Softwire Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 16, 2015

Q. Sun
H. Wang
Y. Cui
Tsinghua University
I. Farrer
Deutsche Telekom AG
M. Boucadair
France Telecom
R. Asati
Cisco Systems, Inc.
February 12, 2015

YANG Data Model for IPv4-in-IPv6 Softwire
draft-sun-softwire-yang-02

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 August 16, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

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 (CPEs).

This document defines a YANG data model that can be used for the configuration and management of IPv4-in-IPv6 Softwire BRs and/or CPEs. To ensure interoperability in mixed vendor environments, it is important that the models can be easily reused between different vendors and implementations.

Due to the inherent similarities of the data plane forwarding, the configuration and management parameters of the different softwire mechanisms are defined in the same YANG model. Parameters that are common to all solutions are abstracted in the common module while specific parameters are defined in individual modules that are specific to a given mechanism (see for example, [I-D.ietf-softwire-unified-cpe]).

Each specific softwire mechanism has their own individual YANG modules:

- 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 CPEs.

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

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].

A simplified graphical representation of the data model is provided in this document. [RFC6087] provides definitions of the symbols used in these diagrams.

1.2. 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 CPEs for the following IPv4-in-IPv6 Softwire mechanisms: Lightweight 4over6, MAP-E and MAP-T.

For the lightweight 4over6, the configure and manage information of lwB4 and lwAFTR are different. The lw4over6 AFTRs needs to maintain the binding table of lwB4s. The lw4o6 lwB4s 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 CPEs such as softwire type, maximum number of softwires, etc.

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 AFTR

The lw4over6 AFTR holds configuration for IPv4-IPv6 address bindings. This is used for the forwarding of traffic originating from lwB4s.

2.3. Lightweight 4over6 lwB4

The lw4over6 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.

2.4. MAP-E

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

2.5. MAP-T

MAP-E elements (BR and CPE) are provisioned with the MAP rules necessary for defining MAP domains and forwarding rules. MAP-T CPEs 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:

```
+--rw softwire-config
|  +--rw enable?                            boolean
|  +--rw description?                       string
|  +--rw tunnel-mtu                         uint32
|  +--rw lw4over6-aftr?
|  +--rw lw4over6-b4?
|  +--rw map-e?
|  +--rw map-t?
|
+--ro softwire-state
   +--ro enable?                            boolean
   +--ro description?                       string
   +--ro tunnel-mtu                         uint32
   +--ro lw4over6-aftr?
   +--ro lw4over6-b4?
   +--ro map-e?
   +--ro map-t?
```

Figure 1: Softwire Common Data Model Structure

The mechanism specific models for lw4over6-aftr, lw4over6-b4, MAP-E and MAP-T are described in detail in the following sections.

3.2. Lightweight 4over6 AFTR Tree Diagrams

Figure 2 defines the softwire data model for Lightweight 4over6 AFTR:

```
      +--rw softwire-config
      |  +--...
      |  +--rw lw4over6-aftr
      |     +--rw enable?                    boolean
      |     +--rw lw4over6-aftr-instances
      |        +--rw lw4over6-aftr-instance* [id]
      |           +--rw id                        uint32
      |           +--rw name?                     string
      |           +--rw softwire-num-threshold    uint32
      |           +--rw binding-table
      |              +--rw binding-entries* [binding-ipv6-addr]
      |                 +--rw binding-ipv6-addr      inet:ipv6-address
      |                 +--rw binding-ipv6-prefix    inet:ipv6-prefix
      |                 +--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 lifetime?              uint32
      |                 +--rw active?                boolean
      +--ro softwire-state
         +--...
         +--ro lw4over6-aftr
            +--ro enable?                    boolean
            +--ro lw4over6-aftr-instances
               +--ro lw4over6-aftr-instance* [id]
                  +--ro id                     uint32
                  +--ro name?                  string
                  +--ro active-softwire-num    uint32
                  +--ro binding-table
                     +--ro binding-entries* [binding-ipv6-addr]
                        +--ro binding-ipv6-addr      inet:ipv6-address
                        +--ro binding-ipv6-prefix    inet:ipv6-prefix
                        +--ro binding-ipv4-addr      inet:ipv4-address
                        +--ro port-set
                        |  +--ro offset      uint8
                        |  +--ro psid        uint16
                        |  +--ro psid-len    uint8
                        +--ro lwaftr-ipv6-addr?      inet:ipv6-prefix
                        +--ro active?                boolean
```

Figure 2: Softwire Lightweight 4over6 AFTR Data Model Structure

- Node "softwire-num-threshold" is used to set the maximum number of tunnels that can be created on the lw4over6 device simultaneously.

- Node "active-softwire-num" is used to present the number of tunnels currently provisioned on the device.

- Node "offset" is used to set the number of offset bits as in the PSID algorithm.

- Node "psid" is used to algorithmically identify a set of ports for a specific softwire.

- Node "active" is used to add or delete a particular binding-entry.

3.3. Lightweight 4over6 lwB4 Tree Diagrams

Figure 3 defines the softwire data model for a Lightweight 4over6 lwB4 element:

```
   module: ietf-softwire
      +--rw softwire-config
      |  +--...
      |  +--rw lw4over6-b4
      |     +--rw enable?                  boolean
      |     +--rw lw4over6-b4-instances
      |        +--rw lw4over6-b4-instance* [binding-ipv6-addr]
      |           +--rw name?                    string
      |           +--rw b4-ipv6-addr-format    boolean
      |           +--rw binding-ipv6-addr      inet:ipv6-address
      |           +--rw binding-ipv6-prefix    inet:ipv6-prefix
      |           +--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 lifetime?              uint32
      |           +--rw nat-table
      |              +--...
      +--ro softwire-state
         +--...
         +--ro lw4over6-b4
            +--ro enable?                  boolean
            +--ro lw4over6-b4-instances
               +--ro lw4over6-b4-instance* [binding-ipv6-addr]
                  +--ro name?                    string
                  +--ro b4-ipv6-addr-format    boolean
                  +--ro binding-ipv6-addr      inet:ipv6-address
                  +--ro binding-ipv6-prefix    inet:ipv6-prefix
                  +--ro binding-ipv4-addr      inet:ipv4-address
                  +--ro port-set
                  |  +--ro offset      uint8
                  |  +--ro psid        uint16
                  |  +--ro psid-len    uint8
                  +--ro lwaftr-ipv6-addr?      inet:ipv6-prefix
                  +--ro nat-table
                     +--...
```

Figure 3: Softwire Lightweight 4over6 lwB4 Data Model Structure

o Node "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.

- Node "binding-ipv4-addr" is used to configure an IPv4 address to the lwB4.

- Node "offset" is used to set the number of offset bits.

- Node "psid" is used to algorithmically identifies a set of ports exclusively, it is allocated by vendors and calculated by devices.

- Node "bind-ipv6-prefix" is used to perform a longest prefix match against the active IPv6 addresses configured on the lwB4 so that a suitable tunnel source address prefix can be selected.

- Container "nat-table" is not extended. It means that the focus is on the provisioning of the external IP address and/or port set; other NAT-specific considerations are out of scope.

3.4. MAP-E Tree Diagrams

Figure 4 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-rule-table
      |           |  +--rw map-rules* [id]
      |           |     +--rw id                  uint32
      |           |     +--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
      +--ro softwire-state
         +--...
         +--ro map-e {map-e}?
            +--ro enable?            boolean
            +--ro map-e-instances
               +--ro map-e-instance* [id]
                  +--ro id                uint32
                  +--ro map-rule-table
                  |  +--ro map-rules* [id]
                  |     +--ro id                  string
                  |     +--ro map-rule-type       enumeration
                  |     +--ro rule-ipv6-prefix    inet:ipv6-prefix
                  |     +--ro rule-ipv4-prefix    inet:ipv4-prefix
                  |     +--ro port-set
                  |     |  +--ro offset      uint8
                  |     |  +--ro psid        uint16
                  |     |  +--ro psid-len    uint8
                  |     +--ro ea-len              uint8
                  +--ro br-ipv6-addr      inet:ipv6-address
```

Figure 4: Softwire MAP-E Data Model Structure

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

o Node "offset" is used to set the number of offset bits.

- Node "psid" is used to algorithmically identify a set of ports exclusively for a specific softwire.

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

3.5. MAP-T Tree Diagrams

Figure 5 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-rule-table
      |           |  +--rw map-rules* [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 enable?            boolean
            +--ro map-t-instances
               +--ro map-t-instance* [id]
                  +--ro id                uint32
                  +--ro map-rule-table
                  |  +--ro map-rules* [id]
                  |     +--ro id                   uint32
                  |     +--ro map-rule-type       enumeration
                  |     +--ro rule-ipv6-prefix    inet:ipv6-prefix
                  |     +--ro rule-ipv4-prefix    inet:ipv4-prefix
                  |     +--ro port-set
                  |     |  +--ro offset      uint8
                  |     |  +--ro psid        uint16
                  |     |  +--ro psid-len    uint8
                  |     +--ro ea-len              uint8
                  +--ro map-t-ce {map-t-ce}?
                     +--ro dmr-ipv6-prefix?   inet:ipv6-prefix
```

Figure 5: Softwire MAP-T Data Model Structure

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

- Node "dmr-ipv6-prefix" defines the DMR in MAP-T.
- Node "offset" is used to set the number of offset bits.
- Node "psid" is used to algorithmically identify a set of ports exclusively for a specific softwire.
- Node "ea-len" is used to set the length of the Embedded- Address (EA), which defined in the mapping rule for a MAP domain.

3.6. Notifications for Softwire YANG

This section describes the diagram tree for the notifications. These notifications pertain to configuration and monitoring portions of specific Softwire machanisms. 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-lwaftr-event     {lw4over6-aftr}?
      |  +--ro lwaftr-id              leafref
      |  +--ro exceed-sw-num-limit?   boolean
      |  +--ro invalid-entry*         leafref
      |  +--ro added-entry*           inet:ipv6-address
      |  +--ro modified-entry*        leafref
      +---n softwire-lwb4-event       {lw4over6-b4}?
      |  +--ro lwb4-binding-ipv6-addr-change    inet:ipv6-address
      +---n softwire-map-e-event      {map-e}?
      |  +--ro map-e-id            leafref
      |  +--ro invalid-entry-id*   leafref
      |  +--ro added-entry*        uint32
      |  +--ro modified-entry*     leafref
      +---n softwire-map-t-event      {map-t}?
         +--ro map-t-id            leafref
         +--ro invalid-entry-id*   leafref
         +--ro added-entry*        uint32
         +--ro modified-entry*     leafref
```

Figure 6: Softwire Notifications Data Model Structure

4. Softwire YANG Model

This module imports typedefs from [RFC6991].

```
<CODE BEGINS> file "ietf-softwire@2015-02-10.yang"
```

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

  import ietf-inet-types { prefix inet; }

  organization "Softwire Working Group";

  contact
    "
    Qi Sun sunqi@csnet1.cs.tsinghua.edu.cn
    Hao Wang wangh13@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 2015-02-10 {
    description
      "Add notifications.";
  }

  revision 2015-02-06 {
    description
      "Correct grammar errors; Reuse groupings; Update descriptions.";
  }

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

/*
 * Typedef
 */
```

```
/*
 * 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 CPEs.";
    reference
      "I-D.ietf-softwire-lw4over6";
  }

  feature lw4over6-aftr {
    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 lw4over6-b4 {
    if-feature lw4over6;
    description
      "The B4s (CPEs) 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 CPE. It sources IPv4 conncections 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 indepent binding. This
feature
      indicates the instance functions as a MAP-E instance.
      ";
    reference
      "I-D.ietf-softwire-map";
  }

  feature map-e-ce {
    if-feature map-e;
    description
      "Indicates the instance functions as a MAP-E CPE.";
      //Not sure if this is needed.
  }

  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";
  }

  feature map-t-ce {
    if-feature map-t;
    description
      "Indicates the instance functions as a MAP-T CPE.";
      //Not sure if this is needed.
  }


/*
 * 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 defaul 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.
      ";
    leaf binding-ipv6-addr {
      mandatory true;
      type inet:ipv6-address;
      description
        "The /128 IPv6 address of the lwB4, which is used to bind the
        IPv4 address and port-set and source the tunnel.
        ";
    }
    leaf binding-ipv6-prefix {
      mandatory true;
      type inet:ipv6-prefix;
      description
        "The operator-assigned IPv6 prefix of the lwB4. ";
    }
    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 NAPT44.
        ";
    }
    container 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.";
    }
  }

  grouping nat-table {
    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-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 differe mapping rule set.
      ";
    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 retrived 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.
        ";
    }
    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.
        ";
    }
  }

/*
 * Configuration Data Nodes
 */

  container softwire-config {
    description
      "The configuration data for Softwire instances. ";
    leaf enable {
      type boolean;
      default "true";
      description
        "Enable/disable the Softwire function.";
    }
    leaf description {
      type string;
      description
        "A textual description of Softwire.";
    }
```

```
    leaf tunnel-mtu {
      mandatory true;
      type uint32;
      description
        "The MTU for softwire tunnel.";
    }
    container lw4over6-aftr {
      if-feature lw4over6-aftr;
      description
        "Indicate this instance supports the lwAFTR function. The
        instances advertise the lw4over6-aftr feature through
        the capability exchange mechanism when a NETCONF session
        is established.";
      leaf enable {
        type boolean;
        description
          "Enable/disable the lwAFTR function.";
      }
      container lw4over6-aftr-instances {
        description
          "A set of lwAFTRs to be configured.";
        list lw4over6-aftr-instance {
          key "id";
          leaf id {
            type uint32;
          }
          leaf name {
            type string;
            description "The name for the lw4over6-aftr.";
          }
          leaf softwire-num-threshold {
            mandatory true;
            type uint32;
            description
              "The maximum number of tunnels that can be created on
              the lwAFTR.";
          }
          container binding-table {
            list binding-entries {
              key "binding-ipv6-addr";
              uses binding-entry;
              leaf lifetime {
                type uint32;
                units seconds;
                description
                  "The lifetime for the entry.";
              }
              leaf active {
```

```
                type boolean;
                default true;
                description
                  "Establish or tear down the tunnel.";
              }
            }
          }
        }
      }
    }
    container lw4over6-b4 {
      if-feature lw4over6-b4;
      description
        "Indicate this instance supports the lwB4 function. The instances
         advertise the lw4over6-b4 feature through the capability
        exchange mechanism when a NETCONF session is established.";
      leaf enable {
        type boolean;
        description
          "Enable/disable the lwB4 function.";
      }
      container lw4over6-b4-instances {
        description
          "A set of lwB4s to be configured.";
        list lw4over6-b4-instance {
          key "binding-ipv6-addr";
          leaf name {
            type string;
            description "The lw4over6-b4 name.";
          }
          leaf b4-ipv6-addr-format {
            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.";
          }
          uses binding-entry;
          leaf lifetime {
            type uint32;
            units seconds;
          }
          container nat-table {
            uses nat-table;
            description "To be extended.";
```

```
          }
        }
      }
    }
    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, including BRs and CPEs.";
        list map-e-instance {
          key "id";
          leaf id {
            type uint32;
          }
          leaf name {
            type string;
          }
          container map-rule-table {
            list map-rules {
              key "id";
              leaf id {
                type uint32;
              }
              uses map-rule;
            }
          }
          leaf br-ipv6-addr {
            //if-feature map-e-ce;
            type inet:ipv6-address;
            description
              "The IPv6 address of the MAP-E BR.";
          }
        }
      }
    }
    container 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, including BRs
          and CPEs.";
        list map-t-instance {
          key "id";
          leaf id {
            type uint32;
          }
          leaf name {
            type string;
          }
          container map-rule-table {
            list map-rules {
              key "id";
              leaf id {
                type uint8;
              }
              uses map-rule;
            }
          }
          leaf dmr-ipv6-prefix {
            //if-feature map-t-ce;
            type inet:ipv6-prefix;
            description
              "The IPv6 prefix of the MAP-T BR. ";
              //I think both the BR and CE should be configured with this
parameter for consistence.
          }
        }
      }
    }
  }

/*
 * Operational state Data Nodes
 */

  container softwire-state {
    config false;
    description
      "The operational state data for Softwire instances. ";
    leaf enable {
      type boolean;
      description
        "Status of the Softwire function.";
    }
    leaf description {
      type string;
      description
        "A textual description of the softwire instances.";
    }
    leaf tunnel-mtu {
      mandatory true;
      type uint32;
      description
        "The tunnel MTU for softwire instances.";
    }
    container lw4over6-aftr {
      if-feature lw4over6-aftr;
      config false;
      description
        "Indicate this instance supports the lwAFTR function. The instances
         advertise the lw4over6-aftr feature through the capability
        exchange mechanism when a NETCONF session is established.";
      leaf enable {
        type boolean;
        description
          "Status of the lwAFTR function.";
      }
      container lw4over6-aftr-instances {
        description
          "A set of lwAFTRs.";
        list lw4over6-aftr-instance {
          key "id";
          leaf id {
            type uint32;
          }
          leaf name {
            type string;
            description "The name for this lw4over6-aftr.";
          }
          leaf active-softwire-num {
            mandatory true;
            type uint32;
            description
              "The number of currently active tunnels on the lw4over6
instance.";
```

```
          }
          container binding-table {
            list binding-entries {
              key "binding-ipv6-addr";
              uses binding-entry;
              leaf active {
                type boolean;
                description
                  "Status of a specific tunnel.";
              }
            }
          }
        }
      }
    }
    container lw4over6-b4 {
      if-feature lw4over6-b4;
      config false;
      description
        "Indicate this instance supports the lwB4 function. The instances
         advertise the lw4over6-b4 feature through the capability
        exchange mechanism when a NETCONF session is established.";
      leaf enable {
        type boolean;
        description
          "Status of the lwB4 function.";
      }
      container lw4over6-b4-instances {
        description
          "A set of lwB4s.";
        list lw4over6-b4-instance {
          key "binding-ipv6-addr";
          leaf name {
            type string;
          }
          leaf b4-ipv6-addr-format {
            mandatory true;
            type boolean;
            description
              "The format of lwB4 IPv6 address. If the parameter is true,
              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 false, the lwB4 is using any /128 address from the
assigned
              IPv6 prefix.";
          }
          uses binding-entry;
          container nat-table {
            uses nat-table;
          }
```

```
        }
      }
    }
    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.";
      leaf enable {
        type boolean;
        description
          "Status of the MAP-E function.";
      }
      container map-e-instances {
        description
          "A set of MAP-E instances, including BRs and CPEs.";
        list map-e-instance {
          key "id";
          leaf id {
            type uint32;
          }
          container map-rule-table {
            list map-rules {
              key "id";
              leaf id {
                type string;
              }
              uses map-rule;
            }
          }
          leaf br-ipv6-addr {
            if-feature map-e-ce;
            mandatory true;
            type inet:ipv6-address;
            description
              "The IPv6 address of the MAP-E BR.";
              // Where should this be, inside or outside the list??
          }
        }
      }
    }
    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.";
      leaf enable {
        type boolean;
        description
          "Status of the MAP-T function.";
      }
      container map-t-instances {
        description
          "A set of the MAP-T instances, including BRs and CPEs.";
        list map-t-instance {
          key "id";
          leaf id {
            type uint32;
          }
          container map-rule-table {
            list map-rules {
              key "id";
              leaf id {
                type uint32;
              }
              uses map-rule;
            }
          }
          container map-t-ce {
            if-feature map-t-ce;
            leaf dmr-ipv6-prefix {
              type inet:ipv6-prefix;
              description
                "The IPv6 prefix of the DMR (default mapping rule).";
            }
          }
        }
      }
    }
  }
  /*
   * Notifications
   */
  notification softwire-lwaftr-event {
    if-feature lw4over6-aftr;
    leaf lwaftr-id {
      mandatory true;
      type leafref {
        path
          "/softwire-state/lw4over6-aftr/lw4over6-aftr-instances/"
          + "lw4over6-aftr-instance/id";
      }
```

```
     }
     leaf exceed-sw-num-limit {
       type boolean;
       default false;
     }
     leaf-list invalid-entry {
       type leafref {
         path
           "/softwire-state/lw4over6-aftr/lw4over6-aftr-instances/"
           + "lw4over6-aftr-instance[id=current()/../lwaftr-id]/"
           + "binding-table/binding-entries/binding-ipv6-addr";
       }
       description
         "Notify the client that a specific binding entry has been
         expired/invalid. The binding-ipv6-addr 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-state/lw4over6-aftr/lw4over6-aftr-instances/"
             + "lw4over6-aftr-instance[id=current()/../lwaftr-id]/"
             + "binding-table/binding-entries/binding-ipv6-addr";
         }
     }
   }
   notification softwire-lwb4-event {
     if-feature lw4over6-b4;
     leaf lwb4-binding-ipv6-addr-change {
       mandatory true;
       type inet:ipv6-address;
       description
         "The sourch 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-state/map-e/map-e-instances/map-e-instance/id";
      }
    }
    leaf-list invalid-entry-id {
      type leafref {
        path
          "/softwire-state/map-e/map-e-instances/"
          + "map-e-instance[id=current()/../map-e-id]/map-rule-table/"
          + "map-rules/id";
      }
    }
    leaf-list added-entry {
      type uint32;
    }
    leaf-list modified-entry {
      type leafref {
        path
          "/softwire-state/map-e/map-e-instances/"
          + "map-e-instance[id=current()/../map-e-id]/map-rule-table/"
          + "map-rules/id";
      }
    }
  }
  notification softwire-map-t-event {
    if-feature map-t;
    leaf map-t-id {
      mandatory true;
      type leafref {
        path
          "/softwire-state/map-t/map-t-instances/map-t-instance/id";
      }
    }
    leaf-list invalid-entry-id {
      type leafref {
        path
          "/softwire-state/map-t/map-t-instances/"
          + "map-t-instance[id=current()/../map-t-id]/map-rule-table/"
          + "map-rules/id";
      }
    }
    leaf-list added-entry {
      type uint32;
    }
    leaf-list modified-entry {
      type leafref {
```

```
        path
          "/softwire-state/map-t/map-t-instances/"
          + "map-t-instance[id=current()/../map-t-id]/map-rule-table/"
          + "map-rules/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:

```
    <rpc message-id="101" xmlns:nc="urn:params:xml:ns:yang:ietf-softwire:1.0">
  // replace with IANA namespace when assigned.
    <edit-config>
      <target>
        <running/>
      </target>
    <softwire-config>
      <lw4over6-aftr>
        <lw4over6-aftr-instances>
          <lw4over6-aftr-instance>
            <aftr-ipv6-addr>2001::2</aftr-ipv6-addr>
            <binding-table>
              <binding-entry>
                <binding-ipv4-addr>123.1.1.1</binding-ipv4-addr>
                <port-set>
                  <psid>1234</psid>
                </port-set>
                <binding-ipv6-addr>2001::1</binding-ipv6-addr>
                <active>1</active>
              </binding-entry>
            </binding-table>
          </lw4over6-aftr-instance>
        </lw4over6-aftr-instances>
      </lw4over6-aftr>
    </softwire-config>
```

Figure 7: Lw4over6 Binding-Table Configuration XML

6. Security Considerations (TBD)

TBD

7. IANA Considerations (TBD)

TBD

8. Acknowledgements

The authors would like to thank Lishan Li and Rajiv Asati for their contributions to this work.

9. References

9.1. Normative References

[I-D.ietf-softwire-lw4over6]
Cui, Y., Qiong, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the DS-Lite Architecture", draft-ietf-softwire-lw4over6-13 (work in progress), November 2014.

[I-D.ietf-softwire-map]
Troan, O., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, "Mapping of Address and Port with Encapsulation (MAP)", draft-ietf-softwire-map-12 (work in progress), November 2014.

[I-D.ietf-softwire-map-t]
Li, X., Bao, C., Dec, W., Troan, O., Matsushima, S., and T. Murakami, "Mapping of Address and Port using Translation (MAP-T)", draft-ietf-softwire-map-t-08 (work in progress), December 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.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 2011.

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

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

9.2. Informative References

[I-D.ietf-softwire-unified-cpe]
Boucadair, M., Farrer, I., Perreault, S., and S. Sivakumar, "Unified IPv4-in-IPv6 Softwire CPE", draft-ietf-softwire-unified-cpe-01 (work in progress), May 2013.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

[RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, February 2012.

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