OPEN NETWORKING
FOUNDATION

# OpenFlow Switch Errata

Version 1.0.2
November 1, 2013

OpenFlow™

## Disclaimer

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Without limitation, ONF disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and ONF disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

No license, express or implied, by estoppel or otherwise, to any Open Networking Foundation or Open Networking Foundation member intellectual property rights is granted herein.

Except that a license is hereby granted by ONF to copy and reproduce this specification for internal use only.

Contact the Open Networking Foundation at `http://www.opennetworking.org` for information on specification licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

## Contents

# 1 Introduction

This document describes version 1.0.2 of the OpenFlow specification. This version of the specification is released as a set of erratum for version 1.0.0 of the OpenFlow switch specification. Those errata amend, correct or supplement the text of that version of specification.

The original 1.0.0 specification can be found at the following URL :

`http://www.openflow.org/wp/documents/`

This document includes various issues found during implementation and interoperability testing of the 1.0.0 specification. The resolution of those issues was carefully designed to minimise specification changes and compatibility issues with existing implementations of 1.0.0. Those issues are grouped in three main sections based on their impact on the specification.

# 2 Common implementation mistakes

This section contains common implementation mistakes which are not due to specification problems. These informative descriptions highlights specific section of the specification. However they are not errata because no specification change or clarification is included.

## 2.1 Default behaviour on table-miss

The 1.0.0 specification only defines a single behaviour for packet that do not match any flow in the flow table, which is sending it to the controller using a packet-in message. This is specified in various places.

OpenFlow 1.0.0, Section 2:

> If no match is found, the packet is forwarded to the controller over the secure channel.

OpenFlow 1.0.0, Section 3.4:

> If no matching entry can be found for a packet, the packet is sent to the controller over the secure channel.

OpenFlow 1.0.0, Section 4.1.2:

> Packet-in: For all packets that do not have a matching flow entry, a packet-in event is sent to the controller (or if a packet matches an entry with a "send to controller" action).

Many controllers and controller applications depend on the proper implementation of that behaviour, for example to discover topology and traffic.

Other behaviour for table-miss may be supported as a vendor extension of OpenFlow, such as dropping the packet or forwarding it using the normal pipeline. However, by default the switch must implement the OpenFlow compliant behaviour specified in 1.0.0, without the need of any workaround or special flow entry.

Sending packet to the controller on table-miss may overload the switch or the controller, and some of those packet may have to be dropped, and this may be an issue in some deployments. In most cases the controller can avoid this behaviour by setting a low priority wildcard rule matching all packets that implement an alternate behaviour.

Some switches use a segregation mechanism outside OpenFlow to decide if packets are processed by a specific OpenFlow instance or not. This segregation mechanism must be independent of the content of the flow table and the flow table must offer a self consistent abstraction when matching packets. For any packet that can be matched by the flow table, if it does not match any flow entry in the flow table, it must be sent to the controller. In other words, the segregation mechanism cannot be used to bypass this requirement.

# 3 Clarification of ambiguous sections

This section contains errata to the specification clarifying parts that are under-specified, ambiguous or inconsistent. While those errata don't change the intent of the specification, they invalidate some legitimate interpretations of the specification.

## 3.1 Port number 0

The 1.0.0 specification defines that switch port enumeration start at the value 1. This is specified in two places.

OpenFlow 1.0.0, Section 5.2.1:

> /* Port numbering. Physical ports are numbered starting from 1. */

OpenFlow 1.0.0, Table 3:

> Ingress Port : Numerical representation of incoming port, starting at 1.

The Changelog for version 0.9 says:

> Previous releases of OpenFlow had port numbers start at 0, release 0.9 changes them to start at 1.

However, some other section mention that 0 is a valid port. This could be interpreted as meaning that port number 0 can be used by the switch as a physical port.

OpenFlow 1.0.0, Section 5.3.3, section 5.3.5:

> Note that to disable output port filtering, `out_port` must be set to `OFPP_NONE`, since 0 is a valid port id.

The intent of the specification is clear, switch port enumeration must start with port number 1.

However, we suggest that both switch and controller should avoid using port number 0 as a special indication (such as invalid), both internally and externally, to avoid compatibility issues with implementations that don't implement properly this part of the specification.

The following text is added in the OpenFlow 1.0.1 specification :

> The switch physical ports are numbered starting from 1. Port number 0 is reserved by the specification and must not be used for a switch physical port.

### 3.2 No padding of error messages

The 1.0.0 specification defines that, in the error message, the data field is of variable length, with no minimum size. This means that it should not be padded.

OpenFlow 1.0.0, Section 5.4.4:

> The data is variable length and interpreted based on the type and code; in most cases this is the message that caused the problem.

However, some language in that section is ambiguous, and could be read as requiring a minimum size of 64 bytes for that field.

OpenFlow 1.0.0, Section 5.4.4:

> The data field contains at least 64 bytes of the failed request.

The specification is ambiguous. This is fixed in subsequent version of the specification, version 1.1 of the specification says that this field is not padded. There is no rationale for padding. Padding actually decreases the efficiency of the protocol and may make it harder for the controller to figure out the message that caused the error.

The intention for this field was to have no minimum size and no padding, and such padding could have adverse consequences, therefore implementations of 1.0.0 must not pad the data portion of error messages.

The following text is added in the OpenFlow 1.0.1 specification :

> The `data` field of the error message is variable length and interpreted based on the `type` and `code`. 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.

### 3.3 Matching packet with no VLAN header

The text of the specification does not explain how to match packets that don't have a VLAN header. The OpenFlow header file `openflow.h`, which is part of the specification, does define it.

openflow.h, line 364:

```
/* The VLAN id is 12 bits, so we can use the entire 16 bits to indicate
special conditions. All ones is used to match that no VLAN id was
set. */
#define OFP_VLAN_NONE 0xffff
```

The content of the header file should be reflected properly in the text of the specification.

The following text is added in the OpenFlow 1.0.1 specification :

> Setting the OFPFW_DL_VLAN bit in the wildcards field specifies that a flow should match packets regardless of whether they contain a VLAN tag. Setting the special value OFP_VLAN_NONE in the dl_vlan field of the match specifies that a flow should match packets without a VLAN tag.

### 3.4 Ignoring fields in the match

The specification does not define the rules by which fields of the match are used or ignored when doing matching. It only defines which fields of the packets are used.

OpenFlow 1.0.0, Section 3.4:

> Header fields used for the table lookup depend on the packet type as described below. A packet matches a flow table entry if the values in the header fields used for the lookup (as defined above) match those defined in the flow table. If a flow table field has a value of ANY, it matches all possible values in the header.

However, the fields used in the matching process not only depends on the packet headers, it also depend on the match and in particular which match fields are wildcarded. In particular, when wildcarding a protocol, all fields of that protocol must be ignored.

The following text is added in the OpenFlow 1.0.1 specification :

> Protocol-specific fields within ofp_match must be ignored when the corresponding protocol is not specified in the match. The IP header and transport header fields must be ignored unless the Ethertype is specified as either IPv4 or ARP. The tp_src and tp_dst fields must be ignored unless the network protocol is set to TCP, UDP or ICMP. The dl_vlan_pcp field must be ignored when the OFPFW_DL_VLAN wildcard bit is set or when the dl_vlan value is set to OFP_VLAN_NONE. Fields that are ignored don't need to be wildcarded and should be set to 0.

### 3.5 Generating flow removed messages

A first issue is that the 1.0.0 specification contains ambiguous text that could be interpreted as the generation of a flow removed message when deleting a message being an optional operation.

OpenFlow 1.0.0, Section 4.1.2:

> The flow modify message also specifies whether the switch should send a flow removed message to the controller when the flow expires. Flow modify messages which delete flows may also cause flow removed messages.

The generation of such message is dictated by a flag associated with the flow entry.

OpenFlow 1.0.0, Section 4.6:

> For DELETE requests, if flow entries match, and must be deleted, then each normal entry with the OFPFF_SEND_FLOW_REM flag set should generate a flow removed message. Deleted emergency flow entries generate no flow removed messages.

The intent of the specification is clear, the generation of flow removed messages when the flow is deleted is mandated when the flag is set.

A related issue is that the 1.0.0 specification does not define if a flow removed message is generated when a flow is removed due to the insertion of an identical flow.

OpenFlow 1.0.0, Section 4.6:

> For valid (non-overlapping) ADD requests, or those with no overlap checking, the switch must insert the flow entry. If a flow entry with identical header fields and priority already resides in any table, then that entry, including its counters, must be removed, and the new flow entry added.

The generation of flow removed messages is explicitly described when a flow expires (section 4.7) or when the controller deletes flows (section 4.6). The specification also specifies that the removal of emergency flows does not generates flow removed messages (section 4.6). Flow removed messages are not generated in all cases of flow removal. The specification does not defines if they are generated when a flow is removed due to the insertion of an identical flow, and it could be interpreted both ways.

Subsequent versions of the specification explicitly state that no flow removed is generated when a flow is removed due to the insertion of an identical flow. Most existing implementations will not generate a flow removed message in that case.

The following text is added in the OpenFlow 1.0.1 specification :

> When a flow entry is removed, either by the controller deleting the flow or the flow expiry mechanism, the switch must check the flow entry's `OFPFF_SEND_FLOW_REM` flag. If this flag is set, the switch must send a flow removed message to the controller.
>
> 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.

# 4 Specification changes

This section contains errata to the specification changing the intent of the specification. The errata are specifying behaviour and features which are not compliant with the OpenFlow switch specification version 1.0.0.

## 4.1 Source Port for Controller Generated Packet

When the controller sends a packet using a Packet-out message, the Packet-out message includes a field to specify the input port that should be associated with the packet.

The first issue is that the 1.0.0 specification only defines the use of that field with the `OFPP_TABLE` virtual port.

OpenFlow 1.0.0, Section 5.3.6:

> If `OFPP_TABLE` is specified as the output port of an action, the `in_port` in the `packet_out` message is used in the flow table lookup.

This cannot be correct, because this field must be used when processing the packet via the OFPP_IN_PORT, OFPP_NORMAL, OFPP_FLOOD, and OFPP_ALL virtual ports, and also when sending packet to a physical output port, because most of those actions prevent a packet being sent to its incoming interface.

The 1.0.0 specification says that this field may be set with the packet input port, or OFPP_NONE.

OpenFlow 1.0.0, Section 5.3.6:

> uint16_t in_port; /* Packet's input port (OFPP_NONE if none). */

The specification does not define what are the valid packet's input port, this may be interpreted to include other virtual ports or not.

Some controller implementations of 1.0.0 do use OFPP_CONTROLLER as the input port for controller generated packets, which conforms to some interpretations of the specification text, but is contrary to its intent. Other controller implementation use OFPP_NONE.

Similarly, there is variation amongst switch implementations of 1.0.0. Some do not accept OFPP_CONTROLLER and some do not accept OFPP_NONE as the packet input port. The controller must be prepared to generate one or the other depending on the switch.

The usage of OFPP_NONE is potentially problematic for the input port for controller generated packets. OFPP_NONE is not a real port, it is a special value used to wildcard the egress port when requesting flow statistics, and to request port statistics for all ports. For these reasons, some implementations may not allow to set OFPP_NONE in the match, and therefore they do not offer a way to match those packets generated by the controller.

Similarly, usage of OFPP_TABLE, OFPP_NORMAL, OFPP_FLOOD and OFPP_ALL as the packet input port is potentially problematic, as those are not real ports.

The usage of OFPP_CONTROLLER is more logical, which is why many implementations started using it even if it is contrary to the intent of the 1.0 specification.

Subsequent versions of the OpenFlow specification fixed many of those issues. This erratum changes the specification behaviour to be closer to the behaviour of subsequent versions. Those changes should help more predictable implementations and easier forward compatibility to later versions of the spec.

When a packet at the controller can be associated with a physical port of the switch, for example if the packet was received via a packet-in, we recommend that the controller sets the in_port field of the packet-out message with that physical port. We also recommend that controller uses OFPP_NONE in preference to OFPP_CONTROLLER to be compatible with the intent of the 1.0.0 specification. Controller should be ready to use OFPP_NONE with switches compliant with 1.0.0 that don't support OFPP_CONTROLLER.

The following changes are made in the OpenFlow 1.0.1 specification :

> The virtual ports OFPP_IN_PORT, OFPP_TABLE, OFPP_NORMAL, OFPP_FLOOD, and OFPP_ALL cannot be used as an input port.
>
> The in_port field in the packet-out message is the input port that must be associated with the packet for OpenFlow processing. It must be set to either a switch physical port, OFPP_LOCAL, OFPP_CONTROLLER or OFPP_NONE.

### 4.2 Malformed packets

The 1.0.0 specification does not define the behaviour of the OpenFlow switch for malformed or corrupted packets.

The only reasonable option is to define that this is not part of the specification, as no reasonable behaviour can be defined for all cases and hardware will behave differently.

The following text is added in the OpenFlow 1.0.1 specification :

> This version of the specification does *not* define the expected behavior when a switch receives a malformed or corrupted packet.

### 4.3 Controller connection failure behaviour

The 1.0.0 specification defines the behaviour for the OpenFlow switch when the controller connection fails or is terminated and cannot connect to a backup controller. In such a case, the switch is mandated to go in emergency mode.

OpenFlow 1.0.0, Section 4.3:

> If some number of attempts to contact a controller (zero or more) fail, the switch must enter "emergency mode" and immediately reset the current TCP connection. In emergency mode, the matching process is dictated by the emergency flow table entries (those marked with the emergency bit when added to the switch). All normal entries are deleted when entering emergency mode.

There are various issues with emergency mode, some related to implementation and some related to specification.

1. Emergency mode is not widely implemented, the only known implementation is the 1.0 reference implementation. Most implementations of OpenFlow do not support it. This seems to imply that most people deploying OpenFlow don't see this feature as valuable enough to demand its implementation.

2. People familiar with implementation claim that it adds significant complexity to the implementation. It is effectively a parallel flow table that needs to be managed, and the rules for emergency flow entries are different than regular flow entries, emergency flow entries have no timeout, they don't generate flow removed messages...

3. The transition from emergency mode to normal mode is poorly specified. There are two ways to interpret how emergency mode might work. Either packets are directed into an alternate table (the emergency flow table), or the emergency flow entries are copied into the main flow tables. The specification actually hints at both interpretations. Both interpretations would behave quite differently when the switch reconnects to a controller.

4. Some other feature interactions related to emergency mode are not properly specified. For example, the specification does not specify if overlap checking takes into account emergency flow entries or not.

For those reasons, emergency mode was removed from later versions of the OpenFlow specification. OpenFlow 1.1 and later specifies the controller disconnection behaviour as either ”fail-secure” or ”fail-standalone”. Fail-secure was also the behaviour for early OpenFlow specification, including 0.8.9. Fail-secure is very simple to implement and is supported by the vast majority of OpenFlow implementations.

The current OpenFlow 1.0.0 specification mandates the implementation of emergency mode, and does not allow any alternative in handling failure of controller connection. OpenFlow 1.0.1 adds fail-secure as an alternative to emergency mode and allows switches to chose between those two behaviours.

The following changes are made in the OpenFlow 1.0.1 specification :

> If some number of attempts to contact a controller (zero or more) fail, the switch must enter either ”emergency mode” or ”fail-secure mode”, depending upon the switch implementation and configuration, and immediately reset the current TCP connection. Other failure modes may be implemented as a vendor extension, however the default failure mode must be either ”emergency mode” or ”fail-secure mode”.
>
> 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”.

## 4.4 Default TCP port

The 1.0.0 specification defines the default TCP port to transport OpenFlow as port 6633.

OpenFlow 1.0.0, Section 4.4:

> The TLS connection is initiated by the switch on startup to the controller’s server, which is located by default on TCP port 6633.

The OpenFlow header file `openflow.h`, which is part of the specification, defines the same port.

openflow.h, line 74:

```
#define OFP_TCP_PORT 6633
#define OFP_SSL_PORT 6633
```

This TCP port, 6633, has never been registered with IANA for OpenFlow use. Moreover, this port has been recently allocated by IANA for other services. Therefore, port 6633 must not be used as the default OpenFlow port, because this violates IANA policies, may conflict with those other services and may create problems in some deployments.

IANA allocated port 6653 for OpenFlow communications. Port 6653 should be used as the default TCP port for OpenFlow communication, and every implementations should default to using this port. Using this port will avoid conflicts and will help network diagnostics. The switch or controller may use a different TCP port than this default port based on the user need, deployment context, switch implementation and IANA policies, such alternate port would need to be explicitly configured locally on the switch or using a configuration protocol such as OF-Config.

The following changes are made in the OpenFlow 1.0.2 specification :

> The default TCP port for any OpenFlow communication, either using TLS or plain TCP, is 6653. The TLS connection is initiated by the switch on startup to the controller's server, which is located by default on TCP port 6653.

The following changes are made in openflow.h :

```
#define OFP_TCP_PORT 6653
#define OFP_SSL_PORT 6653
```

# Appendix A Release Notes

This section contains release notes highlighting the main changes between the main versions of this errata document.

## A.1 OpenFlow version 1.0.1

Release date : June 7, 2012

Please refers to the bug tracking ID for more details on each change

- Default behaviour on table-miss (EXT-180)
- Port number 0 (EXT-174)
- No padding of error messages (EXT-175)
- Matching packet with no VLAN header (EXT-196)
- Ignoring fields in the match (EXT-177)
- Generating flow removed messages (EXT-176)
- Source Port for Controller Generated Packet (EXT-183)
- Malformed packets (EXT-179)
- Controller connection failure behaviour (EXT-178)

## A.2 OpenFlow version 1.0.2

Release date : October, 2013

Please refers to the bug tracking ID for more details on each change

- Default TCP port (EXT-375/EXT-133)