



OPEN NETWORKING
FOUNDATION

Table synchronisation Extension

Version 0.1

December 23, 2014



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.

WITHOUT LIMITING THE DISCLAIMER ABOVE, THIS SPECIFICATION OF THE OPEN NETWORKING FOUNDATION ("ONF") IS SUBJECT TO THE ROYALTY FREE, REASONABLE AND NONDISCRIMINATORY ("RANDZ") LICENSING COMMITMENTS OF THE MEMBERS OF ONF PURSUANT TO THE ONF INTELLECTUAL PROPERTY RIGHTS POLICY. ONF DOES NOT WARRANT THAT ALL NECESSARY CLAIMS OF PATENT WHICH MAY BE IMPLICATED BY THE IMPLEMENTATION OF THIS SPECIFICATION ARE OWNED OR LICENSABLE BY ONF'S MEMBERS AND THEREFORE SUBJECT TO THE RANDZ COMMITMENT OF THE MEMBERS.

Contents

1	Introduction	3
2	How it works	3
3	Table synchronisation Experimenter ID	4
4	Table synchronisation table feature property	4
5	Table synchronisation error	5

1 Introduction

This document describes an ONF extension for OpenFlow version 1.3.X that supports table synchronisation through a table feature property.

2 How it works

Using this extension, a flow table may be *synchronised* with another flow table. When a flow table is synchronised, its content is automatically updated by the switch to reflect changes in the flow table it is synchronised with. This mechanism enables to perform multiple matches on different views of the same data at different points of the OpenFlow pipeline.

Flows tables can be synchronised bidirectionally or unidirectionally, for any synchronised table a table property describes the source flow table it synchronise from (see below). When a flow table is synchronised, when a flow entry is added, modified or removed in the source flow table, a corresponding flow entry must be automatically added, modified or removed in the synchronised flow table by the switch. If a flow entry is added in the source flow table, and the corresponding flow entry has identical match and priority to a older flow entry already residing in the synchronised flow table, the old and new flow entries must be merged by the switch. If a flow entry is modified or removed in the source flow table, and if the corresponding flow entry no longer exist in the synchronised flow table, no flow entry should be modified or created in the synchronised flow table.

The flow entry created in the synchronised table may not be identical to the corresponding flow entry in the source flow table, as they are often different views on the same data. The translation between those synchronised flow entries is not specified by the OpenFlow protocol and depends on the switch implementation and configuration. For example, synchronised flow entries may have a different instruction set and actions, and the match fields may be transposed (source and destination inverted). The counters of a flow entry must be kept independantly of counters in the source flow table, those counters are not synchronised. In general, synchronised flow entries are created as permanent flow entries (both expiry timers set to zero) so that their lifetime is properly synchronised with the flow entries in the source flow table.

The switch may allow the controller to modify or delete those flow entries automatically added in the synchronised table, and may also allow the controller to create new entries in the synchronised

table. This enables the controller to customised those synchronised flow entries beyond the switch automatic translation. Some of these changes may be lost if the source flow entry is modified. If flow table synchronisation is bidirectional (both flow table synchronised on each other), changes done by the controller on the synchronised flow entry need to be reflected on the source flow entry.

A common example of synchronised tables is the mapping of a Ethernet learning/forwarding hardware table as a set of two synchronised flow tables. In this example, a first flow table expresses the learning lookup and matches on Ethernet source address and source port (and optionally other fields such as VLAN ID). A second flow table expresses the forwarding lookup and matches on Ethernet destination address and set the output port. The two tables synchronise from each other, with the Ethernet addresses transposed in the match. Flow entries in the first table are set to expire based on the learning ageing timer, flow entries in the second table as permanent. The controller can add a flow entry in either of those two flow tables, and then modify the instructions of the flow entry automatically added in the other flow table, alternatively the controller can add a flow entry in each flow table concurrently. Another example of synchronised tables is mapping RPF checks for multicast packets as a flow table.

3 Table synchronisation Experimenter ID

The Experimenter ID of this extension is:

```
ONF_EXPERIMENTER_ID = 0x4F4E4600
```

4 Table synchronisation table feature property

The following table feature properties are defines by this extension.

```
/* Message types */
enum onf_table_exp_type {
    /* Do something. */
    ONF_TT_TABLE_SYNC_FROM = 2320,
};
```

The ONF_TT_TABLE_SYNC_FROM table feature property uses the following property structure :

```
/* Table synchronisation table feature property */
struct onf_table_feature_prop_table_sync_from {
    uint16_t      type;           /* OFPTFPT_EXPERIMENTER. */
    uint16_t      length;         /* Length in bytes of this property. */
    uint32_t      experimenter;   /* ONF_EXPERIMENTER_ID. */
    uint32_t      exp_type;       /* ONF_TT_TABLE_SYNC_FROM. */
    /* Followed by:
     * - Exactly (length - 4) bytes containing the table_ids, then
     * - Exactly (length + 7)/8*8 - (length) (between 0 and 7)
     *   bytes of all-zero bytes */
    uint8_t      table_sync_from_ids[0];
};
OFP_ASSERT(sizeof(struct onf_table_feature_prop_table_sync_from) == sizeof(struct ofp_table_feature_prop_experimenter
```

The `type` field must be set to `OFPTFPT_EXPERIMENTER`.

The `experimenter` field is the Experimenter ID (see 3).

The `exp_type` field is set to `ONF_TT_TABLE_SYNC_FROM`.

The `table_sync_from_ids` field is the array of tables the present table is synchronising content from (see above). When a flow entry is added, modified or removed in one of the flow tables listen in the array, a corresponding flow entry is automatically added, modified or removed in the present flow table.

5 Table synchronisation error

The following errors are defined by this extension.

```
/* Error codes */
enum onf_error_exp_type {
    ONFERR_ET_CANT_SYNC = 2320,          /* Can't synchronise flow entry. */
};
```

The error `ONFERR_ET_CANT_SYNC` uses the following structure:

```
/* Message structure for all errors. */
struct onf_error_msg {
    struct ofp_header header;
    uint16_t type;          /* OFPET_EXPERIMENTER. */
    uint16_t exp_type;      /* One of ONFERR_ET_* above. */
    uint32_t experimenter;  /* ONF_EXPERIMENTER_ID. */
    uint8_t data[0];       /* Up to 64 bytes of failed request. */
};
OFP_ASSERT(sizeof(struct onf_error_msg) == sizeof(struct ofp_error_experimenter_msg));
```

The `type` field must be set to `OPPET_EXPERIMENTER`.

The `experimenter` field is the Experimenter ID (see 3).

The `data` fields contains a copy of the failed request message, truncated to 64 bytes.

The `exp_type` field is set to `ONFERR_ET_CANT_SYNC`.