

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: September 15, 2011

D. Thaler  
Microsoft  
March 14, 2011

Source Address Finding (SAF) for IPv6 Translation Mechanisms  
draft-thaler-ipv6-saf-03.txt

## Abstract

There are various recent proposals that would result in IPv6 translation becoming permanent. [RFC 3424](#) discusses UNilateral Self-Address Fixing (UNSAF) mechanisms which are required for applications to work with most translation schemes, points out a number of problems with them, and requires an exit strategy for any UNSAF mechanism. This document discusses an alternative to UNSAF mechanisms should IPv6 translation become permanent.

## 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 September 15, 2011.

## Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4.e](#) of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction . . . . .</a>	<a href="#">3</a>
<a href="#">2.</a>	<a href="#">IPv6 Translation . . . . .</a>	<a href="#">3</a>
<a href="#">3.</a>	<a href="#">IPv6 Translation Without UNSAF . . . . .</a>	<a href="#">4</a>
<a href="#">3.1.</a>	<a href="#">Evaluation of Architectural Issues . . . . .</a>	<a href="#">5</a>
<a href="#">3.2.</a>	<a href="#">Requirements for SAF Mechanisms . . . . .</a>	<a href="#">6</a>
<a href="#">3.3.</a>	<a href="#">DHCPv6 Option as a SAF Mechanism . . . . .</a>	<a href="#">7</a>
<a href="#">3.3.1.</a>	<a href="#">DHCPv6 Server Configuration . . . . .</a>	<a href="#">8</a>
<a href="#">3.3.2.</a>	<a href="#">DHCPv6 Client Behavior . . . . .</a>	<a href="#">9</a>
<a href="#">4.</a>	<a href="#">Security Considerations . . . . .</a>	<a href="#">9</a>
<a href="#">5.</a>	<a href="#">IANA Considerations . . . . .</a>	<a href="#">9</a>
<a href="#">6.</a>	<a href="#">References . . . . .</a>	<a href="#">9</a>
<a href="#">6.1.</a>	<a href="#">Normative References . . . . .</a>	<a href="#">9</a>
<a href="#">6.2.</a>	<a href="#">Informative References . . . . .</a>	<a href="#">10</a>
	<a href="#">Author's Address . . . . .</a>	<a href="#">11</a>

## 1. Introduction

Many applications and protocols use one or more addresses of the local machine, e.g., to send in an application protocol exchange or to advertise a public address at which it will accept connections.

[RFC 2993](#) [[RFC2993](#)] discusses architectural implications of Network Address Translation (NAT). One of the implications of translation is that in general the address that must be used by other nodes to reach a destination is not the address assigned to an interface on the destination, where the destination's applications and protocols would naturally find it. As a result, NAT generally requires a mechanism whereby an endpoint can determine the address by which it is known to other endpoints, and then "fix" its own messages to use that address instead of the one(s) it would normally use. This category of mechanisms is known as UNilateral Self-Address Fixing (UNSAF).

[RFC 3424](#) [[RFC3424](#)] discusses architectural implications of UNSAF mechanisms, and concludes that they are not appropriate as long term fixes and recommends that any UNSAF proposal require, among other things, an exit strategy. Since NAT mechanisms generally require UNSAF mechanisms, an exit strategy for an UNSAF proposal often requires an exit strategy for the NAT mechanism motivating it.

## 2. IPv6 Translation

The notion of IPv4-IPv6 translation (e.g., NAT-PT [[RFC2766](#)]) first introduced the NAT problems into IPv6 and motivated UNSAF mechanisms in IPv6. Although NAT-PT was deprecated ([[RFC4966](#)]), the notion of IPv4-IPv6 translation has become even more important. There is a fairly clear exit strategy (although the timeframe of an exit is not at all clear), which is that IPv4-IPv6 translation use decreases as IPv4-only nodes decrease over time. As a result, the exit strategy of any resulting UNSAF mechanisms is that their use declines as IPv4-IPv6 translation declines.

Recently however there has been discussion of the possibility of IPv6-IPv6 translation (e.g., NPT66 [[I-D.mrw-nat66](#)] to address renumbering pains, Six/One [[I-D.vogt-rrg-six-one](#)] to address routing scalability, etc.). Such proposals, if adopted, are not proposed as short term mechanisms but rather as more permanent changes to the architecture. As such, if UNSAF mechanisms are required, the exit strategy cannot be simply based on declining IPv6-IPv6 translation.

### [3.](#) IPv6 Translation Without UNSAF

In this section, we focus primarily on IPv6-IPv6 translation, although there may be cases where the same concepts might be applicable to IPv4-IPv6 translation or IPv4-IPv4 translation.

While translation in general requires UNSAF mechanisms, some uses of translation do not. UNSAF mechanisms are needed whenever the address reachable by outside parties is not an address of the local machine. Hence any use of translation whereby the address reachable by outside parties is still an address that appears to be assigned to some interface on the machine, does not require UNSAF mechanisms. For example, the Host Identity Protocol (HIP) [[RFC5201](#)] uses translation in this respect. The address seen by applications is in fact not the address used on the wire, but is translated by the HIP layer on both the sender and the receiver.

There are two key requirements for the translation mechanism:

1. The translation is reversible without loss of information, and
2. The address is presented by the host to upper layers in the same way as a normal IP address.

When these requirements are met, reversible translation can be compared to (and contrasted with) a tunnel with header compression. To reverse translation, both translators must have the information necessary to perform the translation, which requires some configuration or per-host signaling mechanism (e.g., DHCP, as opposed to a per-flow signaling mechanism as HIP does) for learning an address to configure on an interface, which obviates the need for

applications to use an UNSAF mechanism above the transport layer. We will refer to this concept as Self-Address Finding (SAF) to distinguish it from UNSAF mechanisms. Note that "finding" is intentionally used here instead of "fixing" as in UNSAF; since the address found is actually used by IP and higher layers, there is nothing to "fix" up higher.

Tunneling mechanisms, however, have deployment incentive issues (as pointed out in [[RFC5218](#)]) in that they require both ends to be changed before either end benefits. Translation mechanisms such as NAT, on the other hand, have the advantage of being unilaterally deployable, at the expense of breaking some applications. For additional discussion, see [[RFC5902](#)].

Reversible IPv6-IPv6 translation can be initially deployed unilaterally (at the expense of breaking some applications) at a translation middlebox without touching end hosts, avoiding the deployment incentive issues with tunneling. End-to-end connectivity can then be restored once the host is able to learn the external

address and configure it on a virtual interface; hence, there is a further incentive built-in that restores the end-to-end model. This provides an exit strategy that does not require an UNSAF mechanism or result in the issues discussed in [[RFC3424](#)].

### 3.1. Evaluation of Architectural Issues

Regarding issues with NAT mechanisms raised in [[RFC2993](#)]:

- o Per-flow state in the middlebox (scaling, multihoming, single point-of-failure, etc): Reversible translation can be done without any per-flow state in the middlebox. NPT66 and Six/One are examples of this.
- o Inhibit IPsec: If translation and reversing can be done below IPsec, IPsec works normally. (Or if translation and reversing is done within IPsec as HIP does, IPsec also works.)
- o Address sharing (NAPT) inhibits other transport protocols: Reversible translation can be done without address sharing, allowing arbitrary transport protocols to work.

Regarding issues with UNSAF mechanisms raised in [[RFC3424](#)]:

- o No unique outside: When nested translators exist, there are multiple outside areas and hence multiple addresses by which one

is reachable by different peers. Reversible translation does not change this. This means that a node must be able to discover the address assigned by each translator in front of it.

- o Circumventing firewalls: Firewalls are orthogonal to reversible translation. SAF mechanisms should not circumvent firewalls. Since translators can be stateless, there is no need for periodic messages purely to maintain state in a translator or to implement a SAF mechanism.
- o Timeout issues of address assignment in middlebox: Since translators can be stateless, there is no state to time out.
- o Fate sharing when a server separate from the middlebox is used: Like UNSAF mechanisms, SAF mechanisms could either use a server separate from the middle box or communicate directly with the middlebox itself. Communicating with a server on the Internet, as protocols such as STUN [[RFC5389](#)] do, without any support from the translator, generally only allows discovering the address assigned by the outermost translator (i.e., the address seen by the server outside), not each cascaded translator. Furthermore, communicating with a remote server results in depending on reachability all the way to that server, whereas the desired communication may be much closer and otherwise be possible even when the server is unreachable. Hence the use of an external server is not recommended for SAF mechanisms.

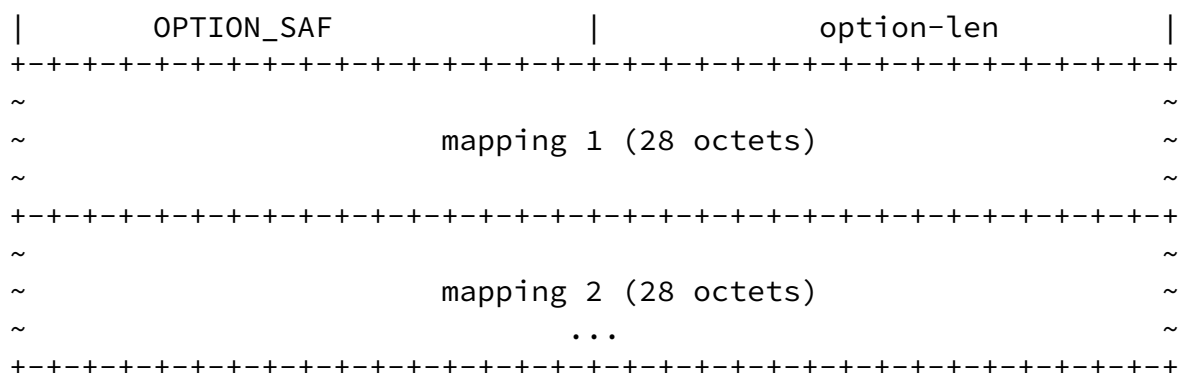
### [3.2.](#) Requirements for SAF Mechanisms

From the above discussion, we obtain the following requirements for SAF mechanisms.

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 [RFC 2119](#) [[RFC2119](#)].

1. Discovery: A SAF mechanism MUST allow a node to find the addresses assigned by all translators it is behind. Specifically, if the node is behind multiple cascaded translators, such that there is no unique "outside", then the SAF mechanism MUST allow the node to learn the address it will appear as within each "outside" area.

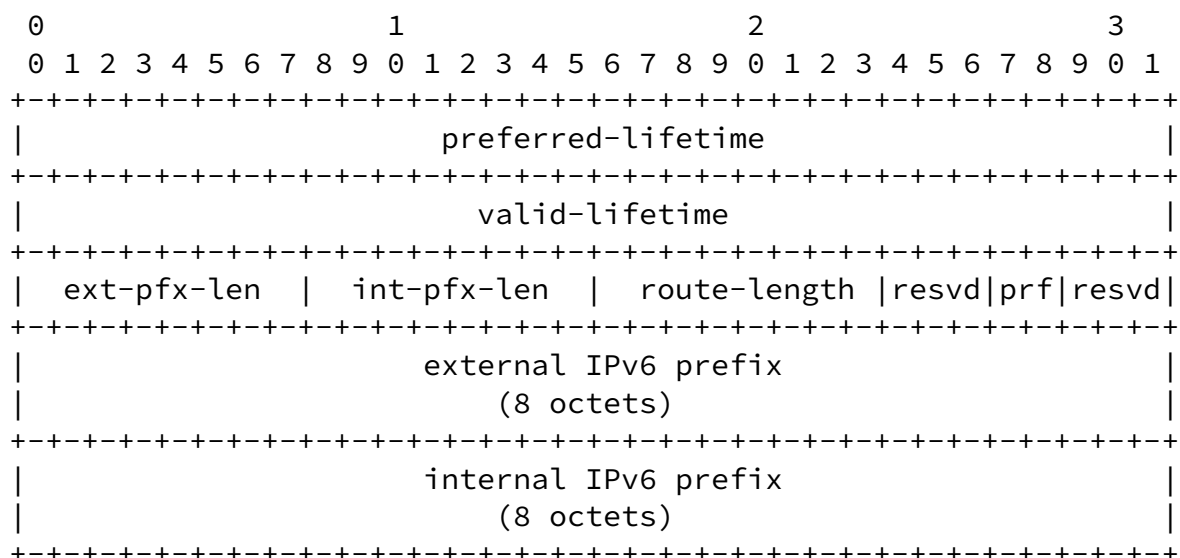




option-code    OPTION\_SAF (TBD).

option-len    Length of the SAF option in octets (not including the size of the option-code and option-len fields). The number of prefix mappings in this option is thus given by (option-len / 28).

mapping n    A 28-byte mapping, as specified below. This is replicated once for each prefix mapping to be included in the option.



preferred-lifetime    The preferred lifetime for the mapping in the



option, expressed in units of seconds. The preferred lifetime of a virtual address is the minimum of this value, and the preferred lifetime of the physical address from which it was derived.

**valid-lifetime** The valid lifetime for the mapping in the option, expressed in units of seconds. The valid lifetime of a virtual address is the minimum of this value, and the valid lifetime of the physical address from which it was derived.

**ext-pfx-len** The prefix length in bits of the external prefix.

**int-pfx-len** The prefix length in bits of the internal prefix.

**route-length** The prefix length in bits of the route for destination addresses usable with virtual addresses created using this mapping.

**resvd** Reserved. MUST be 0, and MUST be ignored by DHCPv6 clients.

**prf** Route Preference. The 2-bit preference of the route. This value is used as specified in [[RFC4191](#)].

**external IPv6 prefix** The external IPv6 prefix, out of which the host can derive virtual addresses.

**internal IPv6 prefix** The internal IPv6 prefix. For each physical IPv6 address a client has in this prefix on the interface over which this option was received, it can construct a virtual address.

### 3.3.1. DHCPv6 Server Configuration

To use this mechanism, a DHCPv6 server is configured with a set of translation configuration information that an end host can use to derive externally-visible addresses from its own physical IPv6 addresses.

When nested translators exist, there are multiple outside areas and hence multiple addresses by which one is reachable by different peers. It is RECOMMENDED that a mapping for each be configured. For example, if internal prefix X is mapped to prefix Y, which is then mapped to prefix Z, the DHCPv6 option should contain mappings for X<->Y with route-length equal to the prefix length of Y, and X<->Z with route-length of 0, so that it can statelessly communicate with other hosts in prefix Y using a virtual address from prefix Y, and also statelessly communicate with hosts on the IPv6 Internet using a virtual address from prefix Z.

### [3.3.2.](#) DHCPv6 Client Behavior

To use this mechanism, a DHCPv6 server is configured with a set of translation configuration information that an end host can use to derive externally-visible addresses from its own physical IPv6 addresses, and then add those externally-visible addresses on a virtual interface. That is, for each mapping between an internal prefix and an external prefix, a DHCPv6 client does the following.

For each physical IPv6 address it has that falls within the internal prefix and is on the interface over which this option was received, it constructs a virtual external address using the translation algorithm specified in [[I-D.mrw-nat66](#)] [section 3.2](#), and assigns the virtual address to a virtual interface.

The client must also use some means to ensure that the virtual addresses are eligible for source address selection when sending to external destinations. For example, the client constructs a route to a destination prefix based on the external IPv6 prefix, truncated or 0-extended to the route-length, and adds the route on the virtual interface.

## [4.](#) Security Considerations

NATs and UNSAF mechanisms generally interfere with security mechanisms because they change the addresses and/or content of messages exchanged. This document discusses requirements for SAF mechanisms that avoid these issues.

## [5.](#) IANA Considerations

IANA is requested to a DHCPv6 option code value of TBD to the SAF option from the DHCPv6 option code space defined in [Section 24.3 of \[RFC3315\]](#).

## [6.](#) References

### [6.1.](#) Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC3041] Narten, T. and R. Draves, "Privacy Extensions for

- [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.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", [RFC 3972](#), March 2005.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", [RFC 4191](#), November 2005.

## [6.2.](#) Informative References

- [I-D.mrw-nat66]  
Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", [draft-mrw-nat66-12](#) (work in progress), March 2011.
- [I-D.vogt-rrg-six-one]  
Vogt, C., "Six/One: A Solution for Routing and Addressing in IPv6", [draft-vogt-rrg-six-one-02](#) (work in progress), October 2009.
- [RFC2766] Tsirtsis, G. and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", [RFC 2766](#), February 2000.
- [RFC2993] Hain, T., "Architectural Implications of NAT", [RFC 2993](#), November 2000.
- [RFC3424] Daigle, L. and IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", [RFC 3424](#), November 2002.
- [RFC4966] Aoun, C. and E. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status", [RFC 4966](#), July 2007.
- [RFC5201] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", [RFC 5201](#), April 2008.

[RFC5218] Thaler, D. and B. Aboba, "What Makes For a Successful Protocol?", [RFC 5218](#), July 2008.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", [RFC 5389](#), October 2008.

[RFC5902] Thaler, D., Zhang, L., and G. Lebovitz, "IAB Thoughts on

Thaler

Expires September 15, 2011

[Page 10]

---

Internet-Draft

SAF in IPv6

March 2011

IPv6 Network Address Translation", [RFC 5902](#), July 2010.

#### Author's Address

Dave Thaler  
Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052  
USA

Phone: +1 425 703 8835  
Email: [dthaler@microsoft.com](mailto:dthaler@microsoft.com)

Thaler

Expires September 15, 2011

[Page 11]