

BESS Working Group

Internet Draft

Intended Status: Proposed Standard

Expires: September 14, 2017

P. Brissette

A. Sajassi

Cisco System

H. Shah

Ciena Corporation

Z. Li

Huawei Technologies

I. Chen

Jabil

K. Tiruveedhula

Juniper Networks

I. Hussain

Infinera Corporation

J. Rabadan

Nokia

March 13, 2017

**Yang Data Model for EVPN
draft-ietf-bess-evpn-yang-02**

Abstract

This document describes a YANG data model for Ethernet VPN services. The model is agnostic of the underlay. It applies to MPLS as well as to VxLAN encapsulation. The model is also agnostic of the services including E-LAN, E-LINE and E-TREE services. Any "add-on" features such as EVPN IRB, EVPN overlay, etc. are for future investigation. This document mainly focuses on EVPN and Ethernet-Segment instance framework.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/1id-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2017 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.

Convention

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](#)].

Table of Contents

1.	Introduction	4
2.	Specification of Requirements	5
3.	EVPN YANG Model	5
3.1.	Overview	5
3.2.	Ethernet-Segment Model	6
3.3.	EVPN Model	6
4.	YANG Module	7
4.1.	Ethernet Segment Yang Module	7
4.2.	EVPN Yang Module	9
5.	Security Considerations	11
6.	IANA Considerations	11
7.	Acknowledgments	11
8.	References	12
8.1.	Normative References	12
8.2.	Informative References	12
	Authors' Addresses	12

1. Introduction

The Network Configuration Protocol (NETCONF) [[RFC6241](#)] is a network management protocol that defines mechanisms to manage network devices. YANG [[RFC6020](#)] is a modular language that represents data structures in an XML or JSON tree format, and is used as a data modeling language for the NETCONF.

This document introduces a YANG data model for Ethernet VPN services (EVPN) [[RFC7432](#)], Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN) [[RFC7623](#)] as well as other WG draft such as EVPN-VPWS, etc. The EVPN services runs over MPLS and VxLAN underlay.

The Yang data model in this document defines Ethernet VPN based services. The model will leverage the definitions used in other IETF Yang draft such as L2VPN Yang.

The goal is to propose a data object model consisting of building blocks that can be assembled in different order to realize different EVPN-based services. The definition work is undertaken initially by a smaller working group with members representing various vendors and service providers. The EVPN basic framework consist of two modules: EVPN and Ethernet-Segment. These models are completely orthogonal. They usually work in pair but user can definitely use one or the other for its own need.

The data model is defined for following constructs that are used for managing the services:

- o Configuration
- o Operational State
- o Executables (Actions)
- o Notifications

The document is organized to first define the data model for the configuration, operational state, actions and notifications of EVPN and Ethernet-Segment.

The EVPN data object model defined in this document uses the instance centric approach whereby EVPN service attributes are specified for a given EVPN instance.

The Ethernet-Segment data object model defined in this document refer to a specific interface. That interface can be a physical interface, a bundle interface or virtual interface. The latter includes

pseudowires. The purpose of creating a separate module is due to the fact that it can be used without having the need to have EVPN configured as layer 2 service. For example, an access node can be dual-homed to two service nodes servicing a VPLS core. The access connectivity can be represented by an Ethernet-Segment where EVPN BGP DF election is performed over both service nodes. The core remains VPLS where no EVPN instance is required.

2. Specification of Requirements

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](#)].

3. EVPN YANG Model

3.1. Overview

Two top level module, Ethernet-Segment and EVPN, are defined. The Ethernet-Segment contains a list of interface to which any Ethernet-Segment attributes are configured/applied.

The EVPN module has 2 main containers: common and instance. The first one has common attributes to all VPNs where as the latter has attributes specific to an EVI. This document state the scope of the EVPN object models definition. The following documents are within the scope. This is not an exhaustive list but a representation of documents that are covered for this work:

- o Requirements for EVPN: [RFC 7209](#)
- o EVPN: [RFC 7432](#)
- o PBB-EVPN: [RFC 7623](#)

The integration with L2VPN instance Yang model is being done as part of the L2VPN Yang model.

Following documents will be covered at that time:

- o VPWS support in EVPN:
[draft-ietf-bess-evpn-vpws](#)
- o E-TREE Support in EVPN & PBB-EVPN:
[draft-ietf-bess-evpn-etree](#)
- o (PBB-)EVPN Seamless Integration with (PBB-)VPLS:
[draft-ietf-bess-evpn-vpls-seamless-integ](#)
- o EVPN Virtual Ethernet Segment:
[draft-sajassi-bess-evpn-virtual-eth-segment](#)

The VxLAN aspect and the work related to Layer 3 is also for future definition. Following documents will be covered at that time:

- o IP Prefix Advertisement in EVPN:
[draft-ietf-bess-evpn-prefix-advertisement](#)
- o VXLAN DCI Using EVPN:
[draft-boutros-l2vpn-vxlan-evpn](#)
- o A Network Virtualization Overlay Solution using EVPN:
[draft-ietf-bess-evpn-overlay-](#)
- o Interconnect Solution for EVPN Overlay networks:
[draft-ietf-bess-dci-evpn-overlay](#)
- o Integrated Routing and Bridging in EVPN:
[draft-ietf-bess-evpn-inter-subnet-forwarding](#)

[3.2 Ethernet-Segment Model](#)

The Ethernet-Segment data model has a list of ES where each refer to an interface. All attributes are optional due to auto-sensing default mode where all values are auto-derive from the network connectivity.

```

module: ietf-ethernet-segment
  +--rw ethernet-segments
  |   +--rw ethernet-segment* [name]
  |       +--rw name string
  |       +--rw (ac-or-pw)?
  |           +--:(ac)
  |           |   +--rw ac? string
  |           +--:(pw)
  |           |   +--rw pw? string
  |       +--rw ethernet-segment-identifier? uint32
  |       +--rw (active-mode)
  |           +--:(single-active)
  |           |   +--rw single-active-mode? empty
  |           +--:(all-active)
  |           |   +--rw all-active-mode? empty
  |       +--rw pbb-parameters {ethernet-segment-pbb-params}?
  |           +--rw backbone-src-mac? yang:mac-address
  |       +--rw bgp-parameters
  |           +--rw common
  |               +--rw rd-rt* [route-distinguisher]
  |                   {ethernet-segment-bgp-params}?
  |                   +--rw route-distinguisher
  |                       rt-types:route-distinguisher
  |                   +--rw vpn-target* [route-target]
  |                       +--rw route-target
  |                           rt-types:route-target
  |                       +--rw route-target-type
  |                           rt-types:route-target-type
  |       +--rw df-election
  |           +--rw (df-election-method)?
  |           |   +--:(highest-random-weight)

```



```

|      | |      +--rw hrw?                boolean
|      | +--rw election-wait-time?      uint32
|      +--rw ead-evi-route?            boolean
+--ro ethernet-segments-state
  +--ro ethernet-segment-state* [name]
    +--ro name                        string
    +--ro service-type?              string
    +--ro status?                    status-type
    +--ro (ac-or-pw)?
      | +--:(ac)
      | | +--ro ac?                  string
      | +--:(pw)
      | +--ro pw?                    string
    +--ro interface-status?          status-type
    +--ro ethernet-segment-identifier? uint32
    +--ro active-mode?               string
    +--ro pbb-parameters {ethernet-segment-pbb-params}?
      | +--ro backbone-src-mac?      yang:mac-address
    +--ro bgp-parameters
      | +--ro common
      |   +--ro rd-rt* [route-distinguisher]
      |   {ethernet-segment-bgp-params}?
      |   +--ro route-distinguisher
      |   rt-types:route-distinguisher
      |   +--ro vpn-target* [route-target]
      |   +--ro route-target
      |   rt-types:route-target
      |   +--ro route-target-type
      |   rt-types:route-target-type
    +--ro df-election
      | +--ro hrw-enabled?            boolean
      | +--ro election-wait-time?    uint32
    +--ro ead-evi-route-enabled?      boolean
    +--ro esi-label?                  string
    +--ro member*
      | +--ro ip-address?            inet:ip-address
    +--ro df*
      +--ro service-identifier?      uint32
      +--ro vlan?                    uint32
      +--ro ip-address?              inet:ip-address

```

3.3 EVPN Model

The evpn-instances container contains a list of evpn-instance. Each entry of the evpn-instance represents a different Ethernet VPN and it is represented by a EVI. Again, mainly all attributes are optional for the same reason as for the Ethernet-Segment module.


```
module: ietf-evpn
  +--rw evpn
  |   +--rw common
  |   |   +--rw (replication-type)?
  |   |   |   +--:(ingress-replication)
  |   |   |   |   +--rw ingress-replication?    boolean
  |   |   |   +--:(p2mp-replication)
  |   |   |   |   +--rw p2mp-replication?        boolean
  |   +--rw evpn-instances
  |   |   +--rw evpn-instance* [name]
  |   |   |   +--rw name                                string
  |   |   |   +--rw evi?                                uint32
  |   |   |   +--rw pbb-parameters {evpn-pbb-params}?
  |   |   |   |   +--rw source-bmac?    yang:hex-string
  |   |   |   +--rw bgp-parameters
  |   |   |   |   +--rw common
  |   |   |   |   |   +--rw rd-rt* [route-distinguisher]
  |   |   |   |   |   |   {evpn-bgp-params}?
  |   |   |   |   |   |   +--rw route-distinguisher
  |   |   |   |   |   |   |   rt-types:route-distinguisher
  |   |   |   |   |   |   +--rw vpn-target* [route-target]
  |   |   |   |   |   |   |   +--rw route-target
  |   |   |   |   |   |   |   |   rt-types:route-target
  |   |   |   |   |   |   |   +--rw route-target-type
  |   |   |   |   |   |   |   |   rt-types:route-target-type
  |   |   |   |   +--rw arp-proxy?                boolean
  |   |   |   |   +--rw arp-suppression?            boolean
  |   |   |   |   +--rw nd-proxy?                  boolean
  |   |   |   |   +--rw nd-suppression?            boolean
  |   |   |   |   +--rw underlay-multicast?        boolean
  |   |   |   |   +--rw flood-unknown-unicast-supression? boolean
  |   +--rw evpn-state
  |   |   +--ro evpn-instances-state
  |   |   |   +--ro evpn-instance*
  |   |   |   |   +--ro name? string
  |   |   |   |   +--ro evi? uint32
  |   |   |   |   +--ro pbb-parameters {evpn-pbb-params}?
  |   |   |   |   |   +--ro source-bmac?    yang:hex-string
  |   |   |   |   +--ro bgp-parameters
  |   |   |   |   |   +--ro common
  |   |   |   |   |   |   +--ro rd-rt* [route-distinguisher]
  |   |   |   |   |   |   |   {evpn-bgp-params}?
  |   |   |   |   |   |   |   +--ro route-distinguisher
  |   |   |   |   |   |   |   |   rt-types:route-distinguisher
  |   |   |   |   |   |   |   +--ro vpn-target* [route-target]
  |   |   |   |   |   |   |   |   +--ro route-target rt-types:route-target
  |   |   |   |   |   |   |   |   +--ro route-target-type
  |   |   |   |   |   |   |   |   |   rt-types:route-target-type
```



```

+--ro advertise-mac-suppression-enabled?      boolean
+--ro arp-proxy-enabled?                      boolean
+--ro arp-suppression-enabled?                boolean
+--ro nd-proxy-enabled?                      boolean
+--ro nd-suppression-enabled?                boolean
+--ro underlay-multicast-enabled?            boolean
+--ro flood-unknown-unicast-suppression-enabled? boolean
+--ro routes
| +--ro ethernet-auto-discovery-route*
| | +--ro rd-rt* [route-distinguisher]
| | | +--ro route-distinguisher
| | | | rt-types:route-distinguisher
| | | +--ro vpn-target* [route-target]
| | | | +--ro route-target    rt-types:route-target
| | +--ro ethernet-segment-identifier?  uint32
| | +--ro ethernet-tag?                  uint32
| | +--ro path*
| | | +--ro next-hop?   inet:ip-address
| | | +--ro label?      rt-types:mpls-label
| | | +--ro detail
| | | | +--ro attributes
| | | | | +--ro extended-community*  string
| | | | +--ro bestpath?              empty
| +--ro mac-ip-advertisement-route*
| | +--ro rd-rt* [route-distinguisher]
| | | +--ro route-distinguisher
| | | | rt-types:route-distinguisher
| | | +--ro vpn-target* [route-target]
| | | | +--ro route-target    rt-types:route-target
| | +--ro ethernet-segment-identifier?  uint32
| | +--ro ethernet-tag?                  uint32
| | +--ro mac-address? yang:hex-string
| | +--ro mac-address-length?            uint8
| | +--ro ip-prefix?   inet:ip-prefix
| | +--ro path*
| | | +--ro next-hop?   inet:ip-address
| | | +--ro label?      rt-types:mpls-label
| | | +--ro label2?     rt-types:mpls-label
| | | +--ro detail
| | | | +--ro attributes
| | | | | +--ro extended-community*  string
| | | | +--ro bestpath?              empty
| +--ro inclusive-multicast-ethernet-tag-route*
| | +--ro rd-rt* [route-distinguisher]
| | | +--ro route-distinguisher
| | | | rt-types:route-distinguisher
| | | +--ro vpn-target* [route-target]
| | | | +--ro route-target    rt-types:route-target

```



```

| | +--ro ethernet-segment-identifier?  uint32
| | +--ro originator-ip-prefix? inet:ip-prefix
| | +--ro path*
| |   +--ro next-hop?  inet:ip-address
| |   +--ro label?     rt-types:mpls-label
| |   +--ro detail
| |     +--ro attributes
| |       | +--ro extended-community*  string
| |       +--ro bestpath?             empty
| +--ro ethernet-segment-route*
| | +--ro rd-rt* [route-distinguisher]
| | | +--ro route-distinguisher
| | |   rt-types:route-distinguisher
| | | +--ro vpn-target* [route-target]
| | |   +--ro route-target  rt-types:route-target
| | +--ro ethernet-segment-identifier?  uint32
| | +--ro originator-ip-prefix?  inet:ip-prefix
| | +--ro path*
| |   +--ro next-hop?  inet:ip-address
| |   +--ro detail
| |     +--ro attributes
| |       | +--ro extended-community*  string
| |       +--ro bestpath?             empty
| +--ro ip-prefix-route*
| | +--ro rd-rt* [route-distinguisher]
| | | +--ro route-distinguisher
| | |   rt-types:route-distinguisher
| | | +--ro vpn-target* [route-target]
| | |   +--ro route-target  rt-types:route-target
| +--ro ethernet-segment-identifier?  uint32
| +--ro ip-prefix?  inet:ip-prefix
| +--ro path*
| | +--ro next-hop?  inet:ip-address
| | +--ro label?     rt-types:mpls-label
| | +--ro detail
| |   +--ro attributes
| |     | +--ro extended-community*  string
| |     +--ro bestpath?             empty
+--ro statistics
  +--ro tx-count?  uint32
  +--ro rx-count?  uint32
  +--ro detail
    +--ro broadcast-tx-count?  uint32
    +--ro broadcast-rx-count?  uint32
    +--ro multicast-tx-count?  uint32
    +--ro multicast-rx-count?  uint32
    +--ro unicast-tx-count?    uint32
    +--ro unicast-rx-count?    uint32

```



```
augment /l2vpn:l2vpn/l2vpn:l2vpn-instances/l2vpn:l2vpn-instance:
  +-rw evpn-instance?    evpn-instance-ref
augment /l2vpn:l2vpn-state/
  l2vpn:l2vpn-instances-state/l2vpn:l2vpn-instance:
  +-ro evpn-instance?    string
```

4. YANG Module

The EVPN configuration container is logically divided into following high level config areas:

4.1 Ethernet Segment Yang Module

```
<CODE BEGINS> file "ietf-ethernet-segment@2017-03-13.yang"
module ietf-ethernet-segment {
  namespace "urn:ietf:params:xml:ns:yang:ietf-ethernet-segment";
  prefix "es";

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

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

  import ietf-routing-types {
    prefix "rt-types";
  }

  organization "ietf";
  contact "ietf";
  description "ethernet segment";

  revision "2017-03-13" {
    description " - Updated to use BGP parameters from " +
      " ietf-routing-types.yang instead of from " +
      " ietf-evpn.yang " +
      """;
    reference """;
  }

  revision "2016-07-08" {
    description " - Added the configuration option to enable or " +
      " disable per-EVI/EAD route " +
      " - Added PBB parameter backbone-src-mac " +
      " - Added operational state branch, initially " +
      " to match the configuration branch" +
      """;
  }
}
```



```
        "";  
    reference    "";  
}  
  
revision "2016-06-23" {  
    description "WG document adoption";  
    reference    "";  
}  
  
revision "2015-10-15" {  
    description "Initial revision";  
    reference    "";  
}  
  
/* Features */  
  
feature ethernet-segment-bgp-params {  
    description "Ethernet segment's BGP parameters";  
}  
  
feature ethernet-segment-pbb-params {  
    description "Ethernet segment's PBB parameters";  
}  
  
/* Typedefs */  
  
typedef status-type {  
    type enumeration {  
        enum up {  
            description "Status is up";  
        }  
        enum down {  
            description "Status is down";  
        }  
    }  
    description "status type";  
}  
  
/* EVPN Ethernet Segment YANG Model */  
  
container ethernet-segments {  
  
    description "ethernet-segment";  
    list ethernet-segment {  
        key "name";  
        leaf name {  
            type string;  
            description "Name of the ethernet segment";  
        }  
    }  
}
```



```
}
choice ac-or-pw {
  description "ac-or-pw";
  case ac {
    leaf ac {
      type string;
      description "Eventual reference to standard " +
        "attachment circuit definition";
    }
  }
  case pw {
    leaf pw {
      type string;
      description "Eventual reference to standard " +
        "pseudowire definition";
    }
  }
}
leaf ethernet-segment-identifier {
  type uint32;
  description "Ethernet segment identifier (esi)";
}
choice active-mode {
  mandatory true;
  description "Choice of active mode";
  case single-active {
    leaf single-active-mode {
      type empty;
      description "single-active-mode";
    }
  }
  case all-active {
    leaf all-active-mode {
      type empty;
      description "all-active-mode";
    }
  }
}
container pbb-parameters {
  if-feature ethernet-segment-pbb-params;
  description "PBB configuration";
  leaf backbone-src-mac {
    type yang:mac-address;
    description "backbone-src-mac, only if this is a PBB";
  }
}
container bgp-parameters {
  description "BGP parameters";
```



```
    container common {
      description "BGP parameters common to all pseudowires";
      list rd-rt {
        if-feature ethernet-segment-bgp-params;
        key "route-distinguisher";
        leaf route-distinguisher {
          type rt-types:route-distinguisher;
          description "Route distinguisher";
        }
        uses rt-types:vpn-route-targets;
        description "A list of route distinguishers and " +
          "corresponding VPN route targets";
      }
    }
  }
  container df-election {
    description "df-election";
    choice df-election-method {
      description "Choice of df election method";
      case highest-random-weight {
        leaf hrw {
          type boolean;
          description "Enable (TRUE) or disable (FALSE) " +
            "highest random weight";
        }
      }
    }
    leaf election-wait-time {
      type uint32;
      description "election-wait-time";
    }
  }
  leaf ead-evi-route {
    type boolean;
    default false;
    description "Enable (true) or disable (false) ead-evi-route";
  }
  description "An ethernet segment";
}

container ethernet-segments-state {
  config false;
  description "Ethernet segmet operational state";
  list ethernet-segment-state {
    key "name";
    leaf name {
      type string;
```



```
    description "Name of the ethernet segment";
  }
  leaf service-type {
    type string;
    description "service-type";
  }
  leaf status {
    type status-type;
    description "Ethernet segment status";
  }
  choice ac-or-pw {
    description "ac-or-pw";
    case ac {
      leaf ac {
        type string;
        description "Name of attachment circuit";
      }
    }
    case pw {
      leaf pw {
        type string;
        description "Name of pseudowire";
      }
    }
  }
  leaf interface-status {
    type status-type;
    description "interface status";
  }
  leaf ethernet-segment-identifier {
    type uint32;
    description "Ethernet segment identifier (esi)";
  }
  leaf active-mode {
    type string;
    description "Single-active-mode/all-active-mode";
  }
  container pbb-parameters {
    if-feature "ethernet-segment-pbb-params";
    description "PBB configuration";
    leaf backbone-src-mac {
      type yang:mac-address;
      description "backbone-src-mac, only if this is a PBB";
    }
  }
  container bgp-parameters {
    description "BGP parameters";
    container common {
```



```
    description "BGP parameters common to all pseudowires";
    list rd-rt {
      if-feature ethernet-segment-bgp-params;
      key "route-distinguisher";
      leaf route-distinguisher {
        type rt-types:route-distinguisher;
        description "Route distinguisher";
      }
      uses rt-types:vpn-route-targets;
      description "A list of route distinguishers and " +
        "corresponding route targets";
    }
  }
}
container df-election {
  description "df-election";
  leaf hrw-enabled {
    type boolean;
    description "hrw-enabled is enabled (TRUE) " +
      "or disabled (FALSE)";
  }
  leaf election-wait-time {
    type uint32;
    description "election-wait-time";
  }
}
leaf ead-evi-route-enabled {
  type boolean;
  description "ead-evi-route is enabled (TRUE) " +
    "or disabled (FALSE)";
}
leaf esi-label {
  type string;
  description "esi-label";
}
list member {
  leaf ip-address {
    type inet:ip-address;
    description "ip-address";
  }
  description "member of the ethernet segment";
}
list df {
  leaf service-identifier {
    type uint32;
    description "service-identifier";
  }
  leaf vlan {
```



```
        type uint32;
        description "vlan";
    }
    leaf ip-address {
        type inet:ip-address;
        description "ip-address";
    }
    description "df of an evpn instance's vlan";
}
description "An ethernet segment";
}
}
}

<CODE ENDS>
```

[4.2](#) EVPN Yang Module

```
<CODE BEGINS> file "ietf-evpn@2017-03-13.yang"
module ietf-evpn {
    namespace "urn:ietf:params:xml:ns:yang:ietf-evpn";
    prefix "evpn";

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

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

    import ietf-l2vpn {
        prefix "l2vpn";
    }

    import ietf-routing-types {
        prefix "rt-types";
    }

    organization "ietf";
    contact "ietf";
    description "evpn";

    revision "2017-03-13" {
        description " - Added an augment to base L2VPN model to " +
            " reference an EVPN instance " +
            " - Reused ietf-routing-types.yang " +
```



```
        "    vpn-route-targets grouping instead of " +
        "    defining it in this module " +
        """;
    reference    "";
}

revision "2016-07-08" {
    description " - Added operational state" +
        " - Added a configuration knob to enable/disable " +
        "    underlay-multicast " +
        " - Added a configuration knob to enable/disable " +
        "    flooding of unknoww unicast " +
        " - Added several configuration knobs " +
        "    to manage ARP and ND" +
        """;
    reference    "";
}

revision "2016-06-23" {
    description "WG document adoption";
    reference    "";
}

revision "2015-10-15" {
    description "Initial revision";
    reference    "";
}

feature evpn-bgp-params {
    description "EVPN's BGP parameters";
}

feature evpn-pbb-params {
    description "EVPN's PBB parameters";
}

/* Typedefs */

typedef evpn-instance-ref {
    type leafref {
        path "/evpn/evpn-instances/evpn-instance/name";
    }
    description "A leafref type to an EVPN instance";
}

/* Groupings */

grouping route-rd-rt-grp {
```



```
description "A grouping for a route's route distinguishers " +
    "and route targets";
list rd-rt {
  key "route-distinguisher";
  leaf route-distinguisher {
    type rt-types:route-distinguisher;
    description "Route distinguisher";
  }
  list vpn-target {
    key "route-target";
    leaf route-target {
      type rt-types:route-target;
      description "BGP route target";
    }
    description "A list of route targets";
  }
  description "A list of route distinguishers and " +
    "corresponding VPN route targets";
}
}

grouping next-hop-label-grp {
  description "next-hop-label-grp";
  leaf next-hop {
    type inet:ip-address;
    description "next-hop";
  }
  leaf label {
    type rt-types:mpls-label;
    description "label";
  }
}

grouping next-hop-label2-grp {
  description "next-hop-label2-grp";
  leaf label2 {
    type rt-types:mpls-label;
    description "label2";
  }
}

grouping path-detail-grp {
  description "path-detail-grp";
  container detail {
    config false;
    description "path details";
    container attributes {
      leaf-list extended-community {
```



```
        type string;
        description "extended-community";
    }
    description "attributes";
}
leaf bestpath {
    type empty;
    description "Indicate this path is the best path";
}
}
}

/* EVPN YANG Model */

container evpn {
    description "evpn";
    container common {
        description "common evpn attributes";
        choice replication-type {
            description "A choice of replication type";
            case ingress-replication {
                leaf ingress-replication {
                    type boolean;
                    description "ingress-replication";
                }
            }
            case p2mp-replication {
                leaf p2mp-replication {
                    type boolean;
                    description "p2mp-replication";
                }
            }
        }
    }
}
container evpn-instances {
    description "evpn-instances";
    list evpn-instance {
        key "name";
        description "An EVPN instance";
        leaf name {
            type string;
            description "Name of EVPN instance";
        }
        leaf evi {
            type uint32;
            description "evi";
        }
        container pbb-parameters {
```



```
    if-feature "evpn-pbb-params";
    description "PBB parameters";
    leaf source-bmac {
        type yang:hex-string;
        description "source-bmac";
    }
}
container bgp-parameters {
    description "BGP parameters";
    container common {
        description "BGP parameters common to all pseudowires";
        list rd-rt {
            if-feature evpn-bgp-params;
            key "route-distinguisher";
            leaf route-distinguisher {
                type rt-types:route-distinguisher;
                description "Route distinguisher";
            }
            uses rt-types:vpn-route-targets;
            description "A list of route distinguishers and " +
                "corresponding VPN route targets";
        }
    }
}
leaf arp-proxy {
    type boolean;
    default false;
    description "Enable (TRUE) or disable (FALSE) ARP proxy";
}
leaf arp-suppression {
    type boolean;
    default false;
    description "Enable (TRUE) or disable (FALSE) " +
        "ARP suppression";
}
leaf nd-proxy {
    type boolean;
    default false;
    description "Enable (TRUE) or disable (FALSE) ND proxy";
}
leaf nd-suppression {
    type boolean;
    default false;
    description "Enable (TRUE) or disable (FALSE) " +
        "ND suppression";
}
leaf underlay-multicast {
    type boolean;
```



```
        default false;
        description "Enable (TRUE) or disable (FALSE) " +
            "underlay multicast";
    }
    leaf flood-unknown-unicast-supression {
        type boolean;
        default false;
        description "Enable (TRUE) or disable (FALSE) " +
            "flood unknown unicast suppression";
    }
}
}
}

container evpn-state {
    description "EVPN operational state";
    container evpn-instances-state {
        config false;
        description "evpn-instances-state";
        list evpn-instance {
            description "The state of an EVPN instance";
            leaf name {
                type string;
                description "Name of EVPN instance";
            }
            leaf evi {
                type uint32;
                description "evi";
            }
            container pbb-parameters {
                if-feature "evpn-pbb-params";
                description "PBB parameters";
                leaf source-bmac {
                    type yang:hex-string;
                    description "source-bmac";
                }
            }
        }
        container bgp-parameters {
            description "BGP parameters";
            container common {
                description "BGP parameters common to all pseudowires";
                list rd-rt {
                    if-feature evpn-bgp-params;
                    key "route-distinguisher";
                    leaf route-distinguisher {
                        type rt-types:route-distinguisher;
                        description "Route distinguisher";
                    }
                }
            }
        }
    }
}
```



```
        uses rt-types:vpn-route-targets;
        description "A list of route distinguishers and " +
            "corresponding VPN route targets";
    }
}
}
leaf advertise-mac-suppression-enabled {
    type boolean;
    description "advertise-mac-suppression " +
        "is enabled (TRUE) " +
        "or disabled (FALSE)";
}
leaf arp-proxy-enabled {
    type boolean;
    description "arp-proxy is enabled (TRUE) " +
        "or disabled (FALSE)";
}
leaf arp-suppression-enabled {
    type boolean;
    description "arp-suppression is enabled (TRUE) " +
        "or disabled (FALSE)";
}
leaf nd-proxy-enabled {
    type boolean;
    description "nd-proxy is enabled (TRUE) " +
        "or disabled (FALSE)";
}
leaf nd-suppression-enabled {
    type boolean;
    description "nd-suppression is enabled (TRUE) " +
        "or disabled (FALSE)";
}
leaf underlay-multicast-enabled {
    type boolean;
    description "underlay-multicast is enabled (TRUE) " +
        "or disabled (FALSE)";
}
leaf flood-unknown-unicast-suppression-enabled {
    type boolean;
    description "flood-unknown-unicast-suppression is " +
        "enabled (TRUE) or disabled (FALSE)";
}
container routes {
    description "routes";
    list ethernet-auto-discovery-route {
        uses route-rd-rt-grp;
        leaf ethernet-segment-identifier {
            type uint32;
```



```
        description "Ethernet segment identifier (esi)";
    }
    leaf ethernet-tag {
        type uint32;
        description "An ethernet tag (etag) indentifying a " +
            "broadcast domain";
    }
    list path {
        uses next-hop-label-grp;
        uses path-detail-grp;
        description "path";
    }
    description "ethernet-auto-discovery-route";
}
list mac-ip-advertisement-route {
    uses route-rd-rt-grp;
    leaf ethernet-segment-identifier {
        type uint32;
        description "Ethernet segment identifier (esi)";
    }
    leaf ethernet-tag {
        type uint32;
        description "An ethernet tag (etag) indentifying a " +
            "broadcast domain";
    }
    leaf mac-address {
        type yang:hex-string;
        description "Route mac address";
    }
    leaf mac-address-length {
        type uint8 {
            range "0..48";
        }
        description "mac address length";
    }
    leaf ip-prefix {
        type inet:ip-prefix;
        description "ip-prefix";
    }
    list path {
        uses next-hop-label-grp;
        uses next-hop-label2-grp;
        uses path-detail-grp;
        description "path";
    }
    description "mac-ip-advertisement-route";
}
list inclusive-multicast-ethernet-tag-route {
```



```
    uses route-rd-rt-grp;
    leaf ethernet-segment-identifier {
        type uint32;
        description "Ethernet segment identifier (esi)";
    }
    leaf originator-ip-prefix {
        type inet:ip-prefix;
        description "originator-ip-prefix";
    }
    list path {
        uses next-hop-label-grp;
        uses path-detail-grp;
        description "path";
    }
    description "inclusive-multicast-ethernet-tag-route";
}
list ethernet-segment-route {
    uses route-rd-rt-grp;
    leaf ethernet-segment-identifier {
        type uint32;
        description "Ethernet segment identifier (esi)";
    }
    leaf originator-ip-prefix {
        type inet:ip-prefix;
        description "originator ip-prefix";
    }
    list path {
        leaf next-hop {
            type inet:ip-address;
            description "next-hop";
        }
        uses path-detail-grp;
        description "path";
    }
    description "ethernet-segment-route";
}
list ip-prefix-route {
    uses route-rd-rt-grp;
    leaf ethernet-segment-identifier {
        type uint32;
        description "Ethernet segment identifier (esi)";
    }
    leaf ip-prefix {
        type inet:ip-prefix;
        description "ip-prefix";
    }
    list path {
        uses next-hop-label-grp;
```



```
        uses path-detail-grp;
        description "path";
    }
    description "ip-prefix route";
}
}
container statistics {
    description "Statistics";
    leaf tx-count {
        type uint32;
        description "transmission count";
    }
    leaf rx-count {
        type uint32;
        description "receive count";
    }
}
container detail {
    description "Detailed statistics";
    leaf broadcast-tx-count {
        type uint32;
        description "broadcast transmission count";
    }
    leaf broadcast-rx-count {
        type uint32;
        description "broadcast receive count";
    }
    leaf multicast-tx-count {
        type uint32;
        description "multicast transmission count";
    }
    leaf multicast-rx-count {
        type uint32;
        description "multicast receive count";
    }
    leaf unicast-tx-count {
        type uint32;
        description "unicast transmission count";
    }
    leaf unicast-rx-count {
        type uint32;
        description "unicast receive count";
    }
}
}
}
}
}
```



```
/* augments */
augment "/l2vpn:l2vpn/l2vpn:l2vpn-instances" +
    "/l2vpn:l2vpn-instance" {
    description "Augment for an L2VPN instance and EVPN association";
    leaf evpn-instance {
        type evpn-instance-ref;
        description "Reference to an EVPN instance";
    }
}

augment "/l2vpn:l2vpn-state" {
    description "Augment for an L2VPN instance's operational " +
        "state of L2VPN and EVPN association";
    leaf evpn-instance {
        type string;
        description "Name of the associated EVPN instance";
    }
}
}

<CODE ENDS>
```

5. Security Considerations

The configuration, state, action and notification data defined in this document are designed to be accessed via the NETCONF protocol [[RFC6241](#)]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [[RFC6242](#)]. The NETCONF access control model [[RFC6536](#)] provides means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

The security concerns listed above are, however, no different than faced by other routing protocols. Hence, this draft does not change any underlying security issues inherent in [I-D.ietf-netmod-routing-cfg]

6. IANA Considerations

None.

7. Acknowledgments

The authors would like to acknowledge TBD for their useful comments.

8. References

[8.1.](#) Normative References

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

[8.2.](#) Informative References

- [[RFC6241](#)] R.Enns et al., "Network Configuration Protocol (NETCONF)", [RFC 6241](#), June 2011
- [RFC6020] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), October 2010.
- [RFC6242] M. Wasserman, "Using the NETCONF Protocol over Secure Shell (SSH)", [RFC 6242](#), June 2011.
- [RFC6536] A. Bierman et al., "Network Configuration Protocol (NETCONF) Access Control Model" [RFC 6536](#), March 2012.
- [RFC7432] Sajassi et al., "BGP MPLS-Based Ethernet VPN", [RFC 7432](#), February 2015.
- [RFC7623] Sajassi et al., "Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN)", [RFC 7623](#), September 2015

Authors' Addresses

Patrice Brissette
Cisco Systems, Inc.
EMail: pbrisset@cisco.com

Ali Sajassi
Cisco Systems, Inc.
EMail: sajassi@cisco.com

Himanshu Shah
Ciena Corporation
EMail: hshah@ciena.com

Zhenbin Li

Huawei Technologies
EMail: lizhenbin@huawei.com

Helen Chen
Jabil
EMail: Ing-Wher_Chen@jabil.com

Kishore Tiruveedhula
Juniper Networks
EMail: kishoret@juniper.net

Iftekar Hussain
Infinera Corporation
EMail: ihussain@infinera.com

Jorge Rabadan
Nokia
EMail: jorge.rabadan@nokia.com

