**OpenFlow v1.0 Messages and Structures**

**Controller-to-Switch Messages**

**Handshake**

***class*ryu.ofproto.ofproto\_v1\_0\_parser.OFPFeaturesRequest(*datapath*)**

Features request message

The controller sends a feature request to the switch upon session establishment.

This message is handled by the Ryu framework, so the Ryu application do not need to process this typically.

Example:

**def** **send\_features\_request**(self, datapath):

ofp\_parser **=** datapath**.**ofproto\_parser

req **=** ofp\_parser**.**OFPFeaturesRequest(datapath)

datapath**.**send\_msg(req)

JSON Example:

{

"OFPFeaturesRequest": {}

}

***class*ryu.ofproto.ofproto\_v1\_0\_parser.OFPSwitchFeatures(*datapath*, *datapath\_id=None*, *n\_buffers=None*, *n\_tables=None*, *capabilities=None*, *actions=None*, *ports=None*)**

Features reply message

The switch responds with a features reply message to a features request.

This message is handled by the Ryu framework, so the Ryu application do not need to process this typically.

| **Attribute** | **Description** |
| --- | --- |
| datapath\_id | Datapath unique ID. |
| n\_buffers | Max packets buffered at once. |
| n\_tables | Number of tables supported by datapath. |
| capabilities | Bitmap of capabilities flag.  OFPC\_FLOW\_STATS  OFPC\_TABLE\_STATS  OFPC\_PORT\_STATS  OFPC\_STP  OFPC\_RESERVED  OFPC\_IP\_REASM  OFPC\_QUEUE\_STATS  OFPC\_ARP\_MATCH\_IP |
| actions | Bitmap of supported OFPAT\_\*. |
| ports | List of **OFPPhyPort** instances. |

Example:

@set\_ev\_cls(ofp\_event**.**EventOFPSwitchFeatures, CONFIG\_DISPATCHER)

**def** **switch\_features\_handler**(self, ev):

msg **=** ev**.**msg

self**.**logger**.**debug('OFPSwitchFeatures received: '

'datapath\_id=0x%016x n\_buffers=%d '

'n\_tables=%d capabilities=0x%08x ports=%s',

msg**.**datapath\_id, msg**.**n\_buffers, msg**.**n\_tables,

msg**.**capabilities, msg**.**ports)

JSON Example:

{

"OFPSwitchFeatures": {

"actions": 2115,

"capabilities": 169,

"datapath\_id": 1095522080376,

"n\_buffers": 0,

"n\_tables": 255,

"ports": {

"6": {

"OFPPhyPort": {

"advertised": 640,

"config": 0,

"curr": 648,

"hw\_addr": "f2:0b:a4:7d:f8:ea",

"name": "Port6",

"peer": 648,

"port\_no": 6,

"state": 2,

"supported": 648

}

},

"7": {

"OFPPhyPort": {

"advertised": 640,

"config": 0,

"curr": 648,

"hw\_addr": "f2:0b:a4:d0:3f:70",

"name": "Port7",

"peer": 648,

"port\_no": 7,

"state": 16,

"supported": 648

}

}

}

}

}

**Switch Configuration**

***class*ryu.ofproto.ofproto\_v1\_0\_parser.OFPSetConfig(*datapath*, *flags=None*, *miss\_send\_len=None*)**

Set config request message

The controller sends a set config request message to set configuraion parameters.

| **Attribute** | **Description** |
| --- | --- |
| flags | One of the following configuration flags.  OFPC\_FRAG\_NORMAL  OFPC\_FRAG\_DROP  OFPC\_FRAG\_REASM  OFPC\_FRAG\_MASK |
| miss\_send\_len | Max bytes of new flow that datapath should send to the controller. |

Example:

**def** **send\_set\_config**(self, datapath):

ofp **=** datapath**.**ofproto

ofp\_parser **=** datapath**.**ofproto\_parser

req **=** ofp\_parser**.**OFPSetConfig(datapath, ofp**.**OFPC\_FRAG\_NORMAL, 256)

datapath**.**send\_msg(req)

***class*ryu.ofproto.ofproto\_v1\_0\_parser.OFPGetConfigRequest(*datapath*)**

Get config request message

The controller sends a get config request to query configuration parameters in the switch.

Example:

**def** **send\_get\_config\_request**(self, datapath):

ofp\_parser **=** datapath**.**ofproto\_parser

req **=** ofp\_parser**.**OFPGetConfigRequest(datapath)

datapath**.**send\_msg(req)

***class*ryu.ofproto.ofproto\_v1\_0\_parser.OFPGetConfigReply(*datapath*)**

Get config reply message

The switch responds to a configuration request with a get config reply message.

| **Attribute** | **Description** |
| --- | --- |
| flags | One of the following configuration flags.  OFPC\_FRAG\_NORMAL  OFPC\_FRAG\_DROP  OFPC\_FRAG\_REASM  OFPC\_FRAG\_MASK |
| miss\_send\_len | Max bytes of new flow that datapath should send to the controller. |

Example:

@set\_ev\_cls(ofp\_event**.**EventOFPGetConfigReply, MAIN\_DISPATCHER)

**def** **get\_config\_reply\_handler**(self, ev):

msg **=** ev**.**msg

dp **=** msg**.**datapath

ofp **=** dp**.**ofproto

**if** msg**.**flags **==** ofp**.**OFPC\_FRAG\_NORMAL:

flags **=** 'NORMAL'

**elif** msg**.**flags **==** ofp**.**OFPC\_FRAG\_DROP:

flags **=** 'DROP'

**elif** msg**.**flags **==** ofp**.**OFPC\_FRAG\_REASM:

flags **=** 'REASM'

**elif** msg**.**flags **==** ofp**.**OFPC\_FRAG\_MASK:

flags **=** 'MASK'

**else**:

flags **=** 'unknown'

self**.**logger**.**debug('OFPGetConfigReply received: '

'flags=%s miss\_send\_len=%d',

flags, msg**.**miss\_send\_len)