Network Working Group
Internet-Draft
Intended status: Informational
Expires: June 25, 2015

S. Zhuang
Z. Li
Huawei Technologies
December 22, 2014

Yang Model for Ethernet VPN
draft-zhuang-bess-evpn-yang-00

Abstract

This document defines a YANG data model that can be used to configure and manage Ethernet VPN.

Requirements Language

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 [RFC2119].

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 June 25, 2015.

Copyright Notice

Copyright (c) 2014 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

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]. YANG is proving relevant beyond its intial confines, as bindings to other interfaces(e.g. ReST) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interface, such as CLI and programatic APIs.

This document defines a YANG data model that can be used to configure and manage Ethernet VPN defined in [I-D.ietf-l2vpn-evpn].

2. Terminology

EVN: Ethernet Virtual Network

EVPN: Ethernet VPN

ESI: Ethernet Segment Identifier

3. Design of Data Model

3.1. Overview

The EVPN Yang module is divided in following containers :

o interfaces : that contains writable configuration objects of interface binded with EVPN.

o evpn : that contains global writable configuration objects of EVPN.

o evn-bgp: that contains writable configuration objects of MP-BGP used for EVPN.

o evn-instances : that contains writable configuration objects of EVPN instance.

The figure below describe the overall structure of the EVPN Yang module :

```
module: evn
   +--rw interfaces
   |  ...
   +--rw evn
   |  ...
   +--rw evn-bgp
   |  ...
   +--rw evn-instances
      ...
```

3.2. EVPN Interface Configuration

EVPN interface configuration includes the interface name and Ethernet Segment Identifier(ESI).

```
   +--rw interfaces
   |  +--rw interface* [name]
   |     +--rw name    leafref
   |     +--rw esi?    string
```

3.3. EVPN Global Configuration

EVPN global configuration includes the global parameters for ARP cache.

```
   +--rw evn
   |  +--rw arp-cache-disable?   boolean
   |  +--rw arp-cache-timeout?   uint32
```

3.4. MP-BGP Configuration for EVPN

The traditional configuration model of BGP is defined in [I-D.zhdankin-netmod-bgp-cfg]. In order to satifsy the requirment of reducing operation cost, this document proposes a new model of MP-BGP configuration for EVPN. A independent evn-bgp container is defined in EVPN Yang model to contains writable configuration objects of MP-BGP used for EVPN. It can directly configure MP-BGP peers for EVPN using the bgpPeers container. In addition, BGP router reflector can be introduced to reduced the configuration work for EVPN since when BGP router reflector is introduced each EVPN BGP client only needs to set up BGP peer with the router reflector. For BGP router reflector used for EVPN, it can enable the dynamic BGP peer setup mode to set up BGP peer with EVPN BGP client through the auto-discovery mechanism. Or it can adopt the traditional method to statically designate the list of EVPN BGP clients. The set-route-reflect-function container contains the writable configuration objects of BGP route reflector used for EVPN.

Besides above configuration, EVPN BGP configuation also includes the parameters of BFD and MAC limit.

```
      +--rw evn-bgp
      |  +--rw bfd
      |  |  +--rw isBfdEnable?   boolean
      |  |  +--rw txInterval?    uint32
      |  |  +--rw rxInterval?    uint32
      |  |  +--rw multiplier?    uint8
      |  +--rw mac-limit-per-peer
      |  |  +--rw mac-limit-value?          uint32
      |  |  +--rw mac-limit-alert-percent?   uint8
      |  |  +--rw (mac-limit-action)?
      |  |     +--:(enable-alert-only)
      |  |     |  +--rw alert-only?                boolean
      |  |     +--:(enable-idle-forever)
      |  |     |  +--rw idle-forever?              boolean
      |  |     +--:(enable-idle-timeout)
      |  |        +--rw idle-timeout?              uint16
      |  +--rw source-address?               inet:ip-address
      |  +--rw bgpPeers
      |  |  +--rw bgpPeer* [peerAddr]
      |  |     +--rw peerAddr    inet:ip-address
      |  +--rw set-route-reflect-function
      |     +--rw (set-type)?
      |     |  +--:(static)
      |     |  |  +--rw bgp-clients
      |     |  |     +--rw bgp-client* [clientAddr]
      |     |  |        +--rw clientAddr    inet:ip-address
      |     |  +--:(dynamic)
      |     |     +--rw server-enable?     boolean
      |     +--rw redundancy-mode?   enumeration
      |     +--rw df-delay-timer?    uint16
      |     +--rw timer
      |        +--rw keepaliveTime?   uint16
      |        +--rw holdTime?        uint16
```

3.5. EVPN Instance Configuration

EVPN instance configuration includes EVPN instance name, EVPN ID, and VLAN IDs in the VPN instance.

```
   +--rw evn-instances
      +--rw evn-instance* [evn-instance-name]
         +--rw evn-instance-name    string
         +--rw evn-id?              uint16
         +--rw vlan-ids
            +--rw vlan-id* [vlan-id-number]
               +--rw vlan-id-number    uint16
```

4. EVPN Yang Module

EVN YANG MODEL
<CODE BEGINS> file "evn@2014-08-17.yang"

```
module evn {
  namespace "urn:huawei:params:xml:ns:yang:evn";
  // replace with IANA namespace when assigned
  prefix "evn";

  import ietf-interfaces {
    prefix if;
    //rfc7223-YANG Interface Management
  }

  import ietf-inet-types {
    prefix inet;
    //RFC6991
  }

  description
    "This YANG module defines the generic configuration data for
     EVN service.

     Terms and Acronyms

     EVN: Ethernet Virtual Network
     EVPN: Ethernet VPN
     ESI: Ethernet Segment Identifier

    ";

  revision 2014-08-17 {
    description
      "Initial revision.";
  }

  /*
   * ethernet segment ID config.
   */
  container interfaces {
    list interface {
      key "name";
      leaf name {
        type leafref {
          path "/if:interfaces/if:interface/if:name";
        }
      }
      leaf esi {
```

```
        description
          "Specify the ethernet segment ID.";

        config "true";
        type string {
          length "24";
          pattern "(^00([0-9a-fA-F]){2}\.(([0-9a-fA-F]){4}\.){3}
          (([0-9a-fA-F]){4})$)";
        }
      }
    }
  }

  /*
   * Enable Ethernet Virtual Network.
   */
  container evnGlobal {

    leaf evnEnable {
      config "true";
      type "boolean";
      default "false";
    }

    leaf arp-cache-disable {
      config "true";
      type boolean;
      default "false";
    }

    leaf arp-cache-timeout {
      config "true";
      type uint32 {
        range "0..100000";
      }
      default "240";
    }

  }
  /*
   * Configuring BFD for EVN BGP.
   */
  container evn-bgp {

    container bfd {
      leaf isBfdEnable {
        description "Enable BFD";
```

```
        config "true";
        type boolean;
        default "false";
      }

      leaf txInterval {
        description "Specify the minimum transmit interval";

        config "true";
        type uint32 {
          range "0..4294967295";
        }
      }
      leaf rxInterval {
        description "Specify the minimum receive interval";

        config "true";
        type uint32 {
          range "0..4294967295";
        }
      }
      leaf multiplier {
        description "Specify the detect multiplier";
        config "true";
        default "3";
        type uint8 {
          range "3..50";
        }
      }


    }

    container mac-limit-per-peer {

      leaf mac-limit-value {
        description
          "Specify Mac route limit value.";

        config "true";
        type uint32 {
          range "1..4294967295";
        }
      }
      leaf mac-limit-alert-percent {
        description
          "Specify maximum percentage value. Start to generate
           warning messages if it reaches maximum percentage
```

```
           value";

        config "true";
        type uint8 {
          range "1..100";
        }
        default "75";
      }

      choice mac-limit-type {

        case enable-alert-only {
          leaf alert-only {
            description
              "Allows the router to generate log message without
               terminating session when the maximum is exceeded.";

            config "true";
            type boolean;
            default "false";
          }
        }
        case enable-idle-forever {
          leaf idle-forever {
            description
              "Do not auto-connect-retry until reset bgp when the
               maximum is exceeded and then terminating session.";

            config "true";
            type boolean;
            default "false";
          }
        }
        case enable-idle-timeout {
          leaf idle-timeout {
            description
              "Specify Value of idle-timeout timer(minutes).
               Auto-connect-retry after timeout when the maximum is
               exceeded and then terminating session.";

            config "true";
            type uint16 {
              range "1..1200";
            }
          }
        }
      }
    }
```

```
    leaf source-address {
      config "true";
      type inet:ip-address;
    }

    /*
     * Configuring an Authentication Mode for EVN BGP.
     */
    container authentication {
      description
        "To improve network security, you can configure MD5 or
         Keychain authentication for EVN BGP peers when they
         set up a TCP connection.
        ";

      leaf cipherPassword {
        config "true";
        type "string";
      }
      leaf keychainName {
        config "true";
        type "string";
      }
    }

    container bgpPeers {
      list bgpPeer {
      key "peerAddr";
      max-elements "unbounded";
      min-elements "0";
        description
          "BGP Peer configure class.";

        leaf peerAddr {
          description
            "The nerighbor address.";
            config "true";
            type inet:ip-address;
            mandatory true;
        }

      }
    }


    container set-route-reflect-function {
      description
        "Configure an EVN BGP RR to reduce the number of EVN BGP
```

```
         peer connections, saving network resources.";

      choice set-type {
        description
          "An EVN BGP RR can be manually specified or dynamically
           configured.

           Static RR:
           After a static RR is configured, you need to manually
           establish peer relationships between the RR and other
           PE devices and specify the PE devices as the RR clients.

           Dynamic RR:
           Only a non-PE device can be configured as a dynamic RR.
           After a device is configured as a dynamic RR, it can
           automatically set up peer relationships with devices
           specified by the peer ip-address command (ip-address is
           the source address of the dynamic RR).
           ";

        case static {
          container bgp-clients {
            list bgp-client {
              key "clientAddr";
              max-elements "unbounded";
              min-elements "0";
              description
                "Configure some peers as route reflector clients.";

              leaf clientAddr {
                description
                  "The client address. A static RR is configured. Only
                   the specified peers can become the RR clients.";

                config "true";
                type inet:ip-address;
              }
            }
          }
        }

        case dynamic {
          leaf server-enable {
            description
              "Enable Server function for dynamic peer. A dynamic RR is
               configured. After a dynamic RR is configured, all PE
               devices that have established peer relationships with the
               RR can become the RR clients.";
```

```
            type boolean;
            default "false";
          }
        }

      }


      leaf redundancy-mode {
        description
          "Specify redundancy-mode.";

        config "true";
        type enumeration {
          enum "single-active";
          enum "all-active";
        }
        default "single-active";
      }


      leaf df-delay-timer{
        description
          "Specify designated forwarder election delay-timer
           value(seconds).";

        config "true";
        type uint32 {
          range "1..1200";
        }
        default "60";
      }

      container timer {
        leaf keepaliveTime {
          description "Specifies the Keepalive interval";
          config "true";
          default "60";
          type uint16 {
            range "0..21845";
          }
        }
        leaf holdTime {
          description "Specifies the Holdtime interval";
          config "true";
          default "180";
          type uint16 {
            range "0..65535";
```

```
          }
        }
      }


    }


  }


  container evnInstances {
    description
      "EVN instance configuration parameters.";

    list evnInstance {
      max-elements "unbounded";
      min-elements "0";
      key "evnName";

      leaf evnName {
        description
          "EVN Instance Name";

        config "true";
        type string {
          length "1..31";
        }
      }

      leaf evnId {
        description
          "Specify the EVN instance id. Each EVN instance has a unique
           ID.";

        config "true";
        type uint32 {
          range "1..65535";
        }
      }

      container vlanList {
        description
          "Specify a vlan list.";

        list vlan-id {
          key "vlan-id-number";
```

```
          leaf vlan-id-number {
            type uint16 {
              range "1..4094";
            }
          }
        }
      }

      container exportAclNameOrId {
        description
          "Filter outgoing routing updates. To accurately control EVN
           routes, configure an export routing policy. The export
           routing policy filters routes before they are advertised to
           other PE devices.";

        choice aclNumOrName {
          case Specify-aclNum {
            leaf aclNum {
              config "true";
              type uint16 {
                range "2000..2999";
              }
            }
          }
          case Specify-aclName {
            leaf aclName {
              config "true";
              type string;
            }
          }
        }
      }

      container importAclNameOrId {
        description
          "Set route filtering policy. To accurately control EVN routes,
           configure an import routing policy. The import routing policy
           filters routes received from other PE devices.";

        choice aclNumOrName {
          case Specify-aclNum {
            leaf aclNum {
              config "true";
              type uint16 {
                range "2000..2999";
              }
            }
          }
          case Specify-aclName {
            leaf aclName {
              config "true";
              type string;
            }
          }
        }
      }

      container importAclNameOrId {
        description
          "Set route filtering policy. To accurately control EVN routes,
           configure an import routing policy. The import routing policy
           filters routes received from other PE devices.";

        choice aclNumOrName {
          case Specify-aclNum {
            leaf aclNum {
              config "true";
              type uint16 {
                range "2000..2999";
              }
            }
          }
          case Specify-aclName {
            leaf aclName {
              config "true";
              type string;
            }
          }
        }
      }
```

```
          case Specify-aclName {
            leaf aclName {
              config "true";
              type string;
            }
          }
        }
      }

      container evnInstanceInfo {
        description
          "Display the information of the evn instance.
           It is intended that this container may be augmented by
           vendors to reflect the vendor-specific operational state
           parameters.";

        leaf exportRT {
          config "false";
          type "string";
        }
        leaf importRT {
          config "false";
          type "string";
        }
        leaf evnRd {
          config "false";
          type "string";
        }
      }

    }

  }


}
</CODE ENDS>
```

5. IANA Considerations

This document makes no request of IANA.

6. Security Considerations

This document does not introduce any new security risk.

7. Acknowledgements

The authors would like to thank Guangying Zheng, Gang Yan for their contributions to this work.

8. References

[I-D.ietf-l2vpn-evpn]
Sajassi, A., Aggarwal, R., Bitar, N., Isaac, A., and J. Uttaro, "BGP MPLS Based Ethernet VPN", draft-ietf-l2vpn-evpn-11 (work in progress), October 2014.

[I-D.zhdankin-netmod-bgp-cfg]
Alex, A., Patel, K., and A. Clemm, "Yang Data Model for BGP Protocol", draft-zhdankin-netmod-bgp-cfg-01 (work in progress), October 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

Authors' Addresses

Shunwan Zhuang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: zhuangshunwan@huawei.com

Zhenbin Li
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com