Softwire WG
Internet-Draft
Intended status: Standards Track
Expires: December 5, 2015

Q. Wang
China Telecom
W. Meng
C. Wang
ZTE Corporation
M. Boucadair
France Telecom
June 3, 2015

RADIUS Extensions for IPv4-Embedded Multicast and Unicast IPv6 Prefixes
draft-hu-softwire-multicast-radius-ext-08

Abstract

This document specifies a new Remote Authentication Dial-In User Service (RADIUS) attribute to carry the Multicast-Prefixes-64 information, aiming to delivery the Multicast and Unicast IPv6 Prefixes to be used to build multicast and unicast IPv4-Embedded IPv6 addresses. this RADIUS attribute is defined based on the equivalent DHCPv6 OPTION_v6_PREFIX64 option.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The solution specified in [I-D.ietf-softwire-dslite-multicast] relies on stateless functions to graft part of the IPv6 multicast distribution tree and IPv4 multicast distribution tree, also uses IPv4-in-IPv6 encapsulation scheme to deliver IPv4 multicast traffic over an IPv6 multicast-enabled network to IPv4 receivers.

To inform the mB4 element of the PREFIX64,a PREFIX64 option may be used. [I-D.ietf-softwire-multicast-prefix-option] defines a DHCPv6 PREFIX64 option to convey the IPv6 prefixes to be used for constructing IPv4-embedded IPv6 addresses.

In broadband environments, a customer profile may be managed by Authentication, Authorization, and Accounting (AAA) servers, together with AAA for users. The Remote Authentication Dial-In User Service (RADIUS) protocol [RFC2865] is usually used by AAA servers to communicate with network elements. Since the Multicast-Prefixes-64 information can be stored in AAA servers and the client configuration is mainly provided through DHCP running between the NAS and the requesting clients, a new RADIUS attribute is needed to send Multicast-Prefixes-64 information from the AAA server to the NAS.

This document defines a new RADIUS attribute to be used for carrying the Multicast-Prefixes-64, based on the equivalent DHCPv6 option already specified in [I-D.ietf-softwire-multicast-prefix-option].

This document makes use of the same terminology defined in [I-D.ietf-softwire-dslite-multicast].

This attribute can be in particular used in the context of DS-Lite Mulitcast, MAP-E Multicast and other IPv4-IPv6 Multicast techniques. However it is not limited to DS-Lite Multicast.

DS-Lite unicast RADIUS extentions are defined in [RFC6519] .

2. Convention and 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 terms DS-Lite multicast Basic Bridging BroadBand element (mB4) and the DS-Lite multicast Address Family Transition Router element (mAFTR) are defined in [I-D.ietf-softwire-dslite-multicast]

3. Multicast-Prefixes-64 Configuration with RADIUS and DHCPv6

Figure 1 illustrates in DS-Lite scenario how the RADIUS protocol and DHCPv6 work together to accomplish Multicast-Prefixes-64 configuration on the mB4 element for multicast service when an IP session is used to provide connectivity to the user.

```
     mB4                                NAS                     AAA
      |                                  |                     Server
      |------ DHCPv6 Solicit --------->  |                       |
      |                                  |                       |
      |                                  |----Access-Request ---->|
      |                                  |                       |
      |                                  |<---Access-Accept-------|
      |                                  |(Multicast-Prefixes-64) |
      |                                  |
                |
      |<-------  DHCPv6 Advertise  ------|                       |
      |   (DHCPv6 OPTION_V6_PREFIX64 )   |                       |
      |                                  |                       |
      |-------  DHCPv6 Request  -------->|                       |
      |                                  |                       |
      |                                  |                       |
      |<----- DHCPv6 Reply ------------- |                       |
      |   (DHCPv6 OPTION_V6_PREFIX64 )   |                       |

                  DHCPv6                          RADIUS
```

Figure 1: RADIUS and DHCPv6 Message Flow for an IP Session

The NAS operates as a client of RADIUS and as a DHCP Server/Relay for mB4. When the mB4 sends a DHCPv6 Solicit message to NAS(DHCP Server/Relay). The NAS sends a RADIUS Access-Request message to the RADIUS server, requesting authentication. Once the RADIUS server receives the request, it validates the sending client, and if the request is approved, the AAA server replies with an Access-Accept message including a list of attribute-value pairs that describe the parameters to be used for this session. This list MAY contain the Multicast-Prefixes-64 attribute (asm-length,ASM_PREFIX64,ssm-length, SSM_PREFIX64,unicast-length,U_PREFIX64). Then, when the NAS receives the DHCPv6 Request message containing the OPTION_V6_PREFIX64 option in its Option Request option,the NAS SHALL use the prefixes returned in the RADIUS Multicast-Prefixes-64 attribute to populate the DHCPv6 OPTION_V6_PREFIX64 option in the DHCPv6 reply message.

NAS MAY be configured to return the configured Multicast-Prefixes-64 by the AAA Server to any requesting client without relaying each received request to the AAA Server.

Figure 2 describes another scenario, which accomplish DS-Lite Multicast-Prefixes-64 configuration on the mB4 element for multicast service when a PPP session is used to provide connectivity to the user. Once the NAS obtains the Multicast-Prefixes-64 attribute from the AAA server through the RADIUS protocol, the NAS MUST store the received Multicast-Prefixes-64 locally. When a user is online and sends a DHCPv6 Request message containing the OPTION_V6_PREFIX64 option in its Option Request option, the NAS retrieves the previously stored Multicast-Prefixes-64 and uses it as OPTION_V6_PREFIX64 option in DHCPv6 Reply message.

```
      mB4                                NAS                    AAA
      |                                  |                    Server
      |                                  |                       |
      |----PPP LCP Config-Request------> |                       |
      |                                  |                       |
      |                                  |----Access-Request ---->|
      |                                  |                       |
      |                                  |<---- Access-Accept-----|
      |                                  | (Multicast-Prefixes-64)|
      |<-----PPP LCP Config-ACK  ------- |                       |
      |                                  |                       |
      |                                  |                       |
      |--- PPP IPv6CP Config-Request --->|                       |
      |                                  |                       |
      |<----- PPP IPv6CP Config-ACK -----|                       |
      |                                  |                       |
      |-------  DHCPv6 Solicit  -------->|                       |
      |                                  |                       |
      |<-------  DHCPv6 Advertise   -----|                       |
      |    (DHCPv6 OPTION_V6_PREFIX64 )  |                       |
      |                                  |                       |
      |-------  DHCPv6 Request  -------->|                       |
      |                                                    |
      |
      |                                  |                       |
      |<-------- DHCPv6 Reply ---------- |                       |
      |    (DHCPv6 OPTION_V6_PREFIX64)   |                       |

                  DHCPv6                          RADIUS
```

Figure 2: RADIUS and DHCPv6 Message Flow for a PPP Session

According to [RFC3315], after receiving the Multicast-Prefixes-64 attribute in the initial Access-Accept packet, the NAS MUST store the received V6_PREFIX64 locally. When the mB4 sends a DHCPv6 Renew message to request an extension of the lifetimes for the assigned address or prefix, the NAS does not have to initiate a new Access-

Request packet towards the AAA server to request the Multicast-Prefixes-64. The NAS retrieves the previously stored Multicast-Prefixes-64 and uses it in its reply.

Also, if the DHCPv6 server to which the DHCPv6 Renew message was sent at time T1 has not responded, the DHCPv6 client initiates a Rebind/Reply message exchange with any available server. In this scenario, the NAS receiving the DHCPv6 Rebind message MUST initiate a new Access-Request message towards the AAA server. The NAS MAY include the Multicast-Prefixes-64 attribute in its Access-Request message.

4. RADIUS Attribute

This section specifies the format of the new RADIUS attribute.

4.1. Multicast-Prefixes-64

The Multicast-Prefixes-64 attribute conveys the IPv6 prefixes to be used in [I-D.ietf-softwire-dslite-multicast] to synthesize IPv4-embedded IPv6 addresses. The NAS SHALL use the IPv6 prefixes returned in the RADIUS Multicast-Prefixes-64 attribute to populate the DHCPv6 PREFIX64 Option [I-D.ietf-softwire-multicast-prefix-option] .

This attribute MAY be used in Access-Request packets as a hint to the RADIUS server, for example, if the NAS is pre-configured with Multicast-Prefixes-64, these prefixes MAY be inserted in the attribute. The RADIUS server MAY ignore the hint sent by the NAS, and it MAY assign a different Multicast-Prefixes-64 attribute.

If the NAS includes the Multicast-Prefixes-64 attribute, but the AAA server does not recognize this attribute, this attribute MUST be ignored by the AAA server.

NAS MAY be configured with both ASM_PREFIX64 and SSM_PREFIX64 or only one of them. Concretely, AAA server MAY return ASM_PREFIX64 or SSM_PREFIX64 based on the user profile and service policies. AAA MAY return both ASM_PREFIX64 and SSM_PREFIX64. When SSM_PREFIX64 is returned by the AAA server, U_PREFIX64 MUST also be returned by the AAA server.

If the NAS does not receive the Multicast-Prefixes-64 attribute in the Access-Accept message, it MAY fall back to a pre-configured default Multicast-Prefixes-64, if any. If the NAS does not have any pre-configured, the delivery of multicast traffic is not supported.

If the NAS is pre-provisioned with a default Multicast-Prefixes-64 and the Multicast-Prefixes-64 received in the Access-Accept message are different from the configured default, then the Multicast-Prefixes-64 attribute received in the Access-Accept message MUST be used for the session.

A summary of the Multicast-Prefixes-64 RADIUS attribute format is shown Figure 3. The fields are transmitted from left to right.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Length    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  asm-length   |               |
+-+-+-+-+-+-+-+-+               :
:    ASM_PREFIX64 (variable)    :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  ssm-length   |               |
+-+-+-+-+-+-+-+-+               :
:    SSM_PREFIX64 (variable)    :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| unicast-length|               |
+-+-+-+-+-+-+-+-+               :
:     U_PREFIX64 (variable)     :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: RADIUS attribute format for Multicast-Prefixes-64

Type:

145 for Multicast-Prefixes-64

Length:

This field indicates the total length in octets of this attribute including the Type and Length fields, and the length in octets of all PREFIX fields.

asm-length:

the prefix-length for the ASM IPv4-embedded prefix, as an 8-bit unsigned integer (0 to 128). This field represents the number of valid leading bits in the prefix.

ASM_PREFIX64:

this field identifies the IPv6 multicast prefix to be used to synthesize the IPv4-embedded IPv6 addresses of the multicast groups in the ASM mode. It is a variable size field with the length of the field defined by the asm-length field and is rounded up to the nearest octet boundary. In such case any additional padding bits must be zeroed. The conveyed multicast IPv6 prefix MUST belong to the ASM range. This prefix is likely to be a /96.

ssm-length:

the prefix-length for the SSM IPv4-embedded prefix, as an 8-bit unsigned integer (0 to 128). This field represents the number of valid leading bits in the prefix.

SSM_PREFIX64:

this field identifies the IPv6 multicast prefix to be used to synthesize the IPv4-embedded IPv6 addresses of the multicast groups in the SSM mode. It is a variable size field with the length of the field defined by the ssm-length field and is rounded up to the nearest octet boundary. In such case any additional padding bits must be zeroed. The conveyed multicast IPv6 prefix MUST belong to the SSM range. This prefix is likely to be a /96.

unicast-length:

the prefix-length for the IPv6 unicast prefix to be used to synthesize the IPv4-embedded IPv6 addresses of the multicast sources, as an 8-bit unsigned integer (0 to 128). This field represents the number of valid leading bits in the prefix.

U_PREFIX64:

this field identifies the IPv6 unicast prefix to be used in SSM mode for constructing the IPv4-embedded IPv6 addresses representing the IPv4 multicast sources in the IPv6 domain. U_PREFIX64 may also be used to extract the IPv4 address from the received multicast data flows. It is a variable size field with the length of the field defined by the unicast-length field and is rounded up to the nearest octet boundary. In such case any additional padding bits must be zeroed. The address mapping MUST follow the guidelines documented in [RFC6052].

5. Table of Attributes

The following tables provide a guide to which attributes may be found in which kinds of packets, and in what quantity.

The following table defines the meaning of the above table entries.

Access-Request	Access-Accept	Access-Reject	Challenge	Accounting-Request	#	Attribute
0-1	0-1	0	0	0-1	145	Multicast-Prefixes-64

CoA-Request	CoA-ACK	CoA-NACK	#	Attribute
0-1	0	0	145	Multicast-Prefixes-64

0 This attribute MUST NOT be present in the packet.
0+ Zero or more instances of this attribute MAY be present in the packet.
0-1 Zero or one instances of this attribute MAY be present in the packet.
1 Exactly one instances of this attribute MAY be present in the packet.

6. Security Considerations

This document has no additional security considerations beyond those already identified in [RFC2865] for the RADIUS protocol and in [RFC5176] for CoA messages.

The security considerations documented in [RFC3315] and [RFC6052] are to be considered.

7. IANA Considerations

Per this document, IANA has allocated a new RADIUS attribute type from the IANA registry "Radius Attribute Types" located at http://www.iana.org/assignments/radius-types.

Multicast-Prefixes-64 - 145

8. Acknowledgments

The authors would like to thank Ian Farrer, Chongfen Xie, Qi Sun, Linhui Sun and Hao Wang for their contributions to this work.

9. Normative References

[I-D.ietf-softwire-dslite-multicast]
Qin, J., Boucadair, M., Jacquenet, C., Lee, Y., and Q. Wang, "Delivery of IPv4 Multicast Services to IPv4 Clients over an IPv6 Multicast Network", draft-ietf-softwire-dslite-multicast-09 (work in progress), March 2015.

[I-D.ietf-softwire-multicast-prefix-option]
Boucadair, M., Qin, J., Tsou, T., and X. Deng, "DHCPv6 Option for IPv4-Embedded Multicast and Unicast IPv6 Prefixes", draft-ietf-softwire-multicast-prefix-option-08 (work in progress), March 2015.

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

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[RFC5176] Chiba, M., Dommety, G., Eklund, M., Mitton, D., and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.

[RFC6519] Maglione, R. and A. Durand, "RADIUS Extensions for Dual-Stack Lite", RFC 6519, February 2012.

Authors' Addresses

Qian Wang
China Telecom
No.118, Xizhimennei
Beijing 100035
China

Email: wangqian@ctbri.com.cn

Wei Meng
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing
China

Email: meng.wei2@zte.com.cn,vally.meng@gmail.com

Cui Wang
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing
China

Email: wang.cui1@zte.com.cn

Mohamed Boucadair
France Telecom
Rennes, 35000
France

Email: mohamed.boucadair@orange.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 10, 2015

B. Sarikaya
Huawei USA
H. Ji
China Telecom

June 8, 2015

Multicast Support for Mapping of Address and Port Protocol and Light Weight 4over6
draft-sarikaya-softwire-map-multicast-04

Abstract

This memo specifies multicast component for MAP and Light Weight 4over6 so that IPv4 hosts can receive multicast data from IPv4 servers over an IPv6 network. The solution developed is based on translation. In the Translation Multicast solution for MAP (MAP-E and MAP-T) and lw4o6, IGMP messages are translated into MLD messages and sent to the network in IPv6. MAP Border Relay/lwAFTR does the reverse translation and joins IPv4 multicast group for the hosts. Border Relay/lwAFTR as multicast router receives IPv4 multicast data and translates the packet into IPv6 multicast data and sends downstream on the multicast tree. Member CEs/lwB4s receive multicast data, translate it back to IPv4 and transmit to the hosts.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 10, 2015.

Table of Contents

1. Introduction

With IPv4 address depletion on the horizon, many techniques are being standardized for IPv6 migration including Mapping of Address and Port (MAP) - Encapsulation, - Translation and 4rd [I-D.ietf-softwire-map], [I-D.ietf-softwire-map-t], [I-D.ietf-softwire-4rd]. MAP/4rd enables IPv4 hosts to communicate with external hosts using IPv6 only ISP network. MAP/4rd Customer Edge (CE) device's LAN side is dual stack and WAN side is IPv6 only. CE tunnels/translates IPv4 packets received from the LAN side to 4rd Border Relays (BR). BRs have anycast IPv6 addresses and receive encapsulated/translated packets from CEs over a virtual interface. MAP/4rd operation is stateless. Packets are received/ sent independent of each other and no state needs to be maintained except for NAT44 operation on IPv4 packets received from the user.

Light Weight 4 Over 6 (lw4o6) is a variant of Dual Stack Lite where carrier grade NAT is moved from AFTR to B4 element, i.e. NAPT is done locally at each B4 called light weight B4 or lwB4. Unicast lw4o6 takes user IPv4 packets from the local LAN and lwB4 does a NAPT and then tunnels the packets in an IPv4-in-IPv6 tunnel to lwAFTR which decapsulates the packet and then sends it to IPv4 network. Incoming packets follow reverse route and are encapsulated at lwAFTR and sent to lwB4 which decapsulates and after NAPT operation transmits to the destination.

It should be noted that there is no depletion problem for IPv4 address space allocated for any source multicast and source specific multicast [RFC3171]. This document is not motivated by the depletion of IPv4 multicast addresses.

MAP-E, MAP-T, 4rd and lw4o6 are unicast only. They do not support multicast. In this document we specify how multicast from home IPv4 users can be supported in MAP-E (as well as MAP-T and 4rd) and lw4o6.

In case IPv6 network is multicast enabled, MAP-T/4rd can provide multicast service to the hosts using MAP-T/4rd Multicast Translation based solution. A Multicast Translator can be used that receives IPv4 multicast group management messages in IGMP and generates corresponding IPv6 group management messages in MLD and sends them to IPv6 network towards MAP-T/4rd Border Relay. We use [I-D.ietf-softwire-map-t] or [I-D.ietf-softwire-4rd] for sending IPv4 multicast data in IPv6 to the CE routers. At MAP-T/4rd CE router another translator is needed to translate IPv6 multicast data into IPv4 multicast data.

It should be noted that if IPv6 network is multicast enabled the translation multicast solution presented in Section 4 can also be used for MAP-E.

In this document we address MAP-E (and MAP-T/4rd) and lw4o6 multicast problem and propose the architecture of Multicast Translation based solution. Section 2 is on terminology, Section 3 is on architecture, Section 4 is on multicast translation protocol, and Section 5 states security considerations.

2. Terminology

This document uses the terminology defined in [I-D.ietf-softwire-map], [I-D.ietf-softwire-lw4over6], [I-D.ietf-softwire-map-t], [I-D.ietf-softwire-4rd], [RFC3810] and [RFC3376].

3. Architecture

In MAP-E, MAP-T and 4rd, there are hosts (possibly IPv4/ IPv6 dual stack) served by MAP-E, MAP-T and 4rd Customer Edge device. CE is dual stack facing the hosts and IPv6 only facing the network or WAN side. MAP-E, MAP-T and 4rd CE may be local IPv4 Network Address and Port Translation (NAPT) box [RFC3022] by assigning private IPv4 addresses to the hosts. MAP-E, MAP-T and 4rd CEs in the same domain may use shared public IPv4 addresses on their WAN side and if they do they should avoid ports outside of the allocated port set for NAPT operation. At the boundary of the network there is MAP-E, MAP-T and 4rd Border Relay. BR receives IPv4 packets tunneled in IPv6 from CE and decapsulates them and sends them out to IPv4 network.

Unicast MAP-E, MAP-T and 4rd are stateless except for the local NAPT at the CE. Each IPv4 packet sent by CE treated separately and different packets from the same CE may go to different BRs or CEs. CE encapsulates IPv4 packet in IPv6 with destination address set to BR address (usually anycast IPv6 address). BR receives the encapsulated packet and decapsulates and send it to IPv4 network. CEs are configured with Rule IPv4 Prefixes, Rule IPv6 Prefixes and with an BR IPv6 anycast address. BR receives IPv4 packets addressed to this ISP and from the destination address it extracts the destination host’s IPv4 address and uses this address as destination address and encapsulates the IPv4 packet in IPv6 and sends it to IPv6-only network.

Unicast Lightweight 4over6 (lw4o6) is a variation of Dual-Stack Lite (DS-Lite) [RFC6333] which moves carrier-grade IPv4-IPv4 NAT from the Address Family Transition Router (AFTR) element to the Basic Bridging BroadBand (B4) element [I-D.ietf-softwire-lw4over6]. The resulting elements are called lwAFTR and lwB4 with NAPT, respectively. Lw4o6 also adopts some features from MAP-E. A+P scheme of public IPv4 address sharing is used by lwB4’s in assigning WAN side IPv4 public addresses with a distinct port set. As in MAP-E, encapsulation of IPv4 packets in IPv6 and decapsulation is according to [RFC2473].

3.1. MAP-T and 4rd Translation Architecture

In case IPv6 only network is multicast enabled, translation multicast architecture can be used. CE implements IGMP Proxy function [RFC4605] towards the LAN side and MLD Proxy on its WAN interface. IPv4 hosts send their join requests (IGMP Membership Report messages) to CE. CE as a MLD proxy sends aggregated MLD Report messages upstream towards BR. CE replies MLD membership query messages with MLD membership report messages based on IGMP membership state in the IGMP/MLD proxy.

BR is MLD querier on its WAN side. On its interface to IPv4 network BR may either have IGMP client or PIM. PIM being able to support both IPv4 and IPv6 multicast should be preferred. BR receives MLD join requests, extracts IPv4 multicast group address and then joins the group upstream, possibly by issuing a PIM join message.

IPv4 multicast data received by the BR as a leaf node in IPv4 multicast distribution tree is translated into IPv6 multicast data by the translator using [I-D.ietf-softwire-map-t], [I-D.ietf-softwire-4rd] and then sent downstream to the IPv6 part of the multicast tree to all downstream routers that are members. IPv6 data packet eventually gets to the CE. At the CE, a reverse [I-D.ietf-softwire-map-t], [I-D.ietf-softwire-4rd] operation takes place by the translator and then IPv4 multicast data packet is sent to the member hosts on the LAN interface. [I-D.ietf-softwire-map-t], [I-D.ietf-softwire-4rd] are modified to handle multicast addresses.

In order to support SSM, IGMPv3 MUST be supported by the host, CE and BR. For ASM, BR MUST be the Rendezvous Point (RP).

MAP-T and 4rd Translation Multicast solution uses the multicast 46 translator in not one but two places in the architecture: at the CE router and at the Border Relay. IPv4 multicast data received at 4rd BR goes through a [I-D.ietf-softwire-4rd] header-mapping into IPv6 multicast data at the BR and another [I-D.ietf-softwire-4rd] header-mapping back to IPv4 multicast data at the CE router. Encapsulation variant of [I-D.ietf-softwire-4rd] is not used. In case of MAP-T, IPv4 data packet is translated using v4 to v6 header translation using multicast addresses instead of the mapping algorithm used in [I-D.ietf-softwire-map-t].

All the elements of MAP-T and 4rd translation-based multicast support system are shown in Figure 1.

```
  Dual Stack Hosts                                                IPv4
   +----+                                                       Network
   | H1 |      +----------+          IPv6       +-------------+
   |    |      |    CE    |                     |     BR      |
   +----+      |Translator|          only       | Translator  |
               |MAP-T/4rd |                     |  MAP-T/4rd  |
   +----+      |          |          network    |        |IGMP|     +
   | H2 |   ---|IGMP-MLD  |---------         -- |MLD     | or |    IPv6
   +----+      |    Proxy |                     |Querier |PIM |   Network
   +----+      +----------+                     +-------------+
   | H3 |
   +----+
```

Figure 1: Architecture of MAP-T and 4rd Translation Multicast

In case IPv6 only network is multicast enabled, translation multicast architecture can also be used for lw4o6 multicast. lwB4 implements IGMP Proxy function [RFC4605] towards the LAN side and MLD Proxy on its WAN interface. IPv4 hosts send their join requests (IGMP Membership Report messages) to lwB4. lwB4 as a MLD proxy sends aggregated MLD Report messages upstream towards lwAFTR. lwB4 replies MLD membership query messages with MLD membership report messages based on IGMP membership state in the IGMP/MLD proxy.

lwAFTR is MLD querier on its WAN side. On its interface to IPv4 network lwAFTR may either have IGMP client or PIM. PIM being able to support both IPv4 and IPv6 multicast should be preferred. lwAFTR receives MLD join requests, extracts IPv4 multicast group address and then joins the group upstream, possibly by issuing a PIM join message.

For multicast data, [I-D.ietf-softwire-dslite-multicast] uses encapsulation of IPv4 multicast data in IPv6 multicast data packet but in this document we use translation. IPv4 multicast data received by the lwAFTR as a leaf node in IPv4 multicast distribution tree is translated into IPv6 multicast data by the translator and then sent downstream to the IPv6 part of the multicast tree to all downstream routers that are members. IPv6 data packet eventually gets to the lwB4. At the lwB4, a reverse translation operation takes place by the translator and then IPv4 multicast data packet is sent to the member hosts on the LAN interface. The translation algorithm in [I-D.ietf-softwire-map-t], [I-D.ietf-softwire-4rd] are modified to handle multicast addresses.

In order to support SSM, IGMPv3 MUST be supported by the host, lwB4 and lwAFTR. For ASM, lwAFTR MUST be the Rendezvous Point (RP).

MAP-T and 4rd Translation Multicast solution uses the multicast 46 translator in not one but two places in the architecture: at the lwB4 router and at the lwAFTR. IPv4 data packet is translated using v4 to v6 header translation using multicast addresses instead of the mapping algorithm used in [I-D.ietf-softwire-map-t].

All the elements of lw4o6 translation-based multicast support system are shown in Figure 2.

```
                                                                 IPv4
   +-------+ IPv4                                               Network
   |       |Multicast
   |       |      +--------------+      +--------------+      +
   |       |      |  lwB4   NAPT |IPv6  |   lwAFTR     |
   |  IPv4 |----->| IGMP    MLD  |----->|IGMP  | IGMP  |     IPv6
   |  LAN  |<-----| Proxy  Proxy |------|Router|  PIM  |    Network
   |       |      +--------------+      +--------------+
   |       |
   +-------+
```

Figure 2: Architecture of lw4o6 Multicast Translation

4. MAP-T and 4rd Translation Multicast Operation

In this section we specify how the host can subscribe and receive IPv4 multicast data from IPv4 content providers based on the architecture defined in Figure 1 in two parts: address translation and protocol translation. Translation details are given in Appendix A.

4.1. Address Translation

IPv4-only host, H1 will join IPv4 multicast group by sending IGMP Membership Report message upstream towards the IGMP Proxy in Figure 1. MLD Proxy first creates a synthesized IPv6 address of IPv4 multicast group address using IPv4-embedded IPv6 multicast address format [I-D.ietf-mboned-64-multicast-address-format]. ASM_MPREFIX64 for any source multicast groups and SSM_MPREFIX64 for source specific multicast groups are used. Both are /96 prefixes.

SSM_MPREFIX64 is set to ff3x:0:8000::/96, with ’x’ set to any valid scope. ASM_MPREFIX64 values are formed as shown in Figure 3. Flag field 1 (ff1) field is defined in [RFC7371] bits M bit MUST BE set to 1. "scop" field is defined in [RFC3956]. Flag field 2 (ff2) is a set of 4 flags rrrM where r bits MUST be set to zero. M bit is set to 1 to indicate that a multicast IPv4 address is embedded in the low-

order 32 bits of the multicast IPv6 address. "sub-group-id" field MUST follow the recommendations specified in [RFC3306] if unicast-based prefix is used or the recommendations specified in [RFC3956] if embedded-RP is used. The default value is all zeros.

```
|   8    |  4 |  4 |  4 |             76               |    32    |
+--------+----+----+----+------------------------------+----------+
|11111111|ff1 |scop|ff2 |         sub-group-id         |v4 address|
+--------+----+----+----+----------------------------------------+
```

Figure 3: ASM_MPREFIX64 Formation

Each translator in the upstream BR is assigned a unique ASM_MPREFIX64 prefix. CE (MLD Proxy in CE) can learn this value by means out of scope with this document. With this, CE can easily create an IPv6 multicast address from the IPv4 group address a.b.c.d that the host wants to join.

Source-Specific Multicast (SSM) can also be supported similar to the Any Source Multicast (ASM) described above. In case of SSM, IPv4 multicast addresses use 232.0.0.0/8 prefix. IPv6 SSM_MPREFIX64 is set to FF3x:0:8000::/96 where ’x’ is any valid scope.

Since SSM translation requires a unique address for each IPv4 multicast source, an IPv6 unicast prefix must be configured to the translator in the upstream BR to represent IPv4 sources. This prefix is prepended to IPv4 source addresses in translated packets.

The join message from the host for the group ASM_MPREFIX64:a.b.c.d or SSM_MPREFIX64:a.b.c.d or an aggregate join message will be received by MLD querier at the BR. BR as multicast anchor checks the group address and recognizes ASM_MPREFIX64 or SSM_MPREFIX64 prefix. It next checks the last 32 bits is an IPv4 multicast address in range 224/8 - 239/8. If all checks succeed, IGMPv4 Client joins a.b.c.d using IGMP on its IPv4 interface.

Joining IPv4 groups can also be done using PIM since PIM supports both IPv4 and IPv6. The advantage of using PIM is that there is no need to enable IGMP support in neighboring IPv4 routers. The advantage of using IGMP is that IGMP is a simpler protocol and it is supported by a wider range of routers. The use of PIM or IGMP is left as an implementation choice.

Address translation described above for MAP-T applies to lw4over6 multicast translation where the entities involved are lwB4 replaces Customer Edge device and lwAFTR replaces BR Figure 2.

4.2. Protocol Translation

The hosts will send their subscription requests for IPv4 multicast groups upstream to the default router, i.e. Costumer Edge device. After subscribing the group, the host can receive multicast data from the CE. The host implements IGMP protocol's host part.

Customer Edge device is IGMP Proxy facing the LAN interface. After receiving the first IGMP Report message requesting subscription to an IPv4 multicast group, a.b.c.d, MLD Proxy in the CE's WAN interface synthesizes an IPv6 multicast group address corresponding to a.b.c.d and sends an MLD Report message upstream to join the group.

When MAP-T or 4rd BR receives IPv4 multicast data for an IPv4 group a.b.c.d it [I-D.ietf-softwire-4rd] translates/encapsulates IPv4 packet into IPv6 multicast packet and sends it to IPv6 synthesized address corresponding to a.b.c.d using ASM_MPREFIX64 or SSM_MPREFIX64. The header mapping described in [I-D.ietf-softwire-4rd] Section 4.2 (using Table 1) is used except for mapping the source and destination addresses. In this document we use the multicast address translation described in Section 4.1 and propose it as a complementary enhancement to the translation algorithm in [I-D.ietf-softwire-4rd].

The IP/ICMP translation translates IPv4 packets into IPv6 using minimum MTU size of 1280 bytes (Section 4.3 in [I-D.ietf-softwire-4rd]) but this can be changed for multicast. Path MTU discovery for multicast is possible in IPv6 so 4rd BR can perform path MTU discovery for each ASM group and use these values instead of 1280. For SSM, a different MTU value MUST be kept for each SSM channel. Because of this 8 bytes added by IPv6 fragment header in each data packet can be tolerated.

Since multicast address translation does not preserve checksum neutrality, [I-D.ietf-softwire-4rd] translator/encapsulator at 4rd BR must however modify the UDP checksum to replace the IPv4 addresses with the IPv6 source and destination addresses in the pseudo-header which consists of source address, destination address, protocol and UDP length fields before calculating the new checksum.

IPv6 multicast data must be translated back to IPv4 at the 4rd CE (e.g. using Table 2 in Section 4.3 of [I-D.ietf-softwire-4rd]). Such a task is much simpler than the translation at 4rd BR because IPv6 header is much simpler than IPv4 header and IPv4 link on the LAN side of 4rd CE is a local link. The packet is sent on the local link to IPv4 group address a.b.c.d for IPv6 group address of ASM_MPREFIX64:a.b.c.d or SSM_MPREFIX64:a.b.c.d.

In case an IPv4 multicast source sends multicast data with the don't fragment (DF) flag set to 1, [I-D.ietf-softwire-4rd] header mapping sets the D bit in IPv6 fragment header before sending the packet downstream as in Fig. 3 in Section 4.3 of [I-D.ietf-softwire-4rd]. This feature of [I-D.ietf-softwire-4rd] preserves the semantics of DF flag at the BR and CE.

Because MAP-T/4rd is stateless, Multicast MAP-T/4rd should stay faithful to this as much as possible. Border Relay acts as the default multicast querier for all CEs that have established multicast communication with it. In order to keep a consistent multicast state between a CE and BR, CE MUST use the same IPv6 multicast prefixes (ASM_MPREFIX64/SSM_REFIX64) until the state becomes empty. After that point, the CE may obtain different values for these prefixes, effectively changing to a different 4rd BR.

Protocol translation described above for MAP-T applies to lw4over6 multicast translation where the entities involved are lwB4 replaces Customer Edge device and lwAFTR replaces BR Figure 2.

4.3. Learning Multicast Prefixes for IPv4-embedded IPv6 Multicast Addresses

CE can be pre-configured with Multicast Prefix64 of ASM_MPREFIX64 and SSM_MPREFIX64 that are supported in their network. However automating this process is also desired.

A new router advertisement option, a Multicast ASM Translation Prefix option, can be defined for this purpose. The option contains IPv6 ASM multicast translation prefix, ASM_MPREFIX64. A new router advertisement option, a Multicast SSM Translation Prefix option, can be defined for this purpose. The option contains IPv6 SSM multicast prefix translation prefix SSM_MPREFIX64.

After the host gets the multicast prefixes, when an application in the host wishes to join an IPv4 multicast group the host MUST use ASM_MPREFIX64 or SSM_MPREFIX64 and then obtain the synthesized IPv6 group address before sending MLD join message.

Source-specific multicast (SSM) group membership message payloads in IGMPv3 and MLDv2 contain address literals and their translation requires another multicast translation prefix option. IPv4 source addresses in IGMPv3 Membership Report message are unicast addresses of IPv4 sources and they have to be translated into unicast IPv6 source addresses in MLDv2 Membership Report message. A new router advertisement option, a Multicast Translation Unicast Prefix option can be defined for this purpose. The option contains IPv6 unicast Network-Specific Prefix U_PREFIX64. The host can be configured by

its default router using router advertisements containing the prefixes [I-D.sarikaya-softwire-6man-raoptions]. 64:ff9b::/96 is the global value called well-known prefix that is assigned to U_PREFIX64 [RFC6052]. Organization specific values called Network-Specific Prefixes can also be used. Since multicast is potentially inter-domain, the use of well-known prefix for U_PREFIX64 is recommended. DHCP servers can also configure hosts with ASM_MPREFIX64, SSM_MPREFIX64 and U_PREFIX64 as in [I-D.ietf-softwire-multicast-prefix-option].

Note that U_PREFIX64 is also used in multicast data packet address translation. Source-specific multicast source address in multicast data packets coming from SSM sources MUST be translated using U_PREFIX64.

4.4. Supporting IPv4 Multicast at CE Router and lwB4

When MAP-E CE router is a NAT or NAPT box assigning private IPv4 addresses to the hosts, IP Multicast requirements for a Network Address Translator (NAT) and a Network Address Port Translator (NAPT) stated in [RFC5135] apply to IGMP messages and IPv4 multicast data packets. The same applies to lwB4s in lw4over6.

On receiving multicast data packets, lwB4 or CE router MUST NOT modify destination IP address or destination port of the packets. Multicast UDP datagrams MUST be forwarded to the local LAN towards the host that is a member of this group.

IGMP membership reports received at lwB4 or CE router may be sent upstream individually for any source multicast but for source specific multicast, e.g. IGMPv3, membership reports MUST be sent after IGMP aggregation.

5. Security Considerations

Multicast control and data message security can be provided by the security architecture, mechanisms, and services described in [RFC4301], [RFC4302] and [RFC4303]. and in [RFC4607] for source specific multicast.

6. IANA Considerations

TBD.

7. Acknowledgements

TBD.

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.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.

[RFC2113] Katz, D., "IP Router Alert Option", RFC 2113, February 1997.

[RFC2711] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, October 1999.

[RFC3171] Albanna, Z., Almeroth, K., Meyer, D., and M. Schipper, "IANA Guidelines for IPv4 Multicast Address Assignments", RFC 3171, August 2001.

[RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, August 2006.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

[RFC3307] Haberman, B., "Allocation Guidelines for IPv6 Multicast Addresses", RFC 3307, August 2002.

[RFC2491] Armitage, G., Schulter, P., Jork, M., and G. Harter, "IPv6 over Non-Broadcast Multiple Access (NBMA) networks", RFC 2491, January 1999.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[RFC2765] Nordmark, E., "Stateless IP/ICMP Translation Algorithm (SIIT)", RFC 2765, February 2000.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC5135] Wing, D. and T. Eckert, "IP Multicast Requirements for a Network Address Translator (NAT) and a Network Address Port Translator (NAPT)", BCP 135, RFC 5135, February 2008.

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.

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

[RFC6275] Perkins, C., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

[RFC7371] Boucadair, M. and S. Venaas, "Updates to the IPv6 Multicast Addressing Architecture", RFC 7371, September 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-13 (work in progress), March 2015.

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

[I-D.ietf-softwire-map-t]
Li, X., Bao, C., Dec, W., Troan, O., Matsushima, S., and T. Murakami, "Mapping of Address and Port using Translation (MAP-T)", draft-ietf-softwire-map-t-08 (work in progress), December 2014.

[I-D.ietf-softwire-4rd]
Despres, R., Jiang, S., Penno, R., Lee, Y., Chen, G., and M. Chen, "IPv4 Residual Deployment via IPv6 - a Stateless Solution (4rd)", draft-ietf-softwire-4rd-10 (work in progress), December 2014.

[I-D.ietf-mboned-64-multicast-address-format]
Boucadair, M., Qin, J., Lee, Y., Venaas, S., Li, X., and M. Xu, "IPv6 Multicast Address With Embedded IPv4 Multicast Address", draft-ietf-mboned-64-multicast-address-format-06 (work in progress), September 2014.

[I-D.ietf-softwire-multicast-prefix-option]
Boucadair, M., Qin, J., Tsou, T., and X. Deng, "DHCPv6 Option for IPv4-Embedded Multicast and Unicast IPv6 Prefixes", draft-ietf-softwire-multicast-prefix-option-08 (work in progress), March 2015.

8.2. Informative references

[RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, August 2002.

[RFC3956] Savola, P. and B. Haberman, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address", RFC 3956, November 2004.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[I-D.ietf-softwire-dslite-multicast]
Qin, J., Boucadair, M., Jacquenet, C., Lee, Y., and Q. Wang, "Delivery of IPv4 Multicast Services to IPv4 Clients over an IPv6 Multicast Network", draft-ietf-softwire-dslite-multicast-09 (work in progress), March 2015.

[I-D.sarikaya-softwire-6man-raoptions]
Sarikaya, B., "IPv6 RA Options for Translation Multicast Prefixes", draft-sarikaya-softwire-6man-raoptions-01 (work in progress), February 2013.

[I-D.perreault-mboned-igmp-mld-translation]
Perreault, S. and T. Tsou, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Translation ("IGMP/MLD Translation")", draft-perreault-mboned-igmp-mld-translation-01 (work in progress), April 2012.

Appendix A. Group Membership Message Translation Details

IGMP Report messages (IGMP type number 0x12 and 0x16, in IGMPv1 and IGMPv2 and 0x22 in IGMPv3) are translated into MLD Report messages (MLDv1 ICMPv6 type number 0x83 and MLDv2 type number 0x8f). IGMP Query message (IGMP type number 0x11) is translated into MLD Query message (ICMPv6 type number 0x82) [I-D.perreault-mboned-igmp-mld-translation].

Destination address in ASM, i.e. IGMPv1, IGMPv2 and MLDv1, is the multicast group address so the destination address in IGMP message is translated into the destination address in MLD message using [I-D.ietf-mboned-64-multicast-address-format].

Destination address in SSM, i.e. IGMPv3 and MLDv2 is translated as follows: it could be all nodes on link, which has the value of 224.0.0.1 (IGMPv3) and ff02::1 (MLDv2), all routers on link, which has the value of 224.0.0.2 (IGMPv3) and ff02::2 (MLDv2), all IGMP/MLD-capable routers on link, which has the value of 224.0.0.22 (IGMPv3) and ff02::16 (MLDv2).

Source address of MLD message that CE sends is set to link-local IPv6 address of CE’s WAN side interface. Source address of MLD message that BR sends is set to link-local IPv6 address of BR’s downstream interface.

Multicast Address or Group Address field in IGMP message payloads is translated using [I-D.ietf-mboned-64-multicast-address-format] as described above into the corresponding field in MLD message.

Source Address in IGMPv3 message payloads is translated using U_PREFIX64, the IPv6 unicast prefix to be used by SSM source. [RFC6052] defines in Section 2.3 the address translation algorithm of embedding an IPv4 source address and obtaining an IPv6 source address using a network specific prefix like U_PREFIX64. At the BR on its upstream interface or at the CE on its LAN interface, IPv4 addresses are extracted from the IPv4-embedded IPv6 addresses.

Maximum Response Time (MRT) field in IGMPv2 and IGMPv3 queries are translated into Maximum Response Delay (MRD) in MLDv1 and MLDv2 queries, respectively. In the corresponding MLD message, MRD is set to 100 times the value of MRT. At the BR on its upstream interface or at the CE on its LAN interface, MRT value is obtained by dividing MRD into 100 and rounding it to the nearest integer.

IGMP messages are sent with a Router Alert IPv4 option [RFC2113]. The translated MLD message are sent with a Router Alert option in a Hop-By-Hop IPv6 extension header [RFC2711]. In both cases, 2-octet value is set to 0.

Authors' Addresses

Behcet Sarikaya
Huawei USA
5340 Legacy Dr. Building 175
Plano, TX 75024

Email: sarikaya@ieee.org

Hui Ji
China Telecom
NO19.North Street
Beijing, Chaoyangmen,Dongcheng District
P.R. China

Email: jihui@chinatelecom.com.cn

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

Q. Sun
H. Wang
Y. Cui
Tsinghua University
I. Farrer
S. Zoric
Deutsche Telekom AG
M. Boucadair
Orange
R. Asati
Cisco Systems, Inc.
July 8, 2016

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

Abstract

This document defines a YANG data model for the configuration and operations (state, notification, RPC etc.) of IPv4-in-IPv6 Softwire Border Routers and Customer Premises Equipment. The model covers the Lightweight 4over6, MAP-E and MAP-T Softwire mechanisms.

Requirements Language

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The IETF Softwire Working Group has developed several IPv4-in-IPv6 Softwire mechanisms to address various deployment contexts and constraints. As a companion to the architectural specification documents, this document focuses on the provisioning of A+P softwire functional elements: Border Routers (BRs) and Customer Premises Equipment (CEs).

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

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

- o Lightweight 4over6 [RFC7596]

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

Two root containers are defined:

1. Container "softwire-config" holds the collection of YANG definitions common to all Softwire element configuration.

2. Container "softwire-state" holds YANG definitions for the operational state of the Softwire elements.

A NETCONF notify module is also included.

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

1.1. Terminology

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

1.2. Tree Diagrams

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

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

- o Braces "{" and "}" enclose feature content.

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

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

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

1.3. YANG Modelling of NAT44 Functionality

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

2. Common

The following sections of the document are structured with the root of the Softwire YANG model (common to all mechanisms) described first. Subsequent sections describe the models relevant to the different softwire mechanisms. All functions are listed, but the YANG models use the "feature" statement to distinguish among the different softwire mechanisms. This document defines a new module named "ietf-softwire" for Softwire data models such that this module aments "ietf-ipv6-unicast-routing" module that is defined in [I-D.ietf-netmod-routing-cfg].

3. Lightweight 4over6

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

The lwB4 is configured with the relevant parameters for establishing the IPv4-in-IPv6 tunnel including an IPv6 address for the lwAFTR and the IPv4 configuration for NAT44.

4. MAP-E and MAP-T

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

5. Softwire YANG Tree Diagrams

5.1. Common Tree Diagrams

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

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

Figure 1: High Level Softwire YANG Tree Organization

5.2. Lightweight 4over6 Tree Diagrams

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

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

```
   |        +--rw enable?                          boolean
   |        +--rw ce-instances
   |           +--rw ce-instance* [binding-ipv6info]
   |              +--rw name?                      string
   |              +--rw tunnel-payload-mtu         uint16
   |              +--rw tunnel-path-mru            uint16
   |              +--rw b4-ipv6-addr-format        boolean
   |              +--rw binding-ipv6info           union
   |              +--rw binding-ipv4-addr          inet:ipv4-address
   |              +--rw port-set
   |              |  +--rw psid-offset             uint8
   |              |  +--rw psid-len                uint8
   |              |  +--rw psid                    uint16
   |              +--rw br-ipv6-addr               inet:ipv6-address
   |              +--rw lifetime?                  uint32
   +--ro softwire-state
      +--...
      +--ro binding {binding}?
         +--ro br {br}?
         |  +--ro br-instances
         |     +--ro br-instance* [id]
         |        +--ro id                        uint32
         |        +--ro name?                     string
         |        +--ro sentPacket?               yang:zero-based-counter64
         |        +--ro sentByte?                 yang:zero-based-counter64
         |        +--ro rcvdPacket?               yang:zero-based-counter64
         |        +--ro rcvdByte?                 yang:zero-based-counter64
         |        +--ro droppedPacket?            yang:zero-based-counter64
         |        +--ro droppedByte?              yang:zero-based-counter64
         |        +--ro active-softwire-num?      uint32
         |        +--ro binding-table
         |           +--ro binding-entry* [binding-ipv6info]
         |              +--ro binding-ipv6info     union
         |              +--ro active?              boolean
         +--ro ce {ce}?
            +--ro ce-instances
               +--ro ce-instance* [binding-ipv6info]
                  +--ro name?                      string
                  +--ro binding-ipv6info           union
                  +--ro sentPacket?                yang:zero-based-counter64
                  +--ro sentByte?                  yang:zero-based-counter64
                  +--ro rcvdPacket?                yang:zero-based-counter64
                  +--ro rcvdByte?                  yang:zero-based-counter64
                  +--ro droppedPacket?             yang:zero-based-counter64
                  +--ro droppedByte?               yang:zero-based-counter64
```

Figure 2: Softwire Lightweight 4over6 Data Model Tree Structure

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

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

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

- o 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].

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

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

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

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

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

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

- o active-tunnel-num (ro): used to present the number of tunnels currently provisioned on the device.

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

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

- o b4-ipv6-addr-format: indicates the format of lwB4 IPv6 address. If set to true, it indicates that the IPv6 source address of the lwB4 is constructed according to the description in Section 6 of [RFC7597]; if set to false, the lwB4 can use any /128 address from the assigned IPv6 prefix.

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

5.3. MAP-E and MAP-T Tree Diagrams

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

```
module: ietf-softwire
   +--rw softwire-config
   |  +--...
   |  +--rw algorithm {algorithm}?
   |     +--rw enable?                       boolean
   |     +--rw algorithm
   |        +--rw algo-instance* [id]
   |           +--rw algo-versioning
   |           |  +--rw algo-version?        uint64
   |           |  +--rw algo-date?           yang:date-and-time
   |           +--rw id                      uint32
   |           +--rw name?                   string
   |           +--rw data-plane              enumeration
   |           +--rw ea-len                  uint8
   |           +--rw rule-ipv6-prefix        inet:ipv6-prefix
   |           +--rw rule-ipv4-prefix        inet:ipv4-prefix
   |           +--rw forwarding              boolean
   |           +--rw psid-offset             uint8
   |           +--rw psid-len                uint8
   |           +--rw tunnel-payload-mtu      uint16
   |           +--rw tunnel-path-mru         uint16
   |           +--rw br-ipv6-addr            inet:ipv6-address
   |           +--rw dmr-ipv6-addr           inet:ipv6-prefix
   +--ro softwire-state
      +--...
      +--ro algorithm {algorithm}?
         +--ro algo-instances
            +--ro algo-instance* [id]
               +--ro id                      int32
               +--ro name?                   string
               +--ro sentPacket?             yang:zero-based-counter64
               +--ro sentByte?               yang:zero-based-counter64
               +--ro rcvdPacket?             yang:zero-based-counter64
               +--ro rcvdByte?               yang:zero-based-counter64
               +--ro droppedPacket?          yang:zero-based-counter64
               +--ro droppedByte?            yang:zero-based-counter64
```

Figure 3: Softwire MAP-E and MAP-T Data Model Structure

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

- o 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.

- 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. See Section 4.1 of [RFC7598].

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

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

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

5.4. Notifications for Softwire YANG

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

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

Figure 4: Softwire Notifications Data Model Structure

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

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

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

6. Softwire YANG Model

This module imports typedefs from [RFC6991].

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

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

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

  organization "Softwire Working Group";

  contact
    "
    Qi Sun <sunqi.ietf@gmail.com>
    Hao Wang <wangh13@mails.tsinghua.edu.cn>
    Yong Cui <yong@csnet1.cs.tsinghua.edu.cn>
    Ian <Farrer ian.farrer@telekom.de>
    Sladjana Zoric <sladjana.zoric@telekom.de>
    Mohamed Boucadair <mohamed.boucadair@orange.com>
    Rajiv <Asati rajiva@cisco.com>
    ";

  description
    "This document defines a YANG data model for the configuration and
    management of A+P Softwire Border Routers (BRs) and Customer
    Premises Equipment (CEs). It covers Lightweight 4over6,
    MAP-E and MAP-T mechanisms.

    Copyright (c) 2016 IETF Trust and the persons identified
    as authors of the code. All rights reserved.
    This version of this YANG module is part of RFC XXX; see the RFC
    itself for full legal notices.";

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

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

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

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

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


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

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

/*
 * Features
 */

  feature binding {
    description
    "Lightweight 4over6 (binding) is an IPv4-over-IPv6 tunnelling
     transition mechanism. Lightweight 4over6 is a solution designed
     specifically for complete independence between IPv6 subnet
```

```
     prefix (and /128 IPv6 address) and IPv4 address with or
     without IPv4 address sharing.

     This is accomplished by maintaining state for
     each softwire (per-subscriber state) in the central lwAFTR and
     a hub-and-spoke forwarding architecture. In order to delegate
     the NAPT function and achieve IPv4 address sharing,
     port-restricted IPv4 addresses needs to be allocated to CEs.

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

   reference
      "RFC7596";
  }

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

  feature ce {
    if-feature binding;
    description
      "The B4 for Lightweight 4over6, so-called lwB4 (CE). This
      feature indicates that a instance functions as a lwB4 (CE). A
      lwB4 (ce) is an IPv4-in-IPv6 tunnel initiator. It is
      dual-stack capable node, either a directly connected end-host
      or a CE. It sources IPv4 connections using the configured
      port-set and the public IPv4 address.";
  }

  feature algorithm {
    description
      "MAP-E is an IPv6 transition mechanism for transporting IPv4
      packets across an IPv6 network using IP encapsulation. MAP-E
      allows for a reduction of the amount of centralized state using
      rules to express IPv4/IPv6 address mappings. This introduces an
      algorithmic relationship between the IPv6 subnet
      and IPv4 address.
      The Mapping of Address and Port - Translation (MAP-T)
      architecture is a double stateless NAT64 based solution. It uses
      the stateless algorithmic address & transport layer port mapping
      scheme defined in MAP-E. The MAP-T solution differs from MAP-E in
```

```
      the use of IPv4-IPv6 translation, rather than encapsulation, as
      the form of IPv6 domain transport.
      This feature indicates the instance functions as a MAP-E or
      MAP-T instance.";
    reference
      "RFC7597 & RFC7599";
  }

/*
 * Grouping
 */

  grouping port-set {
    description
      "Use the PSID algorithm to represent a range of transport layer
      ports.";
    leaf psid-offset {
      type uint8 {
        range 0..16;
      }
      mandatory true;
      description
        "The number of offset bits. In Lightweight 4over6, the default
        value is 0 for assigning one contiguous port range. In MAP-E/T,
        the default value is 6, which excludes system ports by default
        and assigns distributed port ranges. If the this parameter is
        larger than 0, the value of offset MUST be greater than 0.";
    }
    leaf psid-len {
      type uint8 {
        range 0..15;
      }
      mandatory true;
      description
        "The length of PSID, representing the sharing ratio for an
        IPv4 address.";
    }
    leaf psid {
      type uint16;
      mandatory true;
      description
        "Port Set Identifier (PSID) value, which identifies a set
        of ports algorithmically.";
    }
  }

  grouping binding-entry {
    description
```

```
      "The lwAFTR maintains an address binding table that contains
      the binding between the lwB4's IPv6 address, the allocated IPv4
      address and restricted port-set.";
    leaf binding-ipv6info {
      type union {
        type inet:ipv6-address;
        type inet:ipv6-prefix;
      }
      mandatory true;
      description
        "The IPv6 information for a binding entry.
         If it's an IPv6 prefix, it indicates that
         the IPv6 source address of the lwB4 is constructed
         according to the description in RFC7596;
         if it's an IPv6 address, it means the lwB4 uses
         any /128 address from the assigned IPv6 prefix.
         ";
    }
    leaf binding-ipv4-addr {
      type inet:ipv4-address;
      mandatory true;
      description
        "The IPv4 address assigned to the lwB4, which is
         used as the IPv4 external address
         for lwB4 local NAPT44.";
    }
    container port-set {
      description
        "For Lightweight 4over6, the default value
        of offset should be 0, to configure one contiguous
        port range.";
      uses port-set {
        refine psid-offset {
          default "0";
        }
      }
    }
    leaf br-ipv6-addr {
      type inet:ipv6-address;
      mandatory true;
      description
        "The IPv6 address for lwaftr.";
    }
    leaf lifetime {
      type uint32;
      units seconds;
      description "The lifetime for the binding entry";
    }
```

```
  }

/*
  grouping nat-table {

    description
      "Grouping 'nat-table' is not extended. The current mechanism
      is focusing on the provisioning of external IP address and
      port set; other NAT-specific considerations are out of scope.";
  }
*/

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


/*
 * Configuration Data Nodes
 */


  container softwire-config {
    description
```

```
      "The configuration data for Softwire instances. And the shared
      data describes the softwire data model which is common to all of
      the different softwire mechanisms, such as description.";
    leaf description {
      type string;
      description
        "A textual description of Softwire.";
    }
    container binding {
      if-feature binding;
      description
        "lw4over6 (binding) configuration.";
      container br {
        if-feature br;
        description
          "Indicate this instance supports the lwAFTR (BR) function.
          The instances advertise the BR feature through the
          capability exchange mechanism when a NETCONF session is
          established.";
        leaf enable {
          type boolean;
          description
            "Enable/disable the lwAFTR (BR) function.";
        }
        container br-instances {
          description
            "A set of BRs to be configured.";
          list br-instance {
            key "id";
            description
            "A set of lwAFTRs to be configured.";
            container binding-table-version {
              description "binding table’s version";
              leaf binding-table-version{
                type uint64;
                description "Incremental version number
                to the binding table";
              }
              leaf binding-table-date {
                type yang:date-and-time;
                description "Timestamp to the binding
                table";
              }
            }
            leaf id {
              type uint32;
              mandatory true;
              description "An instance identifier.";
```

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

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

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

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

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

```
            description
              "Embedded Address (EA) bits are the IPv4 EA-bits
              in the IPv6 address identify an IPv4
              prefix/address (or part thereof) or
              a shared IPv4 address (or part thereof)
              and a port-set identifier.
              The length of the EA-bits is defined as
              part of a MAP rule for a MAP domain.";
          }
          leaf rule-ipv6-prefix {
            type inet:ipv6-prefix;
            mandatory true;
            description
              "The Rule IPv6 prefix defined in the mapping rule.";
          }
          leaf rule-ipv4-prefix {
            type inet:ipv4-prefix;
            mandatory true;
            description
              "The Rule IPv4 prefix defined in the mapping rule.";
          }
          leaf forwarding {
            type boolean;
            mandatory true;
            description
              "This parameter specifies whether the rule may be used for
              forwarding (FMR). If set, this rule is used as an FMR;
              if not set, this rule is a BMR only and must not be used
              for forwarding.";
          }
          leaf psid-offset {
            type uint8 {
              range 0..16;
            }
            mandatory true;
            description
              "The number of offset bits. In Lightweight 4over6, the default
              value is 0 for assigning one contiguous port range. In MAP-E/T,
              the default value is 6, which excludes system ports by default
              and assigns distributed port ranges. If the this parameter is
              larger than 0, the value of offset MUST be greater than 0.";
          }
          leaf psid-len {
            type uint8 {
              range 0..15;
            }
            mandatory true;
            description
```

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

/*
 * Operational state Data Nodes
 */

  container softwire-state {
    config false;
    description
      "The operational state data for Softwire instances. ";
    leaf description {
      type string;
      description
        "A textual description of the softwire instances.";
    }
    container binding {
      if-feature binding;
      description
        "lw4over6 (binding) state.";
      container br {
        if-feature br;
```

```
        config false;
        description
          "Indicate this instance supports the lwAFTR (BR) function.
          The instances advertise the lwaftr (BR) feature through the
          capability exchange mechanism when a NETCONF session is
          established.";
        container br-instances {
          description
            "A set of BRs.";
          list br-instance {
            key "id";
            description "instances for BR";
            leaf id {
              type uint32;
              mandatory true;
              description "id";
            }
            leaf name {
              type string;
              description "The name for this lwaftr.";
            }
            uses traffic-stat;
            leaf active-softwire-num {
              type uint32;
              description
                "The number of currently active tunnels on the
                lw4over6 (binding) instance.";
            }
            container binding-table {
              description "id";
              list binding-entry {
                key "binding-ipv6info";
                description "An identifier of the binding entry.";
                leaf binding-ipv6info {
                  type union {
                    type inet:ipv6-address;
                    type inet:ipv6-prefix;
                  }
                  mandatory true;
                  description
                    "The IPv6 information used to identify
                      a binding entry. ";
                }
                leaf active {
                  type boolean;
                  description
                    "Status of a specific tunnel.";
                }
```

```
              }
            }
          }
        }
      }


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

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

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

          }
          uses traffic-stat;
        }
      }
    }
  }

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

    leaf br-id {
      type leafref {
        path
          "/softwire-state/binding/br/br-instances/"
          + "br-instance/id";
      }
          description "...";
    }
    leaf-list invalid-entry {
      type leafref {
        path
          "/softwire-config/binding/br/br-instances/"
          + "br-instance[id=current()/../br-id]/"
          + "binding-table/binding-entry/binding-ipv6info";
      }
      description
        "Notify the client that a specific binding entry has been
```

```
        expired/invalid. The binding-ipv6info identifies an entry.";
    }
    leaf-list added-entry {
        type inet:ipv6-address;
        description
          "Notify the client that a binding entry has been added.
          The ipv6 address of that entry is the index. The client
          get other information from the lwaftr about the entry
          indexed by that ipv6 address.
          ";
    }
    leaf-list modified-entry {
        type leafref {
          path
            "/softwire-config/binding/br/br-instances/"
            + "br-instance[id=current()/../br-id]/"
            + "binding-table/binding-entry/binding-ipv6info";
        }
          description "...";
    }
  }

  notification softwire-ce-event {
    if-feature binding;
    if-feature ce;
    description "CE notification";
    leaf ce-binding-ipv6-addr-change {
      type inet:ipv6-address;
      mandatory true;
      description
        "The source tunnel IPv6 address of the lwB4.
         If 'b4-ipv6-addr-format' is false, or the lwb4's
         binding-ipv6-address changes for any reason,
         it SHOULD notify the NETCONF client.";
    }
  }

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

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

7. Example of Configure lw4o6 Binding-Table

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

- o IPv6 Address for a single lwB4
- o Public IPv4 Address
- o Restricted port-set

The entry has two functions: the IPv6 encapsulation of inbound IPv4 packets destined to the lwB4 and the validation of outbound IPv4-in-IPv6 packets received from the lwB4 for de-capsulation.

Let’s consider an example to add an entry that maintains the relationship between 3-tuples of lwB4 (2001:db8::1), ’192.0.2.1’ and ’1234’ in the binding table of the lwAFTR (2001:db8::2). Here is the example binding-table configuration xml:

```
     <rpc message-id="101"
       xmlns:nc="urn:params:xml:ns:yang:ietf-softwire:1.0">
   <!-- replace with IANA namespace when assigned. -->
     <edit-config>
       <target>
         <running/>
       </target>
     <softwire-config>
       <lw4o6-aftr>
         <lw4o6-aftr-instances>
           <lw4o6-aftr-instance>
             <aftr-ipv6-addr>2001:db8::2</aftr-ipv6-addr>
             <binding-table>
               <binding-entry>
                 <binding-ipv4-addr>192.0.2.1</binding-ipv4-addr>
                 <port-set>
                   <psid>1234</psid>
                 </port-set>
                 <binding-ipv6-addr>2001:db8::1</binding-ipv6-addr>
                 <active>1</active>
               </binding-entry>
             </binding-table>
           </lw4o6-aftr-instance>
         </lw4o6-aftr-instances>
       </lw4o6-aftr>
     </softwire-config>
```

Figure 5: lw4o6 Binding-Table Configuration XML

8. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

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

9. IANA Considerations

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

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

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

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

10. Acknowledgements

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

11. References

11.1. Normative References

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

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

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

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

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

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

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

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

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

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

11.2. Informative References

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

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

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

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

Authors' Addresses

Qi Sun
Tsinghua University
Beijing 100084
P.R. China

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

Hao Wang
Tsinghua University
Beijing 100084
P.R. China

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

Yong Cui
Tsinghua University
Beijing 100084
P.R. China

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

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

Email: ian.farrer@telekom.de

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

Email: sladjana.zoric@telekom.de

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Rajiv Asati
Cisco Systems, Inc.
7025 Kit Creek Rd.
RTP, NC 27709
USA

Email: Rajiva@cisco.com