification text
Max bytes of packet that data path should send to the controller. See ofp_controller_max_len for valid values.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.2; pg. 73)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If it is not possible to configure the DUT's default table-miss behavior to trigger an ofp_packet_in message, the result of this test is 'Not Applicable'.

250.140 - MISS_SEND_LEN specifies size of OFP_PACKET_IN

Protocol Messages / SET ofp_switch_config / uint16_t miss_send_len

Purpose
Verify the size of data in OFP_PACKET_IN message specified by MISS_SEND_LEN when action output is not OFFP_CONTROLLER.

Methodology
Configure the DUT's default table-miss behavior to trigger an ofp_packet_in message.

Connect DUT to controller. Send an OFPT_SET_CONFIG command to the switch, setting the miss_send_len value to 0. Send a packet to the data plane, and verify that the OFP_PACKET_IN does not contain any data. Set the miss_send_len to 0xffe5 (65509). Send a packet with 1500 bytes (the maximum Ethernet MTU), and verify that the full packet is encapsulated in the data field of the OFP_PACKET_IN. Set the miss_send_len to 0xffff (65535), and verify that the packet is not buffered and completely encapsulated in the OFP_PACKET_IN.

Specification text
The miss_send_len field defines the number of bytes of each packet sent to the controller by the OpenFlow pipeline when not using an output action to the OFPP_CONTROLLER logical port, for example sending packets with invalid TTL if this message reason is enabled. If this field equals 0, the switch must send zero bytes of the packet in the ofp_packet_in message.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.2; pg. 73)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If it is not possible to configure the DUT's default table-miss behavior to trigger an ofp_packet_in message, the result of this test is 'Not Applicable'.

260 - Flow Mod Protocol Message

Test suite 260 verifies the ofp_flow_mod structure's various fields. Of specific interest are overlapping flow entries, flow removed messages, strict / non strict flow modifications and flow deletions, and additional constraints on modifications and flow deletions (output, and cookie).

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

260.40 - uint64_t cookie; /* Opaque controller-issued identifier. */

Protocol Messages / OFPT_FLOW_MOD / uint64_t cookie

Purpose
Verify matching on a flow's cookie field using an OFPT_FLOW_MOD with a cookie_mask value of -1.

Methodology
Configure and connect DUT to controller. After control channel establishment, add two flows with cookie A and one flow with cookie B. Send another flow deleting cookie A. Verify that both flows get deleted, but the flow with cookie B stays.

Specification text
uint64_t cookie; /* Opaque controller-issued identifier. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 74)

Topology
One control plane connection

Results
Pass / Fail

260.50 - uint64_t cookie_mask;

Protocol Messages / OFPT_FLOW_MOD / uint64_t cookie_mask

Purpose
Verify matching on a flow's cookie field using an OFPT_FLOW_MOD with a cookie_mask value not equal to -1.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install three flows with three different cookies. The first two cookies should differ from each other in the first byte. The third cookie should differ from the first two by the third byte. All other bytes in each cookie should be equal. Create an ofp_flow_mod with a delete command that has a cookie field equal to the first cookie, and a cookie_mask field that masks the third byte. Verify that the first two flows are deleted when this ofp_flow_mod is sent to the DUT.

Specification text
uint64_t cookie_mask; /* Mask used to restrict the cookie bits that must match when the command is OFPFC_MODIFY* or OFPFC_DELETE*. A value of 0 indicates no restriction.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 74)*

Topology
One control plane connection

Results
Pass / Fail

260.60 - Flow mod cookie mask statistics

Protocol Messages / OFPT_FLOW_MOD / uint64_t cookie

Purpose
Verify that flow statistics can be filtered by cookie and cookie mask values.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install three flows with three different cookies. The first two cookies should differ from each other in the first byte. The third cookie should differ from the first two by the third byte. All other bytes in each cookie should be equal. Send a MULTIPART_REQUEST message for flow statistics that has a cookie field equal to the first cookie, and a cookie_mask field that masks the third byte. Verify that the information gathered corresponds to the first two flows when this ofp_flow_mod is sent to the DUT.

Specification text
The cookie field is an opaque data value chosen by the controller. This value appears in messages and flow statistics, and can also be used to filter flow statistics, flow modification and flow deletion (see 6.4). It is not used by the packet processing pipeline and thus doesn't have to reside in hardware.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 75)

Topology
One control plane connection

Results
Pass / Fail

260.70 - Flow mod cookie query

Protocol Messages / OFPT_FLOW_MOD / uint64_t cookie

Purpose
Verify that cookie values are set and reported correctly in ofp_flow_stats messages.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install a flow with cookie A. Send a MULTIPART_REQUEST for flow statistics. Verify that the cookie field in the flow information returned has a value of A.

Specification text
When a flow entry is inserted in a table through an OFPFC_ADD message, its cookie field is set to the provided value.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 75)*

Topology
One control plane connection

Results
Pass / Fail

260.80 - Flow mod cookie modification

Protocol Messages / OFPT_FLOW_MOD / uint64_t cookie

Purpose
Verify that cookie values are not changed when flow entries are modified.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install a flow with cookie A and an output action to port X. Send a flow_mod message to modify the initial flow with cookie B, cookie_mask of zero and an output action to port Y. Send a MULTIPART_REQUEST for flow statistics. Verify that the cookie field in the flow information returned has a value of A.

Specification text
When a flow entry is modified (OFPFC_MODIFY or OFPFC_MODIFY_STRICT messages), its cookie field is unchanged.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 75)

Topology
One control plane connection

Results
Pass / Fail

260.90 - Flow mod cookie restriction

Protocol Messages / OFPT_FLOW_MOD / uint64_t cookie

Purpose
Ensure that using a non-zero cookie value can be used to restrict flow matching.

Methodology
Configure and connect DUT to controller. After control channel establishment, create two flows. The first flow has match fields A and cookie A set, the second flow has match fields A and cookie B set. Send a flow_mod message with a delete command with match fields A and cookie B. Verify that only the second flow was deleted.

Specification text
If the cookie_mask field is non-zero, it is used with the cookie field to restrict flow matching while modifying or deleting flow entries.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 75)

Topology
One control plane connection

Results
Pass / Fail

260.100 - Flow mod add ignore cookie mask

Protocol Messages / OFPT_FLOW_MOD / uint64_t cookie

Purpose
Check that a non-zero cookie mask field is ignored in ofp_flow_mod messages with an add command.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install a flow with cookie A and a cookie_mask field set to X. Send a MULTIPART_REQUEST for flow statistics. Verify that the cookie field in the flow information returned has a value of A.

Specification text
This field is ignored by OFPFC_ADD messages.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 75)

Topology
One control plane connection

Results
Pass / Fail

260.110 - Flow mod delete table

Protocol Messages / OFPT_FLOW_MOD / uint8_t table_id;

Purpose
For ofp_flow_mod messages with a delete or modify command, ensure that table_id can be used to select flows.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow in every table of the test pipeline, starting with table 0. Send a flow_mod message modifying each flow by table_id. After the modification, delete the flow by table_id.

Specification text
The table_id field specifies the table into which the flow entry should be inserted, modified or deleted. Table 0 signifies the first table in the pipeline.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 75)

Topology
One control plane connection

Results
Pass / Fail

Additional remarks
ID of the table to put the flow in. For OFPFC_DELETE_* commands, OFPTT_ALL can also be used to delete matching flows from all tables.

260.120 - Flow mod OFPTT_ALL

Protocol Messages / OFPT_FLOW_MOD / uint8_t table_id;

Purpose
Ensure DUT exhibits correct behavior when table OFPTT_ALL is specified in ofp_flow_mod messages.

Methodology
Configure and connect DUT to controller. After control channel establishment, send a flow_mod add message with the table_id field set to OFPTT_ALL. Verify that the switch sends an error message of type OFPET_FLOW_MOD_FAILED with error code OFPFMFC_BAD_TABLE_ID. Add flows in all tables, and then delete with table_id set to OFPTT_ALL. Verify that all flows were deleted.

Specification text
The use of OFPTT_ALL is only valid for delete requests.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 66)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
Error type is OFPET_FLOW_MOD_FAILED, code is OFPFMFC_BAD_TABLE_ID.

260.190 - Idle time before discarding (seconds).

Protocol Messages / OFPT_FLOW_MOD / uint16_t idle_timeout;

Purpose
Verify that IDLE_TIIMEOUT fields control the length of time a flow remains in the table.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match) with an idle_timeout of 5 seconds and hard_timeout of 0 seconds. Create some data plane traffic matching the flow, and verify that the traffic is forwarded for a set period of time. Stop forwarding data plane traffic, and wait for 5 seconds. Verify that the flow has been removed.

Specification text
The idle_timeout and hard_timeout fields control how quickly flow entries expire (see 5.5). When a flow entry is inserted in a table, its idle_timeout and hard_timeout fields are set with the values from the message.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 75)

Topology
One control plane connection

Results
Pass / Fail

260.200 - Max time before discarding (seconds).

Protocol Messages / OFPT_FLOW_MOD / uint16_t hard_timeout

Purpose
Verify that HARD_TIMEOUT fields control length of time flow remains in the table.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match) with an idle_timeout of 0 seconds and hard_timeout of 5 seconds. Create some data plane traffic matching the flow. Verify that traffic is forwarded for a period of 5 seconds, after which data plane traffic should no longer be forwarded.

Specification text
The idle_timeout and hard_timeout fields control how quickly flow entries expire (see 5.5). When a flow entry is inserted in a table, its idle_timeout and hard_timeout fields are set with the values from the message.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 75)

Topology
One control plane connection

Results
Pass / Fail

260.230 - Flow modification with IDLE_TIMEOUT with HARD_TIMEOUT both set.

Protocol Messages / OFPT_FLOW_MOD / timeout

Purpose
Test flow with IDLE_TIMOUTE and HARD_TIMEOUT both SET.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match) with an idle_timeout of 3 seconds and hard_timeout of 5 seconds. Create some data plane traffic matching the flow. Verify that traffic is forwarded for a period of 5 seconds, after which data plane traffic should no longer be forwarded. Add another flow matching on a named field (under the given Pre-requisites for the match) with an idle_timeout of 3 seconds and hard_timeout of 5 seconds. Create some data plane traffic matching the flow for 1 sec to make sure the flow is work. Stop the traffic for 3 secs and restart it. The traffic should no longer be forwarded.

Specification text
If both idle_timeout and hard_timeout are set, the flow entry will timeout after idle_timeout seconds with no traffic, or hard_timeout seconds, whichever comes first.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 76)

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

260.240 - Flow modification with IDLE_TIMEOUT and HARD_TIMEOUT both = 0.

Protocol Messages / OFPT_FLOW_MOD / timeout

Purpose
Verify that flows with idle and hard timeouts of zero are installed indefinitely.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match) with an idle_timeout of 0 seconds and hard_timeout of 0 seconds. Create some data plane traffic matching the flow. Verify that traffic is forwarded for a set period of time. Verify that the flow has not been removed.

Specification text
If both idle_timeout and hard_timeout are zero, the entry is considered permanent and will never time out. It can still be removed with a flow_mod message of type OFPFC_DELETE.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 76)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

260.250 - Priority level of flow entry.

Protocol Messages / OFPT_FLOW_MOD / uint16_t priority

Purpose
Verify that traffic matches against higher priority rules.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a mac address field with priority N and action output port 2. Add a second flow matching on the ethertype field with a priority N+1 and action output port 3. Verify that the flows are installed in the flow table. Forward data plane traffic matching both flows. Verify that data plane traffic matches against the higher priority flow, and is forwarded correctly.

Specification text
The priority indicates priority within the specified flow table. Higher numbers indicate higher priorities when matching packets (see 5.3). This field is used only for OFPFC_ADD messages when matching and adding flow entries, and for OFPFC_MODIFY_STRICT or OFPFC_DELETE_STRICT messages when matching flow entries.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 76)*

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

260.260 - Buffered packet to apply to, or OFP_NO_BUFFER. Not meaningful for OFPFC_DELETE*

Protocol Messages / OFPT_FLOW_MOD / uint32_t buffer_id;

Purpose
With MAX_LEN of OFPCML_NO_BUFFER set to 0xffff. Verify that packets are sent to the controller in their entirety using the "send to controller" action.

Methodology
230.60

Specification text
The buffer_id refers to a packet buffered at the switch and sent to the controller by a packet-in message. If no buffered packet is associated with the flow mod, it must be set to OFP_NO_BUFFER.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 76)*

Topology
One control plane connection

Results
Pass / Fail

260.270 - Valid BUFFER_ID in FLOW_MOD.

Protocol Messages / OFPT_FLOW_MOD / uint32_t buffer_id;

Purpose
Packets are processed normally if FLOW_MOD contains a valid BUFFER_ID.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a fully wildcarded flow with an action output to port CONTROLLER. Send a matching packet on the data plane. Verify that a packet_in message is triggered and encapsulates the matching packet. Based on the encapsulated packet, create a matching flow mod with a buffer_id set to the buffer_id included in the packet_in message with an action to output to a data plane port. Install the flow_mod and verify that the buffered packet is received on the specified data plane port.

Specification text
A flow mod that includes a valid buffer_id removes the corresponding packet from the buffer and processes it through the entire OpenFlow pipeline after the flow is inserted, starting at the first flow table. This is effectively equivalent to sending a two-message sequence of a flow mod and a packet-out forwarding to the OFPP_TABLE logical port (see 7.3.7), with the requirement that the switch must fully process the flow mod before the packet out.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 76)*

Topology
One control plane connection and three data plane connections

Results
Pass / Fail

260.280 - BUFFER_ID for DELETE messages.

Protocol Messages / OFPT_FLOW_MOD / uint32_t buffer_id;

Purpose
BUFFER_ID in OFP_PACKET_IN is ignored by DELETE messages.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the Pre-requisites for the match) with action as output port CONTROLLER. Send a packet for matching field and verify packet triggers an ofp_packet_in. Send an OFPFC_DELETE request for previous flow with the buffer_id field set to the buffer_id included in the packet_in message. Verify that no error message is received. Send the same packet as earlier and verify that packet is not received by output port and dropped by the switch.

Specification text
This field is ignored by OFPFC_DELETE and OFPFC_DELETE_STRICT flow mod messages.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 76)*

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

260.310 - OFPFC_DELETE* commands, A value of OFPC_ANY and OFPG_ANY disables filtering.

Protocol Messages / OFPT_FLOW_MOD / uint32_t out_group;

Purpose
OFPFC_DELETE for OUT_PORT and OUT_GROUP set to OFPP_ANY and OFPG_ANY respectively.

Methodology
Configure and connect DUT to controller. After control channel establishment, add first flow flow1 matching a named field (under the Pre-requisites for the match) with priority 100 and actions as output port X. Add second flow flow2 with same matching fields but with priority 200 and actions as output port Y. Send a packet matching above flows and verify that packet is received from port Y. Send an OFPFC_DELETE flow_mod request with above match fields, the out_port field as OFPP_ANY, and the out_group field as OFPG_ANY. Verify that both flows are removed from the flow table.

Specification text
Other constraints such as ofp_match structs and priorities are still used; this is purely an additional constraint. Note that to disable output filtering, both out_port and out_group must be set to OFPP_ANY and OFPG_ANY respectively.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 76)

Topology
One control plane connection

Results
Pass / Fail

260.320 - OFPFC_ADD, OFPFC_MODIFY or OFPFC_MODIFY_STRICT ignore OUT_PORT and OUT_GROUP.

Protocol Messages / OFPT_FLOW_MOD / uint32_t out_group;

Purpose
Verify that OFPFC_ADD, OFPFC_MODIFY or OFPFC_MODIFY_STRICT ignore OUT_PORT and OUT_GROUP.

Methodology
Configure and connect DUT to controller. After control channel establishment, add flow1 matching a named field (under the Pre-requisites for the match) with priority 100, out_port=OFPP_ANY, out_group=OFPG_ANY, and actions as output port X. Verify that matching traffic is forwarded to port X. Add a second flow2 with same matching fields but with priority 200, out_port=OFPP_ANY, out_group=OFPG_ANY, and actions as output port Y. Send a packet matching above flows and verify that packet is received from port Y. Send an ofp_flow_mod with command OFPFC_MODIFY_STRICT matching flow1 with an out_port=Z, and an additional match field. Verify that matching traffic is still forwarded to port Y.

Specification text
These fields are ignored by OFPFC_ADD, OFPFC_MODIFY or OFPFC_MODIFY_STRICT messages.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 76)

Topology
One control plane connection

Results
Pass / Fail

260.380 - Don't keep track of packet count

Protocol Messages / OFPT_FLOW_MOD / uint16_t flags; / OFPFF_NO_PKT_COUNTS

Purpose
Check how OFPFF_NO_PKT_COUNTS is handled.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match) with the OFPFF_NO_PKT_COUNTS. Wait a set period of time. Create and forward N matching data plane packets 200 bytes in length, so that the flow counters are increased. Check that the switch can reply to an OFPMP_FLOW multipart request. Verify that packet_count is equal to N, or that packet_count is -1.

Specification text
When the OFPFF_NO_PKT_COUNTS flag is set, the switch does not need to keep track of the flow packet count.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 77)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

260.390 - Don't keep track of byte count

Protocol Messages / OFPT_FLOW_MOD / uint16_t flags; / OFPFF_NO_BYT_COUNTS

Purpose
Check how OFPFF_NO_BYT_COUNTS is handled.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match) with the OFPFF_NO_BYT_COUNTS set. Wait a set period of time. Create and forward N matching data plane packets 200 bytes in length, so that the flow counters are increased. Check that the switch can reply to an OFPMP_FLOW multipart request. Verify that packet_byte is equal to (N * 200), or that packet_byte is -1.

Specification text
When the OFPFF_NO_BYT_COUNTS flag is set, the switch does not need to keep track of the flow byte count. Setting those flags may decrease the processing load on some OpenFlow switches, however those counters may not be available in flow statistics and flow removed messages for this flow entry.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 77)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

260.400 - OFPFF_NO_PKT_COUNTS and OFPFF_NO_BYT_COUNTS flags in flow statistics.

Protocol Messages / OFPT_FLOW_MOD / uint16_t flags;

Purpose
Verify how switch handles OFPFF_NO_PKT_COUNTS and OFPFF_NO_BYT_COUNTS.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match) with the OFPFF_NO_PKT_COUNTS and OFPFF_NO_BYT_COUNTS flags set". Wait a set period of time. Create and forward N matching data plane packets 200 bytes in length, so that the flow counters are increased. Check that the switch can reply to an OFPMP_FLOW multipart request. Verify that the packet_count is equal to N, or that packet_count is -1. Verify that byte_count is equal to (200*N), or that byte_count is -1.

Specification text
A switch is not required to honor those flags and may keep track of a flow count and return it despite the corresponding flag being set. If a switch does not keep track of a flow count, the corresponding counter is not available and must be set to the max field value.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 77)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

260.410 - OFPFF_NO_PKT_COUNTS and OFPFF_NO_BYT_COUNTS flags are ignored.

Protocol Messages / OFPT_FLOW_MOD / uint16_t flags;

Purpose

OFPFF_NO_PKT_COUNTS and OFPFF_NO_BYT_COUNTS are ignored for OFPFC_MODIFY or OFPFC_MODIFY_STRICT.

Methodology

Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given pre-requisites for the match) with the OFPFF_NO_BYT_COUNTS and OFPFF_NO_PKT_COUNTS set. Verify that a single flow is installed. Send an ofp_flow_mod with a modify code matching the first flow with no flags set. Verify that the flow has been modified, but that the flags are equal to the original flow.

Specification text

When a flow entry is inserted in a table, its flags field is set with the values from the message. When a flow entry is matched and modified (OFPFC_MODIFY or OFPFC_MODIFY_STRICT messages), the flags field is ignored.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.4.1; pg. 77)

Topology

One control plane connection

Results

Pass / Fail

300 - Multipart Reply Protocol Message

Test suite 300 verifies the ofp_multipart_request structure's various fields. In particular we define tests for multipart flags, features, statistics, and port descriptions.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

300.40 - Multipart request more flag

Protocol Messages / OFPT_MULTIPART_REQUEST / uint16_t flags; / OFPMPF_REQ_MORE

Purpose

Verify that a multipart request composed of multiple ofp_multipart_request messages is correctly replied to.

Methodology

Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request message of type OFPMP_PORT_DESC. The request should be composed of a minimum of two ofp_multipart_request messages and have the OFPMPF_REQ_MORE flag set on all but the last message. Verify that for each port_desc request a port_desc reply is received. This may be in a single or multiple message ofp_multipart_reply.

Specification text

The body field contains one segment of the request or reply. Every multipart request or reply is defined either as a single structure or as an array of 0 or more structures of the same type. If the multipart request or reply is defined as a single structure, it must use a single multipart message and the whole request or reply must be included in the body. If the multipart request or reply is defined as an array of structures, the body field must contain an integral number of objects, and no object can be split across two messages. To ease implementation, a multipart request or reply defined as an array may use messages with no additional entries (i.e. an empty body) at any point of the multipart sequence.
The flags field controls the segmentation/reassembly process. In a multipart request message, it may have the following values:

```
enum ofp_multipart_request_flags {
   OFPMPF_REQ_MORE  = 1 << 0  /* More requests to follow. */
};
```

In a multipart reply message, it may have the following values:

```
enum ofp_multipart_reply_flags {
   OFPMPF_REPLY_MORE  = 1 << 0  /* More replies to follow. */
};
```

The OFPMPF_REQ_MORE bit and the OFPMPF_REPLY_MORE bit indicate that more requests/replies will follow the current one.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5; pg. 83)

Topology

One control plane connection

Results

Pass / Fail

300.80 - Multipart reply more flag

Protocol Messages / OFPT_MULTIPART_REPLY / uint16_t flags; / OFPMPF_REPLY_MORE

Purpose

Verify that replies composed of multiple ofp_multipart_reply messages have the OFPMPF_REPLY_MORE flag set on all but the last message.

Methodology

Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request message of type OFPMP_PORT_STATS with port number of OFPP_ALL. Verify that replies composed of multiple ofp_multipart_reply messages have the OFPMPF_REPLY_MORE flag set on all but the last message.

Specification text

The only value defined for flags in a request and reply is whether more requests/replies will follow this one - this has the value 0x0001. To ease implementation, the controller is allowed to send requests and the switch is allowed to send replies with no additional entries (i.e. an empty body). However, another message must always follow a message with the more flag set.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5; pg. 83)*

Topology

One control plane connection

Results

Pass / Fail

300.100 - Multipart message xid

Protocol Messages / OFPT_MULTIPART_REPLY / uint16_t flags;

Purpose
Verify that replies composed of multiple ofp_multipart_reply messages have the same xid as the request.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request message of type OFPMP_PORT_DESC. Verify that replies composed of multiple ofp_multipart_request messages have the same xid as the original ofp_multipart_request.

Specification text
A request or reply that spans multiple messages (has one or more messages with the more flag set), must use the same multipart type and transaction id (xid) for all messages in the message sequence. Messages from a multipart request or reply may be interleaved with other OpenFlow message types, including other multipart requests or replies, but must have distinct transaction ids if multiple unanswered multipart requests or replies are in flight simultaneously.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5; pg. 83)

Topology
One control plane connection

Results
Pass / Fail

300.190 - Multipart type group counter statistics

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_multipart_type / OFPMP_GROUP = 6

Purpose
Verify that a valid response is received when requesting group counter statistics.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch. Send an ofp_multipart_request with type OFPMP_GROUP. If groups are not supported verify an ofp_error_msg is returned with type OFPET_BAD_REQUEST and code OFPBRC_BAD_MULTIPART. Otherwise verify that the reply is an array of ofp_group_stats structs.

Specification text
/* Group counter statistics. * The request body is struct ofp_group_stats_request. * The reply is an array of struct ofp_group_stats. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5; pg. 84-85)

Topology
One control plane connection

Results
Pass / Fail

300.200 - Multipart type group description

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_multipart_type / OFPMP_GROUP_DESC = 7

Purpose
Verify that a valid response is received when requesting group descriptions.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch. Send an ofp_multipart_request with type OFPMP_GROUP_DESC. If groups are not supported, verify that an ofp_error_msg is returned with type OFPET_BAD_REQUEST and code OFPBRC_BAD_MULTIPART. Otherwise verify that the reply is an array of ofp_group_desc structs.

Specification text
/* Group description. * The request body is empty. * The reply body is an array of struct ofp_group_desc. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5; pg. 85)

Topology
One control plane connection

Results
Pass / Fail

300.210 - Multipart type group features

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_multipart_type / OFPMP_GROUP_FEATURES = 8

Purpose
Verify that a valid response is received when requesting group features.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch. Send an ofp_multipart_request with type OFPMP_GROUP_FEATURES. If groups are not supported, verify that an ofp_error_msg is returned with type OFPET_BAD_REQUEST and code OFPBRC_BAD_MULTIPART. Otherwise verify that the reply is an array of ofp_group_features structs.

Specification text
/* Group features. * The request body is empty. * The reply body is struct ofp_group_features. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5; pg. 85)

Topology
One control plane connection

Results
Pass / Fail

300.220 - Multipart type meter statistics

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_multipart_type / OFPMP_METER = 9

Purpose
Verify that a valid response is received when requesting meter statistics.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch. Send an ofp_multipart_request with type OFPMP_METER. If meters are not supported, verify that an ofp_error_msg is returned with type OFPET_BAD_REQUEST and code OFPBRC_BAD_MULTIPART. Otherwise, verify that the reply is an array of ofp_meter_stats structs.

Specification text
/* Meter statistics. * The request body is struct ofp_meter_multipart_requests. * The reply body is an array of struct ofp_meter_stats. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5; pg. 85)

Topology
One control plane connection

Results
Pass / Fail

300.230 - Multipart type meter configuration

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_multipart_type / OFPMP_METER_CONFIG = 10

Purpose
Verify that a valid response is received when requesting meter configurations.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch. Send an ofp_multipart_request with type OFPMP_METER_CONFIG. If meters are not supported, verify an ofp_error_msg is returned with type OFPET_BAD_REQUEST and code OFPBRC_BAD_MULTIPART. Otherwise, verify that the reply is an array of ofp_meter_config structs.

Specification text
/* Meter configuration. * The request body is struct ofp_meter_multipart_requests. * The reply body is an array of struct ofp_meter_config. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5; pg. 85)*

Topology
One control plane connection

Results
Pass / Fail

300.240 - Multipart type meter features

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_multipart_type / OFPMP_METER_FEATURES = 11

Purpose
Verify that a valid response is received when requesting meter features.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch. Send an ofp_multipart_request with type OFPMP_METER_FEATURES. If meters are not supported, verify that an ofp_error_msg is returned with type OFPET_BAD_REQUEST and code OFPBRC_BAD_MULTIPART. Otherwise verify that the reply is an array of ofp_meter_features structs.

Specification text
/* Meter features. * The request body is empty. * The reply body is struct ofp_meter_features. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5; pg. 85)

Topology
One control plane connection

Results
Pass / Fail

300.260 - Multipart type port description

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_multipart_type / OFPMP_PORT_DESC = 13

Purpose

Verify that a response composed of multiple ofp_port structs is received when requesting port descriptions.

Methodology

Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request message of type OFPMP_PORT_DESC. Verify that replies composed of multiple ofp_multipart_request messages have the same xid as the original ofp_multipart_request.

Specification text

/* Port description. * The request body is empty. * The reply body is an array of struct ofp_port. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5; pg. 85)

Topology

One control plane connection

Results

Pass / Fail

300.270 - Multipart type experimenter extension

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_multipart_type / OFPMP_EXPERIMENTER = 0xffff

Purpose
Verify that a valid response is received when requesting an experimenter extension.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_experimenter. The request should contain a valid header, but a bogus experimenter part. Verify that the switch returns an error OFPET_BAD_REQUEST and OFPBRC_BAD_EXPERIMENTER.

Specification text
/* Experimenter extension. * The request and reply bodies begin with * struct ofp_experimenter_multipart_header. * The request and reply bodies are otherwise experimenter-defined. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5; pg. 85)

Topology
One control plane connection

Results
Pass / Fail

300.280 - Multipart request buffer overflow

Protocol Messages / OFPT_MULTIPART_REPLY

Purpose
If a multipart request contains more data than a device can buffer, verify that a bad request error with a multipart buffer overflow code is generated.

Methodology
Configure and connect DUT to controller. After connection establishment, send a request that is larger than any imaginable legal request (10 million port stats in one request for example). If the switch does not generate an error, verify that the response is correct.

Specification text
If a multipart request spans multiple messages and grows to a size that the switch is unable to buffer, the switch must respond with an error message of type OFPET_BAD_REQUEST and code OFPBRC_MULTIPART_BUFFER_OVERFLOW.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5; pg. 84)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Test recommendations
Send an ofp_port_stats multipart request with 10 million stats requests to the same physical port.

If a device responds to each ofp_multipart_request structure, and does not wait for the final ofp_multipart structure of the multipart request, it may not be possible to trigger this error message. If the error cannot be reliably triggered, the test MAY be recorded at ‘Not Applicable’.

300.290 - Multipart message unsupported type

Protocol Messages / OFPT_MULTIPART_REPLY

Purpose

If a multipart request contains a type that is not supported, the switch must respond with an error message of type OFPET_BAD_REQUEST and code OFPBRC_BAD_MULTIPART.

Methodology

Configure and connect DUT to controller. Add a flow entry to the switch. Send an ofp_multipart_request with an invalid type. Verify that an ofp_error_msg is returned with type OFPET_BAD_REQUEST and code OFPBRC_BAD_MULTIPART.

Specification text

If a multipart request contains a type that is not supported, the switch must respond with an error message of type OFPET_BAD_REQUEST and code OFPBRC_BAD_MULTIPART.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5; pg. 84)

Topology

One control plane connection

Results

Pass / Fail

310 - Multipart Reply Section One

Test suite 310 verifies the correct implementation of the fields contained in each of the following message structs; ofp_desc, ofp_flow_stats_request, ofp_flow_stats, ofp_aggregate_stats_request, and ofp_aggregate_stats_reply.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

310.60 - Flow statistics

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_flow_stats_request

Purpose
Verify that the switch can reply to the OFPMP_FLOW multipart request.

Methodology
Configure and connect DUT to controller. After control channel establishment, enter a certain number of flows into the flow table. Send several ofp_multipart_requests with type OFPMP_FLOW. Check that the switch can reply to OFPMP_FLOW multipart request.

Specification text
Information about individual flow entries is requested with the OFPMP_FLOW multipart request type
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.1; pg. 86)

Topology
One control plane connection

Results
Pass / Fail

310.70 - Flow statistics table id

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_flow_stats_request / uint8_t table_id; / ID of table to read (from ofp_table_stats), OFPTT_ALL for all tables. */*

Purpose
Verify that the switch can reply to the OFPMP_FLOW multipart request, according to the table_id field.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch. Send an ofp_multipart_request of type OFPMP_FLOW with table_id set to 0. Send a second ofp_multipart_request with table_id set to OFPTT_ALL and a third ofp_multipart_request with table_id set to any other table_id value from ofp_table_stats. Check that the DUT can reply to each OFPMP_FLOW multipart request and the table_id is the same as the specific value in the multipart request message or includes all the values supported by the DUT when the table_id is OFPTT_ALL in the multipart request message.

Specification text
The table_id field indicates the index of a single table to read, or OFPTT_ALL for all tables.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.1; pg. 86)

Topology
One control plane connection

Results
Pass / Fail

310.80 - Flow statistics out port

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_flow_stats_request

Purpose
Verify that the switch can reply to the OFPMP_FLOW multipart request, according to the out_port field.

Methodology
Configure and connect DUT to controller. Add three flow entries to the switch, two of them forwarding packets to only one of the two output ports, and the third flow outputting packets to both output ports. Send N packets matching flow ONE, M packets matching flow TWO, and P packets matching flow THREE. Send an ofp_multipart_request with type OFPMP_FLOW, out_port field specific set to OFPP_ANY and verify that the answer contains all flows. Send one request filtering one specific output port. Check that the switch only replies with the two flows containing that out_port.

Specification text
uint32_t out_port; /* Require matching entries to include this as an output port. A value of OFPP_ANY indicates no restriction. */

The out_port and out_group fields optionally filter by output port and group. If either out_port or out_group contain a value other than OFPP_ANY and OFPG_ANY respectively, it introduces a constraint when matching. This constraint is that the flow entry must contain an output action directed at that port or group.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.1; pg. 86)*

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
Verify that flows exist using data plane verification. Recommend flow matches on eth_src. OXM fields should be fully wildcarded.

310.100 - Flow statistics cookie

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_flow_stats_request

Purpose
Verify that the switch can reply to the OFPMP_FLOW multipart request, according to the cookie field.

Methodology
Configure and connect DUT to controller. Add two flow entries to the switch with cookies set as 1 and 2 respectively. Send an ofp_multipart_request with type OFPMP_FLOW, cookie field set to 1(cookie_mask set to 0xFFFFFFFFFFFFFFFF). Check that the switch can reply to OFPMP_FLOW multipart request, match the flow entry with the cookie set as 1, and correctly set the cookie_field in the multipart reply message to 1.

Specification text
uint64_t cookie; /* Require matching entries to contain this cookie value */

The usage of the cookie and cookie_mask fields is defined in Section 6.4.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.2; pg. 86)

Topology
One control plane connection

Results
Pass / Fail

310.110 - Flow statistics cookie mask

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_flow_stats_request

Purpose
Verify that the switch can reply to the OFPMP_FLOW multipart request, according to the cookie_mask field.

Methodology
Configure and connect DUT to controller. Add three flow entries to the switch with cookies separately as 1,2 and 3. Send an ofp_multipart_request with type OFPMP_FLOW, cookie field set to 2 and cookie_mask set to 0xFFFFFFFFFFFFFFFE. Check that the switch can reply to OFPMP_FLOW multipart request, match the flow entries with their cookies set as 2 and 3, and correctly set the cookie_field in the multipart reply messages as 2 and 3 respectively.

Specification text
uint64_t cookie_mask; /* Mask used to restrict the cookie bits that must match. A value of 0 indicates no restriction. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.2; pg. 86)

Topology
One control plane connection

Results
Pass / Fail

310.150 - Flow statistics nano duration

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_flow_stats

Purpose
Verify that the switch can send the OFPMP_FLOW multipart reply with the right duration_nsec field.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch. Send an ofp_multipart_request with type OFPMP_FLOW in one second with one time per 100ms. Check that the switch can send OFPMP_FLOW multipart reply and duration_nsec field is increasing. If the switch doesn't support the counter it should be set to -1, otherwise it should report the correct value.

Specification text
uint32_t duration_nsec; /* Time flow has been alive in nanoseconds beyond duration_sec. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.2; pg. 87)*

Topology
One control plane connection

Results
Pass / Fail

310.160 - Flow statistics priority

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_flow_stats

Purpose
Verify that the switch can send the OFPMP_FLOW multipart reply with the right priority field.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch with priority N. Send an ofp_multipart_request with type OFPMP_FLOW. Check that the switch can send OFPMP_FLOW multipart reply and priority field is N.

Specification text
uint16_t priority; /* Priority of the entry. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.2; pg. 87)

Topology
One control plane connection

Results
Pass / Fail

310.170 - Flow statistics idle timeout

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_flow_stats

Purpose
Verify that the switch can send the OFPMP_FLOW multipart reply with the right idle_timeout field.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch with idle_timeout as N. Send an ofp_multipart_request with type OFPMP_FLOW. Check that the switch can send OFPMP_FLOW multipart reply and idle_timeout field is N.

Specification text
uint16_t idle_timeout; /* Number of seconds idle before expiration. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.2; pg. 87)

Topology
One control plane connection

Results
Pass / Fail

310.180 - Flow statistics hard timeout

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_flow_stats

Purpose
Verify that the switch can send the OFPMP_FLOW multipart reply with the right hard_timeout field.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch with hard_timeout as N. Send an ofp_multipart_request with type OFPMP_FLOW. Check that the switch can send OFPMP_FLOW multipart reply and hard_timeout field is N.

Specification text
uint16_t hard_timeout; /* Number of seconds before expiration. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.2; pg. 87)

Topology
One control plane connection

Results
Pass / Fail

310.190 - Flow statistics OFPFF_* flags

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_flow_stats

Purpose
Verify that the switch can send the OFPMP_FLOW multipart reply with the right flags field.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch and OFPFF_* flags set to 1. Send an ofp_multipart_request with type OFPMP_FLOW. Check that the switch can send OFPMP_FLOW multipart reply and flags field is 1.

Specification text
uint16_t flags; /* Bitmap of OFPFF_* flags. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.2; pg. 87)

Topology
One control plane connection

Results
Pass / Fail

310.230 - Flow statistics match

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_flow_stats

Purpose
Verify that the switch can send the OFPMP_FLOW multipart reply with the right match field.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch. Send an ofp_multipart_request with type OFPMP_FLOW. Check that the switch can send OFPMP_FLOW multipart reply, and match fields are same with the flow entry.

Specification text
struct ofp_match match; /* Description of fields. Variable size. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.2; pg. 87)

Topology
One control plane connection

Results
Pass / Fail

310.240 - Aggregate statistics

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_aggregate_stats_request

Purpose
Verify that the switch can reply to the OFPMP_AGGREGATE multipart request.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_AGGREGATE. Check that the switch can reply to OFPMP_AGGREGATE multipart request.

Specification text
Aggregate information about multiple flow entries is requested with the OFPMP_AGGREGATE multipart request type
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.3; pg. 87)

Topology
One control plane connection

Results
Pass / Fail

310.250 - Aggregate statistics table id

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_aggregate_stats_request

Purpose
Verify that the switch can reply to the OFPMP_AGGREGATE multipart request, according to the table_id field.

Methodology
Configure and connect DUT to controller. Add a flow entry for each table reported by the switch. For each table reported by the switch, send an ofp_multipart_request with type OFPMP_AGGREGATE, and the table_id field set to a reported table. Check that the switch can reply correctly to each OFPMP_AGGREGATE multipart request. Verify that the flow_count is correct. Additionally, send an ofp_multipart_request request with type OFPMP_AGGREGATE, and the table_id field set to OFPTT_ALL. Verify that the number of ofp_aggregate_stats_reply structs is equal to the number of tables when table_id is OFPTT_ALL in the multipart request message.

Specification text
uint8_t table_id; /* ID of table to read (from ofp_table_stats) OFPTT_ALL for all tables. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.3; pg. 87)*

Topology
One control plane connection

Results
Pass / Fail

310.260 - Aggregate statistics outport

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_aggregate_stats_request

Purpose
Verify that the switch can reply to the OFPMP_AGGREGATE multipart request, according to the out_port field.

Methodology
Configure and connect DUT to controller. Add three flow entries to the switch, two forwarding to only one of the two output ports, and one flow outputting both output ports. Send N matching packets to flow 1 and M matching packets to flow two, and P matching packets on flow three. Send an ofp_multipart_request with type OFPMP_AGGREGATE, out_port field separately set to OFPP_ANY and verify that the answer contains three flows. Send one request filtering one specific output port. Check the switch only replies with the two flows containing that out_port .

Specification text
uint32_t out_port; /* Require matching entries to include this as an output port. A value of OFPP_ANY indicates no restriction. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.3; pg. 87)*

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

310.280 - Aggregate statistics cookie

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_aggregate_stats_request

Purpose
Verify that the switch can reply to the OFPMP_AGGREGATE multipart request, according to the cookie field.

Methodology
Configure and connect DUT to controller. Add two flow entry to the switch with cookies separately as 1 and 2. Send an ofp_multipart_request with type OFPMP_AGGREGATE, cookie field set to 1(cookie_mask set to 0xFFFFFFFFFFFFFFFF). Check that the switch can reply to OFPMP_AGGREGATE multipart request, and flow_count is 1.

Specification text
uint64_t cookie; /* Require matching entries to contain this cookie value */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.3; pg. 87)

Topology
One control plane connection

Results
Pass / Fail

310.290 - Aggregate statistics cookie mask

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_aggregate_stats_request

Purpose
Verify that the switch can reply to the OFPMP_AGGREGATE multipart request,according to the cookie_mask field.

Methodology
Configure and connect DUT to controller. Add three flow entry to the switch with cookies separately as 1,2 and 3. Send an ofp_multipart_request with type OFPMP_AGGREGATE, cookie field set to 2 and cookie_mask set to 0xFFFFFFFFFFFFFFFE. Check the switch can reply to OFPMP_AGGREGATE multipart request, and flow_count is 2.

Specification text
uint64_t cookie_mask; /* Mask used to restrict the cookie bits that must match. A value of 0 indicates no restriction. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.3; pg. 87)*

Topology
One control plane connection

Results
Pass / Fail

310.310 - Aggregate statistics packet count

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_aggregate_stats_reply

Purpose
Verify that the switch can send the OFPMP_AGGREGATE multipart reply with the right packet_count field.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch. Send N matching packets. Send an ofp_multipart_request with type OFPMP_AGGREGATE. Check the switch can send OFPMP_AGGREGATE multipart reply, and packet_count is N.

Specification text
uint64_t packet_count; /* Number of packets in flows. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.3; pg. 87)*

Topology
One control plane connection

Results
Pass / Fail

310.320 - Aggregate statistics byte count

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_aggregate_stats_reply

Purpose
Verify that the switch can send the OFPMP_AGGREGATE multipart reply with the right byte_count field.

Methodology
Configure and connect DUT to controller. Add a flow entry to the switch. Send matching packets with N bytes. Send an ofp_multipart_request with type OFPMP_AGGREGATE. Check that the switch can send OFPMP_AGGREGATE multipart reply, and byte_count is N.

Specification text
uint64_t byte_count; /* Number of bytes in flows. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.3; pg. 87)

Topology
One control plane connection

Results
Pass / Fail

310.330 - Aggregate statistics flow count

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_aggregate_stats_reply

Purpose
Verify that the switch can send the OFPMP_AGGREGATE multipart reply with the right flow_count field.

Methodology
Configure and connect DUT to controller. Add N flow entry to the switch. Send an ofp_multipart_request with type OFPMP_AGGREGATE. Check that the switch can send OFPMP_AGGREGATE multipart reply, and flow_count is N.

Specification text
uint32_t flow_count; /* Number of flows. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.3; pg. 87)

Topology
One control plane connection

Results
Pass / Fail

320 - Multipart Reply Section Two

Test suite 320 verifies the correct implementation of the fields contained in each of the following message structs; ofp_table_stats, ofp_table_features, ofp_table_feature_prop_type, ofp_table_feature_prop_instructions, ofp_table_feature_prop_next_tables, ofp_table_feature_prop_actions, and ofp_action_header_action_ids.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

320.10 - Table Statistics Count

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_stats

Purpose
Verify that the n_tables ofp_table_stats messages are returned in response to a multipart table request.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE. Check that the switch can send OFPMP_TABLE ofp_multipart_reply, and that N ofp_table_stats are returned.

Specification text
Information about tables is requested with the OFPMP_TABLE multipart request type. The request does not contain any data in the body. The body of the reply consists of an array of the following:

```
/* Body of reply to OFPMP_TABLE request. */
struct ofp_table_stats {
uint8_t table_id; /* Identifier of table. Lower numbered tables
are consulted first. */
uint8_t pad[3]; /* Align to 32-bits. */
uint32_t active_count; /* Number of active entries. */
uint64_t lookup_count; /* Number of packets looked up in table. */
uint64_t matched_count; /* Number of packets that hit table. */
};
OFP_ASSERT(sizeof(struct ofp_table_stats) == 24);
```
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.4; pg. 88)

Topology
One control plane connection

Results
Pass / Fail

320.20 - /* Identifier of table. Lower numbered tables are consulted first. */

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_stats / uint8_t table_id;

Purpose
Verify that the n_tables ofp_table_stats messages are returned in response to a multipart table request from lowest table_id to the highest supported table_id.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE. Check that the switch can send OFPMP_TABLE ofp_multipart_reply, and that N ofp_table_stats are returned. Verify that ofp_table_stats are reported in order from lowest table_id to highest table_id.

Specification text
The array has one structure for each table supported by the switch. The entries are returned in the order that packets traverse the tables.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.4; pg. 88)

Topology
One control plane connection

Results
Pass / Fail

320.30 - /* Number of active entries. */

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_stats / uint32_t active_count;

Purpose
Verify that the correct active_count number is returned.

Methodology
Configure and connect DUT to controller. Randomly add between 1 and 3 flows with a hard timeout of 0 to each table in the switch. Send an ofp_multipart_request with type OFPMP_TABLE. Check that the switch can send OFPMP_TABLE ofp_multipart_reply, and that N ofp_table_stats are returned. Verify that the value of active_count in each ofp_table_stats returned is equal to the number of flows added to that table.

Specification text
struct ofp_table_stats {
uint8_t table_id; /* Identifier of table. Lower numbered tables are consulted first. */
uint8_t pad[3]; /* Align to 32-bits. */
uint32_t active_count; /* Number of active entries. */
uint64_t lookup_count; /* Number of packets looked up in table. */
uint64_t matched_count; /* Number of packets that hit table. */
};
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.4; pg. 88)

Topology
One control plane connection

Results
Pass / Fail

320.40 - /* Number of packets looked up in table. */

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_stats / uint64_t lookup_count;

Purpose
Verify that the switch replies with the correct packet lookup_count number.

Methodology
Configure and connect DUT to controller. Add a flow with a hard timeout of 0 to the switch. Send M non-matching packets on the data plane. Send an ofp_multipart_request with type OFPMP_TABLE. Check that the switch can send OFPMP_TABLE ofp_multipart_reply, and that N ofp_table_stats are returned. Verify that the value of lookup_count is equal to M; otherwise verify the value of lookup_count is equal to -1 (counter is not supported).

Specification text
struct ofp_table_stats {
uint8_t table_id; /* Identifier of table. Lower numbered tables are consulted first. */
uint8_t pad[3]; /* Align to 32-bits. */
uint32_t active_count; /* Number of active entries. */
uint64_t lookup_count; /* Number of packets looked up in table. */
uint64_t matched_count; /* Number of packets that hit table. */
};

- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.4; pg. 88)

Topology
One control plane connection

Results
Pass / Fail

320.50 - /* Number of packets that hit table. */

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_stats / uint64_t matched_count;

Purpose
Verify that the swith replies with the correct matched_count number.

Methodology
Configure and connect DUT to controller. Add a flow with a hard timeout of 0 to the switch. Send M matching packets on the data plane. Send an ofp_multipart_request with type OFPMP_TABLE. Check that the switch can send OFPMP_TABLE ofp_multipart_reply, and that N ofp_table_stats are returned. Verify that the value of matched_count is equal to M; otherwise verify the value of matched_count is equal to -1 (counter is not supported).

Specification text
struct ofp_table_stats {
uint8_t table_id; /* Identifier of table. Lower numbered tables are consulted first. */
uint8_t pad[3]; /* Align to 32-bits. */
uint32_t active_count; /* Number of active entries. */
uint64_t lookup_count; /* Number of packets looked up in table. */
uint64_t matched_count; /* Number of packets that hit table. */
};

- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.4; pg. 88)

Topology
One control plane connection

Results
Pass / Fail

320.60 - OFPMP_TABLE_FEATURES Table Reconfiguration

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_features

Purpose
Verify that the oft_multipart_reply returns correct information without error.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES with empty body. Check that the switch can send OFPMP_TABLE_FEATURES ofp_multipart_reply. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and body including the same table features reported by the device in the first ofp_multipart_reply. Verify that if modifying table features is not supported an OFPET_BAD_REQUEST error is generated with an OFPBRC_BAD_LEN code. If modifying table features is disabled, an OFPET_TABLE_FEATURES_FAILED error with an OFPTFFC_EPERM code should be received instead. Otherwise, verify that no errors are returned.

Specification text
The OFPMP_TABLE_FEATURES multipart type allows a controller to both query for the capabilities of existing tables, and to optionally ask the switch to reconfigure its tables to match a supplied configuration. In general, the table feature capabilities represents all possible features of a table, however some features may be mutually exclusive and the current capabilities structures do not allow us to represent such exclusions.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5; pg. 88)*

Topology
One control plane connection

Results
Pass / Fail

320.70 - Table Features request and reply

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_features

Purpose
Verify that an empty table features request message generates a response detailing all configured tables.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES with empty body. Check that switch can send OFPMP_TABLE_FEATURES ofp_multipart_reply, and that N ofp_table_features are returned.

Specification text
If the OFPMP_TABLE_FEATURES request body is empty, the switch will return an array of struct ofp_table_features containing the capabilities of the currently configured flow tables.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.1; pg. 89)

Topology
One control plane connection

Results
Pass / Fail

320.140 - Table features unique id

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_features

Purpose
Verify that table features are reported for each table and that their table numbers are unique.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES. Check that switch can send OFPMP_TABLE_FEATURES ofp_multipart_reply, and that N ofp_table_features are returned. Verify that all ofp_table_features structs' table_id fields are unique.

Specification text
Requests and replies containing ofp_table_features are expected to meet the following minimum requirements: Each ofp_table_features struct's table_id field value should be unique amongst all ofp_table_features structs in the message
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.1; pg. 89)

Topology
One control plane connection

Results
Pass / Fail

320.150 - Table features all property types

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_features

Purpose
Verify that all table feature prop types are reported for each table.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES with empty body. Check that switch can send OFPMP_TABLE_FEATURES ofp_multipart_reply. Verify that exactly one of each ofp_table_feature_prop_type properties is reported for each table. Ignore omitted OFPTFPT_*_MISS properties and omitted _EXPERIMENTER properties.

Specification text
The properties field included in each ofp_table_features structure must contain exactly one of each of the ofp_table_feature_prop_type properties, with two exceptions.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.1; pg. 89)

Topology
One control plane connection

Results
Pass / Fail

320.160 - Table features omitting miss

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_features

Purpose
Verify that action miss types can be modified by including only the *_ACTIONS property, and omitting the *_ACTIONS_MISS property.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES with empty body. Check that switch can send OFPMP_TABLE_FEATURES ofp_multipart_reply. Send a second ofp_multipart_request with type OFPMP_TABLE_FEATURES including the body that omits the ofp_table_feature_prop_header with type OFPTFPT_APPLY_ACTIONS_MISS from each ofp_table_features struct. Verify that if modifying table features is not supported an OFPET_BAD_REQUEST error is generated with an OFPBRC_BAD_LEN code. If modifying table features is disabled an OFPET_TABLE_FEATURES_FAILED error with an OFPTFFC_EPERM code should be received instead. Otherwise verify through a final ofp_multipart_request with type OFPMP_TABLE_FEATURES that the response reports the correct apply action types.

Specification text
First, properties with the _MISS suffix may be ommited if it is the same as the corresponding property for regular flow entries.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.1; pg. 89)*

Topology
One control plane connection

Results
Pass / Fail

320.170 - Table features omitting experimenters

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_features

Purpose
Verify that experimenter miss types can be disabled by omitting only the *_EXPERIMENTER, and *_EXPERIMENTER_MISS properties.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES with empty body. Check that switch can send OFPMP_TABLE_FEATURES ofp_multipart_reply. Send a second ofp_multipart_request with type OFPMP_TABLE_FEATURES including a body that omits the ofp_table_feature_prop_header with types OFPTFPT_EXPERIMENTER and OFPTFPT_EXPERIMENTER_MISS from each ofp_table_features struct. Verify that if modifying table features is not supported, an OFPET_BAD_REQUEST error is generated with an OFPBRC_BAD_LEN code. If modifying table features is disabled an OFPET_TABLE_FEATURES_FAILED error with an OFPTFFC_EPERM code should be received instead. Otherwise, verify through a final ofp_multipart_request with type OFPMP_TABLE_FEATURES that the response does not report the experimenter table feature property types.

Specification text
Second, properties of type OFPTFPT_EXPERIMENTER and OFPTFPT_EXPERIMENTER_MISS may be omitted or included many times. Ordering is unspecified, but implementers are encouraged to use the ordering listed in the specification (see 7.3.5.5.2).
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.1; pg. 89)

Topology
One control plane connection

Results
Pass / Fail

320.180 - Table features omitting matches

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_features

Purpose
Verify that an attempt to set table feature properties without including the match property triggers an error message.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES with empty body. Check that switch replies with an OFPMP_TABLE_FEATURES ofp_multipart_reply message. Send a second ofp_multipart_request with type OFPMP_TABLE_FEATURES including the body that omits the ofp_table_feature_prop_header with type OFPTFPT_MATCH from each ofp_table_features struct. Verify that if modifying table features is not supported an OFPET_BAD_REQUEST error is generated with an OFPBRC_BAD_LEN code. If modifying table features is disabled, an OFPET_TABLE_FEATURES_FAILED error with an OFPTFFC_EPERM code should be received instead. Otherwise, verify an ofp_error message of type OFPET_TABLE_FEATURE_FAILED is returned with an error code of OFPTFFC_BAD_LEN.

Specification text
A switch receiving a request that does not meet these requirements should return an error of type OFPET_TABLE_FEATURES_FAILED with the appropriate error code.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.1; pg. 89)

Topology
One control plane connection

Results
Pass / Fail

320.190 - Table features order

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_features / uint8_t table_id;

Purpose
Verify that table features are reported from the lowest table number to the highest table number.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and no body. Check that switch can send OFPMP_TABLE_FEATURES ofp_multipart_reply, and that N ofp_table_features are returned. Verify that ofp_table_features are reported in order from lowest table_id to highest table_id.

Specification text
The array has one structure for each flow table supported by the switch. The entries are always returned in the order that packets traverse the flow tables.
OFPMP_TABLE_FEATURES./
* Body of reply to OFPMP_TABLE_FEATURES request. */
struct ofp_table_features {
uint16_t length; /* Length is padded to 64 bits. */
uint8_t table_id; /* Identifier of table. Lower numbered tables are consulted first. */
uint8_t pad[5]; /* Align to 64-bits. */
char name[OFP_MAX_TABLE_NAME_LEN];
uint64_t metadata_match; /* Bits of metadata table can match. */
uint64_t metadata_write; /* Bits of metadata table can write. */
uint32_t config; /* Bitmap of OFPTC_* values */
uint32_t max_entries; /* Max number of entries supported. */
/* Table Feature Property list */
struct ofp_table_feature_prop_header properties[0]; /* List of properties */
};
OFP_ASSERT(sizeof(struct ofp_table_features) == 64);
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.1; pg. 90)

Topology
One control plane connection

Results
Pass / Fail

320.200 - Table features name modification

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_features / char name[OFP_MAX_TABLE_NAME_LEN];

Purpose
Verify that table names can be modified through a table freatures request.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES including the body that sets the name of a table with a length of OFP_MAX_TABLE_NAME_LEN. Verify that if modifying table features is not supported an OFPET_BAD_REQUEST error is generated with an OFPBRC_BAD_LEN code. If modifying table features is disabled an OFPET_TABLE_FEATURES_FAILED error with an OFPTFFC_EPERM code should be received instead. Otherwise verify through a final ofp_multipart_request with type OFPMP_TABLE_FEATURES that the response reports the new table name.

Specification text
/* Body for ofp_multipart_request of type OFPMP_TABLE_FEATURES./
* Body of reply to OFPMP_TABLE_FEATURES request. */
struct ofp_table_features {
uint16_t length; /* Length is padded to 64 bits. */
uint8_t table_id; /* Identifier of table. Lower numbered tables are consulted first. */
uint8_t pad[5]; /* Align to 64-bits. */
char name[OFP_MAX_TABLE_NAME_LEN];
uint64_t metadata_match; /* Bits of metadata table can match. */
uint64_t metadata_write; /* Bits of metadata table can write. */
uint32_t config; /* Bitmap of OFPTC_* values */
uint32_t max_entries; /* Max number of entries supported. */
/* Table Feature Property list */
struct ofp_table_feature_prop_header properties[0]; /* List of properties */
};
OFP_ASSERT(sizeof(struct ofp_table_features) == 64);

- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.1; pg. 90)

Topology
One control plane connection

Results
Pass / Fail

320.210 - Table features metadata match

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_features / uint64_t metadata_match;

Purpose

If support for matching on metadata is reported in table features, verify that the metadata_match field is not equal to zero.

Methodology

Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES with empty body. Check that the switch can send an ofp_multipart_reply with type OFPMP_TABLE_FEATURES. Check if the ofp_table_feature_prop of type OFPTFPT_MATCH supports matching on the metadata field. If so, verify the metadata_match field is greater than 0. Otherwise verify that the metadata_match field is equal to zero.

Specification text

The metadata_match field indicates the bits of the metadata field that the table can match on, when using the metadata field of struct ofp_match. A value of 0xFFFFFFFFFFFFFFFF indicates that the table can match the full metadata field.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.1; pg. 90)

Topology

One control plane connection

Results

Pass / Fail

320.220 - Table features metadata write

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_features / uint64_t metadata_write;

Purpose

If support for writing metadata is reported in table features, verify that the metadata_write field is not equal to zero.

Methodology

Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES with empty body. Check that the switch can send an ofp_multipart_reply with type OFPMP_TABLE_FEATURES. Check if the ofp_table_feature_prop of type OFPTFPT_INSTRUCTIONS or OFPTFPT_INSTRUCTIONS_MISS supports writing the metadata field. If so, verify the metadata_write field is greater than 0. Otherwise verify that the metadata_write field is equal to zero.

Specification text

The metadata_write field indicates the bits of the metadata field that the table can write using the OFPIT_WRITE_METADATA instruction. A value of 0xFFFFFFFFFFFFFFFF indicates that the table can write the full metadata field.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.1; pg. 90)

Topology

One control plane connection

Results

Pass / Fail

320.230 - Table features configuration

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_features / uint32_t config;

Purpose
Verify that the config field of a table features message does not set invalid configuration bits.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and no body. Check that the config field is equal to OFPTC_DEPRECATED_MASK or zero.

Specification text
The config field is the table configuration that was set on the table via a table configuration message (see 7.3.3).
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.1; pg. 90)

Topology
One control plane connection

Results
Pass / Fail

320.240 - Table features max entries

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_features / uint32_t max_entries;

Purpose
Verify max_entries is reported correctly.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and no body. Attempt to add max_entries flow matching on a named field (under the given Pre-requisites for the match), with different priorities, with an output action to a data plane port. Verify that an ofp_error_msg is NOT received until 90% of max_entries installed flows are exceeded.

Delete all flows.

Attempt to add max_entries flow matching on all supported fields, with different priorities, with an output action to a data plane port. Verify that an ofp_error_msg is NOT received until 90% of max_entries installed flows are exceeded.

Specification text
The max_entries field describes the maximum number of flow entries that can be inserted into that table. Due to limitations imposed by modern hardware, the max_entries value should be considered advisory and a best effort approximation of the capacity of
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.1; pg. 90)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
Test with a primitive flow and the most complex flow supported by the device. Check if variable size action sets influence flow table capacity.

320.260 - Table features max name length

Protocol Messages / OFPT_MULTIPART_REPLY

Purpose
Verify that the table's name can be set to up to 32 characters in length.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and a body including at least one ofp_table_features message with the name field set to "something". Verify that a response is received and the ofp_table_features' name field is set to "something".

Send a second ofp_multipart_request with type OFPMP_TABLE_FEATURES, and a body including at least one ofp_table_features message with the name field set to a string of length 32 (the 32nd byte should not be the null byte). Verify that if modifying table features is not supported an OFPET_BAD_REQUEST error is generated with an OFPBRC_BAD_LEN code. If modifying table features is disabled, an OFPET_TABLE_FEATURES_FAILED error with an OFPTFFC_EPERM code should be received instead. Otherwise, verify a table_features_reply is received and the ofp_table_features' is set to all but the last character, or that no change occurred.

Specification text
OFP_MAX_TABLE_NAME_LEN is 32
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.1; pg. 90)

Topology
One control plane connection

Results
Pass / Fail

320.470 - Table features property required instructions

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_feature_prop_instructions

Purpose
Verify that all require instructions are supported.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and no body. Check that all the required instruction fields are supported in one table. Document all tables and their instruction capabilities.

Specification text
The instruction_ids is the list of instructions supported by this table (see 5.9). The elements of that list are of variable size to enable expressing experimenter instructions, non-experimenter instructions are 4 bytes.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 91)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
We need to document all the tables and their instruction fields. This allows us later to use the correct instruction types in the correct tables.

320.480 - Table features property next tables

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_feature_prop_next_tables / uint8_t next_table_ids[0];

Purpose
Verify that the next_tables_id field correctly reports all table ids.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and no body. Check that the switch can send an ofp_multipart_reply with type OFPMP_TABLE_FEATURES. If more than 1 ofp_table_features structs are returned in the ofp_multipart_reply, verify that the length of next_table_ids in the ofp_table_feautre_prop_next_tables struct is not zero. If only 1 ofp_table_feautres struct is returned in the ofp_multipart_reply, verify that the length of next_table_ids in the ofp_table_features_prop_next_tables is zero.

Specification text
The next_table_ids is the array of tables that can be directly reached from the present table using the OFPIT_GOTO_TABLE instruction (see 5.1).
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 91)

Topology
One control plane connection

Results
Pass / Fail

320.500 - Table features property write actions miss

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_feature_prop_actions / OFPTFPT_WRITE_ACTIONS_MISS,

Purpose
Verify that all reported actions can be used with the write actions instruction on table miss entries.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES and no message body. Verify that all reported actions under OFPTFPT_WRITE_ACTIONS_MISS can be used with a table_miss flow entry. This can be done by installing one table_miss flow entry per reported action, and checking that no error message is generated (given the correct prerequisites included in the match). If the OFPTFPT_WRITE_ACTIONS_MISS table property is omitted, verify that all reported actions under OFPTFPT_WRITE_ACTIONS can be used with a table_miss flow entry. If the OFPTFPT_WRITE_ACTIONS_MISS table property includes an empty action_ids list, verify that all reported actions under OFPTFPT_WRITE_ACTIONS generate an OFPT_ERROR message with type OFPET_BAD_ACTION and code OFPBAC_BAD_TYPE when used while installing a table_miss flow entry.

Specification text
The OFPTFPT_WRITE_ACTIONS and OFPTFPT_WRITE_ACTIONS_MISS properties describe actions supported by the table using the OFPIT_WRITE_ACTIONS instruction, whereas the OFPTFPT_APPLY_ACTIONS and OFPTFPT_APPLY_ACTIONS_MISS properties describe actions supported by the table using the OFPIT_APPLY_ACTIONS instruction.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 92)

Topology
One control plane connection

Results
Pass / Fail

320.510 - Table features property apply actions

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_feature_prop_actions / OFPTFPT_APPLY_ACTIONS,

Purpose
Verify that all reported actions can be used with the apply actions instruction.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES and no message body. Verify that all reported actions under OFPTFPT_APPLY_ACTIONS can be used with a regular flow entry. This can be done by installing one flow entry per reported action, and checking that no error message is generated (given the correct prerequisites included in the match). If the OFPTFPT_APPLY_ACTIONS table property is omitted, the device shall fail this test case. If the OFPTFPT_APPLY_ACTIONS table property includes an empty action_ids list, verify that OFPIT_APPLY_ACTIONS is not reported under OFPTFPT_INSTRUCTIONS.

Specification text
The OFPTFPT_WRITE_ACTIONS and OFPTFPT_WRITE_ACTIONS_MISS properties describe actions supported by the table using the OFPIT_WRITE_ACTIONS instruction, whereas the OFPTFPT_APPLY_ACTIONS and OFPTFPT_APPLY_ACTIONS_MISS properties describe actions supported by the table using the OFPIT_APPLY_ACTIONS instruction.

- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 92)

Topology
One control plane connection

Results
Pass / Fail

320.520 - Table features property apply actions miss

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_feature_prop_actions / OFPTFPT_APPLY_ACTIONS_MISS.

Purpose
Verify that all reported actions can be used with the apply actions instruction on table miss entries.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES and no message body. Verify that all reported actions under OFPTFPT_APPLY_ACTIONS_MISS can be used with a table_miss flow entry. This can be done by installing one table_miss flow entry per reported action, and checking that no error message is generated (given the correct prerequisites included in the match). If the OFPTFPT_APPLY_ACTIONS_MISS table property is omitted, verify that all reported actions under OFPTFPT_APPLY_ACTIONS can be used with a table_miss flow entry. If the OFPTFPT_APPLY_ACTIONS_MISS table property includes an empty action_ids list, verify that all reported actions under OFPTFPT_APPLY_ACTIONS generate an OFPT_ERROR message with type OFPET_BAD_ACTION and code OFPBAC_BAD_TYPE when used while installing a table_miss flow entry.

Specification text
The OFPTFPT_WRITE_ACTIONS and OFPTFPT_WRITE_ACTIONS_MISS properties describe actions supported by the table using the OFPIT_WRITE_ACTIONS instruction, whereas the OFPTFPT_APPLY_ACTIONS and OFPTFPT_APPLY_ACTIONS_MISS properties describe actions supported by the table using the OFPIT_APPLY_ACTIONS instruction.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 92)

Topology
One control plane connection

Results
Pass / Fail

320.540 - Table features property write actions

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_action_header action_ids[0] / OFPTFPT_WRITE_ACTIONS

Purpose
Verify that all reported actions can be used with the write actions instruction.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES and no message body. Verify that all reported actions under OFPTFPT_WRITE_ACTIONS can be used with a regular flow entry. This can be done by installing one flow entry per reported action, and checking that no error message is generated (given the correct prerequisites included in the match). If the OFPTFPT_WRITE_ACTIONS table property is omitted, the device shall fail this test case. If the OFPTFPT_WRITE_ACTIONS table property includes an empty action_ids list, verify that OFPIT_WRITE_ACTIONS is not reported under OFPTFPT_INSTRUCTIONS.

Specification text
The OFPTFPT_WRITE_ACTIONS and OFPTFPT_WRITE_ACTIONS_MISS properties describe actions supported by the table using the OFPIT_WRITE_ACTIONS instruction, whereas the OFPTFPT_APPLY_ACTIONS and OFPTFPT_APPLY_ACTIONS_MISS properties describe actions supported by the table using the OFPIT_APPLY_ACTIONS instruction.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 92)

Topology
One control plane connection

Results
Pass / Fail

330 - Multipart Reply Section Three

Test suite 330 verifies the correct implementation of the fields contained in each of the following message structs: ofp_table_feature_prop_oxm, and oxm_ids.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

330.20 - Table features wildcards

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_feature_prop_oxm

Purpose
Verify that all reported wildcard OXMs can be matched against.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and no body. Identify all fields reported as supported. Then for each field received from the switch in the reply under OFPTFPT_WILDCARDS, send a flow message with that field wildcarded. Verify that errors are not returned.

Specification text
OFPTFPT_WILDCARDS
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 91)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
Test with multiple iterations if required to cover mutually exclusive fields. For fields that are not advertised by the DUT as supported under OFPTFPT_WILDCARDS property, use properly formatted non-wildcarded OXM types and test with standard formatted packets using expected pre-requisites.

330.30 - Table features write set fields

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_feature_prop_oxm

Purpose
Verify that all reported write set field OXMs can be set.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and no body. Verify that all reported fields under OFPTFPT_WRITE_SETFIELD can be set (given the correct prerequisites included in the match) using the write instruction.

Specification text
OFPTFPT_WRITE_SETFIELD
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 91)

Topology
One control plane connection

Results
Pass / Fail

330.40 - Table features write set fields miss

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_feature_prop_oxm

Purpose
Verify that all reported write set field OXMs can be set on a table miss entry.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and no body. Verify that all reported fields under OFPTFPT_WRITE_SETFIELD_MISS can be set for the table_miss flow entry using the write instruction.

Specification text
OFPTFPT_WRITE_SETFIELD_MISS
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 91)

Topology
One control plane connection

Results
Pass / Fail

330.50 - Table features apply set fields

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_feature_prop_oxm

Purpose
Verify that all reported apply set field OXMs can be set.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and no body. Verify that all reported fields under OFPTFPT_APPLY_SETFIELD can be matched on using the apply instruction.

Specification text
OFPTFPT_APPLY_SETFIELD
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 91)

Topology
One control plane connection

Results
Pass / Fail

330.60 - Table features apply set fields miss

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_table_feature_prop_oxm

Purpose
Verify that all reported apply set field OXMs can be set on a table miss entry.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and no body. Verify that all reported fields under OFPTFPT_APPLY_SETFIELD_MISS can be matched on for the table_miss flow entry using the apply instruction.

Specification text
OFPTFPT_APPLY_SETFIELD_MISS
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 91)

Topology
One control plane connection

Results
Pass / Fail

330.70 - Table features match

Protocol Messages / OFPT_MULTIPART_REPLY / uint32_t oxm_ids[0]

Purpose
Verify that all reported match OXMs can be matched against.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and no body. Then for each advertised field received from the switch in the reply under OFPTFPT_MATCH send a flow message matching on that field. Verify that any error messages with type OFPET_BAD_MATCH and code OFPBMC_BAD_TYPE are not returned.

Specification text
OFPTFPT_MATCH

The oxm_ids field is the list of OXM types for the feature (see 7.2.3.2). The elements of that list are 32-bit OXM headers for non-experimenter OXM fields or 64-bit OXM headers for experimenter OXM fields. The OFPTFPT_MATCH property indicates the fields for which that particular table supports matching on (see 7.2.3.7).
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 92)

Topology
One control plane connection

Results
Pass / Fail

330.80 - Table features match and wildcard

Protocol Messages / OFPT_MULTIPART_REPLY / uint32_t oxm_ids[0]

Purpose
Verify that all reported match OXMs can be wildcarded, and matched against.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request with type OFPMP_TABLE_FEATURES, and no body. Then for each advertised field received from the switch in the reply under OFPTFPT_MATCH, send a flow message matching on that field with proper masking if HASMASK bit is set. Verify that any error messages with type OFPET_BAD_MATCH and code OFPBMC_*_MASK are not returned.

Specification text
If the HASMASK bit is set on the OXM header then the switch must support masking for the given type.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 93)

Topology
One control plane connection

Results
Pass / Fail

330.110 - Table features read only

Protocol Messages / OFPT_MULTIPART_REPLY / uint32_t oxm_ids[0]

Purpose
Verify that the max_entries field is read only.

Methodology
Configure and connect DUT to controller. Verify through a table_feature_request message that the max_entries field is read only. Verify that if modifying table features is not supported, an OFPET_BAD_REQUEST error is generated with an OFPBRC_BAD_LEN code. If modifying table features is disabled, an OFPET_TABLE_FEATURES_FAILED error with an OFPTFFC_EPERM code should be received instead. Otherwise, verify that a table_features_failed error is returned, or that the max_entries field reflects no change.

Specification text
All fields in ofp_table_features may be requested to be changed by the controller with the exception of the max_entries field, this is read only and returned by the switch.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.5.2; pg. 93)

Topology
One control plane connection

Results
Pass / Fail

340 - Multipart Reply Section Four

Test suite 340 verifies the correct implementation of the fields contained in each of the following message structs: ofp_port_stats_request, ofp_port_stats, and ofp_port.

Some counters may not be reliably triggered on every device. In these cases only the existence of the counter and the reasonableness of the return value will be verified. For example, install a flow, send one valid ip packet and check the crc error counter. The counter should be "0", or "-1" if not supported.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

340.20 - Port filter reserved

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats_request / uint32_t port_no;

Purpose

The port_no field optionally filters the stats request to the given port. To request all port statistics, port_no must be set to OFPP_ANY. The response is reported in ofp_port_stats structs.

Methodology

Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, with a port_no equal to OFPP_ANY. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Verify that each configured test port's statistics are reported using the port_no field in the reported ofp_port_stats structs.

Specification text

OFPMP_PORT message must request statistics either for a single port (specified in port_no) or for all ports (if port_no == OFPP_ANY).
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 94)

Topology

One control plane connection and Data plane connections as needed (at least one)

Results

Pass / Fail

340.40 - Port filter standard

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint32_t port_no;

Purpose
Verify that the port_no field optionally filters the stats request to the given port.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured test port number. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Verify that a single ofp_port_stats struct is returned and that the port_no field of the response is equal to the configured test port's number.

Specification text
OFPMP_PORT message must request statistics either for a single port (specified in port_no) or for all ports (if port_no == OFPP_ANY).
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 94)

Topology
One control plane connection and data plane connections as needed (at least one)

Results
Pass / Fail

340.50 - Received packets

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint64_t rx_packets;

Purpose
Verify that the received packets counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the rx_packets field of the ofp_port_stats struct sent by the device. If the rx_packets field is -1 the rx_packets counter is not supported. On the configured data plane test port send N packets. Send a second ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to the configured data plane test port. Verify that that the reported rx_packets field of the ofp_port_stats struct is equal to the recorded rx_packets field plus N.

Specification text
Number of received packets.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

340.60 - Transmitted packets

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint64_t tx_packets;

Purpose
Verify that the transmitted packets counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the tx_packets field of the ofp_port_stats struct sent by the device. If the tx_packets field is -1, then the tx_packets counter is not supported. Install a fully wildcarded ofp_flow_mod with an apply_actions instruction with an output port equal to the configured data plane test port. On a second configured data plane test port send N packets. Send a second ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to the configured data plane test port. Verify that that the reported tx_packets field of the ofp_port_stats struct is equal to the recorded tx_packets field plus N.

Specification text
Number of transmitted packets.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

340.70 - Received bytes

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint64_t rx_bytes;

Purpose
Verify that the received bytes counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the rx_bytes field of the ofp_port_stats struct sent by the device. If the rx_bytes field is -1 the rx_bytes counter is not supported. On the configured data plane test port send N packets of length 150. Send a second ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to the configured data plane test port. Verify that that the reported rx_bytes field of the ofp_port_stats struct is equal to the recorded rx_bytes field plus (N * 150).

Specification text
Number of received bytes.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

340.80 - Transmitted bytes

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint64_t tx_bytes;

Purpose
Verify that the transmitted bytes counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the tx_bytes field of the ofp_port_stats struct sent by the device. If the tx_bytes field is -1 the tx_bytes counter is not supported. Install a fully wildcarded ofp_flow_mod with an apply_actions instruction with an output port equal to the configured data plane test port. On a second configured data plane test port send N packets with a length of 150. Send a second ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to the configured data plane test port. Verify that that the reported tx_bytes field of the ofp_port_stats struct is equal to the recorded tx_bytes field plus (N * 150).

Specification text
Number of transmitted bytes.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

340.90 - Received dropped

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint64_t rx_dropped;

Purpose
Verify that the received drops counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the rx_dropped field of the ofp_port_stats struct sent by the device. If the rx_dropped field is -1 the rx_dropped counter is not supported. Configure port flags to enable OFPPF_NO_RECV. Attempt to forward N packets to the configured port. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. If the rx_dropped field of the response is -1 the rx_dropped counter is not supported. Otherwise verify rx_dropped has increased by N.

Specification text
Number of packets dropped by RX.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

Test recommendations
The DUT should pass 210.60 for this test to be run

340.100 - Transmitted dropped

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint64_t tx_dropped;

Purpose
Verify that the transmitted drops counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the tx_dropped field of the ofp_port_stats struct sent by the device. Configure port flags to enable OFPPF_NO_FORWARD. Attempt to forward N packets out the configured port. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. If the tx_dropped field of the response is -1 the tx_dropped counter is not supported. Otherwise verify tx_dropped has increased by N.

Specification text
Number of packets dropped by TX.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

Test recommendations
The DUT should pass 210.70 for this test to be run

340.110 - Received errors

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint64_t rx_errors;

Purpose
Verify that the received errors counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the rx_errors field of the ofp_port_stats struct sent by the device. If the rx_errors field is -1 the rx_errors counter is not supported. Otherwise verify rx_errors is equal to the sum of all rx_*_err fields.

Specification text
Number of receive errors. This is a superset of more specific receive errors and should be greater than or equal to the sum of all rx_*_err values.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

340.120 - Transmitted errors

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint64_t tx_errors;

Purpose
Verify that the transmitted errors counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the tx_errors field of the ofp_port_stats struct sent by the device. If the tx_errors field is -1 the tx_errors counter is not supported. Otherwise verify tx_errors is equal to the sum of all tx_*_err fields.

Specification text
Number of transmit errors. This is a superset of more specific transmit errors and should be greater than or equal to the sum of all tx_*_err values (none currently defined.)
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

340.130 - Received frame errors

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint64_t rx_frame_err;

Purpose
Verify that the received frame alignment errors counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the rx_frame_err field of the ofp_port_stats struct sent by the device. If the rx_frame_err field is -1 the rx_frame_err counter is not supported, otherwise add a flow matching on a named field (under the given Pre-requisites for the match). Send matching traffic on the data plane which triggers N rx_frame_errs. Using a second ofp_multipart_request verify the rx_frame_err field was incremented by N.

Specification text
Number of frame alignment errors.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

340.140 - Received overrun errors

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint64_t rx_over_err;

Purpose
Verify that the received overrun errors counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the rx_over_err field of the ofp_port_stats struct sent by the device. If the rx_over_err field is -1 the rx_over_err counter is not supported, otherwise add a flow matching on a named field (under the given Pre-requisites for the match). Send matching traffic on the data plane which triggers N rx_over_errs. Using a second ofp_multipart_request verify the rx_over_err field was incremented by N.

Specification text
Number of packets with RX overrun.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

340.150 - Received CRC errors

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint64_t rx_crc_err;

Purpose
Verify that the received CRC errors counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the rx_crc_err field of the ofp_port_stats struct sent by the device. If the rx_crc_err field is -1 the rx_crc_err counter is not supported, otherwise add a flow matching on a named field (under the given Pre-requisites for the match). Send matching traffic on the data plane which triggers N rx_crc_errs. Using a second ofp_multipart_request verify the rx_crc_err field was incremented by N.

Specification text
Number of CRC errors.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

340.160 - Collision errors

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint64_t collisions;

Purpose
Verify that the collisions counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the collisions field of the ofp_port_stats struct sent by the device. If the collisions field is -1 the collisions counter is not supported, otherwise add a flow matching on a named field (under the given Pre-requisites for the match). Send matching traffic on the data plane which triggers N collisions. Using a second ofp_multipart_request verify the collisions field was incremented by N.

Specification text
Number of collisions.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

340.170 - Port duration in seconds

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint32_t duration_sec;

Purpose
Verify that the duration in seconds counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the duration_sec field. Wait N seconds. Send a second ofp_multipart_request with type OFPMP_PORT_STATS, with a port_no equal to the previous data plane port. Verify that the duration_sec field of the response is equal to duration_sec + N.

Specification text
Time port has been alive in seconds.

The duration_sec and duration_nsec fields indicate the elapsed time the port has been configured into the OpenFlow pipeline. The total duration in nanoseconds can be computed as duration_sec*10^9 + duration_nsec. Implementations are required to provide second precision; higher precision is encouraged where available.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

340.180 - Port duration in nanoseconds

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port_stats / uint32_t duration_nsec;

Purpose
Verify that the duration in nanoseconds counter increments correctly.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to a configured data plane test port. Check that the switch can send an ofp_multipart_reply with type OFPMP_PORT_STATS. Record the duration_sec*10^9 + duration_nsec (if nsec counter is supported). Wait N seconds. Send a second ofp_multipart_request with type OFPMP_PORT_STATS, with a port_no equal to the previous data plane port. Verify that the duration_sec*10^9 + duration_nsec (if nsec counter is supported) is roughly equal to the previously recorded value plus N seconds.

Specification text
Time port has been alive in nanoseconds beyond duration_sec.

The duration_sec and duration_nsec fields indicate the elapsed time the port has been configured into the Openflow pipeline. The total duration in nanoseconds can be computed as duration_sec*10^9 + duration_nsec. Implementations are required to provide second precision; higher precision is encouraged where available.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.6; pg. 95)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

340.200 - Unique port number

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port / uint32_t port_no;

Purpose
Verify that all ports reported in response to an OFPMP_PORT_DESC have a unique non-negative port number.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request of type OFPMP_PORT_DESC. Verify that all data plane test ports have a unique non-negative port number greater than zero.

Specification text
The port description request OFPMP_PORT_DESCRIPTION enables the controller to get a description of all the standard ports of the OpenFlow switch (see 4.2). The request body is empty. The reply body consists of an array of the following:
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.7; pg. 95)

Topology
One control plane connection and data plane connections as needed (at least one)

Results
Pass / Fail

Test recommendations
All port related tests should be run at several different port types, speeds and configurations (Fiber / Copper / Autoneg / 10/100/1000). It is recommended to trigger as many different supported port types as possible on the device.

Additional remarks
The report must indicate all port types and configurations tested.

340.220 - Unique hardware address

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port / uint8_t hw_addr

Purpose
Verify that all ports reported in response to an OFPMP_PORT_DESC have a valid hardware address.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request of type OFPMP_PORT_DESC. Verify that all data plane test ports have a valid hw_addr value. Send an ofp_port_mod with a body including at least one ofp_port struct with the hw_addr field set to a unique hardware address. Verify that an ofp_error_msg is received. Otherwise the port's hw_addr field is not changed.

Specification text
uint8_t hw_addr[OFP_ETH_ALEN];
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.7; pg. 95)

Topology
One control plane connection and data plane connections as needed (at least one)

Results
Pass / Fail

Test recommendations
All port related tests should be run at several different port types, speeds and configurations (Fiber / Copper / Autoneg / 10/100/1000). It is recommended to trigger as many different supported port types as possible on the device.

340.240 - Port name

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port / char name

Purpose
Verify that an OFPMP_PORT_DESC message can be used to set the name field on a port.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request of type OFPMP_PORT_DESC. Verify that all data plane test ports have a name value with a length less than 16 (16th byte is reserved for null).

Specification text
char name[OFP_MAX_PORT_NAME_LEN]; /* Null-terminated */
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.7; pg. 95)

Topology
One control plane connection and data plane connections as needed (at least one)

Results
Pass / Fail

Test recommendations
All port related tests should be run at several different port types, speeds and configurations (Fiber / Copper / Autoneg / 10/100/1000). It is recommended to trigger as many different supported port types as possible on the device.

340.260 - Port state

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port / uint32_t state;

Purpose
Verify that each port's state is correctly reported.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request of type OFPMP_PORT_DESC. Verify that all data plane test ports have a state value of zero. Verify that all other reported interfaces have a state field with the OFPPS_LINK_DOWN bit set. Disconnect one of the data plane test ports. Send an ofp_multipart_request of type OFPMP_PORT_DESC. Verify that disconnected data plane test port has a state field with the OFPPS_LINK_DOWN bit set. Reconnect the port.

Specification text
Bitmap of OFPPS_* flags.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.7; pg. 95)

Topology
One control plane connection and data plane connections as needed (at least one)

Results
Pass / Fail

Test recommendations
All port related tests should be run at several different port types, speeds and configurations (Fiber / Copper / Autoneg / 10/100/1000). It is recommended to trigger as many different supported port types as possible on the device. When using logical ports there should be a meaningful and complete mapping between the logical port, and the underlying transport. For example, if a logical port is a tunnel, the port state represents whether the tunnel is up or down.

340.270 - Current features

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port / uint32_t curr;

Purpose
Verify that curr is set to the features negotiated on a port's link.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request of type OFPMP_PORT_DESC. Verify that all data plane test ports have a uint32_t curr value equal to the features reported by the test tool connected to the port (the features negotiated by the two link endpoints). Verify that unconnected ports report an empty current feature set.

Specification text
Current features.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.7; pg. 95)*

Topology
One control plane connection and data plane connections as needed (at least one)

Results
Pass / Fail

Test recommendations
All port related tests should be run at several different port types, speeds and configurations (Fiber / Copper / Autoneg / 10/100/1000). It is recommended to trigger as many different supported port types as possible on the device.

340.280 - Advertised features

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port / uint32_t advertised;

Purpose
Verify that advertised is set to the features configured to be advertised by the device's port.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request of type OFPMP_PORT_DESC. Verify that the switch advertises the expected features per port, and that the current features are part of the advertised features. The DUT must return 0x0 (Advertised = all 0) if it does not support the "advertised" field.

Specification text
Features being advertised by the port.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.7; pg. 95)

Topology
One control plane connection and data plane connections as needed (at least one)

Results
Pass / Fail

Test recommendations
All port related tests should be run at several different port types, speeds and configurations (Fiber / Copper / Autoneg / 10/100/1000). It is recommended to trigger as many different supported port types as possible on the device.

340.290 - Supported features

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port / uint32_t supported;

Purpose
Verify that supported is set to the features supported by the device's port.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request of type OFPMP_PORT_DESC. Verify that the switch supports the expected features per port, and that the advertised features are equal or a subset of the supported features. The DUT must return 0x0 (Supported = all 0) if it does not support the "support" field.

Specification text
Features supported by the port.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.7; pg. 95)

Topology
One control plane connection and data plane connections as needed (at least one)

Results
Pass / Fail

Test recommendations
All port related tests should be run at several different port types, speeds and configurations (Fiber / Copper / Autoneg / 10/100/1000). It is recommended to trigger as many different supported port types as possible on the device.

340.300 - Peer's features

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port / uint32_t peer;

Purpose
Verify that peer is set to the features advertised by the peer's port.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request of type OFPMP_PORT_DESC. Verify that the switch reports the features advertised by peer as expected. The DUT must return 0x0 (Peer = all 0) if it does not support the "peer" field. The expected peer values are known from the test tool.

Specification text
Features advertised by peer.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.7; pg. 85, 44)

Topology
One control plane connection and data plane connections as needed (at least one)

Results
Pass / Fail

Test recommendations
All port related tests should be run at several different port types, speeds and configurations (Fiber / Copper / Autoneg / 10/100/1000). It is recommended to trigger as many different supported port types as possible on the device.

340.310 - Current bit rate

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port / uint32_t curr_speed;

Purpose
Verify that the port's current bit rate is correctly reported.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request of type OFPMP_PORT_DESC. Verify that the switch reports the current bit rate as expected.

Specification text
The curr_speed and max_speed fields indicate the current and maximum bit rate (raw transmission speed) of the link in kbps. The number should be rounded to match common usage. For example, an optical 10 Gb Ethernet port should have this field set to 10000000 (instead of 10312500), and an OC-192 port should have this field set to 10000000 (instead of 9953280). The max_speed fields indicate the maximum configured capacity of the link, whereas the curr_speed indicates the current capacity. If the port is a LAG with 3 links of 1Gb/s capacity, with one of the ports of the LAG being down, one port auto-negotiated at 1Gb/s and 1 port auto-negotiated at 100Mb/s, the max_speed is 3 Gb/s and the curr_speed is 1.1 Gb/s.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.2.1; pg. 51)*

Topology
One control plane connection and data plane connections as needed (at least one)

Results
Pass / Fail

Test recommendations
All port related tests should be run at several different port types, speeds and configurations (Fiber / Copper / Autoneg / 10/100/1000). It is recommended to trigger as many different supported port types as possible on the device.

340.320 - Max bitrate

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_port / uint32_t max_speed;

Purpose
Verify that the port's maximum bit rate is correctly reported.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_multipart_request of type OFPMP_PORT_DESC. Verify that the switch reports the max bit rate as expected.

Specification text
The curr_speed and max_speed fields indicate the current and maximum bit rate (raw transmission speed) of the link in kbps. The number should be rounded to match common usage. For example, an optical 10 Gb Ethernet port should have this field set to 10000000 (instead of 10312500), and an OC-192 port should have this field set to 10000000 (instead of 9953280). The max_speed fields indicate the maximum configured capacity of the link, whereas the curr_speed indicates the current capacity. If the port is a LAG with 3 links of 1Gb/s capacity, with one of the ports of the LAG being down, one port auto-negotiated at 1Gb/s and 1 port auto-negotiated at 100Mb/s, the max_speed is 3 Gb/s and the curr_speed is 1.1 Gb/s.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.2.1; pg. 51)*

Topology
One control plane connection and data plane connections as needed (at least one)

Results
Pass / Fail

Test recommendations
All port related tests should be run at several different port types, speeds and configurations (Fiber / Copper / Autoneg / 10/100/1000). It is recommended to trigger as many different supported port types as possible on the device.

380 - Multipart Reply Section Five

Test suite 380 verifies the correct implementation of the fields contained in each of the following message structs: ofp_queue_stats, ofp_get_queue_config_request, and ofp_packet_queue.

Remarks

Queue support

Support for queues are optional, however a device MUST still respond appropriately to such requests. Devices with no queue support should respond with an error message when attempting to use the unsupported queue message types. Otherwise devices are expected to report all configured queues for each requested port.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

380.40 - Queue stats request reserved

Protocol Messages / OFPT_MULTIPART_REQUEST / ofp_queue_stats

Purpose
Check that a queue stats request with a port field set to OFPP_ANY results in a queue stats reply that includes each test port's configured queues.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_QUEUE, with port_no equal to OFPP_ANY. If the switch supports queues check that the switch can send an ofp_multipart_reply with type OFPMP_QUEUE, otherwise verify DUT replies with an error message of type OFPET_BAD_REQUEST and code OFPBRC_BAD_MULTIPART. If the switch supports queues, verify that each configured test port's queues are reported using the port_no field in the ofp_queue_stats structs.

Specification text
The OFPMP_QUEUE multipart request message provides queue statistics for one or more ports and one or more queues. struct ofp_queue_stats_request {
uint32_t port_no; /* All ports if OFPP_ANY. */
uint32_t queue_id; /* All queues if OFPQ_ALL. */
};
OFP_ASSERT(sizeof(struct ofp_queue_stats_request) == 8);
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.8; pg. 96)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

Test recommendations
If a queue has been configured on the device we should expect > 0 ofp_queue_stats structs to be returned. Devices without configured queues should return an empty list.

380.50 - Queue stats request standard

Protocol Messages / OFPT_MULTIPART_REQUEST / ofp_queue_stats / uint32_t port;

Purpose
Check that a queue stats request for a configured test port results in a queue stats reply that includes the test port's configured queues.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_QUEUE, with port_no equal to a configured test port number. Check that the switch can send an ofp_multipart_reply with type OFPMP_QUEUE. Verify that any ofp_queue_stats structure that is returned has a port_no field equal to the configured test port's number.

Specification text
The request body contains a port_no field identifying the OpenFlow port for which statistics are requested, or OFPP_ANY to refer to all ports. The queue_id field identifies one of the priority queues, or OFPQ_ALL to refer to all queues configured at the specified port. OFPQ_ALL is 0xffffffff. struct ofp_queue_stats_request {
uint32_t port_no; /* All ports if OFPP_ANY. */
uint32_t queue_id; /* All queues if OFPQ_ALL. */
};
OFP_ASSERT(sizeof(struct ofp_queue_stats_request) == 8);
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.5.8; pg. 96)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

Test recommendations
If a queue has been configured on the device we should expect > 0 ofp_queue_stats structs to be returned. Devices without configured queues should return an empty list.
no_queue_support => error
no_configured_queue => empty_list
configured_queue => list of queues

380.80 - Queue stats

Protocol Messages / OFPT_MULTIPART_REPLY / ofp_queue_stats / ofp_packet_queue queues[0]

Purpose
Verify that the correct number of queues is reported for each configured test port.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_QUEUE, with port_no equal to a configured test port number. Check that the switch can send an ofp_multipart_reply with type OFPMP_QUEUE. Verify that devices without queue support return an error message. Otherwise verify the length of queues is equal to the number of configured queues per port.

Specification text
The switch replies back with an ofp_queue_get_config_reply message, containing a list of configured
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.6; pg. 101)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

380.90 - Queue config request reserved

Protocol Messages / OFPT_QUEUE_GET_CONFIG_REQUEST / ofp_queue_get_config_request

Purpose

Check that an ofp_queue_get_config_request with a port field set to OFPP_ANY results in an ofp_queue_get_config_reply that includes each test port's configured queues.

Methodology

Configure and connect DUT to controller. Send an ofp_queue_get_config_request with port_no equal to OFPP_ANY. If the device supports queues, verify that the switch sends a reply for each configured port. The device may also return an error message of OFPET_BAD_REQUEST with a code of OFPBRC_BAD_TYPE if queues are not supported.

Specification text

Query for port queue configuration.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.6; pg. 101)

Topology

One control plane connection and at least one data plane connection

Results

Pass / Fail

380.100 - Queue config request standard

Protocol Messages / OFPT_QUEUE_GET_CONFIG_REQUEST / ofp_queue_get_config_request / uint32_t port;

Purpose

Check that an ofp_queue_get_config_request for a configured test port results in an ofp_queue_get_config_reply that includes the test port's configured queues.

Methodology

Configure and connect DUT to controller. Send an ofp_queue_get_config_request with port equal to an existing data plane port. Check that the switch replies with a single empty ofp_queue_config_reply message with port equal to the queried data plane port number. The device may also return an error message of OFPET_BAD_REQUEST with a code of OFPBRC_BAD_TYPE if queues are not supported.

Specification text

Port to be queried. Should refer to a valid physical port (i.e. < =OFPP_MAX), or OFPP_ANY to request all configured queues.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.6; pg. 101)

Topology

One control plane connection and one data plane connection

Results

Pass / Fail

380.130 - Queue configuration

Protocol Messages / OFPT_QUEUE_GET_CONFIG_REPLY / ofp_queue_get_config_reply / ofp_packet_queue queues[0]

Purpose
Verify that the correct number of queues is reported for each configured test port.

Methodology
Configure and connect DUT to controller. Send an ofp_queue_get_config_request with port equal to a configured data plane port. Check that the switch replies with a single ofp_queue_config_reply message with port equal to the configured data plane port number. Verify that devices without queue support return an error message. Otherwise verify that the length of queues is equal to the number of configured queues on that port.

Specification text
The switch replies back with an ofp_queue_get_config_reply message, containing a list of configured queues.

- OpenFlow Switch Specification 1.3.4 (ch. 7.3.6; pg. 101)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

390 - Packet Out Protocol Message

Test suite 390 verifies the device correctly implements the packet out message type.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

390.30 - Packet out in port

Protocol Messages / Packet-Out Message / ofp_header header / uint32_t in_port;

Purpose
Verify that flow matching on in_port OFPP_CONTROLLER and output action data plane port is matched against when a packet_out message is sent with in_port as OFPP_CONTROLLER and output action OFPP_TABLE.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on in_port OFPP_CONTROLLER with an output action to a data plane test port. Send and receive a barrier request and reply. Generate a non-buffered ofp_packet_out message with an in_port set to OFPP_CONTROLLER, and an output action of OFPP_TABLE. Verify that the packet matches on the installed flow.

Specification text
Packet's input port or OFPP_CONTROLLER.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.7; pg. 102)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

Test recommendations
This should not only be tested against standard port numbers, but also against reserved logical ports like OFPP_CONTROLLER

390.70 - Packet out no buffer

Protocol Messages / Packet-Out Message / ofp_action_header actions[0]

Purpose
Verify that all data included in packet_out with buffer_id of OFP_NO_BUFFER is forwarded correctly.

Methodology
Configure and connect DUT to controller. Generate a non-buffered ofp_packet_out including a tcp packet, and an output action to a valid data plane test port. Verify that the tcp packet is received on the correct port.

Specification text
The buffer_id is the same given in the ofp_packet_in message. If the buffer_id is OFP_NO_BUFFER, then the packet data is included in the data array, and the packet encapsulated in the message is processed by the actions of the message.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.7; pg. 102)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

390.80 - Packet out buffer

Protocol Messages / Packet-Out Message / ofp_action_header actions[0]

Purpose
Verify that all data included in packet_out with buffer_id not equal to OFP_NO_BUFFER and valid is forwarded correctly.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the Pre-requisites for the match) with action as output port CONTROLLER. Forward traffic on a data plane port, and verify that an ofp_packet_in message is received on the control plane. Generate an ofp_packet_out with buffer_id equal to the buffer_id of the received ofp_packet_in, and an output action to a second valid data plane test port. Verify that the tcp packet is received on the correct port.

Specification text
OFP_NO_BUFFER is 0xffffffff. If buffer_id is valid, the corresponding packet is removed from the buffer and processed by the actions of the message.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.7; pg. 102)*

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

390.90 - Packet out invalid in port

Protocol Messages / Packet-Out Message / ofp_action_header actions[0]

Purpose
Verify that a packet_out message with an invalid in_port generates an OFPT_ERROR with appropriate type and code.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on in_port N with an output action to a data plane test port. Generate a non-buffered ofp_packet_out message with an in_port set to N, and an output action of OFPP_TABLE. Verify that the packet matches on the installed flow. Generate a non-buffered ofp_packet_out message with an invalid in_port, and an output action of OFPP_TABLE. Verify that an OFPET_BAD_REQUEST error is received with an error code of OFPBRC_BAD_PORT.

Specification text
The in_port field specifies the ingress port that must be associated with the packet for OpenFlow processing. It must be set to either a valid standard switch port (see 4.2) or OFPP_CONTROLLER. For example, this field is used when processing the packet using groups, OFPP_TABLE, OFPP_IN_PORT, and FPP_ALL.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.7; pg. 102)*

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

390.100 - Packet out actions

Protocol Messages / Packet-Out Message / ofp_action_header actions[0]

Purpose
Verify that packet_out message with action list is supported by the switch.

Methodology
Configure and connect DUT to controller. Generate a non-buffered ofp_packet_out including a TCP packet, a set-field action, an output action to port X, and if supported a group action to a group with an output action to port Y. Verify that the modified TCP packet is received on port X, and if groups are supported, port Y.

Specification text
The action field is a list of actions defining how the packet should be processed by the switch. It may include packet modification, group processing and an output port.
- OpenFlow Switch Specification 1.3.4 (ch. 7.3.7; pg. 102)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

390.110 - Packet out action table

Protocol Messages / Packet-Out Message / ofp_action_header actions[0]

Purpose
Verify that packet_out message with action outport OFPP_TABLE is supported by the switch.

Methodology
Configure and connect DUT to controller. Install a high priority flow (flow-1) matching on in_port and eth_src with an output action to OFPP_CONTROLLER. Install a second low priority flow (flow-2) entry with a match on eth_src, output to a data plane test port. Generate and forward a matching data plane packet to the device on in_port. Verify that an ofp_packet_in message is received. Based on the ofp_packet_in's buffer_id, generate an ofp_packet_out using the buffered packet, the in_port set to OFPP_Controller, and an output action of OFPP_TABLE. Verify that the packet matches on the installed flow-2 and is forwarded correctly.

Specification text
The list of actions of an OFPT_PACKET_OUT message can also specify the OFPP_TABLE reserved port as an output action to process the packet through the OpenFlow pipeline, starting at the first flow table (see 4.5). If OFPP_TABLE is specified, the in_port field is used as the ingress port in the flow table lookup.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.3.7; pg. 102)*

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

Test recommendations
If the device does not have any buffer, the test case can be run without using the switch side buffer

410 - Packet In Protocol Message

Test suite 410 verifies that the device correctly implements the packet in message type.

Remarks

Logical Interfaces

Some devices may not support logical interfaces. In this case, the result of logical interface tests MAY be recorded as ‘Not Applicable’.

Table miss behavior

In some cases a device may allow for an optional default packet behavior of “send to controller” or “generate packet in” when no table miss entry is present. In this case the length of the data field contained in the ofp_packet_in message should be no greater than miss_send_len (configured through the ofp_set_switch_config message). Table miss entries (fully wildcarded priority zero flows) that contain an output action to OFPP_CONTROLLER, MUST generate ofp_packet_in messages with data fields no greater than the max_len len field that was used during flow installation.

Basic Single Table Conformance Test Profile Requirements

To satisfy Basic Single Table conformance requirements an OpenFlow-enabled device MUST pass **all** test cases in this test suite except where result is ‘Not Applicable’ based on conditions as outlined in test case Remarks.

410.50 - Default miss send length

Protocol Messages / Packet-In Message

Purpose
Verify that the default miss_send_len is 128 bytes.

Methodology
Configure the DUT's default table-miss behavior to trigger an ofp_packet_in message.

Connect DUT to controller. After control channel establishment send an OFPT_GET_CONFIG_REQUEST message. Verify that the switch responds with an OFPT_GET_CONFIG_REPLY message. Verify that the returned miss_send_len value is 128.

Specification text
The default miss_send_len is 128 bytes.
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.1; pg. 106)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If not possible to configure the DUT's default table-miss behavior to trigger an ofp_packet_in message, the result of this test is 'Not Applicable'.

410.70 - Packet in buffer documentation

Protocol Messages / Packet-In Message

Purpose
Verify that devices expose available buffering to their users.

Methodology
Vendor must provide this documentation to complete the basic conformance test suite. If the proper documentation is not provided, this test case result shall be "fail."

Specification text
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.
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.1; pg. 105)

Topology
One control plane connection

Results
Pass / Fail

410.90 - Packet in buffer timeout

Protocol Messages / Packet-In Message

Purpose
Verify that buffers are protected until they have been used or have timed out.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is output to port CONTROLLER with max_len set to 0. Send a matching packet on the data plane. Note the timeout value (t-timeout) and max number of packets buffered (n-max) from the documentation. Send one data plane packet, and note the buffer_id. Wait 2 seconds, send n-max packets on the data plane. Verify that you get n-1 valid buffer_ids and one packet_in with buffer-id -1 and full packet in payload. Wait (t-timeout)-1s from the first packet, send another data plane packet, verify it is not buffered. Wait two more seconds, send 2 data plane packets. Verify that one is buffered, with the buffer-id from the very first packet_in, the other one is not buffered.

Specification text
A switch should prevent a buffer from being reused until it has been handled by the controller, or some amount of time (indicated in documentation) has passed.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.1; pg. 105)*

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If the device does not support buffering, the result of this test case MAY be ‘Not Applicable’.

If not possible to configure the DUT's default table-miss behavior to trigger an ofp_packet_in message, the result of this test MAY be 'Not Applicable'.

410.140 - Packet in reason action

Protocol Messages / Packet-In Message / enum ofp_packet_in_reason

Purpose
Verify that ofp_packet_in reason is correctly reported.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), action is output to port CONTROLLER. Send a matching packet on the data plane. Verify that a packet_in message encapsulates the matching packet is triggered. Verify that the reason is OFPR_ACTION.

Specification text
OFPR_ACTION = 1, /* Action explicitly output to controller. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.1; pg. 105)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

410.200 - Packet in cookie

Protocol Messages / Packet-In Message

Purpose
Verify that the cookie field of an ofp_packet_in message matches that of the ofp_flow_mod that triggered the ofp_packet_in.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), with an action output to port CONTROLLER. Note the cookie_id in the ofp_flow_mod. Send a matching packet on the data plane. Verify that a packet_in message that encapsulates the matching packet is triggered. Verify that the cookie field matches the cookie field of the ofp_flow_mod.

Specification text
The cookie field contains the cookie of the flow entry that caused the packet to be sent to the controller.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.1; pg. 106)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

410.220 - Packet in cookie negative one

Protocol Messages / Packet-In Message

Purpose
Verify that if a cookie cannot be associated with a flow, the cookie field is set to negative one.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match) with an action output to OFPP_CONTROLLER using the write_actions instruction. Send a matching packet on the data plane. Verify that a packet_in message that encapsulates the matching packet is triggered. Verify that the cookie field is set to -1.

Specification text
This field must be set to -1 (0xffffff) if a cookie cannot be associated with a particular flow. For example, if the packet-in was generated in a group bucket or from the action set.
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.1; pg. 106)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

410.240 - Packet in match

Protocol Messages / Packet-In Message

Purpose
Verify that the match field of a packet_in message is not empty.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), with an action output to OFPP_CONTROLLER. Send a matching packet on the data plane. Verify that a packet_in message that encapsulates the matching packet is triggered. Verify that the match field is not empty.

Specification text
The match field is a set of OXM TLVs containing the pipeline fields associated with a packet. The pipeline fields values cannot be determined from the packet data, and include for example the input port and the metadata value (see 7.2.3.9).
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.1; pg. 106)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

410.250 - In port match

Protocol Messages / Packet-In Message

Purpose
Verify that the match field of an ofp_packet_in contains an in_port OXM.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match) with an action output to OFPP_CONTROLLER. Send a matching packet on the data plane. Verify that a packet_in message that encapsulates the matching packet is triggered. Verify that the match field contains an OFPXMT_OFB_IN_PORT.

Specification text
The set of OXM TLVs must include all pipeline fields associated with that packet, supported by the switch and which value is not all-bits-zero. If OXM_OF_IN_PHY_PORT has the same value as OXM_OF_IN_PORT, it should be omitted from this set.

The set of OXM TLVs must reflect the packet's headers and context when the event that triggers the packet-in message occurred, they should include all modifications made in the course of previous processing.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.1; pg. 106)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

410.260 - Physical port match

Protocol Messages / Packet-In Message

Purpose
Verify that the in_phy_port OXM is included in the match field of ofp_packet_in messages in the correct instances.

Methodology
If logical interfaces are supported, configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match) with an action output to OFPP_CONTROLLER. Send a matching packet on a data plane logical port. Verify that a packet_in message that encapsulates the matching packet is triggered. Verify that the match field contains an OFPXMT_OFB_IN_PHY_PORT.

Specification text
OFPXMT_OFB_IN_PHY_PORT,

- OpenFlow Switch Specification 1.3.4 (ch. 7.4.1; pg. 58)

Topology
One control plane connection and one data plane connection with a logical interface configured.

Results
Pass / Fail / Not Applicable

Test recommendations
Should be tested against logical ports.

Additional remarks
This is to test logical interfaces if supported. If not supported, the result is Not Applicable.

410.280 - Tunnel id match

Protocol Messages / Packet-In Message

Purpose
Verify that the tunnel_id OXM is included in the match field of ofp_packet_in messages in the correct instances.

Methodology
If logical Interfaces are supported, configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), with an action output to OFPP_CONTROLLER. Send a matching packet on a data plane tunnel interface. Verify that a packet_in message that encapsulates the matching packet is triggered. Verify that the match field contains an OFPXMT_OFB_TUNNEL_ID.

Specification text
OFPXMT_OFB_TUNNEL_ID.
The mapping of the optional encapsulation metadata in the Tunnel ID field is defined by the logical port implementation, it is dependent on the type of logical port and it is implementation specific. We recommend that for a packet received via a GRE tunnel including a (32-bit) key, the key is stored in the lower 32-bits and the high bits are zeroed. We recommend that for an MPLS logical port, the lower 20 bits represent the MPLS Label. We recommend that for a VxLAN logical port, the lower 24 bits represent the VNI.
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.1; pg. 59)

Topology
One control plane connection and one data plane connection with a logical interface configured.

Results
Pass / Fail / Not Applicable

Test recommendations
Should be tested against logical tunnel interfaces.

Additional remarks
This is to test logical interfaces if supported. If not supported, the result is Not Applicable.

410.310 - Physical port match omissions

Protocol Messages / Packet-In Message

Purpose
Verify that ofp_packet_in messages generated due to traffic on physical ports report the correct in_port.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match), with an action output to port CONTROLLER. Send a matching packet on a physical (non-logical) data plane port. Verify that a packet_in message that encapsulates the matching packet is triggered. Verify that OFPXMT_OFB_IN_PHY_PORT is omitted, and that OFPXMT_OFB_IN_PORT is equal to the correct data plane port number.

Specification text
When a packet is received directly on a physical port and not processed by a logical port, OFPXMT_OFB_IN_PORT and OFPXMT_OFB_IN_PHY_PORT have the same value, the OpenFlow port_no of this physical port. OFPXMT_OFB_IN_PHY_PORT should be omitted if it has the same value as OFPXMT_OFB_IN_PORT.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.9; pg. 62)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

410.320 - Logical port match

Protocol Messages / Packet-In Message

Purpose
Verify that ofp_packet_in messages generated due to traffic on logical ports report the correct in_port.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match) with an action output to port CONTROLLER. Send a matching packet on a logical port included in the data plane. Verify that a packet_in message that encapsulates the matching packet is triggered. Verify that OFPXMT_OFB_IN_PHY_PORT corresponds to the correct physical port, and that OFPXMT_OFB_IN_PORT is equal to the correct logical data plane port number.

Specification text
When a packet is received on a logical port by way of a physical port, OFPXMT_OFB_IN_PORT is the logical port's port no and OFPXMT_OFB_IN_PHY_PORT is the physical port's port no.
- OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.9; pg. 62)

Topology
One control plane connection and one logical data plane connection

Results
Pass / Fail / Not Applicable

Test recommendations
Should be tested against logical tunnel interfaces.

Additional remarks
If the device does not support logical ports, this test case's result shall be marked as 'Not Applicable'.

420 - Flow Removed Protocol Message

Test suite 420 verifies that the device correctly implements port status and flow removed message types.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

420.20 - Flow removed message fields

Protocol Messages / Flow removed message

Purpose
Verify that the match, cookie, and priority fields are the same as those used in the flow mod request.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the Pre-requisites for the match) with OFPFF_SEND_FLOW_REM flag set and with action as output port X. Send a packet for matching field and verify that packet is received on port X. Send a OFPFC_DELETE request for previous flow. Send same packet as earlier and verify that packet is not received by output port and dropped by the switch. Verify that the controller receives the flow removed message with match, cookie and priority to that of original flow_mod.

Specification text
The match, cookie, and priority fields are the same as those used in the flow mod request.
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.2; pg. 107)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

420.30 - Flow removed message reason idle timeout

Protocol Messages / Flow removed message

Purpose
Verify that when a flow is removed because of an idle timeout, the reason field of a flow removed message is OFPRR_IDLE_TIMEOUT.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the Pre-requisites for the match) with OFPFF_SEND_FLOW_REM flag set, and idle_timeout of 3 seconds, and an action output port X. Send a packet for matching field and verify that packet is received on port X. After 3 seconds verify that the controller receives the flow removed message, and the reason field is set to OFPRR_IDLE_TIMEOUT. Send the same packet as earlier and verify that packet is not received by output port and dropped by the switch.

Specification text
The reason field is one of the following:
OFPRR_IDLE_TIMEOUT = 0, /* Flow idle time exceeded idle_timeout. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.2; pg. 107)*

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

420.40 - Flow removed message reason hard timeout

Protocol Messages / Flow removed message

Purpose

Verify that when a flow is removed because of a hard timeout, the reason field of a flow removed message is OFPRR_HARD_TIMEOUT.

Methodology

Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the Pre-requisites for the match) with OFPFF_SEND_FLOW_REM flag set, and hard_timeout of 3 seconds, and an action output port X. Send a packet for matching field and verify that packet is received on port X. After 3 seconds verify that the controller receives the flow removed message, and the reason field is set to OFPRR_HARD_TIMEOUT. Send the same packet as earlier and verify that packet is not received by output port and dropped by the switch.

Specification text

OFPRR_HARD_TIMEOUT = 1, /* Time exceeded hard_timeout. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.2; pg. 107)

Topology

One control plane connection and two data plane connections

Results

Pass / Fail

420.50 - Flow removed message reason delete

Protocol Messages / Flow removed message

Purpose
Verify that when a flow is removed because of a delete message, the reason field of a flow removed message is OFPRR_DELETE.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the Pre-requisites for the match) with OFPFF_SEND_FLOW_REM flag set and with action as output port X. Send a packet for matching field and verify that packet is received on port X. Send a OFPFC_DELETE request for previous flow. Send the same packet as earlier and verify that packet is not received by output port and dropped by the switch. Verify that the controller receives the flow removed message, and the reason field is set to OFPRR_DELETE.

Specification text
OFPRR_DELETE = 2, /* Evicted by a DELETE flow mod. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.2; pg. 107)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

420.70 - Flow removed message duration

Protocol Messages / Flow removed message

Purpose
Verify that a flow removed message reports the correct flow duration.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the Pre-requisites for the match) with OFPFF_SEND_FLOW_REM flag set, and hard_timeout of 3 seconds, and an action output port X. Send a packet for matching field and verify that packet is received on port X. After 3 seconds verify that the controller receives the flow removed message, and that duration_sec (mandatory for basic conformance) and duration_nsec (optional for basic conformance) fields are correct. Send the same packet as earlier and verify that packet is not received by output port and dropped by the switch.

Specification text
The duration_sec and duration_nsec fields are described in Section 7.3.5.2.
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.2; pg. 107)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

420.80 - Flow removed message reason timeout

Protocol Messages / Flow removed message

Purpose
Verify that the idle and hard timeout fields in a flow removed message are equal to the flow which was installed.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the Pre-requisites for the match) with OFPFF_SEND_FLOW_REM flag set, idle_timeout of 2 seconds, hard_timeout of 3 seconds, and an action output port X. Send a packet for matching field and verify that packet is received on port X. Verify that the controller receives the flow removed message, and that duration_sec and duration_nsec fields are correct. Also verify that the idle and hard timeout fields are equal to the idle and hard timeout fields included in the installed flow_mod.

Specification text
The idle_timeout and hard_timeout fields are directly copied from the flow mod that created this entry. With the above three fields, one can find both the amount of time the flow entry was active, as well as the amount of time the flow entry received traffic.
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.2; pg. 107)

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

420.90 - Flow removed message counters

Protocol Messages / Flow removed message

Purpose
Verify that the packet and byte counters are correctly reported in a flow removed message.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the Pre-requisites for the match) with OFPFF_SEND_FLOW_REM flag set, hard_timeout of 5 seconds, and an action output port P. Send N packets of length X for matching field and verify that packet is received on port P. Verify that the controller receives the flow removed message, and that packet_count is equal to N or -1 if not supported, and that byte_count is equal to (N*X) or -1 if not supported.

Specification text
The packet_count and byte_count indicate the number of packets and bytes that were associated with this flow entry, respectively. Those counters should behave like other statistics counters (see 7.3.5); they are unsigned and should be set to the maximum field value if not available.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.2; pg. 107)*

Topology
One control plane connection and two data plane connections

Results
Pass / Fail

420.100 - Port status reason add

Protocol Messages / Port Status Messages

Purpose
Verify that a port status message is correctly received when adding ports.

Methodology
Prior to control channel establishment, remove a data plane test port from the OpenFlow instance. Configure and connect DUT to controller. After control channel establishment, add the previously removed data plane test port to the OpenFlow instance. Verify that the device generates an ofp_port_stats message with a reason field of OFPPR_ADD.

Specification text
7.4.3 Port Status Message. As ports are added, modified, and removed from the datapath, the controller needs to be informed with the OFPT_PORT_STATUS message:
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.3; pg. 108)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

Test recommendations
May require manual testing.

420.110 - Port status reason delete

Protocol Messages / Port Status Messages

Purpose
Verify that a port status message is correctly received when removing ports.

Methodology
Configure and connect DUT to controller. After control channel establishment, remove a data plane test port from the OpenFlow instance. Verify that the device generates an ofp_port_stats message with a reason field of OFPPR_DELETE. Add the previously removed data plane test port to the OpenFlow instance.

Specification text
OFPPR_DELETE = 1, /* The port was removed. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.3; pg. 108)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

Test recommendations
May require manual testing.

420.120 - Port status reason modify

Protocol Messages / Port Status Messages

Purpose
Verify that a port status message is correctly received when modifying ports.

Methodology
Configure and connect DUT to controller. After control channel establishment, administratively turn down a data plane test port from the OpenFlow instance. Verify that the device generates an ofp_port_stats message with a reason field of OFPPR_MODIFY. Administratively turn up the previously downed data plane test port.

Specification text
OFPPR_MODIFY = 2, /* Some attribute of the port has changed. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.3; pg. 108)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

Test recommendations
May require manual testing.

430 - Error Messages Section One

Test suite 430 verifies that the device correctly implements various required error messages. Some devices may be unable to trigger specific error messages. The results of these test cases MAY be recorded as ‘Not Applicable’.

Remarks

Permission errors

Because permissions error messages are triggered by intermediary devices, test cases related to these error codes are not applicable to Basic Single Table conformance. The result of these test cases MAY be recorded as ‘Not Applicable’

Valid Experimenter ID

Each DUT has a unique experimenter ID that must be provided by the Vendor for test case 430.130.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite, except where result is ‘Not Applicable’, based on conditions as outlined in test case Remarks.

430.20 - Error message data

Protocol Messages / Error Messages

Purpose
Verify that if a request triggers an error message, a portion of the request is included in the data field of the resulting ofp_error_message.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow with matching on named field. Send OFPFC_DELETE flow_mod message for this flow with invalid table-id. Verify that the switch generates an ofp_error_msg message. Verify that the data field of this ofp_error_msg message includes either the full OFPFC_DELETE ofp_flow_mod message, or the first 64 bytes of that message.

Specification text
Unless specified otherwise, the data field contains at least 64 bytes of the failed request that caused the error message to be generated, if the failed request is shorter than 64 bytes it should be the full request without any padding.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 108)*

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
The way to trigger the error message (invalid table id) is not essential for this test case. If the device does allow every possible table id, alternate ways of triggering an error should be implemented. The test case is only checking for the correct data field of the error message.

430.30 - Error message xid

Protocol Messages / Error Messages

Purpose
If a request triggers an error message, check that the error's xid is equal to the original request.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow with matching on named field. Send OFPFC_DELETE flow_mod message for this flow with invalid table-id. Verify that the switch sends the ofp_error_msg with an xid equal to the xid included in the OFPFC_DELETE flow_mode message.

Specification text
If the error message is in response to a specific message from the controller, e.g., OFPET_BAD_REQUEST, OFPET_BAD_ACTION, OFPET_BAD_INSTRUCTION, OFPET_BAD_MATCH, or OFPET_FLOW_MOD_FAILED, then the xid field of the header must match that of the offending message.
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 109)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
The way to trigger the error message (invalid table id) is not essential for this test case. If the device does allow every possible table id, alternate ways of triggering an error should be implemented. The test case is only checking for the correct data field of the error message.

430.70 - Hello failed data

Protocol Messages / Error Messages / HELLO_FAILED / Data

Purpose
Record a hello failure's data string.

Methodology
If existing, send a version (0) that is not supported by switch, and lower than the switch side provided version number. Verify that the switch generates an OFPT_ERROR message. If the device includes a non-empty data field, log the null-terminated error message.

Specification text
The data field contains an ASCII text string that adds detail on why the error occurred.
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 109)

Topology
One control plane connection

Results
Pass / Fail

430.90 - Bad request bad version

Protocol Messages / Error Messages / BAD_REQUEST / OFPBRC_BAD_VERSION

Purpose
Verify that if a request with a bad version number is transmitted, a bad_request error message with a bad version code is generated.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_barrier_request message with the version field set to an unsupported OpenFlow version. Verify that the device replies with an ofp_error_msg with a type field of OFPET_BAD_REQUEST and code field of OFPBRC_BAD_VERSION.

Specification text
OFPBRC_BAD_VERSION = 0, /* ofp_header.version not supported. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 109)

Topology
One control plane connection

Results
Pass / Fail

430.100 - Bad request bad type

Protocol Messages / Error Messages / BAD_REQUEST / OFPBRC_BAD_TYPE

Purpose
If a request uses an undefined type, the device generates a bad request error with a bad type code.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp message with a type field set to an unsupported OpenFlow message type. Verify that the device replies with an ofp_error_msg with a type field of OFPET_BAD_REQUEST and code field of OFPBRC_BAD_TYPE.

Specification text
OFPBRC_BAD_TYPE = 1, /* ofp_header.type not supported. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 109)

Topology
One control plane connection

Results
Pass / Fail

430.120 - Bad request bad experimenter

Protocol Messages / Error Messages / BAD_REQUEST / OFPBRC_BAD_EXPERIMENTER

Purpose

If an unsupported experimenter request is sent to a device, check that the device generates a bad request error with a bad experimenter error.

Methodology

Configure and connect DUT to controller. After control channel establishment, generate an ofp_multipart_request with an invalid experimenter field. Send the message, and verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_REQUEST and code OFPBRC_BAD_EXPERIMENTER.

Specification text

OFPBRC_BAD_EXPERIMENTER = 3, /* Experimenter id not supported * (in ofp_experimenter_header or * ofp_multipart_request or * ofp_multipart_reply). */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 109)

Topology

One control plane connection

Results

Pass / Fail

430.130 - Bad request bad experimenter type

Protocol Messages / Error Messages / BAD_REQUEST / OFPBRC_BAD_EXP_TYPE

Purpose

If an experimenter request with an unsupported experimenter type is sent to a device, check that the device generates a bad request error with a bad experimenter type error.

Methodology

Configure and connect DUT to controller. After control channel establishment, generate an ofp_multipart_request with a valid experimenter field, and an invalid exp_type field. Send the message, and verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_REQUEST and code OFPBRC_BAD_EXP_TYPE.

Specification text

OFPBRC_BAD_EXP_TYPE = 4, /* Experimenter type not supported. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 109)

Topology

One control plane connection

Results

Pass / Fail

Test recommendations

A valid experimenter ID must be provided by the vendor.

430.150 - Bad request bad length

Protocol Messages / Error Messages / BAD_REQUEST / OFPBRC_BAD_LEN

Purpose
If a request's length is incorrectly specified, verify that the device generates a bad request error with a bad length code.

Methodology
Configure and connect DUT to controller. After control channel establishment, send an ofp_barrier_request message with the length field set to 7. Verify that the device replies with an ofp_error_msg with a type field of OFPET_BAD_REQUEST and code field of OFPBRC_BAD_LEN.

Specification text
OFPBRC_BAD_LEN = 6, /* Wrong request length for type. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)*

Topology
One control plane connection

Results
Pass / Fail

430.160 - Bad request buffer empty

Protocol Messages / Error Messages / BAD_REQUEST / OFPBRC_BUFFER_EMPTY

Purpose

If a request specifies a buffer that has already been emptied, verify that the device generates a bad request error with a buffer empty code.

Methodology

Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the Pre-requisites for the match) with action as output port CONTROLLER. Forward traffic on a data plane port, and verify that an ofp_packet_in message is received on the control plane. Generate an ofp_packet_out with buffer_id equal to the buffer_id of the received ofp_packet_in, and an output action to a valid data plane test port. Verify that the tcp packet is received on the correct port. Resend the ofp_packet_out message. Verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_REQUEST and code OFPBRC_BUFFER_EMPTY.

Specification text

OFPBRC_BUFFER_EMPTY = 7, /* Specified buffer has already been used. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)*

Topology

One control plane connection and two data plane connections

Results

Pass / Fail

Test recommendations

Due to various buffer implementations it may not be possible (or make sense) to throw the error code OFPBRC_BUFFER_EMPTY. In these cases it is acceptable for a device to instead return error code OFPBRC_BUFFER_UNKNOWN.

430.170 - Bad request buffer unknown

Protocol Messages / Error Messages / BAD_REQUEST / OFPBRC_BUFFER_UNKNOWN

Purpose

If a request specified a buffer that does not exist, verify that the device generates a bad request error with a buffer unknown code.

Methodology

Configure and connect DUT to controller. After control channel establishment, generate an ofp_packet_out with buffer_id not equal to -1, and an output action to a valid data plane test port. Send the ofp_packet_out message, and verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_REQUEST and code OFPBRC_BUFFER_UNKNOWN.

Specification text

OFPBRC_BUFFER_UNKNOWN = 8, /* Specified buffer does not exist. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)

Topology

One control plane connection and one data plane connection

Results

Pass / Fail

430.200 - Bad request bad port

Protocol Messages / Error Messages / BAD_REQUEST / OFPBRC_BAD_PORT

Purpose

If a request specifies a port number that does not exist, verify that the device generates a bad request error with a bad port code.

Methodology

Configure and connect DUT to controller. Send an ofp_multipart_request with type OFPMP_PORT_STATS, and a port_no equal to an invalid port number. Verify that an ofp_error_msg is returned with type field of OFPET_BAD_REQUEST and code OFPBRC_BAD_PORT or that an empty port stats message is received.

Specification text

OFPBRC_BAD_PORT = 11, /* Invalid port. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)

Topology

One control plane connection

Results

Pass / Fail

430.210 - Bad request bad packet

Protocol Messages / Error Messages / BAD_REQUEST / OFPBRC_BAD_PACKET

Purpose

If an ofp_packet_out includes an invalid packet in its datafield, ensure the device returns a bad request error with a bad packet code.

Methodology

Configure and connect DUT to controller. After control channel establishment, generate an ofp_packet_out with buffer_id equal to -1, a data field of 0x00, and an output action to a valid data plane test port. Send the ofp_packet_out message, and verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_REQUEST and code OFPBRC_BAD_PACKET.

Specification text

OFPBRC_BAD_PACKET = 12, /* Invalid packet in packet-out. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)

Topology

One control plane connection

Results

Pass / Fail

430.230 - Bad request data

Protocol Messages / Error Messages / BAD_REQUEST / Data

Purpose

Verify that when a bad request triggers an error message, a portion of the request is included in the data field of the resulting ofp_error_message.

Methodology

Configure and connect DUT to controller. After control channel establishment, send an ofp_barrier_request message with the version field set to an unsupported OpenFlow version. Verify that the device replies with an ofp_error_msg with a type field of OFPET_BAD_REQUEST and code field of OFPBRC_BAD_VERSION. Verify that the data field of this ofp_error message includes either the full ofp_barrier_request message, or the first 64 bytes of that message.

Specification text

ofp_error_msg 'code' values for OFPET_BAD_REQUEST. 'data' contains at least * the first 64 bytes of the failed request. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 109)

Topology

One control plane connection

Results

Pass / Fail

430.250 - Bad action bad type

Protocol Messages / Error Messages / BAD_ACTION / OFPBAC_BAD_TYPE

Purpose
Verify that when an undefined action type is specified, the device generates a bad action error with a bad type code.

Methodology
Configure and connect DUT to controller. After control channel establishment, generate an ofp_packet_out with buffer_id equal to -1, a valid data field, and an output action with the type field set to an invalid ofp_action_type. Send the ofp_packet_out message, and verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_ACTION and code OFPBAC_BAD_TYPE.

Specification text
OFPBAC_BAD_TYPE = 0, /* Unknown action type. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

430.260 - Bad action bad length

Protocol Messages / Error Messages / BAD_ACTION / OFPBAC_BAD_LEN

Purpose
Verify that when an ofp_action_header specifies an invalid length field, the device generates a bad action error with a bad length code.

Methodology
Configure and connect DUT to controller. After control channel establishment, generate an ofp_packet_out with buffer_id equal to -1, a valid data field, and an output action with the len field set to 14. Send the ofp_packet_out message, and verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_ACTION and code OFPBAC_BAD_LEN.

Specification text
OFPBAC_BAD_LEN = 1, /* Length problem in actions. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

430.270 - Bad action bad experimenter

Protocol Messages / Error Messages / BAD_ACTION / OFPBAC_BAD_EXPERIMENTER

Purpose
Verify that when an unrecognized experimenter action is received, the device generates a bad action error with a bad experimenter code.

Methodology
Configure and connect DUT to controller. After control channel establishment, generate an ofp_packet_out with buffer_id equal to -1, a valid data field, and an experimenter action with an invalid experimenter field. Send the ofp_packet_out message, and verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_ACTION and code OFPBAC_BAD_EXPERIMENTER.

Specification text
OFPBAC_BAD_EXPERIMENTER = 2, /* Unknown experimenter id specified. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)

Topology
One control plane connection

Results
Pass / Fail

430.330 - Bad action bad queue

Protocol Messages / Error Messages / BAD_ACTION / OFPBAC_BAD_QUEUE

Purpose
Verify that if a bad queue_id is specified in a set-queue action, the device generates a bad action error with a bad queue code.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install an ofp_flow_mod matching on a named field (under the Pre-requisites for the match) with a set-queue action specifying an invalid queue_id, and an output action to a data plane port. Verify that an error is returned with an error type of OFPET_BAD_ACTION and code of OFPBAC_BAD_QUEUE.

Specification text
OFPBAC_BAD_QUEUE = 8, /* Problem validating output queue. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)

Topology
One control plane connection

Results
Pass / Fail

430.500 - Bad action bad set type

Protocol Messages / Error Messages / BAD_ACTION / OFPBAC_BAD_SET_TYPE

Purpose
Verify that if an invalid set-field action is specified, the device generates a bad action error with a bad set type code.

Methodology
Configure and connect DUT to controller. After control channel establishment, create an ofp_flow_mod matching on the named field (under the given Pre-requisites for the match), with an output action and an unsupported set_field action (e.g. OXM_OF_VLAN_VID, OXM_OF_ARP_SPA, or OXM_OF_ICMPV4_TYPE). Verify that an error is returned with error type OFPET_BAD_ACTION and error code OFPBAC_BAD_SET_TYPE.

Specification text
OFPBAC_BAD_SET_TYPE = 13, /* Unsupported type in SET_FIELD action. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)

Topology
One control plane connection

Results
Pass / Fail

430.510 - Bad action bad set length

Protocol Messages / Error Messages / BAD_ACTION / OFPBAC_BAD_SET_LEN

Purpose
Verify that if an invalid set-field oxm length is specified, the device generates a bad action error with a bad set length code.

Methodology
Configure and connect DUT to controller. After control channel establishment, create an ofp_flow_mod matching on the named field (under the given Pre-requisites for the match), with an output action and a supported set_field action with a len field set intentionally smaller than required by the given OXM. Verify that an error is returned with error type OFPET_BAD_ACTION and error code OFPBAC_BAD_SET_LEN.

Specification text
OFPBAC_BAD_SET_LEN = 14, /* Length problem in SET_FIELD action. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)

Topology
One control plane connection

Results
Pass / Fail

430.530 - Bad action data

Protocol Messages / Error Messages / BAD_ACTION / Data

Purpose

Verify that when an action error is triggered, the data portion of the error message contains a portion of the initial request.

Methodology

Configure and connect DUT to controller. After control channel establishment, generate an ofp_packet_out with buffer_id equal to -1, a valid data field, and an output action with the type field set to an invalid ofp_action_type. Send the ofp_packet_out message, and verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_ACTION and code OFPBAC_BAD_TYPE. Verify that the data field of this ofp_error message includes either the full ofp_packet_out message, or the first 64 bytes of that message.

Specification text

ofp_error_msg 'code' values for OFPET_BAD_ACTION. 'data' contains at least * the first 64 bytes of the failed request. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)*

Topology

One control plane connection

Results

Pass / Fail

430.590 - Bad instruction bad experimenter

Protocol Messages / Error Messages / BAD_INSTRUCTION / OFPBIC_BAD_EXPERIMENTER

Purpose
Verify that the correct error message is generated when an unknown experimenter id is used.

Methodology
Configure and connect DUT to controller. After control channel establishment, create an ofp_flow_mod matching on in_port, and an experimenter instruction with an invalid experimenter field. Install the ofp_flow_mod message, and verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_INSTRUCTION and code OFPBIC_BAD_EXPERIMENTER.

Specification text
OFPBIC_BAD_EXPERIMENTER = 5, /* Unknown experimenter id specified. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)*

Topology
One control plane connection

Results
Pass / Fail

430.610 - Bad instruction bad length

Protocol Messages / Error Messages / BAD_INSTRUCTION / OFPBIC_BAD_LEN

Purpose
Verify that the correct error message is generated when an instruction's length is incorrect.

Methodology
Configure and connect DUT to controller. After control channel establishment, create an ofp_flow_mod matching on in_port, and an instruction with an invalid len field. Install the ofp_flow_mod message, and verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_INSTRUCTION and code OFPBIC_BAD_LEN.

Specification text
OFPBIC_BAD_LEN = 7, /* Length problem in instructions. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)*

Topology
One control plane connection

Results
Pass / Fail

430.630 - Bad instruction data

Protocol Messages / Error Messages / BAD_INSTRUCTION / Data

Purpose
Verify that bad instruction errors include the first 64 bytes of the offending message.

Methodology
Configure and connect DUT to controller. After control channel establishment, send OFPFC_ADD flow_mod message with unknown instruction. Verify that the switch sends ofp_error_msg with OFPET_BAD_INSTRUCTION type and OFPBIC_UNKNOWN_INST code. Verify that the data field of this ofp_error message includes either the full ofp_barrier_request message, or the first 64 bytes of that message.

Specification text
/* ofp_error_msg 'code' values for OFPET_BAD_INSTRUCTION. 'data' contains at least * the first 64 bytes of the failed request. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 110)*

Topology
One control plane connection

Results
Pass / Fail

430.640 - Bad match type

Protocol Messages / Error Messages / BAD_MATCH / OFPBMC_BAD_TYPE

Purpose
Verify that the correct error message is generated when a bad match type is used.

Methodology
Configure and connect DUT to controller. After control channel establishment, create an ofp_flow_mod matching on an OXM with the field set to an undefined OFPXMT. Install the ofp_flow_mod message, and verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_MATCH and code OFPBMC_BAD_TYPE.

Specification text
OFPBMC_BAD_TYPE = 0, /* Unsupported match type specified by the match */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 111)*

Topology
One control plane connection

Results
Pass / Fail

430.650 - Bad match length

Protocol Messages / Error Messages / BAD_MATCH / OFPBMC_BAD_LEN

Purpose
Verify that the correct error message is generated when a bad match length is specified.

Methodology
Configure and connect DUT to controller. After control channel establishment, create an ofp_flow_mod matching on an OXM with the oxm_length set to three. Install the ofp_flow_mod message, and verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_MATCH and code OFPBMC_BAD_LEN.

Specification text
OFPBMC_BAD_LEN = 1, /* Length problem in match. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 111)*

Topology
One control plane connection

Results
Pass / Fail

430.690 - Bad wildcard match

Protocol Messages / Error Messages / BAD_MATCH / OFPBMC_BAD_WILDCARDS

Purpose
When using masking, it is an error for a 0-bit in oxm_mask to have a corresponding 1-bit in oxm_value.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on an IPv4 source address with a mask where each zero-bit in the mask has a corresponding one-bit in the value, action is forwarding to an output port. Verify that the device responds with an ofp_error_msg with type field of OFPET_BAD_MATCH and code OFPBMC_BAD_WILDCARDS.

Specification text
OFPBMC_BAD_WILDCARDS = 5, /* Unsupported combination of fields masked or omitted in the match. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 111)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If the switch supports all possible combinations of mask and value for all supported match fields, the error cannot be triggered and the test shall be marked as 'Not Applicable'.

440 - Error Message Section Two

Test suite 440 verifies that the device correctly implements various required error messages. In some cases the DUT may not support the tested field or bits. In these cases it MUST respond with the expected error messages. Some devices may be unable to trigger specific error messages. The results of these test cases MAY be marked as ‘Not Applicable’.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite, except where result is ‘Not Applicable’, based on conditions as outlined in test case Remarks.

440.10 - Bad match data

Protocol Messages / Error Messages / BAD_MATCH / Data

Purpose
Verify that the data field of this ofp_error message includes either the full ofp_flow_mod message, or the first 64 bytes of that message.

Methodology
Configure and connect DUT to controller. After control channel establishment, add flows with missing prerequisites according to table 7.2.3.7. Verify that the switch does not accept the flows, and returns the correct error message OFPET_BAD_MATCH.Verify that the correct error message is triggered and the flow is not added to the flow table. Verify that the data field of this ofp_error message includes either the full ofp_flow_mod message, or the first 64 bytes of that message.

Specification text
/* ofp_error_msg 'code' values for OFPET_BAD_MATCH. 'data' contains at least * the first 64 bytes of the failed request. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 111)

Topology
One control plane connection

Results
Pass / Fail

440.80 - Flow mod failed bad command

Protocol Messages / Error Messages / FLOW_MOD_FAILED / OFPFMFC_BAD_COMMAND

Purpose
Verify that a bad flow mod command triggers the correct error message.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install an ofp_flow_mod matching on a named field (under the Pre-requisites for the match), an output action to a data plane port, and an invalid command field. Verify that an error is returned with an error type of OFPET_FLOW_MOD_FAILED and code of OFPFMFC_BAD_COMMAND.

Specification text
OFPFMFC_BAD_COMMAND = 6, /* Unsupported or unknown command. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 111)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

440.90 - Flow mod failed bad flags

Protocol Messages / Error Messages / FLOW_MOD_FAILED / OFPFMFC_BAD_FLAGS

Purpose
Verify that bad flow mod flags trigger the correct error message.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install an ofp_flow_mod matching on a named field (under the Pre-requisites for the match), an output action to a data plane port, and an invalid flag bit set. Verify that an error is returned with an error type of OFPET_FLOW_MOD_FAILED and code of OFPFMFC_BAD_FLAGS.

Specification text
OFPFMFC_BAD_FLAGS = 7, /* Unsupported or unknown flags. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 111)*

Topology
One control plane connection and one data plane connection

Results
Pass / Fail

440.100 - Flow mod failed data

Protocol Messages / Error Messages / FLOW_MOD_FAILED / Data

Purpose
Verify that flow mod failed errors include up to 64 bytes of the offending request.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on a named field (under the given Pre-requisites for the match). Add a second flow with an overlapping match, the check_overlap flag set, and a different priority. Verify that the flow gets installed in the flow table. Install a third flow with the check_overlap flag set, with a match overlapping that of flow 1, and a priority set equal to flow 1. Verify that the correct error message is triggered and the flow is not added to the flow table. Verify that the data field of this ofp_error message includes either the full ofp_flow_mod message, or the first 64 bytes of that message.

Specification text
/* ofp_error_msg 'code' values for OFPET_FLOW_MOD_FAILED. 'data' contains * at least the first 64 bytes of the failed request. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 111)*

Topology
One control plane connection

Results
Pass / Fail

440.250 - Group mod failed

Protocol Messages / Error Messages / GROUP_MOD_FAILED / Data

Purpose
Verify that group mod failed errors include up to 64 bytes of the offending request.

Methodology
Configure and connect DUT to controller. After control channel establishment, create and install an ofp_group_mod message with an invalid command field. Verify that an error is returned with an error type of OFPET_GROUP_MOD_FAILED. Verify that the data field of this ofp_error message includes either the full ofp_group_mod message, or the first 64 bytes of that message.

Specification text
/* ofp_error_msg 'code' values for OFPET_GROUP_MOD_FAILED. 'data' contains * at least the first 64 bytes of the failed request. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 111)

Topology
One control plane connection

Results
Pass / Fail

440.260 - Port mod failed bad port

Protocol Messages / Error Messages / PORT_MOD_FAILED / OFPPMFC_BAD_PORT

Purpose
Verify that the correct error message is generated when a port mod specifies a port number that does not exist.

Methodology
Configure and connect DUT to controller. Send an ofp_port_mod message with an invalid port number. Verify that an ofp_error_msg is returned with type field of OFPET_PORT_MOD_FAILED and code OFPPMFC_BAD_PORT.

Specification text
OFPPMFC_BAD_PORT = 0, /* Specified port number does not exist. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 112)*

Topology
One control plane connection

Results
Pass / Fail

440.270 - Port mod failed bad hw address

Protocol Messages / Error Messages / PORT_MOD_FAILED / OFPPMFC_BAD_HW_ADDR

Purpose

Verify that the correct error message is generated when a port mod specifies a hardware address that does not match the port numbers' hardware address.

Methodology

Configure and connect DUT to controller. Send an ofp_multipart_request message with type OFPMP_PORT_DESC, and record the reported hardware address for each port. Send an ofp_port_mod message with a hardware address not equal to the reported value. Verify that an ofp_error_msg is returned with type field of OFPET_PORT_MOD_FAILED and code OFPPMFC_BAD_HW_ADDR.

Specification text

OFPPMFC_BAD_HW_ADDR = 1, /* Specified hardware address does not * match the port number. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 112)

Topology

One control plane connection and one data plane connection

Results

Pass / Fail

440.280 - Port mod failed bad configuration

Protocol Messages / Error Messages / PORT_MOD_FAILED / OFPPMFC_BAD_CONFIG

Purpose
Verify that the correct error message is generated when a port mod specifies a bad configuration.

Methodology
Configure and connect DUT to controller. Send an ofp_port_mod message with an invalid config bit set to a value of (1<<1). Verify that an ofp_error_msg is returned with type field of OFPET_PORT_MOD_FAILED and code OFPPMFC_BAD_CONFIG.

Specification text
OFPPMFC_BAD_CONFIG = 2, /* Specified config is invalid. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 112)

Topology
One control plane connection

Results
Pass / Fail

440.290 - Port mod failed bad advertise

Protocol Messages / Error Messages / PORT_MOD_FAILED / OFPPMFC_BAD_ADVERTISE

Purpose
Verify that the correct error message is generated when a port mod specifies a bad advertise field.

Methodology
Configure and connect DUT to controller. Send an ofp_port_mod message with ofp_port_features set to a value of (1<<16). Verify that an ofp_error_msg is returned with type field of OFPET_PORT_MOD_FAILED and code OFPPMFC_BAD_ADVERTISE.

Specification text
OFPPMFC_BAD_ADVERTISE = 3, /* Specified advertise is invalid. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 112)

Topology
One control plane connection

Results
Pass / Fail

440.310 - Port mod failed data

Protocol Messages / Error Messages / PORT_MOD_FAILED / Data

Purpose
Verify that port mod failed errors include up to 64 bytes of the offending request.

Methodology
Configure and connect DUT to controller. Send an ofp_port_mod message with an invalid port number. Verify that an ofp_error_msg is returned with type field of OFPET_PORT_MOD_FAILED and code OFPPMFC_BAD_PORT. Verify that the data field of this ofp_error message includes either the full ofp_port_mod message, or the first 64 bytes of that message.

Specification text
/* ofp_error_msg 'code' values for OFPET_PORT_MOD_FAILED. 'data' contains * at least the first 64 bytes of the failed request. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 112)

Topology
One control plane connection

Results
Pass / Fail

440.320 - Table mod failed bad table

Protocol Messages / Error Messages / TABLE_MOD_FAILED / OFPTMFC_BAD_TABLE

Purpose
Verify that the correct error message is generated when a table mod specifies a table number that does not exist.

Methodology
Configure and connect DUT to controller. Send an ofp_table_mod message with an invalid table number. Verify that an ofp_error_msg is returned with type field of OFPET_TABLE_MOD_FAILED and code OFPTMFC_BAD_TABLE.

Specification text
OFPTMFC_BAD_TABLE = 0, /* Specified table does not exist. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 112)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
Because the ofp_table_mod message is deprecated, it is acceptable to return an ofp_error with OFPET_BAD_REQUEST and OFPBRC_BAD_TYPE, or OFPET_TABLE_MOD_FAILED and OFPTFMFC_EPERM when the message is not supported. Otherwise, the error message defined in the methodology is expected.

440.350 - Table mod failed data

Protocol Messages / Error Messages / TABLE_MOD_FAILED / Data

Purpose
Verify that table mod failed errors include up to 64 bytes of the offending request.

Methodology
Configure and connect DUT to controller. Send an ofp_table_mod message with an invalid table number. Verify that an ofp_error_msg is returned with type field of OFPET_TABLE_MOD_FAILED and code OFPTMFC_BAD_TABLE. Verify that the data field of this ofp_error message includes either the full ofp_table_mod message, or the first 64 bytes of that message.

Specification text
/* ofp_error_msg 'code' values for OFPET_TABLE_MOD_FAILED. 'data' contains * at least the first 64 bytes of the failed request. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 112)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
Because the ofp_table_mod message is deprecated, it is acceptable to return an ofp_error with OFPET_BAD_REQUEST and OFPBRC_BAD_TYPE, or OFPET_TABLE_MOD_FAILED and OFPTFMFC_EPERM when the message is not supported. Otherwise, the error message defined in the methodology is expected.

440.360 - Queue operation failed bad port

Protocol Messages / Error Messages / QUEUE_OP_FAILED / OFPQOFC_BAD_PORT

Purpose
Verify that the correct error message is generated when a queue op message specifies a bad port.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request message with type OFPMP_QUEUE_STATS, with an invalid port. Verify that an ofp_error_msg is returned with type field of OFPET_QUEUE_OP_FAILED and code OFPQOFC_BAD_PORT.

Specification text
OFPQOFC_BAD_PORT = 0, /* Invalid port (or port does not exist). */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 112)

Topology
One control plane connection

Results
Pass / Fail

440.370 - Queue operation failed bad queue

Protocol Messages / Error Messages / QUEUE_OP_FAILED / OFPQOFC_BAD_QUEUE

Purpose
Verify that the correct error message is generated when a queue op message specifies a bad queue.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request message with type OFPMP_QUEUE_STATS, with a valid port and queue_id of 0xfffffffe. Verify that an ofp_error_msg is returned with type field of OFPET_QUEUE_OP_FAILED and code OFPQOFC_BAD_QUEUE.

Specification text
OFPQOFC_BAD_QUEUE = 1, /* Queue does not exist. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 112)

Topology
One control plane connection

Results
Pass / Fail

440.390 - Queue operation failed data

Protocol Messages / Error Messages / QUEUE_OP_FAILED / Data

Purpose
Verify that queue op failed errors include up to 64 bytes of the offending request.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request message with type OFPMP_QUEUE_STATS, with an invalid port. Verify that an ofp_error_msg is returned with type field of OFPET_QUEUE_OP_FAILED and code OFPQOFC_BAD_PORT. Verify that the data field of this ofp_error message includes either the full ofp_queue_get_config_request message, or the first 64 bytes of that message.

Specification text
/* ofp_error msg 'code' values for OFPET_QUEUE_OP_FAILED. 'data' contains * at least the first 64 bytes of the failed request */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 112)*

Topology
One control plane connection

Results
Pass / Fail

440.400 - Switch config failed bad flags

Protocol Messages / Error Messages / SWITCH_CONFIG_FAILED / OFPSCFC_BAD_FLAGS

Purpose
Verify that the correct error message is generated when a switch config specifies bad flags.

Methodology
Configure and connect DUT to controller. Send an OFPT_SET_CONFIG_REQUEST message with type with an invalid flags bit set to (1<<4). Verify that an ofp_error_msg is returned with type field of OFPET_SWITCH_CONFIG_FAILED and code OFPSCFC_BAD_FLAGS.

Specification text
OFPSCFC_BAD_FLAGS = 0, /* Specified flags is invalid. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 113)

Topology
One control plane connection

Results
Pass / Fail

440.410 - Switch config failed bad length

Protocol Messages / Error Messages / SWITCH_CONFIG_FAILED / OFPSCFC_BAD_LEN

Purpose
Verify that the correct error message is generated when a switch config specifies a bad length.

Methodology
Configure and connect DUT to controller. Send an OFPT_SET_CONFIG_REQUEST message with ofp_header len set to three. Verify that an ofp_error_msg is returned with type field of OFPET_SWITCH_CONFIG_FAILED and code OFPSCFC_BAD_LEN.

Specification text
OFPSCFC_BAD_LEN = 1, /* Specified len is invalid. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 113)

Topology
One control plane connection

Results
Pass / Fail

440.430 - Switch config failed data

Protocol Messages / Error Messages / SWITCH_CONFIG_FAILED / Data

Purpose
Verify that switch config failed errors include up to 64 bytes of the offending request.

Methodology
Configure and connect DUT to controller. Send an OFPT_SET_CONFIG_REQUEST message with type with an invalid flags bit set. Verify that an ofp_error_msg is returned with type field of OFPET_SWITCH_CONFIG_FAILED and code OFPSCFC_BAD_FLAGS. Verify that the data field of this ofp_error message includes either the full ofp_switch_config message, or the first 64 bytes of that message.

Specification text
/* ofp_error_msg 'code' values for OFPET_SWITCH_CONFIG_FAILED. 'data' contains * at least the first 64 bytes of the failed request.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 113)*

Topology
One control plane connection

Results
Pass / Fail

440.450 - Role request failed unsupported

Protocol Messages / Error Messages / ROLE_REQUEST_FAILED / OFPRRFC_UNSUP

Purpose
Verify that the correct error message is generated when a role request is unsupported.

Methodology
Configure and connect DUT to only one controller. Set this controller in the equal state and verify. Send an ofp_role_request message with role slave. Verify that an ofp_error_msg is returned with type field of OFPET_ROLE_REQUEST_FAILED and code OFPRRFC_UNSUP. If roles are not supported, a higher level error, for example OFPET_BAD_REQUEST and code OFPBRC_BAD_TYPE, may be generated.

Specification text
OFPRRFC_UNSUP = 1, /* Controller role change unsupported. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 113)*

Topology
One control plane connection

Results
Pass / Fail

440.470 - Role request failed data

Protocol Messages / Error Messages / ROLE_REQUEST_FAILED / Data

Purpose
Verify that role request failed errors include up to 64 bytes of the offending request.

Methodology
Configure and connect DUT to controller. Send an ofp_role_request message with an invalid role. Verify that an ofp_error_msg is returned with type field of OFPET_ROLE_REQUEST_FAILED and code OFPRRFC_BAD_ROLE. Verify that the data field of this ofp_error message includes either the full ofp_role_request message, or the first 64 bytes of that message. If roles are not supported, a higher level error, for example OFPET_BAD_REQUEST and code OFPBRC_BAD_TYPE, may be generated.

Specification text
/* ofp_error_msg 'code' values for OFPET_ROLE_REQUEST_FAILED. 'data' contains * at least the first 64 bytes of the failed request.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 113)*

Topology
One control plane connection

Results
Pass / Fail

440.580 - Meter mod failed out of meters

Protocol Messages / Error Messages / METER_MOD_FAILED / OFPMMFC_OUT_OF_METERS

Purpose

When the device is out of meters verify that the correct error message is generated.

Methodology

Configure and connect DUT to controller. Send an ofp_meter_mod message with an add command, and at least one ofp_meter_band_headers. Verify that an ofp_error_msg is returned with type field of OFPET_METER_MOD_FAILED and code OFPMMFC_OUT_OF_METERS; otherwise continue to add meters until the ofp_error message can be triggered. If METERS are not supported, a higher-level error, for example OFPET_BAD_REQUEST and code OFPBRC_BAD_TYPE, may be generated.

Specification text

OFPMMFC_OUT_OF_METERS = 10, /* No more meters available. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 113)

Topology

One control plane connection

Results

Pass / Fail

440.600 - Meter mod failed data

Protocol Messages / Error Messages / METER_MOD_FAILED / Data

Purpose
Verify that meter mod failed errors include up to 64 bytes of the offending request.

Methodology
Configure and connect DUT to controller. Send an ofp_meter_mod message with an add command, and at least one ofp_meter_band_headers. Verify that an ofp_error_msg is returned with type field of OFPET_METER_MOD_FAILED and code OFPMMFC_OUT_OF_METERS; otherwise continue to add meters until the ofp_error message can be triggered. Verify that the data field of this ofp_error message includes either the full ofp_meter_mod message, or the first 64 bytes of that message. If METERS are not supported, a higher-level error for example OFPET_BAD_REQUEST and code OFPBRC_BAD_TYPE may be generated.

Specification text
/* ofp_error_msg 'code' values for OFPET_METER_MOD_FAILED. 'data' contains * at least the first 64 bytes of the failed request. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 113)*

Topology
One control plane connection

Results
Pass / Fail

440.610 - Table features failed bad table

Protocol Messages / Error Messages / TABLE_FEATURES_FAILED / OFPTFFC_BAD_TABLE

Purpose
Verify that the correct error message is generated when a table features request specifies a bad table id.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request message with type OFPMP_TABLE_FEATURES, with an invalid table_id. Verify that an ofp_error_msg is returned with type field of OFPET_TABLE_FEATURES_FAILED and code OFPTFFC_BAD_TABLE.

Specification text
OFPTFFC_BAD_TABLE = 0, /* Specified table does not exist. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 113)

Topology
One control plane connection

Results
Pass / Fail

440.620 - Table features failed bad metadata

Protocol Messages / Error Messages / TABLE_FEATURES_FAILED / OFPTFFC_BAD_METADATA

Purpose

Verify that the correct error message is generated when a table features request specifies an invalid metadata mask.

Methodology

Configure and connect DUT to controller. Send an ofp_multipart_request message with type OFPMP_TABLE_FEATURES. Record supported metadata bitmask. Based on the reported supported metatdata bitmask, send an ofp_multipart_request message with type OFPMP_TABLE_FEATURES, with an invalid metadata write or match mask bit set. If the device does not support multipart ofp_table_features requests, verify that an ofp_error_msg is returned with an error type OFPET_BAD_REQUEST and error code OFPBRC_BAD_LEN. If multipart ofp_table_features requests are disabled verify that an ofp_error_msg is returned with an error type OFPET_TABLE_FEATURES_FAILED and error code OFPTFFC_EPERM. Otherwise, verify that an ofp_error_msg is returned with type field of OFPET_TABLE_FEATURES_FAILED and code OFPTFFC_BAD_METADATA.

Specification text

OFPTFFC_BAD_METADATA = 1, /* Invalid metadata mask. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 113)

Topology

One control plane connection

Results

Pass / Fail / Not Applicable

Additional remarks

If DUT does not support table configuration through ofp_multipart_request messages with type OFPMP_TABLE_FEATURES, then the result of this test case may be 'Not Applicable'

440.630 - Table features failed bad type

Protocol Messages / Error Messages / TABLE_FEATURES_FAILED / OFPTFFC_BAD_TYPE

Purpose
Verify that the correct error message is generated when a table features request specifies a bad property type.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request message with type OFPMP_TABLE_FEATURES, with an ofp_table_features_prop_header that includes an undefined type. Verify that an ofp_error_msg is returned with type field of OFPET_TABLE_FEATURES_FAILED and code OFPTFFC_BAD_TYPE.

Specification text
OFPTFFC_BAD_TYPE = 2, /* Unknown property type. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 114)

Topology
One control plane connection

Results
Pass / Fail

440.640 - Table features failed bad length

Protocol Messages / Error Messages / TABLE_FEATURES_FAILED / OFPTFFC_BAD_LEN

Purpose
Verify that the correct error message is generated when a table features request specifies a bad length.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request message with type OFPMP_TABLE_FEATURES, and a table_feature_prop with an invalid len value. Verify that if modifying table features is not supported, an OFPET_BAD_REQUEST error is generated with an OFPBRC_BAD_LEN code. If modifying table features is disabled, an OFPET_TABLE_FEATURES_FAILED error with an OFPTFFC_EPERM code should be received instead. Otherwise, verify that an ofp_error_msg is returned with type field of OFPET_TABLE_FEATURES_FAILED and code OFPTFFC_BAD_LEN.

Specification text
OFPTFFC_BAD_LEN = 3, /* Length problem in properties. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 114)

Topology
One control plane connection

Results
Pass / Fail

Test recommendations
For example, send a request for exactly one table. Inside the request is the ofp_table_feature_prop_instructions structure. Set this length field to 2 bytes to trigger the error. The minimum length for the header is 4 bytes.

440.650 - Table features failed bad argument

Protocol Messages / Error Messages / TABLE_FEATURES_FAILED / OFPTFFC_BAD_ARGUMENT

Purpose
Verify that the correct error message is generated when a table features request specifies a bad argument.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request message with type OFPMP_TABLE_FEATURES, with an ofp_table_features_prop_header that includes a bad argument. If the device does not support multipart ofp_table_features requests, verify that an ofp_error_msg is returned with an error type OFPET_BAD_REQUEST and error code OFPBRC_BAD_LEN. If multipart ofp_table_features requests are disabled, verify that an ofp_error_msg is returned with an error type OFPET_TABLE_FEATURES_FAILED and error code OFPTFFC_EPERM. Otherwise, verify that an ofp_error_msg is returned with type field of OFPET_TABLE_FEATURES_FAILED and code OFPTFFC_BAD_ARGUMENT.

Specification text
OFPTFFC_BAD_ARGUMENT = 4, /* Unsupported property value. */
- OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 114)

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Test recommendations
For example if the DUT supports configuration of pipeline through table feature request messages, send a OFPMP_TABLE_FEATURES config message with a valid pipeline config for that switch but omit table number 0.

Additional remarks
If DUT does not support table configuration through ofp_multipart_request messages with type OFPMP_TABLE_FEATURES, then the result of this test case may be 'Not Applicable'.

440.670 - Table features failed data

Protocol Messages / Error Messages / TABLE_FEATURES_FAILED / Data

Purpose
Verify that table features failed errors include up to 64 bytes of the offending request.

Methodology
Configure and connect DUT to controller. Send an ofp_multipart_request message with type OFPMP_TABLE_FEATURES, with an invalid table_id. If the device does not support multipart ofp_table_features requests, verify that an ofp_error_msg is returned with an error type OFPET_BAD_REQUEST and error code OFPBRC_BAD_LEN. If multipart ofp_table_features requests are disabled, verify that an ofp_error_msg is returned with an error type OFPET_TABLE_FEATURES_FAILED and error code OFPTFFC_EPERM. Otherwise, verify that an ofp_error_msg is returned with type field of OFPET_TABLE_FEATURES_FAILED and code OFPTFFC_BAD_TABLE. Verify that the data field of this ofp_error message includes either the full ofp_multipart_request message, or the first 64 bytes of that message.

Specification text
/* ofp_error_msg 'code' values for OFPET_TABLE_FEATURES_FAILED. 'data' contains * at least the first 64 bytes of the failed request. */
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 113)*

Topology
One control plane connection

Results
Pass / Fail

440.680 - Experimenter error message

Protocol Messages / Error Messages / EXPERIMENTER

Purpose
Verify that the correct error message is generated when a bad experimenter type is specified.

Methodology
Configure and connect DUT to controller. Generate an invalid experimenter message using the values described at https://rs.opennetworking.org/wiki/display/PUBLIC/ONF+Registry.

Specification text
For the OFPET_EXPERIMENTER error type, the error message is defined by the following structure and fields, followed by experimenter defined data: /* OFPET_EXPERIMENTER: Error message (datapath -> controller). */ struct ofp_error_experimenter_msg { struct o
- *OpenFlow Switch Specification 1.3.4 (ch. 7.4.4; pg. 114)*

Topology
One control plane connection

Results
Pass / Fail / Not Applicable

Additional remarks
If the switch does not support a testable experimenter set, the switch is allowed to return a meaningful basic error message with bad request or similar types. If an error cannot be reliably triggered on a device, the test case shall be marked as 'Not Applicable'.

450 - Symmetric Messages

Test suite 450 verifies that the device correctly implements various symmetric message types including ofp_hello, ofp_echo_request / ofp_echo_reply, and OUIs included in experimenter message types.

Basic Single Table Conformance Test Profile Requirements

To satisfy the Basic Single Table requirements, an OpenFlow-enabled device MUST pass **all** test cases in this test suite.

450.10 - Unknown hello elements

Protocol Messages / Symmetric Messages / OFPT_HELLO / Elements

Purpose
Verify that the device can deal with unknown hello elements and their data.

Methodology
Configure and connect DUT to controller. Send an ofp_hello message that includes a version bitmap with support for version v1.3, and a second ofp_hello_elem_header struct with a type of 5. Verify that the negotiated version is v1.3, and that the device doesn't generate an ofp_error_msg because of the unknown ofp_hello_elem_header type.

Specification text
The OFPT_HELLO message consists of an OpenFlow header plus a set of variable size hello elements.

OFPT_HELLO. This message includes zero or more hello elements having variable size. Unknown elements types must be ignored/skipped to allow for future extensions.
- OpenFlow Switch Specification 1.3.4 (ch. 7.5.1; pg. 114)

Topology
One control plane connection

Results
Pass / Fail

450.30 - Multiple version bitmaps

Protocol Messages / Symmetric Messages / OFPT_HELLO / multiple bitmaps

Purpose
Verify that version negotiation based on multiple bitmaps is successful.

Methodology
Configure and connect DUT to controller. If the switch does not support bitmap, set the Hello version field to 4. Verify that when negotiating with a bitmap specifying an OpenFlow version > 32, the switch negotiates to version 4. Again, if the switch does support bitmaps, set the hello message field to version 0. Verify that when negotiating with a bitmap specifying an OpenFlow version > 32 && OpenFlow version 4, the switch correctly negotiates to v1.3.

Specification text
The number of bitmaps included in the field depend on the highest version number supported : ofp versions 0 to 31 are encoded in the first bitmap, ofp versions 32 to 63 are encoded in the second bitmap and so on. For example, if a switch supports only version 1.0 (ofp version=0x01) and version 1.3 (ofp version=0x04), the first bitmap would be set to 0x00000012.
- *OpenFlow Switch Specification 1.3.4 (ch. 7.5.1; pg. 115)*

Topology
One control plane connection

Results
Pass / Fail

450.50 - Echo request reply with no data

Protocol Messages / Symmetric Messages / Echo Request / empty data field

Purpose
Verify the DUT responds correctly to an ECHO request.

Methodology
Configure and connect DUT to controller. Send an OFPT_ECHO_REQUEST. Verify that an OFPT_ECHO_REPLY is received from the DUT.

Specification text
or zero-size to verify liveness between the switch and controller.
- OpenFlow Switch Specification 1.3.4 (ch. 7.5.2; pg. 115)

Topology
One control plane connection

Results
Pass / Fail

Appendix A: References

OpenFlow Switch Specification v1.3.4; https://www.opennetworking.org

Appendix B: Credits

Spec Contributors in Alphabetical Order:
Stephen Chambers (UNH-IOL), Mrinmoy Das (Ixia), Uwe Dahlmann (InCNTRE, Tallac), Benny A. Eggers (HP), Keisuke Fukumoto (NEC), R.M. Jheng (NBL), Naresh Thukkani (Criterion Networks), Ron Milford (InCNTRE), Ashish Nandy (Ixia), Michael Orr (Marvell), Sibylle Schaller (NEC), Subhash Kumar Singh (Criterion Networks), Jonathan Stout (InCNTRE), Timothy Winters (UNH-IOL)