Softwire Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

Q. Sun
H. Wang
Y. Cui
Tsinghua University
I. Farrer
S. Zoric
Deutsche Telekom AG
M. Boucadair
Orange
R. Asati
Cisco Systems, Inc.

July 8, 2016

A YANG Data Model for IPv4-in-IPv6 Softwires
draft-sun-softwire-yang-05

Abstract

This document defines a YANG data model for the configuration and operations (state, notification, RPC etc.) of IPv4-in-IPv6 Softwire Border Routers and Customer Premises Equipment. The model covers the 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

Copyright Notice

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 of A+P softwire functional elements: Border Routers (BRs) and Customer Premises Equipment (CEs).

This document defines a YANG data model [RFC6020] that can be used to configure and manage A+P Softwire elements using the NETCONF protocol [RFC6241]. DS-Lite YANG data model is defined in [I-D.boucadair-softwire-dslite-yang].

The Softwire YANG model is structured into two sub-models:

- o Lightweight 4over6 [RFC7596]
- o MAP-E [RFC7597] and MAP-T [RFC7599] (combined due to their common configuration parameters).

Two root containers are defined:

1. Container "softwire-config" holds the collection of YANG definitions common to all Softwire element configuration.
2. Container "softwire-state" holds YANG definitions for the operational state of the Softwire elements.

A NETCONF notify module is also included.

This approach has been taken so that the model can be easily extended to support additional Softwire mechanisms, if required.

1.1. Terminology

The reader should be familiar with the concepts and terms defined in [RFC7596], [RFC7597], [RFC7599], and the YANG data modelling language [RFC6020].

1.2. Tree Diagrams

The meaning of the symbols in these diagrams are as follows:

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

The model does not include CPE NAT-specific provisioning parameters that may be used for IPv4 address sharing other than the external IP address and port set which a softwire client may use for NAT44. NAT-specific considerations are out of scope of this document. A YANG model for the configuration and management of NAT gateways is described in [I-D.sivakumar-yang-nat].

2. Common

The following sections of the document are structured with the root of the Softwire YANG model (common to all mechanisms) described first. 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. This document defines a new module named "ietf-softwire" for Softwire data models such that this module aments "ietf-ipv6-unicast-routing" module that is defined in [I-D.ietf-netmod-routing-cfg].

3. Lightweight 4over6

Lightweight 4over6 (binding) includes two elements: lwAFTR (BR) and lwB4 (CE). The lwAFTR holds configuration for IPv4-IPv6 address bindings which are used for the forwarding of traffic originating from lwB4s.

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 NAT44.

4. MAP-E and MAP-T

MAP-E and MAP-T elements are provisioned with the MAP rules necessary for defining MAP domains and forwarding rules. For MAP-T CEs, an additional "ipv6-prefix" parameter is also included. Note that when referring to MAP-E/T (algorithm), the CE and BR shares the same model for configuration and management.

5. Softwire YANG Tree Diagrams

5.1. Common Tree Diagrams

Figure 1 describes the high level softwire YANG data model and the way tree is organized is common to all of the different softwire mechanisms listed in Section 1:

```
      +--rw softwire-config
      |  +--rw description?              string
      |  +--rw binding {binding}?
      |  |  +--rw br {br}?
      |  |  +--rw cr {cr}?
      |  +--rw algorithm {algorithm}?
      |
      +--ro softwire-state
         +--ro description?              string
         +--ro binding {binding}?
         |  +--ro br {br}?
         |  +--ro ce {ce}?
         +--ro algorithm {algorithm}?
```

Figure 1: High Level Softwire YANG Tree Organization

5.2. Lightweight 4over6 Tree Diagrams

Figure 2 defines the softwire data model for lw4o6 (softwire binding mode) which includes lwAFTR (BR) and lwB4 (CE):

```
module: ietf-softwire
   +--rw softwire-config
   |  +--...
   |  +--rw binding {binding}?
   |     +--rw br {br}?
   |     |  +--rw enable?                           boolean
   |     |  +--rw br-instances
   |     |     +--rw br-instance* [id]
   |     |        +--rw binding-table-versioning
   |     |        |  +--rw binding-table-version?  uint64
   |     |        |  +--rw binding-table-date?     yang:date-and-time
   |     |        +--rw id                         uint32
   |     |        +--rw name?                      string
   |     |        +--rw softwire-num-threshold     uint32
   |     |        +--rw tunnel-payload-mtu         uint16
   |     |        +--rw tunnel-path-mru            uint16
   |     |        +--rw binding-table
   |     |           +--rw binding-entry* [binding-ipv6info]
   |     |              +--rw binding-ipv6info     union
   |     |              +--rw binding-ipv4-addr    inet:ipv4-address
   |     |              +--rw port-set
   |     |              |  +--rw psid-offset       uint8
   |     |              |  +--rw psid-len          uint8
   |     |              |  +--rw psid              uint16
   |     |              +--rw br-ipv6-addr         inet:ipv6-address
   |     |              +--rw lifetime?            uint32
   |     +--rw ce {ce}?
```

```
   |        +--rw enable?                              boolean
   |        +--rw ce-instances
   |           +--rw ce-instance* [binding-ipv6info]
   |              +--rw name?                        string
   |              +--rw tunnel-payload-mtu           uint16
   |              +--rw tunnel-path-mru              uint16
   |              +--rw b4-ipv6-addr-format          boolean
   |              +--rw binding-ipv6info             union
   |              +--rw binding-ipv4-addr            inet:ipv4-address
   |              +--rw port-set
   |              |  +--rw psid-offset               uint8
   |              |  +--rw psid-len                  uint8
   |              |  +--rw psid                      uint16
   |              +--rw br-ipv6-addr                 inet:ipv6-address
   |              +--rw lifetime?                    uint32
   +--ro softwire-state
      +--...
      +--ro binding {binding}?
         +--ro br {br}?
         |  +--ro br-instances
         |     +--ro br-instance* [id]
         |        +--ro id                           uint32
         |        +--ro name?                        string
         |        +--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 active-softwire-num?         uint32
         |        +--ro binding-table
         |           +--ro binding-entry* [binding-ipv6info]
         |              +--ro binding-ipv6info     union
         |              +--ro active?              boolean
         +--ro ce {ce}?
            +--ro ce-instances
               +--ro ce-instance* [binding-ipv6info]
                  +--ro name?                        string
                  +--ro binding-ipv6info             union
                  +--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 2: Softwire Lightweight 4over6 Data Model Tree Structure

The data model assumes that each CE/BR instance can: be enable/disabled, be provisioned with a dedicated configuration data, and maintain its own binding table.

Additional information on some of the important lwAFTR nodes is provided below:

- binding-table-versioning: optionally used to add a incremental version number and/or timestamp to the binding table. This can be used for logging/data retention purposes. The version number is incremented and a new timestamp value written whenever a change is made to the contents of the binding table or a new binding table list is created.

- binding-entry: used to define the binding relationship between 3-tuples, which contains the lwB4's IPv6 address/prefix, the allocated IPv4 address and restricted port-set. For detail information, please refer to [RFC7596].

- tunnel-payload-mtu: used to set the IPv4 MTU for the lw4o6 tunnel.

- tunnel-path-mru: used to set the maximum lw4o6 IPv6 encapsulating packet size that can be received.

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

- psid-len: defines the number of ports that will be allocated for the softwire.

- psid: used to identify the set of ports allocated for a specific softwire.

- tunnel-num-threshold: used to set the maximum number of tunnels that can be created on the lw4o6 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.

Additional information on some of the important lwB4 nodes is provided below:

- 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 Section 6 of [RFC7597]; if set to false, the lwB4 can use any /128 address from the assigned IPv6 prefix.

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

5.3. MAP-E and MAP-T Tree Diagrams

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

```
 module: ietf-softwire
    +--rw softwire-config
    |  +--...
    |  +--rw algorithm {algorithm}?
    |     +--rw enable?                        boolean
    |     +--rw algorithm
    |        +--rw algo-instance* [id]
    |           +--rw algo-versioning
    |           |  +--rw algo-version?         uint64
    |           |  +--rw algo-date?            yang:date-and-time
    |           +--rw id                       uint32
    |           +--rw name?                    string
    |           +--rw data-plane               enumeration
    |           +--rw ea-len                   uint8
    |           +--rw rule-ipv6-prefix         inet:ipv6-prefix
    |           +--rw rule-ipv4-prefix         inet:ipv4-prefix
    |           +--rw forwarding               boolean
    |           +--rw psid-offset              uint8
    |           +--rw psid-len                 uint8
    |           +--rw tunnel-payload-mtu       uint16
    |           +--rw tunnel-path-mru          uint16
    |           +--rw br-ipv6-addr             inet:ipv6-address
    |           +--rw dmr-ipv6-addr            inet:ipv6-prefix
    +--ro softwire-state
       +--...
       +--ro algorithm {algorithm}?
          +--ro algo-instances
             +--ro algo-instance* [id]
                +--ro id                       int32
                +--ro name?                    string
                +--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 and MAP-T Data Model Structure

Additional information on some of the important MAP-E and MAP-T nodes is provided below:

- algo-versioning: optionally used to add a incremental version number and/or timestamp to the algorithm. This can be used for logging/data retention purposes. The version number is incremented and a new timestamp value written whenever a change is made to the algorithm or a new instance is created.

- forwarding: specifies whether the rule can be used as a Forward Mapping Rule (FMR). If not set, this rule is a Basic Mapping Rule (BMR) only and must not be used for forwarding. See Section 4.1 of [RFC7598].

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

- dmr-ipv6-prefix: defines the Default Mapping Rule (DMR) for MAP-T. This parameter is optional when configuring a MAP-T BR.

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

5.4. Notifications for Softwire YANG

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

```
module: ietf-softwire
notifications:
   +---n softwire-binding-br-event {binding,br}?
   |  +--ro br-id?             -> /softwire-state/binding/br/.../id
   |  +--ro invalid-entry*     -> /softwire-config/binding/br/.../binding-table/
binding-entry/binding-ipv6info
   |  +--ro added-entry*       inet:ipv6-address
   |  +--ro modified-entry*    -> /softwire-config/binding/br/.../binding-table/
binding-entry/binding-ipv6info
   +---n softwire-binding-ce-event {binding,ce}?
   |  +--ro ce-binding-ipv6-addr-change    inet:ipv6-address
   +---n softwire-algorithm-instance-event {algorithm}?
      +--ro algo-id            -> /softwire-config/algorithm/.../id
      +--ro invalid-entry*     -> /softwire-config/algorithm/.../id
      +--ro added-entry*       -> /softwire-config/algorithm/.../id
      +--ro modified-entry*    -> /softwire-config/algorithm/.../id
```

Figure 4: Softwire Notifications Data Model Structure

Additional information on some of the important notification nodes is listed below:

   o invalid-entry, added-entry, modified-entry: used to notify the
      client that a specific binding entry or MAP rule is expired or
      invalidated, added, or modified.

   o ce-binding-ipv6-addr-change: used to notify that the lwB4's
      binding-ipv6-address has been changed or the value of the
      'b4-ipv6-addr-format' is "False".

6. Softwire YANG Model

   This module imports typedefs from [RFC6991].

```
<CODE BEGINS> file "ietf-softwire@2016-06-04.yang"

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

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

  organization "Softwire Working Group";

  contact
    "
    Qi Sun <sunqi.ietf@gmail.com>
    Hao Wang <wangh13@mails.tsinghua.edu.cn>
    Yong Cui <yong@csnet1.cs.tsinghua.edu.cn>
    Ian <Farrer ian.farrer@telekom.de>
    Sladjana Zoric <sladjana.zoric@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 A+P Softwire Border Routers (BRs) and Customer
    Premises Equipment (CEs). It covers Lightweight 4over6,
    MAP-E and MAP-T mechanisms.

    Copyright (c) 2016 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 2016-06-04 {
    description
      "Version-05: Combined MAP-E/MAP-T into a single tree. Added binding
```

```
          table/alogorthm versioning";
       reference "-05";
  }

  revision 2015-09-30 {
    description
      "Version-04: Fix YANG syntax; Add flags to map-rule; Remove
      the map-rule-type element. ";
       reference "-04";
  }

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

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


  revision 2015-02-06 {
    description
      "Version-01: Correct grammar errors; Reuse groupings; Update
      descriptions.";
       reference "-01";
  }

  revision 2015-02-02 {
    description
      "Initial revision.";
       reference "-00";
  }

/*
 * Features
 */

  feature binding {
    description
    "Lightweight 4over6 (binding) 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.

     Besides lw4o6, this feature also covers MAP in 1:1 mode
     (offset=0, PSID explicit)";

   reference
      "RFC7596";
  }

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

  feature ce {
    if-feature binding;
    description
      "The B4 for Lightweight 4over6, so-called lwB4 (CE). This
      feature indicates that a instance functions as a lwB4 (CE). A
      lwB4 (ce) 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 algorithm {
    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.
      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-E or
      MAP-T instance.";
    reference
      "RFC7597 & RFC7599";
  }

/*
 * Grouping
 */

  grouping port-set {
    description
      "Use the PSID algorithm to represent a range of transport layer
      ports.";
    leaf psid-offset {
      type uint8 {
        range 0..16;
      }
      mandatory true;
      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-len {
      type uint8 {
        range 0..15;
      }
      mandatory true;
      description
        "The length of PSID, representing the sharing ratio for an
        IPv4 address.";
    }
    leaf psid {
      type uint16;
      mandatory true;
      description
        "Port Set Identifier (PSID) value, which identifies a set
        of ports algorithmically.";
    }
  }

  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-ipv6info {
          type union {
            type inet:ipv6-address;
            type inet:ipv6-prefix;
          }
          mandatory true;
          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 RFC7596;
             if it's an IPv6 address, it means the lwB4 uses
             any /128 address from the assigned IPv6 prefix.
             ";
        }
        leaf binding-ipv4-addr {
          type inet:ipv4-address;
          mandatory true;
          description
            "The IPv4 address assigned to the lwB4, which is
             used as the IPv4 external address
             for lwB4 local NAPT44.";
        }
        container port-set {
          description
            "For Lightweight 4over6, the default value
            of offset should be 0, to configure one contiguous
            port range.";
          uses port-set {
            refine psid-offset {
              default "0";
            }
          }
        }
        leaf br-ipv6-addr {
          type inet:ipv6-address;
          mandatory true;
          description
            "The IPv6 address for lwaftr.";
        }
        leaf lifetime {
          type uint32;
          units seconds;
          description "The lifetime 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.";
  }
*/

  grouping traffic-stat {
    description "Traffic statistics";
    leaf sentPacket {
      type yang:zero-based-counter64;
      description "Number of packets sent.";
    }
    leaf sentByte {
      type yang:zero-based-counter64;
      description "Traffic sent, in bytes";
    }
    leaf rcvdPacket {
      type yang:zero-based-counter64;
      description "Number of packets received.";
    }
    leaf rcvdByte {
      type yang:zero-based-counter64;
      description "Traffic received, in bytes";
    }
    leaf droppedPacket {
      type yang:zero-based-counter64;
      description "Number of packets dropped.";
    }
    leaf droppedByte {
      type yang:zero-based-counter64;
      description "Traffic dropped, in bytes";
    }
  }


/*
 * 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 binding {
      if-feature binding;
      description
        "lw4over6 (binding) configuration.";
      container br {
        if-feature br;
        description
          "Indicate this instance supports the lwAFTR (BR) function.
          The instances advertise the BR feature through the
          capability exchange mechanism when a NETCONF session is
          established.";
        leaf enable {
          type boolean;
          description
            "Enable/disable the lwAFTR (BR) function.";
        }
        container br-instances {
          description
            "A set of BRs to be configured.";
          list br-instance {
            key "id";
            description
            "A set of lwAFTRs to be configured.";
            container binding-table-version {
              description "binding table's version";
              leaf binding-table-version{
                type uint64;
                description "Incremental version number
                to the binding table";
              }
              leaf binding-table-date {
                type yang:date-and-time;
                description "Timestamp to the binding
                table";
              }
            }
            leaf id {
              type uint32;
              mandatory true;
              description "An instance identifier.";
```

```
            }
            leaf name {
              type string;
              description "The name for the lwaftr.";
            }
            leaf softwire-num-threshold {
              type uint32;
              mandatory true;
              description
                "The maximum number of tunnels that can be created on
                the lwAFTR.";
            }
            leaf tunnel-payload-mtu {
              type uint16;
              mandatory true;
              description
                "The payload MTU for Lightweight 4over6 tunnel.";
            }
            leaf tunnel-path-mru {
              type uint16;
              mandatory true;
              description
                "The path MRU for Lightweight 4over6 tunnel.";
            }
            container binding-table {
              description "binding table";
              list binding-entry {
                key "binding-ipv6info";
                description "binding entry";
                uses binding-entry;
              }
            }
          }
        }
      }

      container ce {
        if-feature ce;
        description
          "Indicate this instance supports the lwB4 (CE) function.
          The instances advertise the CE feature through the
          capability exchange mechanism when a NETCONF session is
          established.";
        leaf enable {
          type boolean;
          description
            "Enable/disable the lwB4 (CE) function.";
        }
```

```
        container ce-instances {
          description
            "A set of CEs to be configured.";
          list ce-instance {
            key "binding-ipv6info";
            description "instances for CE";
            leaf name {
              type string;
              description "The CE's name.";
            }
            leaf tunnel-payload-mtu {
              type uint16;
              mandatory true;
              description
                "The payload MTU for Lightweight 4over6 tunnel.";
            }
            leaf tunnel-path-mru {
              type uint16;
              mandatory true;
              description
                "The path MRU for Lightweight 4over6 tunnel.";
            }
            leaf b4-ipv6-addr-format {
              type boolean;
              mandatory true;
              description
               "The format of lwB4 (CE) IPv6 address. If set to true,
               it indicates that the IPv6 source address of the lwB4
               is constructed according to the description in
               [RFC7596]; if set to false, the lwB4 (CE)
               can use any /128 address from the assigned IPv6
               prefix.";
            }
            uses binding-entry;
          }
        }
      }
    }

    container algorithm {
      if-feature algorithm;
      description
        "Indicate the instances support the MAP-E and MAP-T function.
        The instances advertise the map-e feature through the
        capability exchange mechanism when a NETCONF session is
        established.";
      leaf enable {
        type boolean;
```

```
        description
          "Enable/disable the MAP-E or MAP-T function.";
      }
      container algo-instances {
        description
          "A set of MAP-E or MAP-T instances to be configured,
           applying to BRs and CEs. A MAP-E/T instance defines a MAP
           domain comprising one or more MAP-CE and MAP-BR";
        list algo-instance {
          key "id";
          description "instance for MAP-E/MAP-T";
          container algo-versioning {
            description "algorithm's version";
            leaf algo-version {
              type uint64;
              description "Incremental version number to
              the algorithm";
            }
            leaf algo-date {
              type yang:date-and-time;
              description "Timestamp to the algorithm";
            }
          }
          leaf id {
            type uint32;
            mandatory true;
            description "Algorithm Instance ID";
          }
          leaf name {
            type string;
            description "The name for the instance.";
          }
          leaf data-plane {
            type enumeration {
              enum "encapsulation" {
                description "encapsulation for MAP-E";
              }
              enum "translation" {
                description "translation for MAP-T";
              }
            }
            description
              "Encapsulation is for MAP-E while translation is
              for MAP-T";
          }
          leaf ea-len {
            type uint8;
            mandatory true;
```

```
            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.";
          }
          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.";
          }
          leaf forwarding {
            type boolean;
            mandatory true;
            description
              "This parameter specifies whether the rule may be used for
              forwarding (FMR). If set, this rule is used as an FMR;
              if not set, this rule is a BMR only and must not be used
              for forwarding.";
          }
          leaf psid-offset {
            type uint8 {
              range 0..16;
            }
            mandatory true;
            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-len {
            type uint8 {
              range 0..15;
            }
            mandatory true;
            description
```

```
              "The length of PSID, representing the sharing ratio for an
              IPv4 address.";
          }
          leaf tunnel-payload-mtu {
            type uint16;
            description
              "The payload MTU for MAP-E tunnel.";
          }
          leaf tunnel-path-mru {
            type uint16;
            description
              "The path MRU for MAP-E tunnel.";
          }
          leaf br-ipv6-addr {
            type inet:ipv6-address;
            mandatory true;
            description
              "The IPv6 address of the MAP-E BR.";
          }
          leaf dmr-ipv6-prefix {
            type inet:ipv6-prefix;
            description
              "The IPv6 prefix of the MAP-T BR. ";
          }
        }
      }
    }
  }

/*
 * Operational state Data Nodes
 */

  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 binding {
      if-feature binding;
      description
        "lw4over6 (binding) state.";
      container br {
        if-feature br;
```

```
        config false;
        description
          "Indicate this instance supports the lwAFTR (BR) function.
          The instances advertise the lwaftr (BR) feature through the
          capability exchange mechanism when a NETCONF session is
          established.";
        container br-instances {
          description
            "A set of BRs.";
          list br-instance {
            key "id";
            description "instances for BR";
            leaf id {
              type uint32;
              mandatory true;
              description "id";
            }
            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 (binding) instance.";
            }
            container binding-table {
              description "id";
              list binding-entry {
                key "binding-ipv6info";
                description "An identifier of the binding entry.";
                leaf binding-ipv6info {
                  type union {
                    type inet:ipv6-address;
                    type inet:ipv6-prefix;
                  }
                  mandatory true;
                  description
                    "The IPv6 information used to identify
                      a binding entry. ";
                }
                leaf active {
                  type boolean;
                  description
                    "Status of a specific tunnel.";
                }
```

```
              }
            }
          }
        }
      }


      container ce {
        if-feature ce;
        config false;
        description
          "Indicate this instance supports the lwB4 (CE) function.
          The instances advertise the lwb4 (CE) feature through the
          capability exchange mechanism when a NETCONF session is
          established.";
        container ce-instances {
          description
            "Status of the configured CEs.";
          list ce-instance {
            key "binding-ipv6info";
            description "a lwB4 (CE) instance.";
            leaf name {
              type string;
              description "The CE's name.";
            }
            leaf binding-ipv6info {
              type union {
                type inet:ipv6-address;
                type inet:ipv6-prefix;
              }
              mandatory true;
              description
                "The IPv6 information used to identify
                 a binding entry. ";
            }
            uses traffic-stat;
          }
        }
      }
    }

    container algorithm {
      if-feature algorithm;
      config false;
      description
        "Indicate the instances support the MAP-E and MAP-T function.
        The instances advertise the map-e/map-t feature through the
        capability exchange mechanism when a NETCONF session is
```

```
          established.";
        container algo-instances {
          description
            "Status of MAP-E instance(s).";
          list algo-instance {
            key "id";
            description "Instances for algorithm";
            leaf id {
              type uint32;
              mandatory true;
              description "id";
            }
            leaf name {
              type string;
              description "The map-e instance name.";

            }
            uses traffic-stat;
          }
        }
      }
    }

  /*
   * Notifications
   */
    notification softwire-br-event {
      if-feature binding;
      if-feature br;
      description "Notification for BR.";

      leaf br-id {
        type leafref {
          path
            "/softwire-state/binding/br/br-instances/"
            + "br-instance/id";
        }
            description "...";
      }
      leaf-list invalid-entry {
        type leafref {
          path
            "/softwire-config/binding/br/br-instances/"
            + "br-instance[id=current()/../br-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/binding/br/br-instances/"
            + "br-instance[id=current()/../br-id]/"
            + "binding-table/binding-entry/binding-ipv6info";
        }
          description "...";
    }
  }

  notification softwire-ce-event {
    if-feature binding;
    if-feature ce;
    description "CE notification";
    leaf ce-binding-ipv6-addr-change {
      type inet:ipv6-address;
      mandatory true;
      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-algorithm-instance-event {
    if-feature algorithm;
    description "Notifications for MAP-E or MAP-T.";
    leaf algo-id {
      type leafref {
        path
          "/softwire-config/algorithm/algo-instances/algo-instance/id";
      }
      mandatory true;
      description "MAP-E or MAP-T event.";
    }
```

```
    leaf-list invalid-entry-id {
      type leafref {
        path
          "/softwire-config/algorithm/algo-instances/algo-instance/id";
      }
      description "Invalid entry event.";
    }
    leaf-list added-entry {
      type leafref {
        path
          "/softwire-config/algorithm/algo-instances/algo-instance/id";
      }
      description "Added entry.";
    }
    leaf-list modified-entry {
      type leafref {
        path
          "/softwire-config/algorithm/algo-instances/algo-instance/id";
      }
      description "Modified entry.";
    }
  }
}
<CODE ENDS>
```

7. Example of Configure lw4o6 Binding-Table

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

o IPv6 Address for a single lwB4

o Public IPv4 Address

o 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.

Let's consider an example to add an entry that maintains the relationship between 3-tuples of lwB4 (2001:db8::1), '192.0.2.1' and '1234' in the binding table of the lwAFTR (2001:db8::2). 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>
       <lw4o6-aftr>
         <lw4o6-aftr-instances>
           <lw4o6-aftr-instance>
             <aftr-ipv6-addr>2001:db8::2</aftr-ipv6-addr>
             <binding-table>
               <binding-entry>
                 <binding-ipv4-addr>192.0.2.1</binding-ipv4-addr>
                 <port-set>
                   <psid>1234</psid>
                 </port-set>
                 <binding-ipv6-addr>2001:db8::1</binding-ipv6-addr>
                 <active>1</active>
               </binding-entry>
             </binding-table>
           </lw4o6-aftr-instance>
         </lw4o6-aftr-instances>
       </lw4o6-aftr>
     </softwire-config>
```

Figure 5: lw4o6 Binding-Table Configuration XML

8. 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.

All data nodes defined in the YANG module which can be created, modified and deleted (i.e., config true, which is the default). These data nodes are considered sensitive. Write operations (e.g., edit-config) applied to these data nodes without proper protection can negatively affect network operations.

9. IANA Considerations

This document requests IANA to register the following URI in the "IETF XML Registry" [RFC3688].

```
URI: urn:ietf:params:xml:ns:yang:softwire
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.
```

This document requests IANA to register the following YANG module in the "YANG Module Names" registry [RFC6020].

```
name: ietf-dslite-aftr
namespace: urn:ietf:params:xml:ns:yang:softwire
prefix: softwire
reference: RFC XXXX
```

10. Acknowledgements

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

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC7596] Cui, Y., Sun, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the Dual-Stack Lite Architecture", RFC 7596, DOI 10.17487/RFC7596, July 2015, <http://www.rfc-editor.org/info/rfc7596>.

[RFC7597] Troan, O., Ed., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, Ed., "Mapping of Address and Port with Encapsulation (MAP-E)", RFC 7597, DOI 10.17487/RFC7597, July 2015, <http://www.rfc-editor.org/info/rfc7597>.

[RFC7598] Mrugalski, T., Troan, O., Farrer, I., Perreault, S., Dec, W., Bao, C., Yeh, L., and X. Deng, "DHCPv6 Options for Configuration of Softwire Address and Port-Mapped Clients", RFC 7598, DOI 10.17487/RFC7598, July 2015, <http://www.rfc-editor.org/info/rfc7598>.

[RFC7599] Li, X., Bao, C., Dec, W., Ed., Troan, O., Matsushima, S., and T. Murakami, "Mapping of Address and Port using Translation (MAP-T)", RFC 7599, DOI 10.17487/RFC7599, July 2015, <http://www.rfc-editor.org/info/rfc7599>.

11.2. Informative References

[I-D.boucadair-softwire-dslite-yang]
Boucadair, M., Jacquenet, C., and S. Sivakumar, "YANG Data Model for the DS-Lite Address Family Transition Router (AFTR)", draft-boucadair-softwire-dslite-yang-04 (work in progress), June 2016.

[I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-22 (work in progress), July 2016.

[I-D.sivakumar-yang-nat]
Sivakumar, S., Boucadair, M., and S. <>, "YANG Data Model for Network Address Translation (NAT)", draft-sivakumar-yang-nat-04 (work in progress), March 2016.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

Authors' Addresses

Qi Sun
Tsinghua University
Beijing 100084
P.R. China

Phone: +86-10-6278-5822
Email: sunqi.ietf@gmail.com

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

Sladjana Zoric
Deutsche Telekom AG
CTO-IPT,Landgrabenweg 151
Bonn, NRW 53227
Germany

Email: sladjana.zoric@telekom.de

Mohamed Boucadair
Orange
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