Abstract

This document defines YANG modules for the configuration and operation of IPv4-in-IPv6 softwire Border Relays and Customer Premises Equipment for the Lightweight 4over6, Mapping of Address and Port with Encapsulation (MAP-E), and Mapping of Address and Port using Translation (MAP-T) softwire mechanisms.

Editorial Note (To be removed by RFC Editor)

Please update these statements within this document with the RFC number to be assigned to this document:

- "This version of this YANG module is part of RFC XXXX;"
- "RFC XXXX: YANG Modules for IPv4-in-IPv6 Address plus Port Softwires;"
- "reference: RFC XXXX"

Please update the "revision" date of the YANG modules.

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 August 2, 2019.
Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction .................................................. 3
2. Terminology ................................................... 3
3. Overview of the Modules ...................................... 4
   3.1. Overall Structure ......................................... 4
   3.2. Additional Components Configuration ..................... 5
4. Softwire CE YANG Tree Diagram .............................. 5
   4.1. CE Tree Diagram .......................................... 5
   4.2. Softwire CE Tree Diagram Description .................... 7
5. Softwire BR YANG Tree Diagram .............................. 8
   5.1. BR Tree Diagram .......................................... 9
   5.2. Softwire BR Tree Diagram Description .................... 12
6. Softwire CE YANG Module .................................... 13
7. BR Softwire YANG Module .................................... 18
8. Common Softwire Element Groups YANG Module .............. 31
9. Security Considerations ...................................... 39
10. IANA Considerations ......................................... 40
11. Acknowledgements ............................................ 41
12. Contributing Authors ........................................ 42
13. Contributors .................................................. 42
14. References .................................................... 43
   14.1. Normative References .................................... 43
   14.2. Informative References .................................. 44
Appendix A. Configuration Examples ............................ 46
   A.1. Configuration Example for a lw4o6 BR Binding-Table .... 46
   A.2. Configuration Example for a MAP-E BR .................... 47
   A.3. lw4o6 CE Configuration Example ............................ 48
Authors’ Addresses .............................................. 51
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 address plus port (A+P) softwire functional elements: Border Routers (BRs) and Customer Premises Equipment (CEs, a.k.a., CPE). The softwire mechanisms covered in this document are Lightweight 4 over 6 (lw4o6) [RFC7596], Mapping of Address and Port with Encapsulation (MAP-E) [RFC7597], and Mapping of Address and Port using Translation (MAP-T) [RFC7599].

This document focuses on A+P mechanisms [RFC6346]; the reader can refer to [RFC8513] for a YANG module for DS-Lite [RFC6333].

This document defines YANG modules [RFC7950] that can be used to configure and manage A+P softwire elements using the NETCONF [RFC6241], or RESTCONF [RFC8040] protocols for:

- Configuration
- Operational State
- Notifications

2. Terminology

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

The YANG modules in this document adopt the Network Management Datastore Architecture (NMDA) [RFC8342]. The meanings of the symbols used in tree diagrams are defined in [RFC8340].

The document uses the abbreviation ‘BR’ as a general term for softwire tunnel concentrators, including both MAP Border Routers [RFC7597] and Lightweight 4over6 1WAFTRs [RFC7596].

For brevity, "algorithm" is used to refer to the "mapping algorithm" defined in [RFC7597].

A network element may support one or multiple instances of a softwire mechanism; each of these instances (i.e., binding instances, MAP-E instances, or MAP-T instances) may have its own configuration and parameters. The term ‘algo-instance’ is used to denote both MAP-E and MAP-T instances.
3. Overview of the Modules

3.1. Overall Structure

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

ietf-softwire-ce Provides configuration and monitoring for softwire CE element. This module is defined as augments to the interface YANG module [RFC8343].

ietf-softwire-br Provides configuration and monitoring for softwire BR element.

In addition, the following module is defined:

ietf-softwire-common Contains groups of common functions that 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.

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 configuration. For each module, a choice statement ‘ce-type’ is included for either ‘binding’ or ‘algorithm’. ‘Binding’ is used for configuring Lightweight 4over6, whereas ‘algorithm’ is used for configuring MAP-T or MAP-E.

In the ‘algo-instances’ container, a choice statement ‘data-plane’ is included to specify MAP-E (encapsulation) or MAP-T (translation). Table 1 shows how these choices are used to indicate the desired softwire mechanism:

| S46 Mechanism     | ce-type? | data-plane? |}
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Lightweight 4over6</td>
<td>binding</td>
<td>n/a</td>
</tr>
<tr>
<td>MAP-E</td>
<td>algorithm</td>
<td>encapsulation</td>
</tr>
<tr>
<td>MAP-T</td>
<td>algorithm</td>
<td>translation</td>
</tr>
</tbody>
</table>

Table 1: Softwire Mechanism Choice Statement Enumeration

NETCONF notifications are also included.
Note: Earlier versions of this specification 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 their role in the architecture (CE or BR) was a better approach as this follows the intended function and existing implementation approaches more closely.

3.2. Additional Components Configuration

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

- IPv6 forwarding and routing configuration, enabling the CE to obtain one or more IPv6 prefixes for softwire usage. A YANG module for routing management is described in [RFC8349].

- IPv4 routing configuration, to add one or more IPv4 destination prefix(es) reachable via the configured softwire. A YANG module for routing management is described in [RFC8349].

- Stateful NAT44/NAPT management, to optionally specify a port set (Port Set Identifier (PSID)) along with its length. A YANG module for NAT management is described in [RFC8512].

- Stateless NAT46 management, required by softwire translation based mechanisms (i.e., the assignment of a Network-Specific Prefix to use for IPv4/IPv6 translation). A YANG module for NAT management is described in [RFC8512].

As YANG modules for the above functions are already defined in other documents, their functionality is not duplicated here and they should be referred to, as needed. Appendix A.3 provides XML examples of how these modules 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 modules defined in [RFC8344] and [RFC8349] may be used.

4. Softwire CE YANG Tree Diagram

4.1. CE Tree Diagram

The CE module provides configuration and monitoring for all of the softwire mechanisms covered in this document (i.e., Lightweight 4over6, MAP-E, and MAP-T).
This module augments "ietf-interfaces", defined in [RFC8343] with an entry for the softwire. This entry can be referenced to configure IPv4 forwarding features for the element. This entry is added only if tunnel type (Section 10) is set to ‘aplusp’.

Figure 1 shows the tree structure of the softwire CE YANG module:

module: ietf-softwire-ce
    augment /if:interfaces/if:interface:
        +--rw softwire-payload-mtu?       uint16
        +--rw softwire-path-mru?          uint16
        +--rw (ce-type)?
            +--:(binding) (binding-mode)?
                |  +--rw binding-ipv6info?       union
                |  +--rw br-ipv6-addr           inet:ipv6-address
            +--:(algo) (map-e or map-t)?
                +--rw algo-instance* [name]
                    +--rw name                string
                    +--rw enable?             boolean
                    +--rw algo-versioning
                        |  +--rw version?           uint64
                        |  +--rw date?              yang:date-and-time
                    +--rw (data-plane)?
                        |  +--:(encapsulation) (map-e)?
                        |     |  +--rw br-ipv6-addr       inet:ipv6-address
                        |  +--:(translation) (map-t)?
                        |     +--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/if:interface/if:statistics:
        +--ro sent-ipv4-packets?
            |  +--ro sent-ipv4-bytes?
        +--ro sent-ipv6-packets?
            |  +--ro sent-ipv6-bytes?
        +--ro rcvd-ipv4-packets?
            |  +--ro rcvd-ipv4-bytes?
        +--ro rcvd-ipv6-packets?
            |  +--ro rcvd-ipv6-bytes?
4.2. Softwire CE Tree Diagram Description

Additional information related to the operation of a CE element 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 Maximum Transit Unit (MTU) 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 payload Maximum Receive Unit (MRU) size automatically (see Section 3.2 of [RFC4213]).
o ce-type: provides a choice statement allowing the binding or
algorithmic softwire mechanisms to be selected.

Further details relevant to binding softwire elements are:

o binding-ipv6info: used to set the IPv6 binding prefix type to
identify which IPv6 address to use as the tunnel source. It can
be 'ipv6-prefix' or 'ipv6-address'.

o br-ipv6-addr: sets the IPv6 address of the remote BR.

Additional details relevant to some of the important algorithmic
elements are provided below:

o algo-versioning: optionally used to associate a version number
and/or timestamp to the algorithm. This can be used for logging/
data retention purposes \[ RFC7422 \]. The version number is selected
to uniquely identify the algorithm configuration and a new value
written whenever a change is made to the algorithm or a new algo-
instance is created.

o 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. Refer to
Section 4.1 of \[ RFC7598 \].

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

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

o br-ipv6-addr: defines the IPv6 address of the BR. This
information is valid for MAP-E.

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

Additional information on the notification node is listed below:

o ce-binding-ipv6-addr-change: if the CE’s binding IPv6 address
changes for any reason, the NETCONF client will be notified.

5. Softwire BR YANG Tree Diagram
5.1. BR Tree Diagram

The BR YANG module provides configuration and monitoring for all of the softwire mechanisms covered in this document (i.e., Lightweight 4over6, MAP-E, and MAP-T).

Figure 2 provides the tree structure of this module:

```
module: ietf-softwire-br
  +--rw br-instances
    +--rw (br-type)?
      +--:(binding) {binding-mode}?
        +--rw binding
          +--rw bind-instance* [name]
            +--rw name                       string
            +--rw binding-table-versioning
              +--rw version?   uint64
              +--rw date?      yang:date-and-time
            +--rw softwire-num-max      uint32
            +--rw softwire-payload-mltu      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 icmpv4-rate?              uint32
              +--rw icmpv6-errors
                +--rw generate-icmpv6-errors?   boolean
                +--rw icmpv6-rate?              uint32
            +--ro traffic-stat
              +--ro discontinuity-time       yang:date-and-time
              +--ro sent-ipv4-packets?
                +--ro sent-ipv4-packets?      yang:zero-based-counter64
                +--ro sent-ipv4-bytes?
                  +--ro sent-ipv4-bytes?      yang:zero-based-counter64
```
yang:zero-based-counter64

---ro sent-ipv6-bytes?

---ro rcvd-ipv4-packets?

---ro rcvd-ipv4-bytes?

---ro rcvd-ipv6-packets?

---ro rcvd-ipv6-bytes?

---ro dropped-ipv4-packets?

---ro dropped-ipv4-bytes?

---ro dropped-ipv4-packets?

---ro dropped-ipv4-bytes?

---ro dropped-ipv6-packets?

---ro dropped-ipv6-bytes?

---ro dropped-ipv4-fragments?

---ro dropped-ipv4-fragment-bytes?

---ro ipv6-fragments-reassembled?

---ro ipv6-fragments-bytes-reassembled?

---ro out-icmpv4-error-packets?

---ro out-icmpv4-error-bytes?

---ro out-icmpv6-error-packets?

---ro out-icmpv6-error-bytes?

---ro dropped-icmpv4-packets?

---ro dropped-icmpv4-bytes?

---ro hairpin-ipv4-packets?

---ro hairpin-ipv4-bytes?

---ro active-softwire-num?

uint32

---: (algo) (map-e or map-t)?

---rw algorithm

---rw algo-instance* [name]
---rw name                      string
---rw enable?                  boolean
---rw algo-versioning
    |    ---rw version?       uint64
    |    ---rw date?         yang:date-and-time
---rw (data-plane)?
    |    ---:(encapsulation) {map-e}?
    |        |    ---rw br-ipv6-addr inet:ipv6-address
    |    ---:(translation) {map-t}?
    |        |    ---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 port-set
    |    ---rw psid-offset?   uint8
    |    ---rw psid-len       uint8
    |    ---rw psid           uint16
---ro traffic-stat
    |    ---ro discontinuity-time yang:date-and-time
    |    ---ro sent-ipv4-packets?
    |        |        |        | yang:zero-based-counter64
    |    ---ro sent-ipv4-bytes?
    |        |        | yang:zero-based-counter64
    |    ---ro sent-ipv6-packets?
    |        | yang:zero-based-counter64
    |    ---ro sent-ipv6-bytes?
    |        | yang:zero-based-counter64
    |    ---ro rcvd-ipv4-packets?
    |        | yang:zero-based-counter64
    |    ---ro rcvd-ipv4-bytes?
    |        | yang:zero-based-counter64
    |    ---ro rcvd-ipv6-packets?
    |        | yang:zero-based-counter64
    |    ---ro rcvd-ipv6-bytes?
    |        | yang:zero-based-counter64
    |    ---ro dropped-ipv4-packets?
    |        | yang:zero-based-counter64
    |    ---ro dropped-ipv4-bytes?
    |        | yang:zero-based-counter64
    |    ---ro dropped-ipv6-packets?
    |        | yang:zero-based-counter64
    |    ---ro dropped-ipv6-bytes?
    |        | yang:zero-based-counter64
    |    ---ro dropped-ipv4-fragments?
    |        | yang:zero-based-counter64
    |    ---ro dropped-ipv4-fragment-bytes?
    |        | yang:zero-based-counter64
5.2.  Softwire BR Tree Diagram Description

The descriptions for leaves which are common with the CE module are provided in Section 4.2. Descriptions for additional elements are provided below:

- binding-table-versioning: optionally used to associate a version number and/or timestamp to the binding table. This can be used for logging or data retention purposes [RFC7422]. The version number is selected to uniquely identify the binding table configuration 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 {lwB4’s IPv6 address/prefix, the allocated IPv4 address, restricted port-set}. For detail information, please refer to [RFC7596].

- softwire-num-max: used to set the maximum number of softwire binding rules that can be created on the lw4o6 element simultaneously. This parameter must not be set to zero because this is equivalent to disabling the BR instance.

- active-softwire-num: holds the number of softwires currently provisioned on the BR element.

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

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

6. Softwire CE YANG Module

This module imports the modules defined in [RFC6991], [RFC8343], and [RFC7224]. It also imports the ‘ietf-softwire-common’ and ‘iana-tunnel-type’ modules [I-D.ietf-softwire-iftunnel].

```yaml
<CODE BEGINS>file "ietf-softwire-ce@2019-01-11.yang"

module ietf-softwire-ce {
    yang-version 1.1;
    prefix softwire-ce;

    import ietf-inet-types {
        prefix inet;
        reference "Section 4 of RFC 6991";
    }
    import ietf-interfaces {
        prefix if;
        reference "RFC 8343: A YANG Data Model for Interface Management";
    }
    import ietf-softwire-common {
        prefix softwire-common;
        reference "RFC XXXX: YANG Modules for IPv4-in-IPv6 Address plus Port Softwires";
    }
    import iana-tunnel-type {
        prefix iana-tunnel-type;
    }

```
This document defines a YANG 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) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";}
revision 2019-01-11 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Modules for IPv4-in-IPv6 Address plus Port Softwires";
}

/*
 * Features
 */

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

    Binding based softwire mechanisms are IPv4-over-IPv6 tunnelling
transition mechanisms specifically intended for complete
independence between the IPv6 subnet prefix (and 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 using
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.

    This feature indicates that the network element can function as
one or more binding based softwire instances.";
  reference
    "RFC7596: Lightweight 4over6: An Extension to the Dual-Stack Lite
Architecture
RFC7597: Mapping of Address and Port with Encapsulation (MAP-E)
RFC7599: Mapping of Address and Port using Translation (MAP-T)";
}

feature map-e {
  description
    "MAP-E is an IPv6 transition mechanism for transporting IPv4
packets across an IPv6 network using IP encapsulation. MAP-E
allows for a reduction of the amount of centralized state using
rules to express IPv4/IPv6 address mappings. This introduces an
algorithmic relationship between the IPv6 subnet and IPv4
address.

    This feature indicates that the network element can function as
one or more MAP-E softwire instances.";
  reference

"RFC7597: Mapping of Address and Port with Encapsulation (MAP-E)";
}

feature map-t {
    description
        "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 that the network element can function as one or more MAP-T softwire instances.";
    reference
        "RFC7599: Mapping of Address and Port using Translation (MAP-T)";
}

// Binding Entry

grouping binding-entry {
    description
        "The binding BR (Border Relay) maintains an address binding table that contains the binding between the CE’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."
            When the IPv6 prefix type is used, the IPv6 source address of the CE is constructed according to the description in RFC7596.
            If the IPv6 address type is used, the CE can use any valid /128 address from a prefix assigned to the CE.";
        reference "Section 5.1 of RFC7596.";
    }
    leaf br-ipv6-addr {
        type inet:ipv6-address;
        mandatory true;
        description
            "The IPv6 address of the binding BR.";
    }
}
// configuration and stateful parameters for softwire CE interface

augment " /if:interfaces /if:interface" {
    when "derived-from(if:type, 'iana-tunnel-type:aplusp')";
    description
        "Softwire CE interface configuration";
    leaf softwire-payload-mtu {
        type uint16;
        units "bytes";
        description
            "The payload IPv4 MTU for the softwire tunnel.";
    }
    leaf softwire-path-mru {
        type uint16;
        units "bytes";
        description
            "The path MRU for the softwire (payload + encapsulation
             overhead).";
        reference
            "RFC 4213: Basic Transition Mechanisms for IPv6 Hosts and
             Routers";
    }
    choice ce-type {
        description
            "Sets the softwire CE mechanism";
        case binding {
            if-feature "binding-mode";
            description
                "CE binding configuration";
            uses binding-entry;
        }
        case algo {
            if-feature "map-e or map-t";
            description
                "CE algorithm configuration";
            container algo-instances {
                description
                    "Collection of MAP-E/MAP-T parameters";
                list algo-instance {
                    key "name";
                    description
                        "MAP forwarding rule instance for
                         MAP-E/MAP-T";
                    leaf name {
                        type string;
                        mandatory true;
                        description
                            "The name is used to uniquely identify an algorithm";
                    };
                };
            };
        };
    };
}
instance.

This name can be automatically assigned or explicitly configured.
}
}
}
}
}
}
}
}
}

augment "/if:interfaces/if:interface/if:statistics" {
    when "derived-from(../if:type, 'iana-tunnel-type:aplusp')"
    description
        "Softwire CE interface statistics."
    uses softwire-common:traffic-stat;
}

notification softwire-ce-event {
    if-feature "binding-mode"
    description
        "CE notification"
    leaf ce-binding-ipv6-addr-change {
        type inet:ipv6-address;
        mandatory true;
        description
            "This notification is generated whenever the CE’s binding IPv6 address changes for any reason."
    }
}

<CODE ENDS>

7. BR Softwire YANG Module

This module imports typedefs from [RFC6991]. It also imports the ‘ietf-softwire-common’ module.

<CODE BEGINS>file "ietf-softwire-br@2019-01-11.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;
    reference "Section 4 of RFC 6991";
}
import ietf-yang-types {
    prefix yang;
    reference "Section 3 of RFC 6991";
}
import ietf-softwire-common {
    prefix softwire-common;
    reference
        "RFC XXXX: YANG Modules for IPv4-in-IPv6 Address plus Port
        Softwires";
}

organization "IETF Softwire Working Group";
contact
    "WG Web: <https://datatracker.ietf.org/wg/softwire/>
    WG List: <mailto:softwire@ietf.org>

    Author: Qi Sun
    <mailto:sunqi.ietf@gmail.com>

    Author: Linhui Sun
    <mailto:lh.sunlinh@gmail.com>

    Author: Yong Cui
    <mailto:yong@csnet1.cs.tsinghua.edu.cn>

    Editor: Ian Farrer
    <mailto:ian.farrer@telekom.de>

    Author: Sladjana Zoric
    <mailto:sladjana.zoric@telekom.de>

    Editor: Mohamed Boucadair
    <mailto:mohamed.boucadair@orange.com>

    Author: Rajiv Asati
    <mailto:rajiva@cisco.com">

description
    "This document defines a YANG module for the configuration and
    management of A+P Softwire Border Routers. It covers Lightweight
    4over6, MAP-E, and MAP-T mechanisms.

    Copyright (c) 2019 IETF Trust and the persons identified as
grouping port-set {
  description
    "Describes a set of layer 4 port numbers.
    This may be a simple port range, or use the Port Set
    Identifier (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 the assigned port ranges are 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, can
be used 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 binding BR maintains an address binding table that
  contains the binding between the CE’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 CE binding entry.
    When the IPv6 prefix type is used,
    the IPv6 source address of the CE is constructed
    according to the description in RFC7596;
    if the IPv6 address type is used, the CE can use
    any valid /128 address from a prefix assigned to
    the CE.";
    reference
    "RFC7596: Lightweight 4over6: An Extension to the Dual-Stack
    Lite Architecture";
  }
  leaf binding-ipv4-addr {
    type inet:ipv4-address;
    description
    "The IPv4 address assigned to the binding CE,
    which is used as the IPv4 external address
    for binding CE local NAPT44.";
  }
  container port-set {
    description
    "For Lightweight 4over6, the default value
for 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 binding BR."
}
}

/*
 * Features
 */

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

  Binding based softwire mechanisms are IPv4-over-IPv6 tunnelling
  transition mechanisms specifically intended for complete
  independence between the IPv6 subnet prefix (and 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 using
  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.

  This feature indicates that the network element can function as
  one or more binding based softwire instances."
  reference  "RFC7596: Lightweight 4over6: An Extension to the Dual-Stack Lite
  Architecture
  RFC7597: Mapping of Address and Port with Encapsulation (MAP-E)
  RFC7599: Mapping of Address and Port using Translation (MAP-T)"
}

feature map-e {
  description  "MAP-E is an IPv6 transition mechanism for transporting IPv4
  packets across an IPv6 network using IP encapsulation. MAP-E
  allows for a reduction of the amount of centralized state using
rules to express IPv4/IPv6 address mappings. This introduces an
algorithmic relationship between the IPv6 subnet and IPv4
address.

This feature indicates that the network element can function as
one or more MAP-E software instances.

reference
"RFC7597: Mapping of Address and Port with Encapsulation (MAP-E)"

feature map-t {
  description
  "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 that the network element can function as
one or more MAP-T software instances."

reference
"RFC7599: Mapping of Address and Port using Translation (MAP-T)"
}

container br-instances {
  description
  "BR instances enabled in a network element."
  choice br-type {
    description
    "Select binding or algorithmic BR functionality."
    case binding {
      if-feature "binding-mode"
      container binding {
        description
        "binding mechanism (binding table) configuration."
        list bind-instance {
          key "name";
          description
          "A set of binding instances to be configured."
          leaf name {
            type string;
            mandatory true;
            description
            "The name for the binding BR. It is used to uniquely
distinguish a binding instance by its name."
          }
        }
        container binding-table-versioning {
          description
leaf version {
  type uint64;
  description "Version number for this binding table."
}
leaf date {
  type yang:date-and-time;
  description "Timestamp when the binding table was activated."
}

A binding instance may be provided with binding entries that may change in time (e.g., increase the size of the port set). When a party who is the victim of abuse presents an external IP address/port, the version of the binding table is important because depending on the version, a distinct customer may be identified.

The timestamp is used as a key to find the appropriate binding table that was put into effect when an abuse occurred."

reference "RFC7422: Deterministic Address Mapping to Reduce Logging in Carrier-Grade NAT Deployments";

leaf softwire-num-max {
  type uint32 {
    range "1..max";
  }
  mandatory true;
  description "The maximum number of softwires that can be created on the binding BR."
}

leaf softwire-payload-mtu {
  type uint16;
  units "bytes";
  mandatory true;
  description "The payload IPv4 MTU for binding softwire."
}

leaf softwire-path-mru {
  type uint16;
  units "bytes";
  mandatory true;
  description
"The path MRU for binding softwire."
reference
"RFC4213: Basic Transition Mechanisms for IPv6 Hosts and Routers"
}
leaf enable-hairpinning {
  type boolean;
  default "true";
  description
  "Enables/disables support for locally forwarding (hairpinning) traffic between two CEs."
  reference "Section 6.2 of RFC7596"
}
container binding-table {
  description
  "binding table"
  list binding-entry {
    key "binding-ipv6info";
    description
    "binding entry"
    uses binding-entry;
  }
}
container icmp-policy {
  description
  "The binding BR can be configured to process or drop incoming ICMP messages, and to generate outgoing ICMP error messages."
  container icmpv4-errors {
    description
    "ICMPv4 error processing configuration"
    leaf allow-incoming-icmpv4 {
      type boolean;
      default "true";
      description
      "Enables the processing of incoming ICMPv4 packets."
      reference
      "RFC7596: Lightweight 4over6: An Extension to the Dual-Stack Lite Architecture"
    }
    leaf icmpv4-rate {
      type uint32;
      description
      "Rate limit threshold in messages per-second for processing incoming ICMPv4 errors messages"
    }
    leaf generate-icmpv4-errors {
type boolean;
default "true";
description
  "Enables the generation of outgoing ICMPv4 error
  messages on receipt of an inbound IPv4 packet
  with no matching binding table entry."
  reference "Section 5.2 of RFC7596.";
} 
}
}
}
}
}
}
}
} container icmpv6-errors {
  description
  "ICMPv6 error processing configuration";
  leaf generate-icmpv6-errors {
    type boolean;
    default "true";
    description
      "Enables the generation of ICMPv6 error messages if
      no matching binding table entry is found for a
      received packet."
      reference "Section 6.2 of RFC7596.";
  }
  leaf icmpv6-rate {
    type uint32;
    description
      "Rate limit threshold in messages per-second
      for sending ICMPv6 errors messages"
      reference "Section 9 of RFC7596.";
  }
}
}
}
}
}
}
}
} container traffic-stat {
  config false;
  description
    "Traffic statistics information for the BR.";
  leaf discontinuity-time {
    type yang:date-and-time;
    mandatory true;
    description
      "The time of the most recent occasion on which the BR
      instance suffered a discontinuity. This must be
      initialized when the BR instance is configured
      or rebooted.";
  }
}
uses softwire-common:traffic-stat;
leaf dropped-icmpv4-packets {
  type yang:zero-based-counter64;
  description
    "ICMPv4 packets that are dropped as a result
of the ICMP policy. Typically, this can be any incoming ICMPv4 packets if ICMPv4 processing is disabled or incoming ICMPv4 packets that exceed the ICMPv4 rate-limit threshold.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ‘discontinuity-time’.

leaf dropped-icmpv4-bytes {
    type yang:zero-based-counter64;
    description
    "ICMPv4 messages, in bytes, that are dropped as a result of the ICMP policy. Typically, it can be any incoming ICMPv4 packets if ICMPv4 processing is disabled or incoming ICMPv4 packets that exceed the ICMPv4 rate-limit threshold.

    Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ‘discontinuity-time’.";
}

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

    Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ‘discontinuity-time’.";
}

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

    Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ‘discontinuity-time’.";
}

leaf active-softwire-num {
type uint32;
config false;
description
"The number of currently active softwires on the binding instance.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ‘discontinuity-time’.";
}
}
}
}
}
case algo {
  if-feature "map-e or map-t";
  container algorithm {
    description
    " A set of parameters used for MAP-E/MAP-T.";
    list algo-instance {
      key "name";
      description
      "Instances of algorithm";
      leaf name {
        type string;
        mandatory true;
        description
        "The name is used to uniquely identify an algorithm instance.
        This name can be automatically assigned or explicitly configured.";
      }
      uses softwire-common:algorithm-instance;
    }
    container port-set {
      description
      "Indicates a set of ports.";
      uses port-set;
    }
    container traffic-stat {
      config false;
      description
      "Traffic statistics information for the BR.";
      leaf discontinuity-time {
        type yang:date-and-time;
        mandatory true;
        description
        "The number of currently active softwires on the binding instance.
        Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ‘discontinuity-time’.";
      }
    }
  }
}
The time of the most recent occasion on which the BR instance suffered a discontinuity. This must be reset to the current date-and-time when the BR instance is configured or rebooted.

notification softwire-binding-instance-event {
  if-feature "binding-mode";
  description
    "Notifications for binding instance when an entry is added, modified, or is not valid anymore.";
  leaf bind-name {
    type leafref {
      path "/br-instances/binding/bind-instance/name";
    } 
    description
      "The name of the binding-instance that generated the notification.";
  }
  leaf-list invalid-entry {
    type leafref {
      path "/br-instances/binding/"
      + "bind-instance[name=current()]/../bind-name/"
      + "binding-table/binding-entry/binding-ipv6info";
    } 
    description
      "Notify the client that a specific binding entry has expired or is 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 gets other information from the binding BR about the entry";
indexed by that ipv6 address."
}
leaf-list modified-entry {
  type leafref {
    path
      "/br-instances/binding/
      + "bind-instance[name=current()/../bind-name]/
      + "binding-table/binding-entry/binding-ipv6info";
  }
  description
    "The binding-table entry that has been modified.";
}
}
notification softwire-algorithm-instance-event {
  if-feature "map-e or map-t";
  description
    "Notifications for algorithm instance when an entry is
    added, modified, or is not valid anymore.";
  leaf algo-name {
    type leafref {
      path "/br-instances/algorithm/algo-instance/name";
    }
    mandatory true;
    description
      "algorithmic instance event.";
  }
  leaf-list invalid-entry {
    type leafref {
      path "/br-instances/algorithm/algo-instance/name";
    }
    description
      "Invalid entry event.";
  }
  leaf-list added-entry {
    type leafref {
      path "/br-instances/algorithm/algo-instance/name";
    }
    description
      "Added entry.";
  }
  leaf-list modified-entry {
    type leafref {
      path "/br-instances/algorithm/algo-instance/name";
    }
    description
      "Modified entry.";
  }
}
8. Common Softwire Element Groups YANG Module

This module imports typedefs from [RFC6991].

The following YANG module contains definitions that are used by both the softwire CE and softwire BR YANG modules.
This document defines a YANG module defining types common to all A+P modules.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.

revision 2019-01-11 {
  description "Initial revision.";
  reference "RFC XXXX: YANG Modules for IPv4-in-IPv6 Address plus Port Softwires";
}

feature map-e {
  description "MAP-E is an IPv6 transition mechanism for transporting IPv4 packets across an IPv6 network using IP encapsulation. MAP-E allows for a reduction of the amount of centralized state using rules to express IPv4/IPv6 address mappings. This introduces an algorithmic relationship between the IPv6 subnet and IPv4 address.

  This feature indicates that the network element can function as one or more MAP-E softwire instances.";
  reference "RFC7597: Mapping of Address and Port with Encapsulation (MAP-E)";
}

feature map-t {
  description "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 that the network element can function as one or more MAP-T softwire instances.

reference
"RFC7599: Mapping of Address and Port using Translation (MAP-T)";

/
* Groupings
*/

grouping algorithm-instance {
  description
  "A collection of parameters that is used for MAP-E/MAP-T."
  leaf enable {
    type boolean;
    description
    "Enable/disable an individual MAP-E or MAP-T rule.";
  }
  container algo-versioning {
    description
    "Version number for this algorithm instance"
    leaf version {
      type uint64;
      description
      "A version number for the mapping algorithm rules provided to the algorithm instance";
    }
    leaf date {
      type yang:date-and-time;
      description
      "Timestamp when the algorithm instance was activated.

      An algorithm instance may be provided with mapping rules that may change in time (for example, increase the size of the port set). When a party who is the victim of abuse presents an external IP address/port, the version of the algorithm is important because depending on the version, a distinct customer may be identified.

      The timestamp is used as a key to find the appropriate algorithm that was put into effect when an abuse occurred."
    }
  }
}

reference
"RFC7422: Deterministic Address Mapping to Reduce
choice data-plane {
  description
      "Selects MAP-E (encapsulation) or MAP-T (translation)";
  case encapsulation {
    if-feature "map-e";
    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 {
    if-feature "map-t";
    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 identifying 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 Basic Mapping Rule (BMR) only and must not be used for forwarding."
}
}


grouping traffic-stat {

description
"Traffic statistics";
leaf sent-ipv4-packets {
type yang:zero-based-counter64;
description
"Number of decapsulated and forwarded IPv4 packets.
Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'."
}
leaf sent-ipv4-bytes {
type yang:zero-based-counter64;
description
"Decapsulated/translated IPv4 traffic sent, in bytes
Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'."
}
leaf sent-ipv6-packets {
type yang:zero-based-counter64;
description
"Number of encapsulated IPv6 packets sent.
Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'."
}
leaf sent-ipv6-bytes {

type yang:zero-based-counter64;
description
"Encapsulated IPv6 traffic sent, in bytes
Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'.";
}
leaf rcvd-ipv4-packets {
  type yang:zero-based-counter64;
description
"Number of IPv4 packets received.
Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'.";
}
leaf rcvd-ipv4-bytes {
  type yang:zero-based-counter64;
description
"IPv4 traffic received, in bytes.
Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'.";
}
leaf rcvd-ipv6-packets {
  type yang:zero-based-counter64;
description
"Number of IPv4-in-IPv6 packets received.
Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'.";
}
leaf rcvd-ipv6-bytes {
  type yang:zero-based-counter64;
description
"IPv4-in-IPv6 traffic received, in bytes.
Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'.";
leaf dropped-ipv4-packets {
  type yang:zero-based-counter64;
  description
    "Number of IPv4 packets dropped at the Internet-facing interface.

    Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'.";
}

leaf dropped-ipv4-bytes {
  type yang:zero-based-counter64;
  description
    "IPv4 traffic dropped at the Internet-facing interface, in bytes.

    Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'.";
}

leaf dropped-ipv6-packets {
  type yang:zero-based-counter64;
  description
    "Number of IPv4-in-IPv6 packets dropped.

    Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'.";
}

leaf dropped-ipv6-bytes {
  type yang:zero-based-counter64;
  description
    "IPv4-in-IPv6 traffic dropped, in bytes.

    Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'.";
}

leaf dropped-ipv4-fragments {
  type yang:zero-based-counter64;
  description
    "Number of fragmented IPv4 packets dropped.

    Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'.";
}
Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ‘discontinuity-time’.

leaf dropped-ipv4-fragment-bytes {
  type yang:zero-based-counter64;
  description
  "Fragmented IPv4 traffic dropped, in bytes."

  Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ‘discontinuity-time’.
}

leaf ipv6-fragments-reassembled {
  type yang:zero-based-counter64;
  description
  "Number of IPv6 fragments successfully reassembled."

  Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ‘discontinuity-time’.
}

leaf ipv6-fragments-bytes-reassembled {
  type yang:zero-based-counter64;
  description
  "IPv6 fragments successfully reassembled, in bytes."

  Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ‘discontinuity-time’.
}

leaf out-icmpv4-error-packets {
  type yang:zero-based-counter64;
  description
  "Internally generated ICMPv4 error packets."

  Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ‘discontinuity-time’.
}

leaf out-icmpv4-error-bytes {
  type yang:zero-based-counter64;
  description

"Internally generated ICMPv4 error messages, in bytes.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'."
}
leaf out-icmpv6-error-packets {
  type yang:zero-based-counter64;
  description
  "Internally generated ICMPv6 error packets.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'."
}
leaf out-icmpv6-error-bytes {
  type yang:zero-based-counter64;
  description
  "Internally generated ICMPv6 error messages, in bytes.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'."
}

9. Security Considerations

The YANG modules defined in this document is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

All data nodes defined in the YANG modules which can be created, modified, and deleted (i.e., config true, which is the default) are considered sensitive. Write operations (e.g., edit-config) applied
to these data nodes without proper protection can negatively affect
network operations. An attacker who is able to access the BR can
undertake various attacks, such as:

- Setting the value of ‘br-ipv6-addr’ on the CE to point to an
  illegitimate BR so that it can intercept all the traffic sent by a
  CE. Illegitimately intercepting users’ traffic is an attack with
  severe implications on privacy.

- Setting the MTU to a low value, which may increase the number of
  fragments ('softwire-payload-mtu').

- Disabling hairpinning (i.e., setting 'enable-hairpinning' to
  'false') to prevent communications between CEs.

- Setting 'softwire-num-max' to an arbitrary high value, which may
  be exploited by a misbehaving user to perform a DoS on the binding
  BR by mounting a massive number of softwires.

- Setting 'icmpv4-rate' or 'icmpv6-rate' to a low value, which may
  lead to the deactivation of ICMP messages handling.

- Accessing to private data maintained by the BR (e.g., the binding
  table or the algorithm configuration). Such data can be misused
  to track the activity of a host.

- Instructing the BR to install entries which in turn will induce a
  DDoS attack by means of the notifications generated by the BR.
  This DDoS can be softened by defining a notification interval, but
  given that this interval parameter can be disabled or set to a low
  value by the misbehaving entity, the same problem will be
  observed.

Security considerations related to lw4o6, MAP-T, and MAP-E are
discussed in [RFC7596], [RFC7597], and [RFC7599] respectively.

Security considerations given in [RFC7950] are also applicable here.

10. IANA Considerations

This document requests IANA to assign a new tunnel type under the
"tunnelType" sub-registry of the "ifType definitions" registry
maintained at [TUNNELTYPE-IANA-REGISTRY] and use the following data
for the new entry:
This document requests IANA to register the following in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

```
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.
```

```
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.
```

```
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" subregistry [RFC7950] within the "YANG Parameters" registry.

```
name: ietf-softwire-ce
prefix: softwire-ce
reference: RFC XXXX
```

```
name: ietf-softwire-br
prefix: softwire-br
reference: RFC XXXX
```

```
name: ietf-softwire-common
prefix: softwire-common
reference: RFC XXXX
```

11. Acknowledgements

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

Thanks to Sheng Jiang for the review.

Special thanks to Tom Petch and Martin Bjorklund for the detailed
review and suggestions.

12. Contributing Authors

The following individuals are co-authors:

Yong Cui
Tsinghua University
Beijing 100084
P.R. China
Phone: +86-10-6260-3059
Email: cuiyong@tsinghua.edu.cn

Qi Sun
Tsinghua University
Beijing 100084
P.R. China
Phone: +86-10-6278-5822
Email: sunqi.ietf@gmail.com

Linhui Sun
Tsinghua University
Beijing 100084
P.R. China
Phone: +86-10-6278-5822
Email: lh.sunlinh@gmail.com

Sladjana Zechlin
Deutsche Telekom AG
Landgrabenweg 151
Bonn, NRW 53227
Germany
Email: sladjana.zechlin@telekom.de

Rajiv Asati
Cisco Systems, Inc.
7025 Kit Creek Rd.
RTP, NC 27709
USA
Email: Rajiva@cisco.com

13. Contributors

The following individual contributed to this document:
14. References

14.1. Normative References

[I-D.ietf-softwire-iftunnel]


Farrer & Boucadair Expires August 2, 2019 [Page 43]
14.2.  Informative References


Farrer & Boucadair Expires August 2, 2019 [Page 45]
Appendix A.  Configuration Examples

The following sections provide examples of how the softwire YANG modules can 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 PSID:                  0x34
lwB4 PSID Length            8
BR IPv6 Address:            2001:db8:1::2
A.2. Configuration Example for a MAP-E BR

A MAP-E BR is configured with forward mapping rules for the CEs 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

Figure 4 provides the example MAP-E BR configuration xml.

```xml
<instances>
  <algorithm>
    <algo-instance>
      <name>myalgo-instance</name>
      <encapsulation>
        <br-ipv6-addr>2001:db8:ffff::1</br-ipv6-addr>
      </encapsulation>
      <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>
      <port-set>
        <psid-offset>6</psid-offset>
        <psid-len>8</psid-len>
      </port-set>
    </algo-instance>
  </algorithm>
</instances>
```

Figure 4: MAP-E FMR Configuration XML

### A.3. lw4o6 CE Configuration Example

This 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 PSID: 0x34
lwB4 PSID Length 8
BR IPv6 Address: 2001:db8:1::2

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

Figure 5: lw4o6 CE Configuration XML

In the example depicted in Figure 5, the interface name is defined for the softwire tunnel. This name is then referenced by the routing configuration for the IPv4 route. Figure 6 provides an example configuration for the CE’s IPv4 routing, using the YANG module described in [RFC8349].
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <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

Figure 7 provides an example configuration for the CE’s NAPT44 function, using the YANG module described in [RFC8512].

<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <nat xmlns="urn:ietf:params:xml:ns:yang:ietf-nat">
    <instances>
      <instance id="1">
        <policy id="1">
          <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>
            <pool-id>1</pool-id>
            <high-threshold>80</high-threshold>
          </notify-pool-usage>
        </policy>
      </instance>
    </instances>
  </nat>
</config>
</policy>
<mapping-limits>
  <limit-per-protocol>
    <protocol-id>1</protocol-id>
    <limit>8</limit>
  </limit-per-protocol>
  <limit-per-protocol>
    <protocol-id>6</protocol-id>
    <limit>32</limit>
  </limit-per-protocol>
  <limit-per-protocol>
    <protocol-id>17</protocol-id>
    <limit>16</limit>
  </limit-per-protocol>
</mapping-limits>
<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>
</instance>
</instances>
</nat>
</config>

Figure 7: lw4o6 NAT Configuration XML

Authors’ Addresses