Softwire Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2015

Q. Sun
H. Wang
Y. Cui
Tsinghua University
I. Farrer
Deutsche Telekom AG

October 20, 2014

YANG Data Model for IPv4-in-IPv6 Softwire
draft-sun-softwire-yang-00

Abstract

This document defines a YANG data model for the configuration and management of IPv4-in-IPv6 Softwire Concentrators containing a Network Configuration Protocol (NETCONF) server. The models cover A+P [RFC6346], encapsulation based softwires.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The IETF Softwire Working Group has developed several IPv4-in-IPv6
Softwire mechanisms for different scenarios. This document defines a
YANG data model that can be used to configure and manage the IPv4-in-
IPv6 Softwire Concentrator (Border Router).

Due to the inherent similarities of the data plane forwarding, the
YANG models that are described in this document are for Lightweight
4o6 [I-D.ietf-softwire-lw4over6] and MAP-E [I-D.ietf-softwire-map].

DISCUSSION POINT: Should the draft be extended to include MAP-T, 4rd
and DS-Lite?

The models define several containers. Container "softwire-config"
holds a collection of YANG definitions common to all softwire
configuration. Container "softwire-state" holds YANG definitions for

the operational state of the Softwire Concentrator.

The softwire mechanism specifics each have their own indivial YANG modules:

o Lightweight 4over6

o MAP-E

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

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The reader should be familiar with the terms define by the IETF Softwire working group, and relevant documents and the YANG data modelling language [RFC6020].

1.2. Tree Diagram

The meaning if the symbols in these diagrams is as follows:

o Brackets "[" and "]" enclose list keys.

o Parentheses "(" and ")" enclose choice and case nodes, and case nodes are also marked with a colon (":").

o Symbols after data node names: "?" means an optional node, and "*" denotes a list and leaf-list.

o Abbreviations before data node names: "rw" means configuration data (read-write), and "ro" means state data (read-only).

2. Objectives

2.1. Common

This YANG model intends to abstract the shared features of different Concentrators such as softwire type, maximum number of softwires, etc.

Since different concentrators have specialised functions, the following sections describe the model objectives for those solutions.

The following model is the root of the softwire YANG model. All functions are listed, but the YANG model uses the "feature" statement to distinguish different softwire mechanisms.

2.2. Lightweight 4over6

The IPv4-IPv6 address binding information needs to be configured to Lightweight AFTR so that the state synchronisation can be achieved between the provisioning system and the lwAFTR.

2.3. MAP-E

The provisioning system or administrator needs to configure the corret MAP rules on the MAP BR.

3. Softwire YANG Data Model

3.1. Common Data Tree

Figure 1 below describes the softwire data model which is common between all of the different softwire mechanisms:

```
   +--rw softwire-config
   |  +--rw enabled?                                   boolean
   |  +--rw name?                                      string
   |  +--rw description?                               string
   |  +--rw softwire-num-threshold                     uint32
   |  +--rw tunnel-mtu                                 uint32
   |  +--rw lw4over6
   |  +--rw map-e
   +--ro softwire-state
      +--ro enabled?                                   boolean
      +--ro name?                                      string
      +--rw description?                               string
      +--ro tunnel-mtu                                 uint32
      +--ro lw4over6
      +--ro map-e
```

Figure 1: Softwire Common Data Model Structure

The mechanism specific models for lw4o6 and MAP-E are described in detail in the following sections.

3.2. Lightweight 4over6 Data Tree

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

```
   +--rw softwire-config
   |  +--...
   |  +--rw lw4over6
   |     +--rw lwaftrs
   |        +--rw lwaftr* [id]
   |           +--rw id                             uint32
   |           +--rw lwaftr-ipv6-addr               inet:ipv6-address
   |           +--rw binding-table
   |              +--rw binding-entry* [id]
   |                 +--rw id                       uint32
   |                 +--rw binding-ipv4-addr        inet:ipv4-address
   |                 +--rw port-set
   |                 |  +--rw offset                uint8
   |                 |  +--rw psid-len              uint8
   |                 |  +--rw psid                  inet:port-number
   |                 +--rw binding-ipv6-addr        inet:ipv6-address
   |                 +--rw active                   boolean
   +--ro softwire-state
      +--...
      +--ro lw4over6
         +--ro lwaftrs
            +--ro lwaftr* [id]
               +--ro id                             uint32
               +--ro lwaftr-ipv6-addr               inet:ipv6-address
               +--ro binding-table
                  +--ro binding-entry* [id]
                     +--ro id                       uint32
                     +--ro binding-ipv4-addr        inet:ipv4-address
                     +--ro port-set
                     |  +--ro offset                uint8
                     |  +--ro psid-len              uint8
                     |  +--ro psid                  inet:port-number
                     +--ro binding-ipv6-addr        inet:ipv6-address
                     +--ro active                   boolean
```

Figure 2: Softwire Lightweight 4over6 Data Model Structure

Node that the "active" item is used to determine whether the binding entry should be deleted.

3.3. MAP-E Data Tree

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

```
   +--rw softwire-config
   |  +--...
   |  +--rw map-e
   |     +--rw map-brs
   |        +--rw map-br* [id]
   |           +--rw id                                unit32
   |           +--rw br-ipv6-addr                      inet:ipv6-address
   |           +--rw map-rule-table
   |              +--rw map-rule-entry* [id]
   |                 +--rw id                          uint8
   |                 +--rw IPv6-prefix                 inet:ipv6-address
   |                 +--rw IPv6-prefix-len             uint8
   |                 +--rw IPv4-prefix                 inet:ipv4-address
   |                 +--rw IPv4-prefix-len             uint8
   |                 +--rw port-set
   |                 |  +--rw offset                   uint8
   |                 |  +--rw psid-len                 uint8
   |                 |  +--rw psid                     inet:port-number
   |                 +--rw ea-len                      uint8
   |                 +--rw active                      boolean
   +--ro softwire-state
      +--...
      +--ro map-e
         +--ro map-brs
            +--ro map-br* [id]
               +--ro id                                uint32
               +--ro br-ipv6-addr                      inet:ipv6-address
               +--ro map-rule-table
                  +--ro map-rule-entry* [id]
                  |  +--ro id                          uint8
                  |  +--ro IPv6-prefix                 inet:ipv6-address
                  |  +--ro IPv6-prefix-len             uint8
                  |  +--ro IPv4-prefix                 inet:ipv4-address
                  |  +--ro IPv4-prefix-len             uint8
                  |  +--ro port-set
                  |  |  +--ro offset                   uint8
                  |  |  +--ro psid-len                 uint8
                  |  |  +--ro psid                     inet:port-number
                  |  +--ro ea-len                      uint8
                  |  +--ro active                      boolean
                  +--ro active-map-rule-num            uint8
```

Figure 3: Softwire MAP-E Data Model Structure

4. Softwire YANG Module

This module imports typedefs from [RFC6991].

```
  <CODE BEGINS> file "ietf-softwire@2014-10-20.yang"

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

    import ietf-inet-types { prefix inet; }

    organization "softwire";

    contact
      "
      Qi Sun sunqi@csnet1.cs.tsinghua.edu.cn
      Hao Wang wangh13@mails.tsinghua.edu.cn
      Yong Cui yong@csnet1.cs.tsinghua.edu.cn
      Ian Farrer ian.farrer@telekom.de
      ";

    description
      "This document defines a YANG data model that can be used to
      configure and manage softwire concentrators.
      Copyright (c) 2014 IETF Trust and the persons identified
      as authors of the code. All rights reserved.
      This version of this YANG module is part of RFC XXX; see the RFC
      itself for full legal notices.";

    revision 2014-10-20 {
      description
        "Initial revision.";
    }

  /*
   * Typedef
   */


  /*
   * Features
   */


    feature lw4over6 {
      description
        "Lightweight 4over6 moves the Network Address and Port
```

```
        Translation (NAPT) function from the centralized DS-Lite tunnel
        concentrator to the tunnel client located in the Customer
        Premises Equipment (CPE).  This removes the requirement for a
        Carrier Grade NAT function in the tunnel concentrator and
        reduces the amount of centralized state that must be held to a
        per-subscriber level.  In order to delegate the NAPT function
        and make IPv4 Address sharing possible, port-restricted IPv4
        addresses are allocated to the CPEs.";
      reference
        "I-D.ietf-softwire-lw4over6";
    }

    feature map-e {
      description
        "MAP-E is a mechanism for transporting IPv4 packets across an
        IPv6 network using IP encapsulation, and a generic mechanism
        for mapping between IPv6 addresses and IPv4 addresses and
        transport layer ports.";
      reference
        "I-D.ietf-softwire-map";
    }

  /*
   * Grouping
   */

    grouping port-set {
      description
        "A range of transport layer ports.";
      leaf offset {
        type uint8;
        default "0";
        description
          "The number of offset bits.";
      }
      leaf psid-len {
        type uint8;
        description
          "The length of Port Set Identifier (PSID).";
      }
      leaf psid {
        type inet:port-number;
        description
          "Algorithmically identifies a set of ports.";
      }
    }

    grouping binding-table {
```

```
      description
        "The lwAFTR maintains an address binding table containing the
        binding between the lwB4's IPv6 address, the allocated IPv4
        address and restricted port-set.";
      list binding-entry {
        key "id";
        leaf id {
          type uint32;
        }
        leaf binding-ipv4-addr {
          type inet:ipv4-address;
          description
            "The IPv4 address assigned to a lwB4, which is used as the
            IPv4 External Address for lwB4 local NAPT44. One of three
            elemnts constructing a binding entry.";
        }
        container port-set {
          uses port-set;
        }
        leaf binding-ipv6-addr {
          type inet:ipv6-address;
          description
            "The IPv6 address of the lwB4, which is used to bind the
            IPv4 address and port-set.";
        }
        leaf active {
          type boolean;
          description
            "Used to delete the inactive binding-entries.";
        }
      }
    }

    grouping map-rule-table {
      description
        "The (conceptual) table containing rule Information for
        a specific mapping rule. It can also be used for row creation.";
      list map-rule-entry {
        key "id";
        leaf id {
          type uint8;
        }
        leaf IPv6-prefix {
          type inet:ipv6-address;
          description
            "The IPv6 prefix defined in the mapping rule which will be
            assigned to CE.";
        }
```

```
        leaf IPv6-prefix-len {
          type uint8;
          description
            "The length of the IPv6 prefix defined in the mapping rule.
            As a parameter for the mapping rule, it will be also assigned
            to CE.";
        }
        leaf IPv4-prefix {
          type inet:ipv4-address;
          description
            "The IPv4 prefix defined in the mapping rule which will be
            assigned to CE.";
        }
        leaf IPv4-prefix-len {
          type uint8;
          description
            "The length of the IPv4 prefix defined in the mapping
            rule. As a parameter for the mapping rule, it will be also
            assigned to CE.";
        }
        container port-set {
          uses port-set;
        }
        leaf ea-len {
          type uint8;
          description
            "The length of the Embedded-Address (EA) defined in
            mapping rule which will be assigned to CE.";
        }
        leaf active {
          type boolean;
          description
              "Used to delete inactive map-rule-entries.";
        }
      }
    }

  /*
   * Configuration Data Nodes
   */

    container softwire-config {
      description
        "The configuration data for concentrators in softwire. ";
      leaf enabled {
        type boolean;
        default "true";
        description
```

```
          "Enable/disable the Softwire concentrator function.";
      }
      leaf name {
        type string;
        description
          "The name of the softwire concentrator.";
      }
      leaf description {
        type string;
        description
          "A textual description of the softwire concentrator.";
      }
      leaf softwire-num-threshold {
        type uint32;
        description
          "The maximum number of tunnels that can be created on
          the concentrator.";
      }
      leaf tunnel-mtu {
        type uint32;
        description
          "The MTU of tunnel payload on the softwire concentrator.";
      }
      container lw4over6 {
        if-feature lw4over6;
        description
          "Indicate this device supports the Lightweight 4over6 function.
          Devices advertise the lw4over6 feature through the capability
          exchange mechanism when a NETCONF session is established.";
        container lwaftrs {
          description
            "An AFTR element (Address Family Transition Router element
            [RFC6333]), which supports Lightweight 4over6 extension.
            An lwAFTR is an IPv4-in-IPv6 tunnel endpoint which maintains
            per-subscriber address binding only and does not perform a
            NAPT44 function.";
          list lwaftr {
            key "id";
            leaf id {
              type uint32;
            }
            leaf lwaftr-ipv6-addr {
              type inet:ipv6-address;
              description
                "The IPv6 address of the lwAFTR.";
            }
            container binding-table {
              uses binding-table;
```

```
            }
          }
        }
      }
      container map-e {
        if-feature map-e;
        description
          "Indicate the devices support the MAP-E function. Devices
          advertise the map-e feature through the capability exchange
          mechanism when a NETCONF session is established.";
        container map-brs {
          description
            "A MAP enabled router managed by the service provider at
            the edge of a MAP domain.
            A Border Relay router has at least an IPv6-enabled interface
            and an IPv4 interface connected to the native IPv4 network.
            A MAP BR may also be referred to simply as a "BR" within the
            context of MAP.";
          list map-br {
            key "id";
            leaf id {
              type uint32;
            }
            leaf br-ipv6-addr {
              type inet:ipv6-address;
              description
                "The IPv6 address of the Border Router.";
            }
            container map-rule-table {
              uses map-rule-table;
            }
          }
        }
      }
    }

  /*
   * Operational state Data Nodes
   */

    container softwire-state {
      config false;
      description
        "The operational state data for concentrators in softwire. ";
      leaf enabled {
        type boolean;
        description
          "Indicate if the Softwire concentrator function is enabled/
```

```
          disbaled.";
      }
      leaf name {
        type string;
        description
          "The name of the softwire concentrator.";
      }
      leaf description {
        type string;
        description
          "A textual description of the softwire concentrator.";
      }
      leaf tunnel-mtu {
        type uint32;
        description
          "The MTU of tunnel payload on the softwire concentrator.";
      }

      container lw4over6 {
        if-feature lw4over6;
        description
          "Indicate the device support the Lightweight 4over6 function.
          Device advertise the lw4over6 feature through the capability
          exchange mechanism when a NETCONF session is established.";
        container lwaftrs {
          description
            "An AFTR element (Address Family Transition Router element
            [RFC6333]), which supports Lightweight 4over6 extension.
            An lwAFTR is an IPv4-in-IPv6 tunnel endpoint which maintains
            per-subscriber address binding only and does not perform a
            NAPT44 function.";
          list lwaftr {
            key "id";
            leaf id {
              type uint32;
            }
            leaf lwaftr-ipv6-addr {
              type inet:ipv6-address;
              description
                "The IPv6 address of the lwAFTR.";
            }
            container binding-table {
              uses binding-table;
            }
          }
        }
      }
      container map-e {
```

```
        if-feature map-e;
        description
          "Indicate the device support the MAP-E function. Device
          advertise the map-e feature through the capability exchange
          mechanism when a NETCONF session is established.";
        container map-brs {
          description
            "A MAP enabled router managed by the service provider at
            the edge of a MAP domain.
            A Border Relay router has at least an IPv6-enabled interface
            and an IPv4 interface connected to the native IPv4 network.
            A MAP BR may also be referred to simply as a "BR" within the
            context of MAP.";
          list map-br {
            key "id";
            leaf id {
              type uint32;
            }
            leaf br-ipv6-addr {
              type inet:ipv6-address;
              description
                "The IPv6 address of the Border Router.";
            }
            container map-rule-table {
              uses map-rule-table;
              leaf active-map-rule-num {
                type uint8;
                description
                "The number of map rules which are active.";
              }
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

5. Security Considerations

TBD

6. IANA Considerations

TBD

7. Acknowledgements

The authors would like to thank Lishan Li for her contributions to this work.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

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

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

8.2. Informative References

[I-D.ietf-softwire-lw4over6]
Cui, Y., Qiong, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the DS-Lite Architecture", draft-ietf-softwire-lw4over6-11 (work in progress), October 2014.

[I-D.ietf-softwire-map]
Troan, O., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, "Mapping of Address and Port with Encapsulation (MAP)", draft-ietf-softwire-map-11 (work in progress), October 2014.

[I-D.ietf-softwire-map-dhcp]
Mrugalski, T., Troan, O., Farrer, I., Perreault, S., Dec, W., Bao, C., leaf.yeh.sdo@gmail.com, l., and X. Deng, "DHCPv6 Options for configuration of Softwire Address and Port Mapped Clients", draft-ietf-softwire-map-dhcp-09 (work in progress), October 2014.

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

Authors' Addresses

Qi Sun
Tsinghua University
Beijing 100084
P.R. China

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

Hao Wang
Tsinghua University
Beijing 100084
P.R. China

Phone: +86-10-6278-5822
Email: wangh13@mails.tsinghua.edu.cn

Yong Cui
Tsinghua University
Beijing 100084
P.R. China

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

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

Email: ian.farrer@telekom.de