BESS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 7, 2015

Kishore Tiruveedhula, Ed.
Tapraj Singh
Juniper Networks
Ali Sajassi
Deepak Kumar
Cisco Systems
Luay Jalil
Verizon

March 6, 2015

YANG Data Model for PBB EVPN protocol
draft-tsingh-bess-pbb-evpn-yang-cfg-00

Abstract

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

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 [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 September 7, 2015.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG data model for PBB EVPN protocol.

This yang data model includes both configuration of an PBB EVPN protocol instance as well as operational states.

2. Terminology

ISID : Instance Service Identifier

ESI : Ethernet Segment Identifier

BMAC : Backbone MAC

C-BD : Customer Bridge domain

B-BD : Backbone Bridge domain

B-Comp: B-Component instance

I-Comp: I-Component instance

CE : Customer Edge device

CIP : Customer Instance Port

PIP : Provider Instance Port

CBP : Customer Backbone Port

PBP : Provider Backbone Port

3. Design of the Data Model

The PBB EVPN YANG module is divided into two containers. One is I-Component container and other one is B-Component container.

The I-component is responsible for mapping of traffic from CE to the customer bridge domain. The customer bridge domain is mapped to an ISID. Within the I-component, there are two different ports, one is customer instance port and other one is provider instance port. The I-component container contains customer bridge domain information, ISID, the customer instance port (CIP) and provider instance port (PIP).

The B-component is responsible to learn and forward packets based on the Backbone MAC addresses (BMACs). Within the B-component, there are two different ports, one is customer backbone port (CBP) and provider backbone port (PBP). The B-component container which contains PBB EVPN specific information and backbone bridge domain information which maps the I-component traffic to B-Component towards the MPLS core.

The figure below describe the overall structure of the PBB EVPN YANG module:

```
module: pbb-evpn
   +--rw interfaces
   |  +--rw interface* [name]
   |     +--rw name     leafref
   |     +--rw esi?     string
   |     +--rw redudancy-mode?   boolean
   |     +--rw auto-source-bmac?   boolean
   |     +--rw source-bmac?   yang:mac-address
   +--rw logicalinterfaces
   |  +--rw interface* [name]
   |     +--rw name     leafref
   |     +--rw encap?   uint16
   +--rw b-comp-instance
   |  +--rw instance-name    string
   |  +--rw route-distinguisher?  string
   |  +--rw auto-route-target?   boolean
   |  +--rw route-target?         string
   |  +--rw protocol?   enumeration
   |  +--rw control-word?      boolean
   |  +--rw cbp interface [name]
   |  +--rw b-bridge-domain*
   |  |  +--rw nvlanid?   uint16
   |  |  +--rw isid*
   |  |     +--rw isid?   uint32
   |  |     +--rw extended?   boolean
   |   ...
   +--rw i-comp-instance
      +--rw pip interface [name]
      +--rw cbd*
      |  +--rw member-interface
      |  |  +--rw memberifs*
      |  +--rw nvlanid?    uint16
      |  +--rw isid?       uint32
      +--rw peer-b-component?   string
```

4. B-Component Configuration

The B-component configuration contains EVPN instance name, route distinguisher, route target and B-component bridge domain configuration.

```
           +--rw b-comp-instance*
              +--rw instance-name    string
              +--rw route-distinguisher   string
              +--rw auto-route-target?   boolean
              +--rw route-target
              +--rw protocol
              +--rw control-word
              +--rw cbp-interface
                 +--rw b-bridge-domain*
                    +.......
```

4.1. Backbone Bridge domain Configuration

The Backbone bridge domain contains the ISIDs and whether those ISIDs are to be extended to PBB EVPN core. The bridge domains which are not extended to PBB EVPN core can be used for local switching purpose.

```
     +--rw b-bridge-domain*
     |  +--rw nvlanid?   uint16
     |  +--rw isid*
     |     +--rw isid?      uint32
     |     +--rw extended?  boolean
```

5. I-Component Configuration

The I-component configuration contains customer bridge domain configuration and B-component instance name to map the I-component to B-component.

```
           +--rw i-comp-instance*
              +--rw pip-interface    [name]
              +--rw mapping-b-comp-instance-name [name]
                 +--rw cbd*
                    +.......
```

5.1. Customer Bridge domain Configuration

The customer bridge domain contains the mapping of interface to ISID.

```
       +--rw cbd*
       |  +--rw isid       uint32
       |  +--rw interface-name?  [name]
```

5.2. BMAC Configuration

For single home case, the multiple ISIDs in the customer bridge domains can share the same source BMAC. For the multi-homing cases, the source BMAC is associated to interface. The source BMAC can also be auto-derived based on LACP info.

```
     +--rw service-groups*
     |  +--rw service-group-name [uint32]
     |  +--rw isid*
     |  +--rw source-bmac
```

5.3. PBB EVPN Interface Configuration

PBB EVPN interface configuration includes the name of the interface, Ethernet Segment Identifier(ESI) and mode of interface, which tells single-active or active-active and source BMAC.

```
      +--rw interfaces
      |  +--rw interface*         [name]
      |     +--rw if_name         string
      |     +--rw esi_value       string
      |     +--rw redundancy-mode string
      |     +--rw source-bmac
```

6. YANG Module

<CODE BEGINS> file "ietf-pbb-evpn@2015-03-6.yang"

```
module pbbevpn {
  namespace "urn:juniper:params:xml:ns:yang:pbbevpn";
  // replace with IANA namespace when assigned

  prefix pevpn;

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

```
     import ietf-inet-types {
       prefix inet;
       //rfc6991
     }
   */

     import ietf-yang-types {
       prefix yang;
     }

     description
       "This YANG module defines the generic configuration data for
        PBB EVPN Service.

        Terms and Acronyms
        EVN: Ethernet Virtual Network
        EVPN: Ethernet VPN
        I-SID: Service Instance Identifier
        B-VID: Backbone VLAN ID
        C-MAC: Customer/Client MAC
        B-MAC: Backbone MAC
        BEB: Backbone Edge Bridge
        ES: Ethernet Segment
        ESI: Ethernet Segment Identifier
        LSP: Label Switched Path
        MP2MP: Multipoint to Multipoint
        MP2P: Multipoint to Point
        P2MP: Point to Multipoint
        P2P: Point to Point
        PE: Provider Edge
        EVPN: Ethernet VPN
        EVI: EVPN Instance
       ";

     revision 2015-03-06 {
       description
         "Initial revision.";
     }

     /*
      * Configuring Ethernet Segment
      */
     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})$)";
          }
        }

        leaf redudancy-mode {
          description
            "Specify Redundancy mode, value are all-active (false),
             single-active (true)";
          config "true";
          type boolean;
        }

        leaf auto-source-bmac {
          description
            "Specify auto derived mode (true) ,
             manual bmac config (false)";
          config "true";
          type boolean;
        }

        leaf source-bmac {
           type yang:mac-address;
        }
      } /* End of Interface */
    } /* End of Container */

    /*
     * Configuring Service Classification
     */
    container logicalinterfaces {
      list interface {
        key "name";
        leaf name {
          type leafref {
            path "/if:interfaces/if:interface/if:name";
          }
        }
        leaf encap {
```

```
          description
            "Vlan ID";
          config "true";
           type uint16 {
             range "1..4094";
           }
        } /* End encap */
      }
    }

    /*
     * Configuring I-component
     */
    container i-component {
      list bridge-domain {
        description
          "Customer Bridge Domain.";
        config "true";
        type uint16;

        container member-interface {
          description
            "member interface.";
          config true;
          list memberifs {
            description
             "member interfaces.";
             config true;
             type if:interface-ref;
          }
        } /* End of member if*/
        leaf nvlanid {
          description
            "Normalized Vlan ID";
          config "true";
           type uint16 {
             range "1..4094";
           }
        }
        leaf isid {
          description
            "I-SID";
          config "true";
           type uint32 {
             range "1..16777215";
           }
        }
      } /*End of List */
```

```
       leaf peer-b-component {
         description
           "Peer Backbone Component.";
         config true;
         type string {
           length "24";
           pattern "(^00([0-9a-fA-F]){2}\.(([0-9a-fA-F]){4}\.){3}
           (([0-9a-fA-F]){4})$)";
         }
       } /*end of peer-b-component */
     }

     /* Configuring Bcomponent */
     container b-component {
       list b-bridge-domain {
         description
           "Backbone Bridge Domain.";
         config "true";
         type uint16;
         leaf nvlanid {
           description
             "Normalized Vlan ID";
           config "true";
            type uint16 {
              range "1..4094";
            }
         }
         list isid {
           description
             "I-SID";
           config "true";
           leaf isid {
             description
               "I-SID";
             config "true";
              type uint32 {
                range "1..16777215";
              }
           }
         }
       } /*End of List */

       leaf route-distinguisher {
         description
           "Route Distinguisher.";
         config true;
         type string;
       } /*end of route-distinguiher */
```

```
      leaf auto-route-target {
          description
            "Specify auto derived route target (true) ,
             manual route target (false)";
          config "true";
          type boolean;
      }

      leaf route-target {
        description
          "Route Target.";
          config true;
          type string;
      } /* end of route target. */

      leaf protocol {
        description
          "Protocol running on Backbone B-Comp.";
        config true;
        type enumeration {
          enum "evpn" {
            value 0;
          }
          enum "pbb-evpn" {
            value 1;
          }
        }
      }

      leaf control-word {
        description
          "Control Word.";
        config true;
        type boolean;
      }
    }
  }
```

<CODE ENDS>

7. Security Considerations

Configuration and state data defined in this document are designed to be accessed via the NETCONF protocol [RFC6241].

Authors recommends to implement NETCONF access control model ([RFC6536]) to restrict access to all or part of the configuration to

specific users. Access control to RPCs is also critical as RPC permits to clear protocol datastructures that would definitively impact the network service. This kind of RPC needs only to be used in specific cases by well-known experienced users.

Authors consider that all the configuration is considered as sensitive/vulnerable as well as RPCs. But security teams can decide to open some part of the configuration to less experienced users depending on the internal organization, for example:

- User FullWrite: would access to the whole data model. This kind of profile may be restricted to few experienced people.

- User PartialWrite: would only access to configuration part within /interfaces/interface. So this kind of profile is restricted to creation/modification/deletion of interfaces. This profile does not have access to RPC.

- User Read: would only access to state part.

Unauthorized access to configuration or RPC may cause high damages to the network service.

When configuring ISIS using the NETCONF protocol, authors recommends the usage of secure transport of NETCONF using SSH ([RFC6242]).

8. Contributors

Authors would like to thank Wen Lin for their major contributions to the draft.

9. Acknowledgements

TBD.

10. IANA Considerations

TBD.

11. References

11.1. Informative 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.ietf-l2vpn-pbb-evpn]
Sajassi, A., Salam, S., Bitar, N., Isaac, A., Henderickx, W., and L. Jin, "PBB-EVPN", draft-ietf-l2vpn-pbb-evpn-09 (work in progress), October 2014.

11.2. Normative References

[I-D.ietf-netmod-routing-cfg]
Lhotka, L., "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-15 (work in progress), May 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.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

Appendix A. Example: NETCONF <get> Reply

This section gives an example of a reply to the NETCONF <get> request for a device that implements the data model defined in this document. The example is written in XML.

Authors' Addresses

Kishore Tiruveedhula (editor)
Juniper Networks
10 Technology Park Drive
Westford MA 01886
USA

Email: kishoret@juniper.net

Tapraj Singh
Juniper Networks
1194 N Mathilda Ave
Sunnyvale CA 94089
USA

Email: tsingh@juniper.net

Ali Sajassi
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sajassi@cisco.com

Deepak Kumar
Cisco Systems
510 McCarthy Blvd
Milpitas CA 95035
USA

Email: dekumar@cisco.com

Luay Jalil
Verizon
400 International Parkway
Richardson, TX 75081
USA

Email: luay.jalil@verizon.com