Softwire Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

Q. Sun
H. Wang
Y. Cui
Tsinghua University
I. Farrer
S. Zechlin
Deutsche Telekom AG
M. Boucadair
Orange
R. Asati
Cisco Systems, Inc.
October 30, 2017

A YANG Data Model for IPv4-in-IPv6 Address plus Port Softwires
draft-ietf-softwire-yang-02

Abstract

This document defines YANG data models for the configuration and operation of IPv4-in-IPv6 softwire Border Relays 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

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 https://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 May 3, 2018.

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). The softwire mechanisims covered in this document are Lightweight 4 over 6 [RFC7596], MAP-E [RFC7597], and MAP-T [RFC7599].

This document defines YANG data models [RFC6020] that can be used to configure and manage A+P softwire elements using the NETCONF protocol [RFC6241] for:

Configuration

Operational State

Notifications

1.1. Terminology

The reader should be familiar with the concepts and terms defined in [RFC7596], [RFC7597], [RFC7599], and the YANG data modelling language defined in [RFC6020] and [RFC7950].

The meaning of the symbols in tree diagrams is defined in [I-D.ietf-netmod-yang-tree-diagrams].

2. Overview of the Models

The document defines these two YANG data modules for the configuration and monitoring of softwire functional elements:

ietf-softwire-ce	Provides configuration and monititoring for softwire CE element.
ietf-softwire-br	Provides configuration and monititoring for softwire BR element.

In addition, the following module is also defined:

ietf-softwire-common	Contains groups of functions that are common and are imported into the CE and BR modules.

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

The modules are defined as augments to the interface YANG module [RFC7223].

Within the BR and CE modules, the YANG "feature" statement is used to distinguish which of the different softwire mechanism(s) is relevant

for a specific element's configuraiton. For each module, a choice statement is included for either 'binding' or 'algorithmic'. Binding is used for configuring Lightweight 4over6, whereas Algorithmic is used for configuring MAP-T or MAP-E.

In the 'algo-instances' container, a choice statement is included to specify MAP-E (encapsulation) or MAP-T (translation). The following table shows the how these choices are used to indicate the desired softwire mechanism:

S46 Mechanism	ce-type?	data-plane?
Lightweight 4over6	binding	n/a
MAP-E	algorithm	encapsulation
MAP-T	algorithm	translation

Table 1: Softwire Mechanism Choice Statement Enumeration

NETCONF notifications also included.

Note: Earlier versions of this document combined the softwire mechanisms by their associated technologies rather than their function in the architecture. As the document was revised, it became apparent that dividing the modules by by their role in the architecture (CE or BR) was a better approach as this follows the intended function and existing implmemention approaches more closely.

2.1. Additional Configuration

The softwire module only aims to provide configuration relevant for softwires. In order to fully specify a CE element, the following may also be necessary:

- IPv6 routing configuration, to enable CE to obtain one or more IPv6 prefixes for softwire usage. YANG model for routing management is described in [RFC8022]

- IPv4 routing configuration, to add one or more IPv4 destination prefix(es) reachable via the configured softwire. YANG model for routing management is described in [RFC8022]

- Stateful NAT44 / NAPT management, to optionally specify a port set (PSID) along with its length. YANG model for NAT management is described in [I-D.ietf-opsawg-nat-yang] Note that

- Stateless NAT46 management, required by softwire translation based mechanisms (i.e. the assignment of a Network-Specific prefix to use for IPv4/IPv6 translation). YANG model for NAT management is described in [I-D.ietf-opsawg-nat-yang]

As YANG modules for the configuration for these functions are already defined in other documents, they are not repeated, but imported here, as needed. Appendix A.3 provides XML examples of how these models can be used together.

The CE must already have minimal IPv6 configuration in place so it is reachable by the Netconf client to obtain softwire configuration. If additional IPv6 specific configuration is necessary, the YANG models defined in [RFC7277] and [RFC8022] may be used.

3. Softwire YANG CE Tree Diagram

Figure 1 describes the softwire YANG module for CE elements. This module augments "ietf-interfaces", defined in [RFC7223] with an entry for the softwire. This entry can be referenced to configure IPv4 routing for the element.

The module provides configuration and monitoring for all of the softwire mechanisms listed in Section 1.

```
module: ietf-softwire-ce
  augment /if:interfaces/if:interface:
    +--rw softwire-payload-mtu?   uint16
    +--rw softwire-path-mru?      uint16
    +--rw (ce-type)?
       +--:(binding) {binding}?
       |  +--rw binding-ipv6info?       union
       |  +--rw br-ipv6-addr?           inet:ipv6-address
       +--:(algorithm) {algorithm}?
          +--rw algo-instances
             +--rw algo-instance* [id]
                +--rw enable?              boolean
                +--rw algo-versioning
                |  +--rw version?   uint64
                |  +--rw date?      yang:date-and-time
                +--rw id                   uint32
                +--rw name?                string
                +--rw (data-plane)?
                |  +--:(encapsulation)
                |  |  +--rw br-ipv6-addr        inet:ipv6-address
                |  +--:(translation)
                |     +--rw dmr-ipv6-prefix?    inet:ipv6-prefix
                +--rw ea-len               uint8
```

```
               +--rw rule-ipv6-prefix    inet:ipv6-prefix
               +--rw rule-ipv4-prefix    inet:ipv4-prefix
               +--rw forwarding          boolean
  augment /if:interfaces-state/if:interface:
    +--ro ce-interface
       +--ro name?                                  string
       +--ro type?                                  identityref
       +--ro sent-ipv4-packet?                      yang:zero-based-counter64
       +--ro sent-ipv4-byte?                        yang:zero-based-counter64
       +--ro sent-ipv6-packet?                      yang:zero-based-counter64
       +--ro sent-ipv6-byte?                        yang:zero-based-counter64
       +--ro rcvd-ipv4-packet?                      yang:zero-based-counter64
       +--ro rcvd-ipv4-byte?                        yang:zero-based-counter64
       +--ro rcvd-ipv6-packet?                      yang:zero-based-counter64
       +--ro rcvd-ipv6-byte?                        yang:zero-based-counter64
       +--ro dropped-ipv4-packet?                   yang:zero-based-counter64
       +--ro dropped-ipv4-byte?                     yang:zero-based-counter64
       +--ro dropped-ipv6-packet?                   yang:zero-based-counter64
       +--ro dropped-ipv6-byte?                     yang:zero-based-counter64
       +--ro dropped-ipv4-fragments?                yang:zero-based-counter64
       +--ro dropped-ipv4-bytes?                    yang:zero-based-counter64
       +--ro ipv6-fragments-reassembled?            yang:zero-based-counter64
       +--ro ipv6-fragments-bytes-reassembled?      yang:zero-based-counter64
       +--ro out-icmpv4-error-packets?              yang:zero-based-counter64
       +--ro out-icmpv6-error-packets?              yang:zero-based-counter64

  notifications:
    +---n softwire-ce-event {binding}?
       +--ro ce-binding-ipv6-addr-change    inet:ipv6-address
```

Figure 1: Softwire YANG CE Tree Diagram

3.1. Softwire CE Tree Diagram Descriptions

Additional information on some of the important CE elements is provided below:

- softwire-payload-mtu: optionally used to set the IPv4 MTU for the softwire. Needed if the softwire implementation is unable to correctly calculate the correct IPv4 size automatically.

- softwire-path-mru: optionally used to set the maximum IPv6 softwire packet size that can be received, including the encapsulation/translation overhead. Needed if the softwire implementation is unable to correctly calculate the correct IPv4 size automatically.

- ce-type: provides a choice statement allowing the binding or

algorithmic softwire mechanisms to be selected.

Additional details relevant to binding softwire elements are:

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

- br-ipv6-addr: indicates the IPv6 of the remote BR.

Additional details relevant to some of the important algorithmic elements are 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.

- data-plane: provides a choice statement for either encapsulation (MAP-E) or translation (MAP-T).

- br-ipv6-addr: defines the IPv6 address of the BR for MAP-E.

- dmr-ipv6-prefix: defines the Default Mapping Rule (DMR) IPv6 prefix of the BR for MAP-T.

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

Additional information on the notification node is listed below:

- ce-binding-ipv6-addr-change: if the CE's binding-ipv6-address changes for any reason, it SHOULD notify the NETCONF client.

4. Softwire BR YANG Tree Diagram

4.1. BR Tree Diagrams

Figure 2 describes the high level softwire YANG module for BRs. The module provides configuration and monitoring for all of the softwire mechanisms listed in Section 1.

```
module: ietf-softwire-br
    +--rw br-instances
       +--rw (br-type)?
          +--:(binding) {binding}?
          |  +--rw binding {binding}?
          |     +--rw br-instance* [id]
          |        +--rw binding-table-versioning
          |        |  +--rw version?   uint64
          |        |  +--rw date?      yang:date-and-time
          |        +--rw id                          uint32
          |        +--rw name?                       string
          |        +--rw softwire-num-threshold      uint32
          |        +--rw softwires-payload-mtu       uint16
          |        +--rw softwire-path-mru           uint16
          |        +--rw enable-hairpinning?         boolean
          |        +--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 icmp-policy
          |        |  +--rw icmpv4-errors
          |        |  |  +--rw allow-incoming-icmpv4?    boolean
          |        |  |  +--rw generate-icmpv4-errors?   boolean
          |        |  +--rw icmpv6-errors
          |        |     +--rw generate-icmpv6-errors?   boolean
          |        |     +--rw icmpv6-errors-rate?       uint16
          |        +--ro traffic-stat
          |        |  +--ro sent-ipv4-packet?
          |        |  |       yang:zero-based-counter64
          |        |  +--ro sent-ipv4-byte?
          |        |  |       yang:zero-based-counter64
          |        |  +--ro sent-ipv6-packet?
          |        |  |       yang:zero-based-counter64
          |        |  +--ro sent-ipv6-byte?
          |        |  |       yang:zero-based-counter64
          |        |  +--ro rcvd-ipv4-packet?
          |        |  |       yang:zero-based-counter64
```

```
|       |  +--ro rcvd-ipv4-byte?
|       |  |       yang:zero-based-counter64
|       |  +--ro rcvd-ipv6-packet?
|       |  |       yang:zero-based-counter64
|       |  +--ro rcvd-ipv6-byte?
|       |  |       yang:zero-based-counter64
|       |  +--ro dropped-ipv4-packet?
|       |  |       yang:zero-based-counter64
|       |  +--ro dropped-ipv4-byte?
|       |  |       yang:zero-based-counter64
|       |  +--ro dropped-ipv6-packet?
|       |  |       yang:zero-based-counter64
|       |  +--ro dropped-ipv6-byte?
|       |  |       yang:zero-based-counter64
|       |  +--ro dropped-ipv4-fragments?
|       |  |       yang:zero-based-counter64
|       |  +--ro dropped-ipv4-bytes?
|       |  |       yang:zero-based-counter64
|       |  +--ro ipv6-fragments-reassembled?
|       |  |       yang:zero-based-counter64
|       |  +--ro ipv6-fragments-bytes-reassembled?
|       |  |       yang:zero-based-counter64
|       |  +--ro out-icmpv4-error-packets?
|       |  |       yang:zero-based-counter64
|       |  +--ro out-icmpv6-error-packets?
|       |          yang:zero-based-counter64
|       +--rw hairpin-ipv4-bytes?         yang:zero-based-counter64
|       +--rw hairpin-ipv4-packets?       yang:zero-based-counter64
|       +--ro active-softwire-num?        uint32
+--:(algorithm) {algorithm}?
   +--rw algorithm {algorithm}?
      +--rw algo-instance* [id]
         +--rw id                uint32
         +--rw name?             string
         +--rw algo-instances
         |  +--rw algo-instance* [id]
         |     +--rw enable?             boolean
         |     +--rw algo-versioning
         |     |  +--rw version?   uint64
         |     |  +--rw date?      yang:date-and-time
         |     +--rw id                  uint32
         |     +--rw name?               string
         |     +--rw (data-plane)?
         |     |  +--:(encapsulation)
         |     |  |  +--rw br-ipv6-addr        inet:ipv6-address
         |     |  +--:(translation)
         |     |     +--rw dmr-ipv6-prefix?    inet:ipv6-prefix
         |     +--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 traffic-stat
                       +--rw sent-ipv4-packet?
                       |       yang:zero-based-counter64
                       +--rw sent-ipv4-byte?
                       |       yang:zero-based-counter64
                       +--rw sent-ipv6-packet?
                       |       yang:zero-based-counter64
                       +--rw sent-ipv6-byte?
                       |       yang:zero-based-counter64
                       +--rw rcvd-ipv4-packet?
                       |       yang:zero-based-counter64
                       +--rw rcvd-ipv4-byte?
                       |       yang:zero-based-counter64
                       +--rw rcvd-ipv6-packet?
                       |       yang:zero-based-counter64
                       +--ro rcvd-ipv6-byte?
                       |       yang:zero-based-counter64
                       +--rw dropped-ipv4-packet?
                       |       yang:zero-based-counter64
                       +--rw dropped-ipv4-byte?
                       |       yang:zero-based-counter64
                       +--rw dropped-ipv6-packet?
                       |       yang:zero-based-counter64
                       +--rw dropped-ipv6-byte?
                       |       yang:zero-based-counter64
                       +--rw dropped-ipv4-fragments?
                       |       yang:zero-based-counter64
                       +--rw dropped-ipv4-bytes?
                       |       yang:zero-based-counter64
                       +--rw ipv6-fragments-reassembled?
                       |       yang:zero-based-counter64
                       +--rw ipv6-fragments-bytes-reassembled?
                       |       yang:zero-based-counter64
                       +--rw out-icmpv4-error-packets?
                       |       yang:zero-based-counter64
                       +--rw out-icmpv6-error-packets?
                               yang:zero-based-counter64

  notifications:
    +---n softwire-br-event {binding}?
    |  +--ro br-id?            -> /br-instances/binding/br-instance/id
    |  +--ro invalid-entry*    leafref
    |  +--ro added-entry*      inet:ipv6-address
    |  +--ro modified-entry*   leafref
```

```
    +---n softwire-algorithm-instance-event {algorithm}?
       +--ro algo-id             -> /br-instances/algorithm/algo-instance/id
       +--ro invalid-entry-id*   -> /br-instances/algorithm/algo-instance/id
       +--ro added-entry*        -> /br-instances/algorithm/algo-instance/id
       +--ro modified-entry*     -> /br-instances/algorithm/algo-instance/id
```

Figure 2: Softwire YANG BR Tree

4.2. Softwire BR Tree Diagram Descriptions

The descriptions for some of the leaves in the BR module are the same as in Figure 1. Information on the additional elements are provided below:

- binding-table-versioning: optionally used to add an 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].

- softwire-num-threshold: used to set the maximum number of softwires that can be created on the lw4o6 element simultaneously.

- active-softwire-num (ro): used to present the number of softwires currently provisioned on the element.

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

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

- 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.

5. Softwire CE YANG Model

This module imports typedefs from [RFC6991].

```
  <CODE BEGINS>   file "ietf-softwire-ce@2017-10-19.yang"
module ietf-softwire-ce {
  yang-version 1.1;
```

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

  import ietf-inet-types {prefix inet; }
  import ietf-interfaces {prefix if; }
  import iana-if-type {prefix ianaift; }
  import ietf-softwire-common {prefix softwire-common; }

  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 module for the configuration and
    management of A+P Softwire Customer Premises Equipment (CEs). It
    covers Lightweight 4over6, MAP-E, and MAP-T mechanisms.

    Copyright (c) 2017 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 2017-10-19 {
    description
      "Initial version of standalone CE model, with updates for
      importing groups from ietf-softwire-common and
      augmenting ietf-interfaces.";
    reference "-02";
  }

  /*
   * Features
   */

  feature binding {
    description
      "Binding is used for configuring Lightweight 4over6 mechanism.

      Binding softwire mechanisms are IPv4-over-IPv6 tunnelling transition
```

```
      mechanisms 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 Border Relay (BR) and a
      hub-and-spoke forwarding architecture. In order to delegate the
      NAPT function and achieve IPv4 address sharing, port-restricted
      IPv4 addresses needs to be allocated to CEs.";

    reference
      "RFC7596, RFC7597 & RFC7599";
  }

  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.

      MAP-T is an IPv6 transition mechanism for transporting IPv4 packets
      across an IPv6 network using IP translation. It leverages a double
      stateless NAT64 based solution as well as the stateless
      algorithmic address & transport layer port mapping algorithm defined
      for MAP-E.

      This feature indicates the instance functions as a MAP-E or
      MAP-T instance.";

    reference
      "RFC7597 & RFC7599";
  }

  // Binding Entry
  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;
      }
```

```
      description
        "The IPv6 information for a binding entry.
        If this is type IPv6 prefix, it indicates that
        the IPv6 source address of the CE is constructed
        according to the description in RFC7596;
        if it is type IPv6 address, it means the CE uses
        any valid /128 address from a prefix assigned to
        the CE.";
    }

    leaf br-ipv6-addr {
      type inet:ipv6-address;
      description
        "The IPv6 address for lwaftr.";
    }
  }

  // configuration parameters for CE softwire interface
  augment "/if:interfaces/if:interface" {
    when "if:type = 'ianaift:tunnel'";
    description "CE Softwire interface configuration";

    leaf softwire-payload-mtu {
      type uint16;
      units bytes;
      description
        "The payload MTU for the Softwire tunnel.";
    }

    leaf softwire-path-mru {
      type uint16;
      units bytes;
      description
        "The path MRU for the softwire (payload + encapsulation overhead).";
    }

    choice ce-type {
      description "Sets the CE softwire mechanism";

      case binding {
        if-feature binding;
        description "CE binding configuration";
        uses binding-entry;
      }

      case algorithm {
        if-feature algorithm;
        description "CE algorithm configuration";
```

```
        uses softwire-common:algorithm;
      }
    }
  }

  // operational state parameters for CE softwire binding interface
  augment "/if:interfaces-state/if:interface" {
    when "if:type = 'ianaift:tunnel'";
    description "CE Softwire binding interface operational state";

    container ce-interface {
      config false;
      description
        "Data nodes for the operational state of interfaces.";

      leaf name {
        type string;
        description
          "The name of the interface.";
        reference
          "RFC 2863: The Interfaces Group MIB - ifName";
      }

      leaf type {
        type identityref {
          base if:interface-type;
        }
        description
          "The type of the interface.";
        reference
          "RFC 2863: The Interfaces Group MIB - ifType";
      }
      uses softwire-common:traffic-stat;
    }
  }


  /*
   * Notifications
   */

  notification softwire-ce-event {
    if-feature binding;
    description "CE notification";
    leaf ce-binding-ipv6-addr-change {
      type inet:ipv6-address;
      mandatory true;
      description
```

```
        "If the CE's binding-ipv6-address changes for any reason,
        it SHOULD notify the NETCONF client.";
    }
  }
}
<CODE ENDS>
```

6. BR Softwire YANG Model

This module imports typedefs from [RFC6991].

```
<CODE BEGINS> file "ietf-softwire-br@2017-10-19.yang"
module ietf-softwire-br {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-softwire-br";
  prefix "softwire-br";

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

  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 module for the configuration and
    management of A+P Softwire Border Routers. It covers Lightweight 4over6,
    MAP-E, and MAP-T mechanisms.

    Copyright (c) 2017 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 2017-10-19 {
    description
      "Update...";
    reference "-02";
```

```
  }

  revision 2017-06-14 {
    description
      "Monolithic version of ietf-softwire divided into separate CE and BR
      models. Added icmp handling and improved counters.";
    reference "-06";
  }
  revision 2016-06-04 {
    description
      "Version-05: Combined MAP-E/MAP-T into a single tree. Added binding
      table/algorthm 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
      "Binding is used for configuring Lightweight 4over6 mechanism.

      Binding softwire mechanisms are IPv4-over-IPv6 tunnelling transition
mechanisms 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 Border Relay (BR) and a hub-and-spoke
forwarding architecture. In order to delegate the NAPT function and achieve
IPv4 address sharing, port-restricted IPv4 addresses needs to be allocated to
CEs.";

    reference
      "RFC7596, RFC7597 & RFC7599";
  }

  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.

      MAP-T is an IPv6 transition mechanism for transporting IPv4 packets
across an IPv6 network using IP translation. It leverages double stateless
NAT64 based solution as well as the stateless algorithmic address & transport
layer port mapping algorithm defined for MAP-E.

      This feature indicates the instance functions as a MAP-E or
      MAP-T instance.";

    reference
      "RFC7597 & RFC7599";
  }

  container br-instances {
    description
      "BR Instances";

    choice br-type {
```

```
      description
        "Select binding or algorithmic BR functionality.";
      case binding {
        if-feature binding;
        container binding {
          if-feature binding;
          description
            "lw4over6 (binding table) configuration.";
```

```
          list br-instance {
            key "id";
            description
              "A set of lwAFTRs to be configured.";
            container binding-table-versioning {
              description "binding table's version";
              leaf version{
                type uint64;
                description "Incremental version number of the binding
                  table";
              }
              leaf date {
                type yang:date-and-time;
                description "Timestamp of 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 softwires that can be created on
                the lwAFTR.";
            }
            leaf softwires-payload-mtu {
              type uint16;
              units bytes;
              mandatory true;
              description
                "The payload MTU for Lightweight 4over6 softwire.";
            }
            leaf softwire-path-mru {
              type uint16;
              units bytes;
              mandatory true;
              description
                "The path MRU for Lightweight 4over6 softwire.";
            }
            leaf enable-hairpinning {
```

```
              type boolean;
              default true;
              description
                "Enables/disables support for locally forwarding
                (hairpinning) traffic between two CEs (RFC7596
                    Section 6.2)";
            }
            container binding-table {
              description "binding table";
              list binding-entry {
                key "binding-ipv6info";
                description "binding entry";
                uses softwire-common:binding-entry;
              }
            }
            container icmp-policy {
              description
                "The lwAFTR can be configured to process or drop incoming ICMP
                messages, and to generate outgoing ICMP error messages or
                not.";

              container icmpv4-errors {
                description
                  "ICMPv4 error processing configuration";
                leaf allow-incoming-icmpv4 {
                  type boolean;
                  default true;
                  description
                    "Whether to allow processing of incoming ICMPv4 packets.
                    (RFC7596 )";
                }

                leaf generate-icmpv4-errors {
                  type boolean;
                  default true;
                  description
                    "Whether to generate outgoing ICMP error messages on
                    receipt of an inbound IPv4 packet with no matching
                    binding table entry (RFC7596 Seciton 5.2).";
                }
              }

              container icmpv6-errors {
                description
                  "ICMPv6 error processing configuration";
                leaf generate-icmpv6-errors {
                  type boolean;
                  default true;
```

```
                description
                  "Whether to generate ICMPv6 errors messages if no
                  matching binding table entry is found (RFC7596
                      Section 6.2)";
              }
              leaf icmpv6-errors-rate {
                type uint16;
                description
                  "Rate limit threshold in messages per-second
                  for sending ICMPv6 errors messages (RFC7596
                      Section 9.)";
              }
            }
          }

          container traffic-stat {
            config false;
            description
              "traffic-stat";
            uses softwire-common:traffic-stat;
          }

          leaf hairpin-ipv4-bytes {
            type yang:zero-based-counter64;
            description "IPv4 packets locally routed between two CEs
              (hairpinned).";
          }

          leaf hairpin-ipv4-packets {
            type yang:zero-based-counter64;
            description "IPv4 bytes locally routed between two CEs
              (hairpinned).";
          }

          leaf active-softwire-num {
            type uint32;
            config false;
            description
              "The number of currently active softwires on the
              lw4over6 (binding) instance.";
          }
        }
      }
    }
    case algorithm {
      if-feature algorithm;
      container algorithm {
        if-feature algorithm;
```

```
          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.";
          list algo-instance {
            key "id";
            description "Instances of algorithm";
            leaf id {
              type uint32;
              mandatory true;
              description "id";
            }
            leaf name {
              type string;
              description "The MAP instance name.";
            }
            uses softwire-common:algorithm {
              augment "algo-instances/algo-instance"{
                description "Augments the port-set group for the algorithm.";
                uses softwire-common:port-set;
            }
            }
            container traffic-stat {
              description
                "traffic-stat";
              uses softwire-common:traffic-stat;
            }
          }
        }
      }
    }

    /*
     * Notifications
     */
    notification softwire-br-event {
      if-feature binding;
      description "Notifications for BR.";
      leaf br-id {
        type leafref {
          path
            "/br-instances/binding/"
            + "br-instance/id";
        }
        description "...";
      }
```

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

  notification softwire-algorithm-instance-event {
    if-feature algorithm;
    description "Notifications for MAP-E or MAP-T.";
    leaf algo-id {
      type leafref {
        path
          "/br-instances/algorithm/algo-instance/id";
      }
      mandatory true;
      description "MAP-E or MAP-T event.";
    }
    leaf-list invalid-entry-id {
      type leafref {
        path
          "/br-instances/algorithm/algo-instance/id";
      }
      description "Invalid entry event.";
```

```
    }
    leaf-list added-entry {
      type leafref {
        path
          "/br-instances/algorithm/algo-instance/id";
      }
      description "Added entry.";
    }
    leaf-list modified-entry {
      type leafref {
        path
          "/br-instances/algorithm/algo-instance/id";
      }
      description "Modified entry.";
    }
  }
}

<CODE ENDS>
```

7. Common Softwire Element Groups YANG

The following YANG model contains definitions that are used by both the softwire CE and softwire BG YANG models.

```
<CODE BEGINS> file "ietf-softwire-common@2017-10-19.yang"
module ietf-softwire-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-softwire-common";
  prefix "softwire-common";

  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 Customer Premises Equipment (CEs). It
    covers Lightweight 4over6, MAP-E and MAP-T mechanisms.

    Copyright (c) 2017 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 2017-10-19 {
    description
      "Initial version of containing a model for common softwire elements.";
    reference "-02";
  }

  /*
   * Grouping
   */

  grouping port-set {
    description
      "Indicates a set of ports.
      It may be a simple port range, or use the PSID algorithm
      to represent a range of transport layer ports which will
      be used by a NAPT.";

    leaf psid-offset {
      type uint8 {
        range 0..16;
      }
      description
        "The number of offset bits. In Lightweight 4over6,
        the default value is 0 for assigning one contiguous
          port range. In MAP-E/T, the default value is 6,
        which means the system ports (0-1023) are excluded by
        default and assigns port ranges distributed across the
        entire port space, depending on either psid-len or the
        number of contiguous ports.";
    }

    leaf psid-len {
      type uint8 {
        range 0..15;
      }
      mandatory true;
      description
        "The length of PSID, representing the sharing
        ratio for an IPv4 address. This, along with ea-len, also
```

```
        helps to calculate the number of contiguous ports per
        port range";
    }

    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;
      }
      description
        "The IPv6 information for a binding entry.
        If this is an IPv6 prefix, it indicates that
        the IPv6 source address of the CE is constructed
        according to the description in RFC7596;
      if it is an IPv6 address, it means the CE uses
        any /128 address from the assigned CE prefix.
          ";
    }
    leaf binding-ipv4-addr {
      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 {
      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;
      description
        "The IPv6 address for lwaftr.";
    }
  }

  grouping 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.";
    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 "MAP forwarding rule instance for MAP-E/MAP-T";
        leaf enable {
          type boolean;
          description
            "Enable/disable individual MAP-E or MAP-T rule.";
        }
        container algo-versioning {
          description "algorithm's version";
          leaf version {
            type uint64;
            description "Incremental version number for the algorithm";
          }
          leaf 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.";
        }
```

```
        choice data-plane {
          description "Selects MAP-E (encapsulation) or MAP-T (translation)";
          case encapsulation {
            description "encapsulation for MAP-E";
            leaf br-ipv6-addr {
              type inet:ipv6-address;
              mandatory true;
              description
                "The IPv6 address of the MAP-E BR.";
            }
          }
          case translation {
            description "translation for MAP-T";
            leaf dmr-ipv6-prefix {
              type inet:ipv6-prefix;
              description
                "The IPv6 prefix of the MAP-T BR. ";
            }
          }
        }
        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.";
        }
      }
    }
  }

  grouping traffic-stat {
    description "Traffic statistics";
    leaf sent-ipv4-packet {
      type yang:zero-based-counter64;
      description "Number of decapsulated/translated IPv4 packets sent.";
    }
    leaf sent-ipv4-byte {
      type yang:zero-based-counter64;
      description "Decapsulated/translated IPv4 traffic sent, in bytes";
    }
    leaf sent-ipv6-packet {
      type yang:zero-based-counter64;
      description "Number of encapsulated/translated IPv6 packets sent.";
    }
    leaf sent-ipv6-byte {
      type yang:zero-based-counter64;
      description "Encapsulated/translated IPv6 traffic sent, in bytes";
    }
    leaf rcvd-ipv4-packet {
      type yang:zero-based-counter64;
      description "Number of IPv4 packets received for processing.";
    }
    leaf rcvd-ipv4-byte {
      type yang:zero-based-counter64;
      description "IPv4 traffic received for processing, in bytes";
    }
    leaf rcvd-ipv6-packet {
      type yang:zero-based-counter64;
      description "Number of IPv6 packets received for processing.";
    }
    leaf rcvd-ipv6-byte {
      type yang:zero-based-counter64;
      config false;
      description "IPv6 traffic received for processing, in bytes";
    }
    leaf dropped-ipv4-packet {
      type yang:zero-based-counter64;
      description "Number of IPv4 packets dropped.";
    }
    leaf dropped-ipv4-byte {
      type yang:zero-based-counter64;
```

```
      description "IPv4traffic dropped, in bytes";
    }
    leaf dropped-ipv6-packet {
      type yang:zero-based-counter64;
      description "Number of IPv4 packets dropped.";
    }
    leaf dropped-ipv6-byte {
      type yang:zero-based-counter64;
      description "IPv4 traffic dropped, in bytes";
    }
    leaf dropped-ipv4-fragments {
      type yang:zero-based-counter64;
      description "Number of fragmented IPv4 packets dropped";
    }
    leaf dropped-ipv4-bytes {
      type yang:zero-based-counter64;
      description "Fragmented IPv4 traffic dropped, in bytes";
    }
    leaf ipv6-fragments-reassembled {
      type yang:zero-based-counter64;
      description "Number of IPv6 fragments successfully reassembled";
    }
    leaf ipv6-fragments-bytes-reassembled {
      type yang:zero-based-counter64;
      description "IPv6 fragments successfully reassembled, in bytes";
    }
    leaf out-icmpv4-error-packets {
      type yang:zero-based-counter64;
      description "Internally generated ICMPv4 error packets.";
    }
    leaf out-icmpv6-error-packets {
      type yang:zero-based-counter64;
      description "Internally generted ICMPv6 error packets.";
    }
  }
}
<CODE ENDS>
```

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 URIs in the "IETF XML Registry" [RFC3688].

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

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

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

This document requests that IANA registers the following YANG modules in the "YANG Module Names" registry [RFC6020].

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

name: ietf-softwire-br
namespace: urn:ietf:params:xml:ns:yang:softwire-br
prefix: softwire-br
reference: RFC XXXX

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

10. Acknowledgements

The authors would like to thank Lishan Li, Bert Wijnen, Giles Heron, Ole Troan, and Leo Tietz 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://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, <https://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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc6536>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <https://www.rfc-editor.org/info/rfc7223>.

[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, <https://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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc7599>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

11.2. Informative References

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-02 (work in progress), October 2017.

[I-D.ietf-opsawg-nat-yang]
Boucadair, M., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Data Model for Network Address Translation (NAT) and Network Prefix Translation (NPT)", draft-ietf-opsawg-nat-yang-06 (work in progress), October 2017.

[I-D.ietf-softwire-dslite-yang]
Boucadair, M., Jacquenet, C., and S. Sivakumar, "YANG Data Modules for the DS-Lite", draft-ietf-softwire-dslite-yang-07 (work in progress), October 2017.

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

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <https://www.rfc-editor.org/info/rfc7277>.

[RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, DOI 10.17487/RFC8022, November 2016, <https://www.rfc-editor.org/info/rfc8022>.

Appendix A. Configuration Examples

The following sections of the document provide examples on how these YANG models could be used for configuring softwire elements.

A.1. Configuration Example for a lw4o6 BR 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.

Consider an example for the following lw4o6 binding table entry:

lwB4 Binding IPv6 Address: 2001:db8::1

lwB4 Binding IPv4 Address: 192.0.2.1

lwB4 IPv6 Address: 123

lwB4 PSID Length 8

BR IPv6 Address: 2001:db8:1::2

```
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <softwire-config xmlns="urn:ietf:params:xml:ns:yang:ietf-softwire-br">
    <br-instances>
      <binding>
          <br-instance>
            <id>1</id>
            <binding-table>
              <binding-entry>
                <binding-ipv6info>2001:db8::1</binding-ipv6info>
                <binding-ipv4-addr>192.0.2.1</binding-ipv4-addr>
                <port-set>
                  <psid>123</psid>
                  <psid-len>8</psid-len>
                </port-set>
                <br-ipv6-addr>2001:db8:1::2</br-ipv6-addr>
              </binding-entry>
            </binding-table>
            <softwire-num-threshold>1024</softwire-num-threshold>
            <softwire-path-mru>1540</softwire-path-mru>
            <softwire-payload-mtu>1500</softwire-payload-mtu>
          </br-instance>
      </binding>
    </br-instances>
  </softwire-config>
</config>
```

Figure 3: lw4o6 Binding-Table Configuration XML

A.2. Configuration Example for a MAP-E BR

A MAP-E BR is configured with forward mapping rules for the clients it is serving. In this example (taken from [RFC7597], Appendix A, Example 2), the following parameters are required:

- Rule IPv6 Prefix
- Rule IPv4 Prefix
- Rule EA-bit bit length
- IPv6 Address of MAP-BR

The mapping rule has two functions: identifying the destination CE IPv6 address for encapsulating inbound IPv4 packets and the validation of outbound IPv4-in-IPv6 packets received from the CE for de-capsulation.

The transport type for the data plane also needs to be configured for encapsulation to enable MAP-E and forwarding needs to be enabled.

Consider an example for the following MAP-E Forwarding Mapping Rule:

Data plane: encapsulation

Rule IPv6 Prefix: 2001:db8::/40

Rule IPv4 Prefix: 192.0.2.0/24

Rule EA-bit Length: 16

BR IPv6 Address: 2001:db8:ffff::1

Here is the example MAP-E BR configuration xml:

```
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <softwire-config xmlns="urn:ietf:params:xml:ns:yang:ietf-softwire-br">
    <br-instances>
      <algorithm>
          <algo-instance>
            <id>42</id>
            <algo-instances>
              <algo-instance>
                <id>1234</id>
                <data-plane>encapsulation</data-plane>
                <ea-len>16</ea-len>
                <rule-ipv4-prefix>192.0.2.0/24</rule-ipv4-prefix>
                <rule-ipv6-prefix>2001:db8::/40</rule-ipv6-prefix>
                <forwarding>true</forwarding>
                <br-ipv6-addr>2001:db8:ffff::1</br-ipv6-addr>
                <psid-offset>6</psid-offset>
                <psid-len>8</psid-len>
              </algo-instance>
            </algo-instances>
          </algo-instance>
      </algorithm>
    </br-instances>
  </softwire-config>
</config>
```

Figure 4: MAP-E FMR Configuration XML

A.3. lw4o6 CE Configuration Example

The following section provides XML examples for configuring a lw4o6 CE. Examples for routing and NAT44 are also provided for convienience.

Consider an example for the following lw4o6 CE Configuration:

lwB4 Binding IPv6 Address: 2001:db8::1

lwB4 Binding IPv4 Address: 192.0.2.1

lwB4 IPv6 Address: 123

lwB4 PSID Length 8

BR IPv6 Address: 2001:db8:1::2

```
  <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    <interface>
      <name>lw4o6-wan</name>
      <type xmlns:iana="urn:ietf:params:xml:ns:yang:iana-if-type">iana:tunnel</
type>
      <ce-interface xmlns="urn:ietf:params:xml:ns:yang:ietf-softwire-ce">
        <br-ipv6-addr>2001:db8:1::2</br-ipv6-addr>
        <binding-ipv6info>2001:db8::1</binding-ipv6info>
      </ce-interface>
    </interface>
  </interfaces>
</config>
```

Figure 5: lw4o6 CE Configuration XML

In the above example, the interface name is defined for the softwire tunnel. This name is then referenced by the routing configuration for the IPv4 route. The following section provides example configuration for the CE's IPv4 routing, using the YANG model described in [RFC8022].

```
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type>static</type>
        <name>v4</name>
        <static-routes>
          <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing">
            <route>
              <destination-prefix>0.0.0.0/0</destination-prefix>
              <next-hop>
                <outgoing-interface>lw4o6-wan</outgoing-interface>
              </next-hop>
            </route>
          </ipv4>
        </static-routes>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```

Figure 6: lw4o6 CE Routing Configuration XML

The following section provides example configuration for the CE's NAPT44 function, using the YANG model described in [I-D.ietf-opsawg-nat-yang].

```
 <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <nat-module xmlns="urn:ietf:params:xml:ns:yang:ietf-nat">
     <nat-instances>
       <nat-instance>
         <id>1</id>
         <nat-policy>
         <policy-id>1</policy-id>
         <external-ip-address-pool>
           <pool-id>1</pool-id>
           <external-ip-pool>192.0.2.1</external-ip-pool>
         </external-ip-address-pool>
         <port-set-restrict>
           <port-set-algo>
             <psid-offset>6</psid-offset>
             <psid-len>8</psid-len>
             <psid>52</psid>
           </port-set-algo>
         </port-set-restrict>
         <notify-pool-usage>
           <notify-pool-hi-threshold>80</notify-pool-hi-threshold>
```

```
        </notify-pool-usage>
        </nat-policy>
        <connection-limit>
          <limit-per-icmp>8</limit-per-icmp>
          <limit-per-tcp>32</limit-per-tcp>
          <limit-per-udp>16</limit-per-udp>
          <limit-per-instance>1024</limit-per-instance>
        </connection-limit>
        <logging-info>
          <logging-enable>false</logging-enable>
          <destination-address>127.0.0.1/32</destination-address>
          <destination-port>12345</destination-port>
        </logging-info>
        <mapping-limit>
          <limit-per-icmp>8</limit-per-icmp>
          <limit-per-tcp>32</limit-per-tcp>
          <limit-per-udp>16</limit-per-udp>
          <limit-per-instance>1024</limit-per-instance>
        </mapping-limit>
        <mapping-table>
          <mapping-entry>
            <index>1</index>
            <external-src-address>192.0.2.1/32</external-src-address>
            <internal-src-address>192.168.1.0/24</internal-src-address>
            <transport-protocol>6</transport-protocol>
          </mapping-entry>
          <mapping-entry>
            <index>2</index>
            <external-src-address>192.0.2.1/32</external-src-address>
            <internal-src-address>192.168.1.0/24</internal-src-address>
            <transport-protocol>17</transport-protocol>
          </mapping-entry>
          <mapping-entry>
            <index>3</index>
            <external-src-address>192.0.2.1/32</external-src-address>
            <internal-src-address>192.168.1.0/24</internal-src-address>
            <transport-protocol>1</transport-protocol>
          </mapping-entry>
        </mapping-table>
      </nat-instance>
    </nat-instances>
  </nat-module>
</config>
```

Figure 7: lw4o6 NAT Configuration XML

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 Zechlin
Deutsche Telekom AG
Landgrabenweg 151
Bonn, NRW 53227
Germany

Email: sladjana.zechlin@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