NVO3 WG
Internet-Draft
Intended status: Standards Track
Expires: December 9, 2016

Fangwei. Hu
Ran. Chen
ZTE Corporation
Mallik. Mahalingam
Springpath
Qiang. Zu
Ericsson
S. Davari
yahoo

June 7, 2016

YANG Data Model for VxLAN Protocol
draft-chen-nvo3-vxlan-yang-03.txt

Abstract

This document defines a YANG data model for VxLAN protocol.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 9, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Terminology 2
3. YANG Data Model for VxLAN Configuration 2
3.1. VxLAN Multicast IP Address 2
3.2. Inner VLAN Tag Handling Mode 2
4. Design Tree of Configuration Data Model 3
5. Design Tree of Operational State Model 3
6. VxLAN YANG Model 4
7. Security Considerations 14
8. Acknowledgements 14
9. IANA Considerations 14
10. Normative References 14
Authors' Addresses 15

1. Introduction

YANG[RFC6020]is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. This document defines a YANG data model for the configuration of VxLAN protocol [RFC7348].

2. Terminology

3. YANG Data Model for VxLAN Configuration

3.1. VxLAN Multicast IP Address

The vxlan-multicast-ip is used to configure the IP multicast group, which the VxLAN VNI of the VTEP is mapping to.

3.2. Inner VLAN Tag Handling Mode

There are two handling modes for the inner VLAN tag: discard-inner-vlan mode and no-discard-inner-vlan mode. If the VTEP interface works in the discard-inner-vlan mode, the VxLAN is only mapped to one VLAN. The inner VLAN tag will be stripped when encapsulating the VxLAN frame. On the decapsulation side, if VTEP receives the VxLAN frame with inner VLAN tag, it will discard the frame in this work mode. If the VTEP receives the VxLAN frame without VLAN tag, it will fill in the VLAN tag based on the VxLAN and VLAN mapping.

If the VTEP interface works in the no-discard-inner-vlan mode, the VxLAN could be mapped to several VLANs. The inner VLAN tag will not

stripped when encapsulating the VxLAN frame in the VxLAN encapsulation side. On the decapsulation side, if VTEP receives the VxLAN frame, it will strip the VxLAN header, and keep the VLAN frame.

4. Design Tree of Configuration Data Model

```
module: ietf-vxlan
   +--rw vxlan
   |  +--rw global-enable             empty
   |  +--rw vxlan-instance* [vxlan-id]
   |  |  +--rw vxlan-id               vxlan-id
   |  |  +--rw description?           string
   |  |  +--rw unknow-unicast-drop?   enumeration
   |  |  +--rw filter-vrrp?           enumeration
   |  |  +--rw vxlan-evpn
   |  |     +--rw route-distinguisher?   string
   |  |     +--rw vpn-targets* [rt-value]
   |  |        +--rw rt-value    string
   |  |        +--rw rt-type     bgp-rt-type
   |  +--rw vtep-instances* [vtep-id]
   |  |  +--rw vtep-id                     uint32
   |  |  +--rw vtep-name?                  string
   |  |  +--rw source-interface?           if:interface-ref
   |  |  +--rw multicast-ip                inet:ip-address
   |  |  +--rw inner-vlan-handling-mode?   inner-vlan-handling-mode
   |  |  +--rw bind-vxlan-id* [vxlan-id]
   |  |     +--rw vxlan-id    vxlan-id
   |  +--rw static-vxlan-tunnel* [vxlan-tunnel-id]
   |  |  +--rw vxlan-tunnel-id      uint32
   |  |  +--rw vxlan-tunnel-name?   string
   |  |  +--rw address-family* [af]
   |  |     +--rw af                       address-family-type
   |  |     +--rw tunnel-source-ip?        address-family-type
   |  |     +--rw tunnel-destination-ip?   address-family-type
   |  |     +--rw bind-vxlan-id* [vxlan-id]
   |  |        +--rw vxlan-id    vxlan-id
   |  +--rw redundancy-group-binds
   |     +--rw redundancy-group-bind* [vxlan-id redundancy-group]
   |        +--rw vxlan-id            uint32
   |        +--rw redundancy-group    uint32
```

5. Design Tree of Operational State Model

```
          +--ro vxlan-state
             +--ro vxlan
                +--ro vxlan-tunnels
                   +--ro vxlan-tunnel* [local-ip remote-ip]
                      +--ro local-ip            address-family-type
                      +--ro remote-ip           address-family-type
                      +--ro static-tunnel-id?   uint32
                      +--ro evpn-tunnel-id?     uint32
                      +--ro statistics
                         +--ro tunnel-statistics
                         |  +--ro in-bytes?      string
                         |  +--ro out-bytes?     string
                         |  +--ro in-packets?    string
                         |  +--ro out-packets?   string
                         +--ro tunnel-vni-statistics
                            +--ro tunnel-vni-statistic* [vxlan-id]
                               +--ro vxlan-id       uint32
                               +--ro in-bytes?      string
                               +--ro out-bytes?     string
                               +--ro in-packets?    string
                               +--ro out-packets?   string
```

6. VxLAN YANG Model

```
<CODE BEGINS> file "ietf-vxlan@2016-06-02.yang"
module ietf-vxlan {
    namespace "urn:ietf:params:xml:ns:yang:ietf-vxlan";
    prefix "vxlan";

    import ietf-interfaces {
      prefix "if";
     }

    import ietf-inet-types {
       prefix "inet";
     }

    import ietf-yang-types {
       prefix yang;
     }

    organization
      "IETF NVO3(Network Virtualization Overlays) Working Group";

      contact
```

```
       "
         WG List:  <mailto:nvo3@ietf.org>

         WG Chair: Matthew Bocci
                  <mailto:matthew.bocci@alcatel-lucent.com>

         WG Chair: Benson Schliesser
                  <mailto:bensons@queuefull.net>

         Editor:   Fangwei Hu
                   <mailto:hu.fangwei@zte.com.cn>

         Editor:   Ran Chen
                   <mailto:chen.ran@zte.com.cn>

         Editor:   Mallik Mahalingam
                   <mailto:mallik_mahalingam@yahoo.com>

         Editor:   Zu Qiang
                   <mailto:Zu.Qiang@Ericsson.com>
        ";

    description
       "The YANG module defines a generic configuration
         model for VxLAN protocol";

    revision 2016-06-02 {
       description
         "03 revision. Update the YANG data model based on thec comments of
IETF 96th meeting.";
       reference
         "draft-chen-nvo3-vxlan-yang-02";
     }

    revision 2015-12-01 {
       description
         "02 revision.";
       reference
         "draft-chen-nvo3-vxlan-yang-02";
     }

    revision 2015-10-12 {
       description
         "01 revision.";
       reference
         "draft-chen-nvo3-vxlan-yang-01";
     }

    revision 2015-05-05 {
```

```
      description "Initial revision";
      reference
         "draft-chen-nvo3-vxlan-yang-00";
    }

    /* Typedefs  */

    typedef bgp-rt-type {
      type enumeration {
        enum import {
        description "For import";
       }
       enum export {
         description "For export";
       }
       enum both {
         description "For both import and export";
       }
     }
     description "BGP route-target type. Import from BGP YANG";
    }


    typedef vlan {
      type uint16 {
        range 1..4094;
      }
      description
      "Typedef for VLAN";
    }

    typedef vxlan-id {
      type uint32;
      description
        "Typedef for VxLAN ID.";
    }

    typedef inner-vlan-handling-mode {
       type enumeration {
         enum discard-inner-vlan {
              description
                "Discard inner-VLAN.";
         }
         enum no-discard-inner-vlan {
              description
                "No discard inner-VLAN.";
         }
       }
```

```
       default discard-inner-vlan ;
       description
        "Typedef for inner-vlan-handling-mode";
    }

    typedef address-family-type {
       type enumeration {
         enum ipv4 {
           description
             "IPv4";
         }
         enum ipv6 {
           description
             "IPv6";
         }
       }
       description
         "Typedef for address family type.";
     }

    /* Configuration Data */

      container vxlan{

        leaf global-enable {
          type empty ;
          mandatory true ;
          description 'VXLAN global enble.';
        }

        list vxlan-instance {
           key vxlan-id ;
           leaf vxlan-id {
             type vxlan-id;
             description "VxLAN ID.";
           }

           leaf description {
            type string {
              length 0..64 {
                description 'VXLAN instance description information.';
              }
            }
            description 'The description information of VXLAN instance.';
           }

           leaf unknow-unicast-drop {
            type enumeration {
```

```
              enum enable {
                value 1 ;
                description 'Unknown unicast drop enable.';
              }
              enum disable {
                value 2 ;
                description 'Unknown unicast drop disable.';
              }
            }
            default enable ;
            description 'Unknow unicast drop configuration of VXLAN instance.';
           }

           leaf filter-vrrp {
             type enumeration {
              enum enable {
                value 1 ;
                description 'VRRP packets filter.';
              }
              enum disable {
                value 2 ;
                description 'VRRP packets not filter.';
              }
            }
            default enable ;
            description 'VRRP packets filter configuration of VXLAN instance.';
           }

           container vxlan-evpn {
              leaf route-distinguisher {
                type string;
                description "VxLAN EVPN BGP RD";
              }

              list vpn-targets {
                key rt-value;
                description "VxLAN EVPN Route Targets";
                leaf rt-value {
                  type string;
                  description "Route-Target value";
                }

                leaf rt-type {
                  type bgp-rt-type;
                  mandatory true;
                  description "Type of RT";
                }
              }
```

```
              description
                 "EVPN VxLAN parameters.";
            }
          description
              "VxLAN instance.";
        }


        list vtep-instances {
            key vtep-id ;
            leaf vtep-id {
               type uint32;
               description
               "VTEP ID.";
            }

            leaf vtep-name{
               type string;
               description
                "VTEP instance name.";
            }

            leaf source-interface {
               type if:interface-ref;
               description
                     "Source interface name.";
            }

            leaf multicast-ip {
                type inet:ip-address;
                mandatory true ;
                description
                  "VxLAN multicast IP address.";
            }

            leaf inner-vlan-handling-mode {
                type inner-vlan-handling-mode;
                description
                   "The inner vlan tag handling mode.";
            }

            list bind-vxlan-id {
                key vxlan-id;
                leaf vxlan-id {
                  type vxlan-id;
                  description
                       "VxLAN ID.";
                }
```

```
                description
                  "VxLAN ID list for the VTEP.";
            }
            description
                "VTEP instance.";
        }

        list static-vxlan-tunnel{
            key vxlan-tunnel-id;
            leaf vxlan-tunnel-id {
               type uint32;
               description
                     "Static VxLAN tunnel ID.";
            }

            leaf vxlan-tunnel-name {
               type string;
               description
               "Name of the static VxLAN tunnel.";
            }

            list address-family {
               key "af";
               leaf af {
                 type address-family-type;
                 description
                 "Address family type value.";
               }

               leaf tunnel-source-ip {
                 type address-family-type;
                 description
                 "Source IP address for the static VxLAN tunnel";
               }

               leaf tunnel-destination-ip {
                  type address-family-type;
                  description
                   "Destination IP address for the static VxLAN tunnel";
               }

               list bind-vxlan-id {
                  key vxlan-id;
                  leaf vxlan-id {
                    type vxlan-id;
                    description
                       "VxLAN ID.";
                  }
```

```
                  description
                     "VxLAN ID list for the VTEP.";
               }

                description
                   "Per-af params.";
             }
            description
              "Configure the static VxLAN tunnel";
        }

        container redundancy-group-binds {
           list redundancy-group-bind {
              key 'vxlan-id redundancy-group';
              leaf vxlan-id {
                type uint32 {
                  range 1..16777215 {
                    description 'The value of VXLAN,it must between 1 to
16777215.';
                  }
                }
                description 'VXLAN ID binding by redundancy group.';
              }

              leaf redundancy-group {
                type uint32 {
                   range 1..4294967293 {
                      description 'The value of redundancy group,it must
between 1 to'
                           + ' 4294967293.';
                    }
                }
                description 'Redundancy group ID.';
              }
              description 'Redundancy group bind table.';
           }
           description 'Redundancy group bind table.';
        }

     description
       "VxLAN configure model.";
    }

      /* Operational  data */
    container vxlan-state{
      config false ;
      container vxlan {
        container vxlan-tunnels {
```

```
          list vxlan-tunnel {
             key 'local-ip remote-ip';
```

```
             leaf local-ip {
               type address-family-type ;
               description 'Local IP of tunnel.';
             }

             leaf remote-ip {
               type address-family-type ;
               description 'Remote IP of tunnel.';
             }

             leaf static-tunnel-id {
               type uint32 ;
               description 'Static tunnel ID.';
             }

             leaf evpn-tunnel-id {
               type uint32 ;
               description 'EVPN tunnel ID.';
             }

            container statistics {
               container tunnel-statistics {
                 leaf in-bytes {
                    type string {
                      length 0..24 ;
                    }
                    description 'Total bytes received.';
                 }

                 leaf out-bytes {
                   type string {
                     length 0..24 ;
                   }
                   description 'Total bytes sent.';
                 }

                 leaf in-packets {
                   type string {
                      length 0..24;
                   }
                   description 'Total packets received.';
                 }

                 leaf out-packets {
                   type string {
                     length 0..24 ;
                   }
                   description 'Total packets sent.';
```

```
                 }
                 description 'Total tunnel statistics.';
               }

               container tunnel-vni-statistics {
                list tunnel-vni-statistic {
                 key vxlan-id ;
                 leaf vxlan-id {
                   type uint32 ;
                   description 'The VXLAN in tunnel.';
                 }

                 leaf in-bytes {
                   type string {
                     length 1..24 ;
                   }
                   description 'Total bytes received.';
                 }

                 leaf out-bytes {
                   type string {
                     length 1..24 ;
                   }
                   description 'Total bytes sent.';
                 }

                 leaf in-packets {
                   type string {
                     length 1..24 ;
                   }
                   description 'Total packets received.';
                 }

                 leaf out-packets {

                   type string {
                     length 1..24 ;
                   }
                   description 'Total packets sent.';
                 }
                 description 'Statistics in VXLAN tunnel.';
               }
               description 'Statistics in VXLAN tunnel.';
             }
              description 'Tunnel statistics.' ;
           }
             description 'VXLAN tunnel info.';
         }
```

```
          description 'VXLAN tunnel Info.';
      }
      description 'Information of VXLAN state.';
    }
    description 'Information of VXLAN state.';
   }

  }
 <CODE ENDS>
```

7. Security Considerations

8. Acknowledgements

9. IANA Considerations

This document requires no IANA Actions. Please remove this section before RFC publication.

10. Normative References

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <http://www.rfc-editor.org/info/rfc7348>.

Authors' Addresses

Fangwei Hu
ZTE Corporation
No.889 Bibo Rd
Shanghai 201203
China

Phone: +86 21 68896273
Email: hu.fangwei@zte.com.cn

Ran Chen
ZTE Corporation
No.50 Software Avenue,Yuhuatai District
Nanjing, Jiangsu Province 210012
China

Phone: +86 025 88014636
Email: chen.ran@zte.com.cn

Mallik Mahalingam
Springpath
640 W. California Ave, Suite #110
Sunnyvale, CA 94086
USA

Email: mallik_mahalingam@yahoo.com

Zu Qiang
Ericsson
8400, boul. Decarie
Ville Mont-Royal, QC
Canada

Email: Zu.Qiang@Ericsson.com

Davari Shahram
yahoo

Email: davarish@yahoo.com