YANG Data Model for IPv4-in-IPv6 Softwire
draft-sun-softwire-yang-03

Abstract

This document defines a YANG data model for the configuration and management of IPv4-in-IPv6 Softwire Border Routers and Customer Premises Equipment. It covers Lightweight 4over6, MAP-E and MAP-T Softwire mechanisms.

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 October 9, 2015.
1. Introduction

The IETF Softwire Working Group has developed several IPv4-in-IPv6 Softwire mechanisms to address various deployment contexts and constraints. As a companion to the architectural specification documents, this document focuses on the provisioning aspects for
softwire functional elements that are: Border Routers (BRs) and Customer Premises Equipment (CEs).

This document defines a YANG data model that can be used to configure and manage IPv4-in-IPv6 Softwire BRs and/or CEs via NETCONF protocol [RFC6241]. To ensure interoperability in mixed vendor environments, it is important that the models can be easily reused between different vendors and implementations.

There are three different mechanisms in this YANG model. Each specific mechanism has their separate YANG modules respectively:

- Lightweight 4over6 [I-D.ietf-softwire-lw4over6]
- MAP-E [I-D.ietf-softwire-map]
- MAP-T [I-D.ietf-softwire-map-t]

This model is structured into two root containers:

1. Container "softwire-config" holds the collection of YANG definitions common to all softwire configuration of BRs and CEs.

2. Container "softwire-state" holds YANG definitions for the operational state of the Softwire BRs and CEs.

The model also includes a notification module. The aim is to notify the client that a specific status has been changed.

This approach has been taken so that the model can be easily extended in the future to support additional softwire mechanism, should this be necessary.

1.1. Terminology

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].

The reader should be familiar with the terms defined in [I-D.ietf-softwire-lw4over6] [I-D.ietf-softwire-map] [I-D.ietf-softwire-map-t], and the YANG data modelling language [RFC6020].
1.2. YANG Tree Diagrams

Softwire YANG tree diagrams provide a concise representation of a YANG module to help readers understand the module structure. The meaning if the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Braces "{" and "}" enclose feature content.
- Parentheses "(" and ")" enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Symbols after data node names: "?" means an optional node, and "*" denotes a list and leaf-list.
- Abbreviations before data node names: "rw" means configuration data (read-write), and "ro" means state data (read-only).

1.3. YANG Modelling of NAT44 Functionality

This documented model does not include NAT-specific provisioning parameters other than the external IP address and port set which a softwire client may use for NAT44. Additional NAT-specific considerations are out of scope.

2. Objectives

This document defines a YANG data model that can be used to configure and manage BRs and CEs for the following IPv4-in-IPv6 Softwire mechanisms: Lightweight 4over6, MAP-E and MAP-T.

For the Lightweight 4over6, the configuration and management information of lwB4 and lwAFTTR are different. The Lightweight 4over6 AFTTRs needs to maintain the binding table of lwB4s. The Lightweight 4over6 B4s need to maintain the NAPT table of hosts.

For the MAP-T and MAP-T, CE and BR both need to maintain the map-rule table. Thus, there is no need to distinguish BR and CE.

2.1. Common

This common model abstracts the shared features of different BRs and CEs. The model defined a single node softwire description for common feature now.

The following sections of the document are structured with the root of the softwire YANG model (common to all mechanisms) described
first. The subsequent sections describe the models relevant to the different softwire mechanisms. All functions are listed, but the YANG models use the "feature" statement to distinguish among the different softwire mechanisms.

2.2. Lightweight 4over6

Lightweight 4over6 include two elements: lwAFTR and lwB4. The lwAFTR holds configuration for IPv4-IPv6 address bindings which used for the forwarding of traffic originating from lwB4s. And the lwB4 is configured with the relevant parameters for establishing the IPv4 in IPv6 tunnel including an IPv6 address for the lwAFTR and the IPv4 configuration for NAPT44.

2.3. MAP-E

MAP-E elements (BR and CE) are provisioned with the MAP rules necessary for defining MAP domains and forwarding rules.

2.4. MAP-T

MAP-E elements (BR and CE) are provisioned with the MAP rules necessary for defining MAP domains and forwarding rules. MAP-T CEs an additional "ipv6-prefix" parameter is also configured.

3. Softwire YANG Tree Diagrams

3.1. Common Tree Diagrams

Figure 1 describes the softwire data model which is common to all of the different softwire mechanisms listed in Section 1:
The mechanism specific models for lw4over6, MAP-E and MAP-T are described in detail in the following sections.

3.2. Lightweight 4over6 Tree Diagrams

Figure 2 defines the softwire data model for Lightweight 4over6 which include lwAFR and lwB4:

```yang
top-level-module
  +--rw softwire-config
  |    +--rw description? string
  |    +--rw lw4over6 {lw4over6}?
  |       +--rw lwaftr {lwaftr}?
  |          +--rw lwb4 {lwb4}?
  |          +--rw map-e {map-e}?
  |          +--rw map-t {map-t}?
  +--ro softwire-state
      +--ro description? string
      +--ro lw4over6 {lw4over6}?
      |      +--ro lwaftr {lwaftr}?
      |      +--ro lwb4 {lwb4}?
      +--ro map-e {map-e}?
      +--ro map-t {map-t}?
```

Figure 1: Softwire Common Data Model Structure
Internet-Draft              YANG for softwire                 April 2015

---rw tunnel-mtu           uint32
---rw fragment-mru         uint16
---rw tunnel-num-threshold uint32

++-rw lwb4 {lwb4}?
   ---rw enable?           boolean
   ---rw lwb4-instances
      ++-rw lwb4-instance* [binding-ipv6info]
         ---rw name?         string
         ---rw b4-ipv6-addr-format boolean
         ---rw binding-ipv6info union
         ---rw binding-ipv4-addr inet:ipv4-address
         ---rw port-set
        |   ---rw offset         uint8
        |   ---rw psid           uint16
        |   ---rw psid-len       uint8
        |   ---rw lwaftr-ipv6-addr? inet:ipv6-prefix
      ++-rw tunnels
         ++-rw tunnel* [tunnel-id]
            |   ---rw tunnel-id     uint8
            |   ---rw tunnel-mtu    uint32
            |   ---rw fragment-mru uint16

---ro softwire-state

---ro lw4over6 {lw4over6}?
   ---ro lwaftr {lwaftr}?
      ---ro lwaftr-instances
         ++-ro lwaftr-instance* [id]
            ---ro id           uint32
            ---ro name?       string
            ---ro active-tunnel-num? uint32
            ---ro binding-table
               |   ++-ro binding-entry* [binding-ipv6-addr]
               |      ---ro binding-ipv6-addr inet:ipv6-address
               |      ---ro active?  boolean
            ---ro traffic-stat
               |   ---ro sentPacket?  yang:zero-based-counter64
               |   ---ro sentByte?    yang:zero-based-counter64
               |   ---ro rcvdPacket?  yang:zero-based-counter64
               |   ---ro rcvdByte?    yang:zero-based-counter64
               |   ---ro droppedPacket?  yang:zero-based-counter64
               |   ---ro droppedByte? yang:zero-based-counter64

---ro lwb4 {lwb4}?
   ---ro lwb4-instances
      ++-ro lwb4-instance* [binding-ipv6info]
         ---ro name?        string
         ---ro binding-ipv6info union
         ---ro traffic-stat
Introduction of important lwAFTR nodes:

- binding-entry: used to define the binding relationship between 3-tuples, which contains the lwB4’s IPv6 address, the allocated IPv4 address and restricted port-set. For detail information, please refer to [I-D.ietf-softwire-lw4over6].

- tunnel-mtu: used to set the value of MTU for Lightweight 4over6 tunnel.

- fragment-mru: used to the value of fragment for Lightweight 4over6 tunnel.

- tunnel-num-threshold: used to set the maximum number of tunnels that can be created on the lw4over6 device simultaneously.

- active-tunnel-num (ro): used to present the number of tunnels currently provisioned on the device.

- active (ro): used to show the status of particular binding-entry.

- stat-count (ro): use to show the numbers of packets and bytes information of specific device respectively.

Introduction of important lwB4 nodes:

- b4-ipv6-addr-format: indicates the format of lwB4 IPv6 address. If set to true, it indicates that the IPv6 source address of the lwB4 is constructed according to the description in [I-D.ietf-softwire-lw4over6]; if set to false, the lwB4 can use any /128 address from the assigned IPv6 prefix.

- binding-ipv6info: used to set the IPv6 address type which is combined in a binding entry, for a complete address or a prefix.

- stat-count (ro): use to show the numbers of packets and bytes information of specific device respectively.
3.3. MAP-E Tree Diagrams

Figure 3 defines the softwire data model for MAP-E:

module: ietf-softwire
  +--rw softwire-config
    |   +--...  
    |      +--rw map-e (map-e)?
    |          |   +--rw enable?   boolean
    |          |   +--rw map-e-instances
    |          |       +--rw map-e-instance* [id]
    |          |          |   +--rw id           uint32
    |          |          |   +--rw name?        string
    |          |          |   +--rw map-rules
    |          |          |       +--rw map-rule* [id]
    |          |          |          |   +--rw id           uint8
    |          |          |          |   +--rw map-rule-type enumeration
    |          |          |          |   +--rw rule-ipv6-prefix inet:ipv6-prefix
    |          |          |          |   +--rw rule-ipv4-prefix inet:ipv4-prefix
    |          |          |          |   +--rw port-set
    |          |          |          |       |   +--rw offset       uint8
    |          |          |          |       |   +--rw psid          uint16
    |          |          |          |       |   +--rw psid-len      uint8
    |          |          |          |       |   +--rw ea-len        uint8
    |          |          |       +--rw br-ipv6-addr? inet:ipv6-address
    |          |       +--rw tunnels
    |          |       |   +--rw tunnel* [tunnel-id]
    |          |       |       +--rw tunnel-id    uint8
    |          |       |       +--rw tunnel-mtu   uint32
    |          |       |       +--rw fragment-mru uint16
    |   +--...  
  +--ro softwire-state  
    +--...  
    +--ro map-e (map-e)?
    |   +--ro map-e-instances
    |       +--ro map-e-instance* [id]
    |          |   +--ro id           uint32
    |          |   +--ro name?        string
    |          |   +--ro traffic-stat
    |          |       +--ro sentPacket? yang:zero-based-counter64
    |          |       +--ro sentByte?  yang:zero-based-counter64
    |          |       +--ro rcvdPacket? yang:zero-based-counter64
    |          |       +--ro rcvdByte?  yang:zero-based-counter64
    |          |       +--ro droppedPacket? yang:zero-based-counter64
    |          |       +--ro droppedByte? yang:zero-based-counter64
    |   +--...  

Figure 3: Softwire MAP-E Data Model Structure
Introduction of important MAP-E nodes:

- map-rule-type: used to define the type of map rule. The data type is enumeration, which are "BMR" and "FMR".

- offset: used to set the number of offset bits.

- psid: used to algorithmically identify a set of ports exclusively for a specific softwire.

- ea-len: used to set the length of the Embedded-Address (EA), which defined in the mapping rule for a MAP domain.

- tunnel-mtu: used to set the value of MTU for MAP-E tunnel.

- fragment-mru: used to the value of fragment for MAP-E tunnel.

- stat-count (ro): use to show the numbers of packets and bytes information of specific device respectively.

3.4. MAP-T Tree Diagrams

Figure 4 defines the softwire data model for MAP-T:
module: ietf-softwire
   +--rw softwire-config
      +--...
      +--rw map-t {map-t}?
         +--rw enable?  boolean
      +--rw map-t-instances
         +--rw map-t-instance* [id]
            +--rw id  uint32
            +--rw name?  string
         +--rw map-rules
            +--rw map-rule* [id]
               +--rw id  uint8
               +--rw map-rule-type  enumeration
               +--rw rule-ipv6-prefix  inet:ipv6-prefix
               +--rw rule-ipv4-prefix  inet:ipv4-prefix
               +--rw port-set
                  +--rw offset  uint8
                  +--rw psid  uint16
                  +--rw psid-len  uint8
               +--rw ea-len  uint8
               +--rw dmr-ipv6-prefix?  inet:ipv6-prefix
      +--ro softwire-state
         +--...
         +--ro map-t {map-t}?
            +--ro map-t-instances
               +--ro map-t-instance* [id]
                  +--ro id  uint32
                  +--ro name?  string
               +--ro traffic-stat
                  +--ro sentPacket?  yang:zero-based-counter64
                  +--ro sentByte?  yang:zero-based-counter64
                  +--ro rcvdPacket?  yang:zero-based-counter64
                  +--ro rcvdByte?  yang:zero-based-counter64
                  +--ro droppedPacket?  yang:zero-based-counter64
                  +--ro droppedByte?  yang:zero-based-counter64

Figure 4: Softwire MAP-T Data Model Structure

Introduction of important MAP-T nodes:

- map-rule-type: is used to define the type of map rule. The data type is enumeration, which are "BMR" and "FMR".

- offset: is used to set the number of offset bits.

- psid: is used to algorithmically identify a set of ports exclusively for a specific softwire.
**3.5. Notifications for Softwire YANG**

This section describes the diagram tree for the notifications. These notifications pertain to configuration and monitoring portions of specific Softwire mechanisms. The logic is that, the softwire instance notifies the NETCONF client with the index for a mapping entry and then the NETCONF client retrieves the related information from the operational datastore of that instance.
module: ietf-softwire

notifications:

  +++-n softwire-lwafr-event   {lwafr}?
     |  +--ro lwafr-id       -> /softwire-state/lwafr/lwafr-instances/
     |     |                  lwafr-instance/id
     |  +--ro invalid-entry*   -> /softwire-config/lwafr/lwafr-instances/
     |     |                  lwafr-instance[id=current()]/../lwafr-id/
     |     |                    binding-table/binding-entry/binding-ipv6-addr
     |  +--ro added-entry*      inet:ipv6-address
     |  +--ro modified-entry*   -> /softwire-config/lwafr/lwafr-instances/
     |     |                  lwafr-instance[id=current()]/../lwafr-id/
     |     |                    binding-table/binding-entry/binding-ipv6-addr
          |  +---n softwire-lwb4-event   {lwb4}?
            |     |  +--ro lwb4-binding-ipv6-addr-change inet:ipv6-address
          |  +---n softwire-map-e-event   {map-e}?
            |     |  +--ro map-e-id       -> /softwire-config/map-e/map-e-instances/
            |     |     |                  map-e-instance/id
            |     |  +--ro invalid-entry-id*   -> /softwire-config/map-e/map-e-instances/
            |     |     |                  map-e-instance[id=current()]/../map-e-id/
            |     |     |                    map-rules/map-rule/id
            |     |  +--ro added-entry*   uint32
            |     |  +--ro modified-entry*   -> /softwire-config/map-e/map-e-instances/
            |     |     |                  map-e-instance[id=current()]/../map-e-id/
            |     |     |                    map-rules/map-rule/id
            |  +---n softwire-map-t-event   {map-t}?
            |     |  +--ro map-t-id       -> /softwire-config/map-t/map-t-instances/
            |     |     |                  map-t-instance/id
            |     |  +--ro invalid-entry-id*   -> /softwire-config/map-t/map-t-instances/
            |     |     |                  map-t-instance[id=current()]/../map-t-id/
            |     |     |                    map-rules/map-rule/id
            |     |  +--ro added-entry*   uint32
            |     |  +--ro modified-entry*   -> /softwire-config/map-t/map-t-instances/
            |     |     |                  map-t-instance[id=current()]/../map-t-id/
            |     |     |                    map-rules/map-rule/id

Figure 5: Softwire Notifications Data Model Structure

Introduction of important notification nodes:

- invalid-entry, added-entry, modified-entry: use to notify the client that a specific binding entry has been expired/invalid, added, or modified respectively. Same meaning for map rule list.

- lwb4-binding-ipv6-addr-change: use to present the lWB4’s binding-ipv6-address has been changed or the value of the ‘b4-ipv6-addr-format’ is "false".
4. Softwire YANG Model

This module imports typedefs from [RFC6991].

<CODE BEGINS> file "ietf-softwire@2015-04-07.yang"

module ietf-softwire {
  namespace "urn:ietf:params:xml:ns:yang:softwire";
  prefix "softwire";

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

  organization "Softwire Working Group";

  contact
    "Qi Sun sunqi@csnet1.cs.tsinghua.edu.cn
    Hao Wang wanghai3@mails.tsinghua.edu.cn
    Yong Cui yong@csnet1.cs.tsinghua.edu.cn
    Ian Farrer ian.farrer@telekom.de
    Mohamed Boucadair mohamed.boucadair@orange.com
    Rajiv Asati rajiva@cisco.com"

  description
    "This document defines a YANG data model for the configuration and
management of IPv4-in-IPv6 Softwire Border Routers and Customer
Premises Equipment. It covers Lightweight 4over6, MAP-E and MAP-T
Softwire mechanisms.

Copyright (c) 2014 IETF Trust and the persons identified
as authors of the code. All rights reserved.
This version of this YANG module is part of RFC XXX; see the RFC
itself for full legal notices."

revision 2015-04-07 {
  description
    "Version-03: Integrate lw4over6; Update state nodes; Correct
    grammar errors; Reuse groupings; Update descriptions.
    Simplify the model.";
}

revision 2015-02-10 {
  description
    "Version-02: Add notifications.";
}

Sun, et al. Expires October 9, 2015 [Page 14]
revision 2015-02-06 {
  description
    "Version-01: Correct grammar errors; Reuse groupings; Update descriptions.";
}

revision 2015-02-02 {
  description
    "Initial revision.";
}

/*
 * Features
 */

feature lw4over6 {
  description
    "Lightweight 4over6 (lw4over6) is an IPv4-over-IPv6 tunnelling transition mechanism. Lightweight 4over6 is a solution designed specifically for complete independence between IPv6 subnet prefix (and /128 IPv6 address) and IPv4 address with or without IPv4 address sharing. This is accomplished by maintaining state for each softwire (per-subscriber state) in the central lwAFTR and a hub-and-spoke forwarding architecture. In order to delegate the NAPT function and achieve IPv4 address sharing, port-restricted IPv4 addresses needs to be allocated to CEs.";
  reference
    "I-D.ietf-softwire-lw4over6";
}

feature lwaftr {
  if-feature lw4over6;
  description
    "The AFTRs (BRs) for Lightweight 4over6, so-called lwAFTR. This feature indicates that a instance functions as a lwAFTR. A lwAFTR is an IPv4-in-IPv6 tunnel concentrator that maintains per-subscriber IPv4-IPv6 address binding.";
}

feature lwb4 {
  if-feature lw4over6;
  description
    "The B4s (CEs) for Lightweight 4over6, so-called lwB4. This feature indicates that a instance functions as a lwB4. A lwB4 is an IPv4-in-IPv6 tunnel initiator. It is dual-stack capable node, either a directly connected end-host or a CE. It sources IPv4 connections using the configured port-set and the public IPv4 address.";
}
feature map-e {

description
"MAP-E is an IPv6 transition mechanism for transporting IPv4 packets across an IPv6 network using IP encapsulation. MAP-E allows for a reduction of the amount of centralized state using rules to express IPv4/IPv6 address mappings. This introduces an algorithmic relationship between the IPv6 subnet and IPv4 address. This relationship also allows the option of direct, meshed connectivity between users. Alternatively, MAP-E can be configured to support IPv4/IPv6 independent binding. This feature indicates the instance functions as a MAP-E instance.";

reference
"I-D.ietf-softwire-map";
}

feature map-t {

description
"The Mapping of Address and Port - Translation (MAP-T) architecture is a double stateless NAT64 based solution. It uses the stateless algorithmic address & transport layer port mapping scheme defined in MAP-E. The MAP-T solution differs from MAP-E in the use of IPv4-IPv6 translation, rather than encapsulation, as the form of IPv6 domain transport. This feature indicates the instance functions as a MAP-T instance.";

reference
"I-D.ietf-softwire-map-t";
}

/*
 * Grouping
 */

grouping port-set {

description
"Use the PSID algorithm to represent a range of transport layer ports.";

leaf offset {
mandatory true;

type uint8 {
range 0..16;
}

description
"The number of offset bits. In Lightweight 4over6, the default value is 0 for assigning one contiguous port range. In MAP-E/T, the default value is 6, which excludes system ports by default and assigns distributed port ranges. If the this parameter is
larger than 0, the value of offset MUST be greater than 0.

leaf psid {
  mandatory true;
  type uint16;
  description
  "Port Set Identifier (PSID) value, which identifies a set of
  ports algorithmically.";
}

leaf psid-len {
  mandatory true;
  type uint8 {
    range 0..16;
  }
  description
  "The length of PSID, representing the sharing ratio for a IPv4
  address.";
}

grouping binding-entry {
  description
  "The lwAFTR maintains an address binding table that contains the
  binding between the lwB4’s IPv6 address, the allocated IPv4
  address and restricted port-set.";
  choice binding-ipv6info {
    description
    "Allow for the binding table to be specified in one of two ways,
    either with an explicit /128 or a /64 with the IID automatically
    constructed from the IPv4 address and port set.";
    case ipv6addr {
      leaf binding-ipv6-addr {
        mandatory true;
        type inet:ipv6-address;
        description
        "The /128 IPv6 address of the lwB4, which is used to bind
        the address and port-set and source the tunnel.";
      }
    }
    case ipv6pref {
      leaf binding-ipv6-prefix {
        mandatory true;
        type inet:ipv6-prefix;
        description
        "The operator-assigned IPv6 prefix of the lwB4. The lwAFTR
        uses the prefix to construct the /128 IPv6 address to complete
        the binding entry.";
      }
    }
  }
}
leaf binding-ipv6info {
  mandatory true;
  type union {
    type inet:ipv6-address;
    type inet:ipv6-prefix;
  }
  description "The IPv6 information for a binding entry. If it’s an IPv6 prefix, it indicates that the IPv6 source address of the lwB4 is constructed according to the description in [I-D.ietf-softwire-lw4over6]; if it’s an IPv6 address, it means the lwB4 uses any /128 address from the assigned IPv6 prefix."
}

leaf binding-ipv4-addr {
  mandatory true;
  type inet:ipv4-address;
  description "The IPv4 address assigned to the lwB4, which is used as the IPv4 External Address for lwB4 local NAT44."
}

counter port-set {
  uses port-set {
    refine offset {
      default "0";
    }
  }
}

leaf lwaftr-ipv6-addr {
  type inet:ipv6-prefix;
  description "The IPv6 address for lwaftr. Optional for the binding entry."
}

leaf lifetime {
  type uint32;
  units seconds;
}

/*
   grouping nat-table {

Sun, et al. Expires October 9, 2015 [Page 18]
description
"Grouping 'nat-table' is not extended. The current mechanism is focusing on the provisioning of external IP address and port set; other NAT-specific considerations are out of scope for this model."
}
*/

grouping map-instance {
  leaf id {
    mandatory true;
    type uint32;
  }
  leaf name {
    type string;
  }
  container map-rules {
    list map-rule {
      description
      "A set of parameters describing the mapping between an IPv4 prefix, IPv4 address or shared IPv4 address and an IPv6 prefix or address. Each domain uses a different mapping rule set."
      key "id";
      leaf id {
        type uint8;
      }
      leaf map-rule-type {
        mandatory true;
        type enumeration {
          enum "BMR";
          enum "FMR";
        }
        description
        "The BMR and FMR share the rule format. BMR is used for a node to configure itself with IPv4 information retrieved from the rule. FMR is designed for the in-domain 4-in-6 routing, used in mesh mode. A BMR can be FMR in some case. The DMR for map-t is defined separately."
      }
      leaf rule-ipv6-prefix {
        type inet:ipv6-prefix;
        mandatory true;
        description
        "The Rule IPv6 prefix defined in the mapping rule."
      }
      leaf rule-ipv4-prefix {
        type inet:ipv4-prefix;
        mandatory true;
        description
        "The Rule IPv4 prefix defined in the mapping rule."
      }
    }
  }
}
description
   "The Rule IPv4 prefix defined in the mapping rule."
}
container port-set {
    uses port-set {
        refine offset {
            default "6";
        }
    }
    leaf ea-len {
        mandatory true;
        type uint8;
        description
            "Embedded Address (EA) bits are the IPv4 EA-bits in the IPv6
            address identify an IPv4 prefix/address (or part thereof) or
            a shared IPv4 address (or part thereof) and a port-set
            identifier. The length of the EA-bits is defined as part of a
            MAP rule for a MAP domain.";
    }
}
}
grouping traffic-stat {
    leaf sentPacket {
        type yang:zero-based-counter64;
    }
    leaf sentByte {
        type yang:zero-based-counter64;
    }
    leaf rcvdPacket {
        type yang:zero-based-counter64;
    }
    leaf rcvdByte {
        type yang:zero-based-counter64;
    }
    leaf droppedPacket {
        type yang:zero-based-counter64;
    }
    leaf droppedByte {
        type yang:zero-based-counter64;
    }
}
/*
* Configuration Data Nodes
*/
container softwire-config {
    description
    "The configuration data for Softwire instances. And the shared
    data describes the softwire data model which is common to all of
    the different softwire mechanisms, such as description.";
    leaf description {
        type string;
        description
        "A textual description of Softwire.";
    }
}

container lwaftr {
    if-feature lwaftr;
    description
    "Indicate this instance supports the lwAFTR function. The
    instances advertise the lwaftr feature through the
capability exchange mechanism when a NETCONF session is
established.";
    leaf enable {
        type boolean;
        description
        "Enable/disable the lwAFTR function.";
    }
}

container lwaftr-instances {
    description
    "A set of lwAFTRs to be configured.";
    list lwaftr-instance {
        key "id";
        leaf id {
            type uint32;
        }
        leaf name {
            type string;
            description "The name for the lwaftr.";
        }
        leaf softwire-num-threshold {
            mandatory true;
            type uint32;
            description
            "The maximum number of tunnels that can be created on
            the lwAFTR.";
        }
        leaf tunnel-mtu {
            mandatory true;
            type uint32;
            description
            "The MTU for Lightweight 4over6 tunnel.";
        }
        leaf fragment-mru {
mandatory true;
type uint16;
description
"The fragmentation MRU for Lightweight 4over6 tunnel.";
}
container binding-table {
  list binding-entry {
    key "binding-ipv6info";
    uses binding-entry;
  }
}
}

container lwb4 {
  if-feature lwb4;
description
"Indicate this instance supports the lwB4 function. The
instances advertise the lwB4 feature through the
capability exchange mechanism when a NETCONF session is
established.";
leaf enable {
  type boolean;
description
"Enable/disable the lwB4 function.";
}
container lwb4-instances {
  description
"A set of lwB4s to be configured.";
list lwb4-instance {
  key "binding-ipv6info";
  leaf name {
    type string;
description "The lwb4 name.";
  }
  leaf tunnel-mtu {
    mandatory true;
type uint32;
description
"The MTU for Lightweight 4over6 tunnel.";
  }
  leaf fragment-mru {
    mandatory true;
type uint16;
description
"The fragment MRU for Lightweight 4over6 tunnel.";
  }
  leaf b4-ipv6-addr-format {

Sun, et al. Expires October 9, 2015 [Page 22]
type boolean;
mandatory true;
description
"The format of lwB4 IPv6 address. If set to true, it
indicates that the IPv6 source address of the lwB4 is
constructed according to the description in
[I-D.ietf-softwire-lw4over6]; if set to false, the lwB4
can use any /128 address from the assigned IPv6 prefix.
[DISCUSS] Is this redundant? ";
}
uses binding-entry;
}
}
container map-e {
if-feature map-e;
description
"Indicate the instances support the MAP-E function. The
instances advertise the map-e feature through the capability
exchange mechanism when a NETCONF session is established.";
leaf enable {
  type boolean;
default "true";
description
  "Enable/disable the MAP-E function.";
}
container map-e-instances {
description
  "A set of MAP-E instances to be configured, applying to BRs and
CEs."
list map-e-instance {
  key "id";
  uses map-instance;
  leaf tunnel-mtu {
    mandatory true;
type uint32;
description
    "The MTU for MAP-E tunnel.";
  }
  leaf fragment-mru {
    mandatory true;
type uint16;
description
    "The fragment MRU for MAP-E tunnel.";
  }
  leaf br-ipv6-addr {
    type inet:ipv6-address;
description
"..."
"The IPv6 address of the MAP-E BR."
)
)
}
counter map-t {
  if-feature map-t;
  description
    "Indicate the instances support the MAP-T function. The
    instances advertise the map-t feature through the capability
    exchange mechanism when a NETCONF session is established.";
  leaf enable {
    type boolean;
    default "true";
    description
      "Enable/disable the MAP-T function.";
  }
}

container map-t-instances {
  description
    "A set of the MAP-T instances to be configured, applying to BRs
    and CEs."
  list map-t-instance {
    key "id";
    uses map-instance;
    leaf dmr-ipv6-prefix {
      type inet:ipv6-prefix;
      description
        "The IPv6 prefix of the MAP-T BR. ";
    }
  }
}

namespace  
container softwire-state {  
  config false;
  description
    "The operational state data for Softwire instances. ";
  leaf description {
    type string;
    description
      "A textual description of the softwire instances.";
  }
}

container lwaftr {
if-feature lwaftr;
config false;
description
 "Indicate this instance supports the lwAFTR function. The
 instances advertise the lwaftr feature through the
 capability exchange mechanism when a NETCONF session is
 established.";
container lwaftr-instances {
    description
     "A set of lwAFTRs.";
    list lwaftr-instance {
        key "id";
        leaf id {
            type uint32;
        }
        leaf name {
            type string;
            description "The name for this lwaftr.";
        }
        uses traffic-stat;
        leaf active-softwire-num {
            type uint32;
            description
             "The number of currently active tunnels on the lw4over6
              instance.";
        }
    }
    container binding-table {
        list binding-entry {
            key "binding-ipv6info";
            leaf binding-ipv6info {
                mandatory true;
                type union {
                    type inet:ipv6-address;
                    type inet:ipv6-prefix;
                }
                description
                 "The IPv6 information used to identify a binding entry. ";
            }
            leaf active {
                type boolean;
                description
                 "Status of a specific tunnel.";
            }
        }
    }
}
container lwb4 {
    if-feature lwb4;
    config false;
    description
        "Indicate this instance supports the lWB4 function. The
        instances advertise the lwb4 feature through the
        capability exchange mechanism when a NETCONF session is
        established.";
    container lwb4-instances {
        description
            "Status of the configured lWB4s.";
        list lwb4-instance {
            key "binding-ipv6info";
            leaf name {
                type string;
                description "The lwb4 name.";
            }
            leaf binding-ipv6info {
                mandatory true;
                type union {
                    type inet:ipv6-address;
                    type inet:ipv6-prefix;
                }
                description
                    "The IPv6 information used to identify a binding entry. ";
            }
            uses traffic-stat;
        }
    }
}
container map-e {
    if-feature map-e;
    config false;
    description
        "Indicate the instances support the MAP-E function. The
        instances advertise the map-e feature through the capability
        exchange mechanism when a NETCONF session is established.";
    container map-e-instances {
        description
            "Status of MAP-E instance(s).";
        list map-e-instance {
            key "id";
            leaf id {
                type int32;
            }
            leaf name {
                type string;
                description "The map-e instance name.";
            }
        }
    }
}
uses traffic-stat;
}
}
)
}
container map-t {
    if-feature map-t;
    config false;
    description
        "Indicate the instances support the MAP-T function. The instances advertise the map-t feature through the capability exchange mechanism when a NETCONF session is established.";
    container map-t-instances {
        description
            "Status of MAP-T instances.";
        list map-t-instance {
            key "id";
            leaf id {
                type int32;
            }
            leaf name {
                type string;
                description "The map-t instance name.";
            }
            uses traffic-stat;
        }
    }
}
/*
 * Notifications
 */
notification softwire-lwaftr-event {
    if-feature lwaftr;
    leaf lwaftr-id {
        mandatory true;
        type leafref {
            path
                "/softwire-state/lwaftr/lwaftr-instances/"
                + "lwaftr-instance/id";
        }
    }
    leaf-list invalid-entry {
        type leafref {
            path
                "/softwire-config/lwaftr/lwaftr-instances/"
                + "lwaftr-instance[id=current()]/../lwaftr-id/";
+ "binding-table/binding-entry/binding-ipv6info";
}  
description  
"Notify the client that a specific binding entry has been  
expired/invalid. The binding-ipv6info identifies an entry.";
}
leaf-list added-entry {  
type inet:ipv6-address;  
description  
"Notify the client that a binding entry has been added.  
The ipv6 address of that entry is the index. The client  
get other information from the lwaftr about the entry  
indexed by that ipv6 address."
};
}
leaf-list modified-entry {  
type leafref {  
path  
"/softwire-config/lwaftr/lwaftr-instances/"  
+ "lwaftr-instance[id=current()//lwaftr-id]/"  
+ "binding-table/binding-entry/binding-ipv6info";
 }
}
}
notification softwire-lwb4-event {  
if-feature lwb4;  
leaf lwb4-binding-ipv6-addr-change {  
mandatory true;  
type inet:ipv6-address;  
description  
"The source tunnel IPv6 address of the lwb4. If 'b4-ipv6-addr-format'  
is false, or the lwb4’s binding-ipv6-address changes for any reason,  
it SHOULD notify the NETCONF client.";
 }
}
notification softwire-map-e-event {  
if-feature map-e;  
leaf map-e-id {  
mandatory true;  
type leafref {  
path  
"/softwire-config/map-e/map-e-instances/map-e-instance/id";
 }
}  
leaf-list invalid-entry-id {  
type leafref {  
path  
"/softwire-config/map-e/map-e-instances/"  
"
leaf-list added-entry {
    type uint32;
}

leaf-list modified-entry {
    type leafref {
        path
            "/softwire-config/map-e/map-e-instances/
            + map-e-instance[id=current()//../map-e-id]/map-rules/
            + map-rule/id";
    }
}

notification softwire-map-t-event {
    if-feature map-t;
    leaf map-t-id {
        mandatory true;
        type leafref {
            path
                "/softwire-config/map-t/map-t-instances/map-t-instance/id";
        }
    }
    leaf-list invalid-entry-id {
        type leafref {
            path
                "/softwire-config/map-t/map-t-instances/
                + map-t-instance[id=current()//../map-t-id]/map-rules/
                + map-rule/id";
        }
    }
    leaf-list added-entry {
        type uint32;
    }
    leaf-list modified-entry {
        type leafref {
            path
                "/softwire-config/map-t/map-t-instances/
                + map-t-instance[id=current()//../map-t-id]/map-rules/
                + map-rule/id";
        }
    }
}

<CODE ENDS>
5. Example of Configure Lw4over6 Binding-Table

The lwAFTR maintains an address binding table which contains the following 3-tuples:

- IPv6 Address for a single lwB4
- Public IPv4 Address
- Restricted port-set

The entry has two functions: the IPv6 encapsulation of inbound IPv4 packets destined to the lwB4 and the validation of outbound IPv4-in-IPv6 packets received from the lwB4 for de-capsulation.

Requirement: Add an entry that maintain the relationship between 3-tuples of lwB4 (2001::1) in binding-table, which on the lwAFTR (2001::2). The data value of this 3-tuples are '2001::1', '123.1.1.1' and '1234' respectively.

Here is the example binding-table configuration xml:
6. Security Considerations

The YANG module defined in this memo is 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 the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in this YANG module which are writable/creatable/deletable (i.e. config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g. edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

Figure 6: Lw4over6 Binding-Table Configuration XML
subtrees and data nodes and state why they are sensitive

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g. via get, get-config or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

subtrees and data nodes and state why they are sensitive

7. IANA Considerations

A registry for standard YANG modules shall be set up. This document registers one URI for the YANG XML namespace in the IETF XML registry [RFC3688].


8. Acknowledgements

The authors would like to thank Lishan Li, Bert Wijnen and Giles Heron for their contributions to this work.

9. References

9.1. Normative References

[I-D.ietf-softwire-lw4over6]

[I-D.ietf-softwire-map]

[I-D.ietf-softwire-map-t]

Internet-Draft              YANG for softwire                 April 2015

[ RFC3688 ]  Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,

[ 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.

              Protocol (NETCONF) Access Control Model", RFC 6536,
              March 2012.

9.2.  Informative References

[ RFC6991 ]  Schoenwaelder, J., "Common YANG Data Types", RFC 6991,
              July 2013.

Authors’ Addresses

Qi Sun
Tsinghua University
Beijing  100084
P.R. China

Phone: +86-10-6278-5822
Email: sunqi@csnet1.cs.tsinghua.edu.cn

Hao Wang
Tsinghua University
Beijing  100084
P.R. China

Phone: +86-10-6278-5822
Email: wangh13@mails.tsinghua.edu.cn
Yong Cui  
Tsinghua University  
Beijing  100084  
P.R. China  

Phone: +86-10-6260-3059  
Email: yong@csnet1.cs.tsinghua.edu.cn

Ian Farrer  
Deutsche Telekom AG  
CTO-ATI,Landgrabenweg 151  
Bonn, NRW  53227  
Germany  

Email: ian.farrer@telekom.de

Mohamed Boucadair  
France Telecom  
Rennes  35000  
France  

Email: mohamed.boucadair@orange.com

Rajiv Asati  
Cisco Systems, Inc.  
7025 Kit Creek Rd.  
RTP, NC  27709  
USA  

Email: Rajiva@cisco.com