

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: February 28, 2014

Z. Cao  
China Mobile  
A. Ding  
Cambridge University / Helsinki University  
August 27, 2013

Service Discovery in a Multiple Connection Environment: Problem  
Statement  
draft-cao-mif-srv-dis-ps-03

Abstract

This document analyzes the problems of service discovery in a multiple connection environment. A multiple connection environment consists of multiple-interfaced nodes connecting to multiple networks or multiple provisioning domains. Given a type of service a multiple-interfaced client is looking for, the discovery progress ought to return a correct pointer to the service instance that the client is able to access without trying every available channel.

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 February 28, 2014.

Copyright Notice

Copyright (c) 2013 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

1. Introduction . . . . .	2
2. Requirements and Terminology . . . . .	3
2.1. Requirements . . . . .	3
2.2. Terminology . . . . .	3
3. Scenarios . . . . .	3
3.1. Mif Scenario . . . . .	3
3.2. Homenet Scenario . . . . .	5
4. Problem Analysis . . . . .	5
5. IANA Considerations . . . . .	9
6. Security Considerations . . . . .	9
7. References . . . . .	9
7.1. Normative References . . . . .	9
7.2. Informative References . . . . .	9
Authors' Addresses . . . . .	10

## 1. Introduction

A multihomed host has multiple provisioning domains via physical and/or virtual interfaces. A multihomed host receives node configuration information from each of its access networks, through various mechanisms such as DHCP, PPP and IPv6 Router Advertisements. When the received node-scoped configuration objects have different values from each administration domains, such as different DNS servers IP addresses, different default gateways or different address selection policies, the node has to decide which it will use or how it will merge them.

Issues regarding how the multi-homed host uses the configuration objects have been addresses in [RFC6418]. Current practices of how the various implementations handle these problems are introduced in [RFC6419]. [RFC6731] extends DHCPv6 to inform the host which DNS server it ought to select to send the query request, and DNS based Service Discovery (DNS-SD) has been specified in [RFC6763].

This document analyzes the problem of service discovery in a multiple connection environment. A multiple connection environment consists of multiple-interfaces nodes connecting to multiple networks or multiple provisioning domains. Given a type of service a multiple-interfaced client is looking for, the discovery progress ought to return a correct pointer to the service instance that the a client is able to access.

## 2. Requirements and Terminology

### 2.1. Requirements

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].

### 2.2. Terminology

#### Service Domain

A set of services that can be accessed by users. Besides providing services, a service domain is responsible for delivering configuration and pointers that ensure a guaranteed service access.

#### Service Discovery

Procedure to acquire information that is necessary to access service.

#### Multiple Connection Environment

Consists of multiple-interfaced nodes that connect to multiple networks or multiple provisioning domains.

## 3. Scenarios

We describe two scenarios in this section, one related to Multiple Interfaces, and the other one related to Home Networks (homenet).

### 3.1. Mif Scenario

The service discovery process can be summarized as the following five steps.

1. Service Discovery Preparation: the host determines which interface it should send a query request based on the configuration information.
2. Service Query Request: the host sends a query request to find a service. The query should include a description of the service, for example, a full-qualified domain name, a URI, or an application-specific naming of the service.
3. Service Request Handling: any entity that receives the query request should handle the request. The entity should understand

the meaning of the request, and check the semantics of the request language before giving an answer back.

4. **Service Query Response:** the entity that receives the query request should reply with an answer to the query. The answer should include a pointer to the service.
5. **Service Access:** the host accesses the service via the pointer provided in the query response. The host is supposed to be able to get the service instance via the pointer under a successful and efficient service discovery mechanism, unless the servers in such service domain encounter problems e.g. a web server is down.

Figure 1 shows a typical scenario for service discovery in a multiple connection environment. It is common in today's mobile Internet that a host is equipped with multiple network interfaces. On the service domain, different services are deployed and some services may not be accessible to a certain interface on the host due to security concern or access policy. The connectivity each interface provides may not be restricted to Internet access. For instance, WLAN and bluetooth can offer direct access to potential services e.g. printers via local ad-hoc connectivity. In such multiple connection environment, the service discovery process should return a correct point to the host and ensure that the host can access the service via this pointer. This situation makes the multiple interface service discovery different from the typical one-interface Internet access scenario. Furthermore, the growing usage of IPv6 in the homenet environment has made service discovery more challenging that requires thorough investigation.

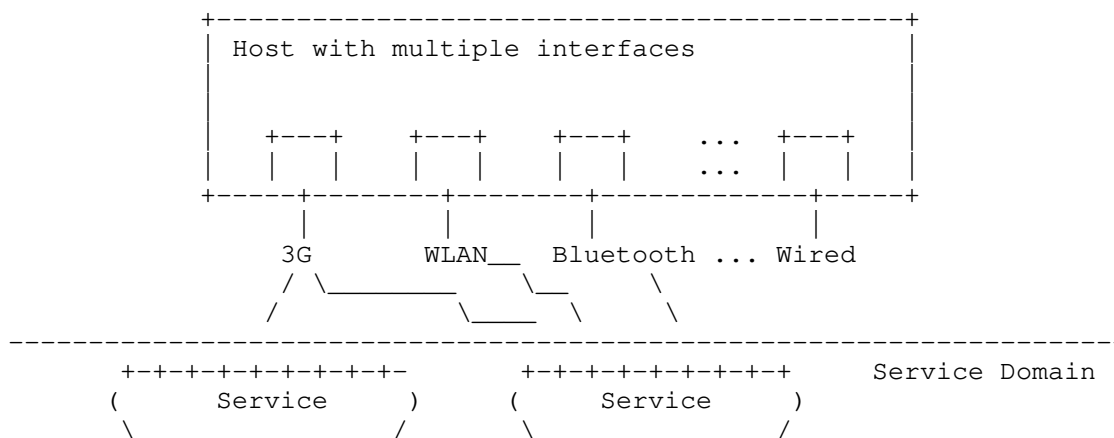


Figure 1: Multiple Interface Host with Multiple Available Services

### 3.2. Homenet Scenario

We also describe the issues related to the homenet architecture [I-D.ietf-homenet-arch], as depicted in Figure 2.

Suppose one MIF host is connected to three domains: homenet domain, 3gpp domain and a WiFi or enterprise domain. There is one service that is named with the private domain name, say 'temperature.ietf', which is only resolvable via the domain name service residing inside the homenet and is supported by the multicast dns service [RFC6762].

There are several problems in this scenario. First of all, since the host has two unicast dns domains configured over the 3GPP and WiFi, and as well as a multicast service discovery domain within the homenet, the host does not know which domain it should send a dns resolution request. Secondly, even if coupled with the split dns solution [RFC6731], the configuration information obtained from DHCP supports only those two unicast dns domains, but not the homenet domain which is normally considered as 'zero-configuration'. Third, the service discovery problem will become more complicated if the host is connecting to more than one home networks, i.e., multiple multicast dns domains.

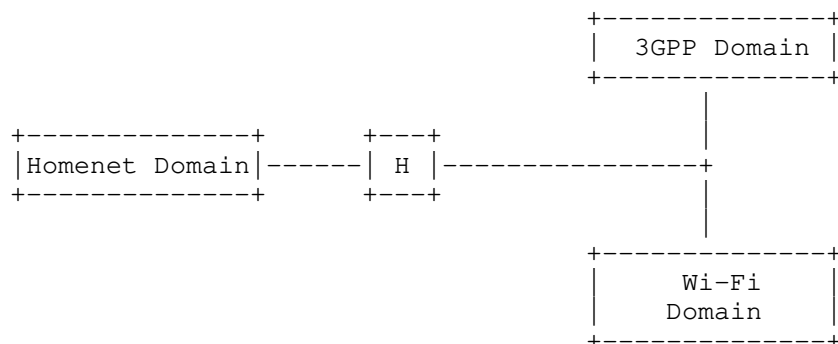


Figure 2: Homenet Scenario

### 4. Problem Analysis

The problems that a multiple-interfaced host may meet during the service discovery include:

1. How the query requests are sent? Because there are multiple interfaces available and multiple service rendezvous existing,

the host should decide which destination it ought to send the query to. And if there is a round-robin mechanism, the host should determine the order of the query request.

2. How to handle the query and reply? Some pointers to the service are restricted to a local scope or a certain interface, e.g., an office printer may not be accessible to the 3G-interface. The service discovery process is supposed to return a pointer that is accessible to the host.
3. How to access the service? Given the pointer to the service, the host should be able to determine from which interface it can access the service.

The existing work of [RFC6418] and [RFC6419] have covered the general problems encountered by hosts accessing multiple provisioning domains, but the focus is on connectivity and configuration. Proposal of Happy Eyeball in [I-D.ietf-mif-happy-eyeballs-extension] allows a host with multiple interfaces to pick a suitable one for access and enables automatic fallback. In a DNS based service discovery [RFC6763], the problem of domain split is analyzed in the [RFC6731]. The document defines an extension to the DHCPv4 and DHCPv6 to inform the MIF host which domain scope the Recursive DNS Server(RDNSS) is serving for, so that the "service query request" can be sent to the correct RDNSS to get an answer.

The existing proposals resolve the partial problem in the service discovery process mentioned above. To highlight the missing blocks, Figure 3 provides a 'gap' analysis. In the figure, we compare three existing solutions on service discovery, DNS-SD[RFC6763], DNS-Server-Selection [RFC6731], and MIF Happy Eyeball [I-D.ietf-mif-happy-eyeballs-extension], from three aspects as mentioned above. The DNS-Srv-Sel solution uses the defined DHCP option for the MIF host to select the corresponding DNS Server, and MIF-HE inherits this method in its most updated version. The MIF-HE can help host failover to the workable interface during service access while DNS-Srv-Sel does not handle this particular issue. The DNS-SD is not designed for a multiple interfaces environment and DNS server selection and request handling are based on standard DNS behaviors.

Aspects \ Sol	DNS-SD	DNS-Srv-Sel	MIF-HE
How to Send Query	Std. DNS behavior	DHCP Option informed	Same as DNS-Srv-Sel

How to Handle Queries	Std. DNS server behavior	selection based on option	Same as DNS-Srv-Sel
How to Access service	no guarantee of connectivity	not possible if ports rejected	Failover to the Happiest one

Figure 3: Gap Analysis of Existing Service Discovery Methods

In a complicated network as shown in Figure 4 , the host connects to the enterprise network via the wired interface, a WLAN network with the 802.11 interface, and an operator's network via the cellular interface. The three intranets have their own Firewall policies to the global Internet. On the enterprise network, many outgoing ports are restricted, and on the WLAN and operator's public network, there is more freedom. If the MIF host makes a DNS-SRV query to a service in a global domain, all the RDNS servers have the corresponding records. But say the service port number has been blocked by the enterprise network administrator, the DNS has no such information. Even if the DNS returns a pointer to the MIF host, the MIF host cannot access this service via the wired interface.

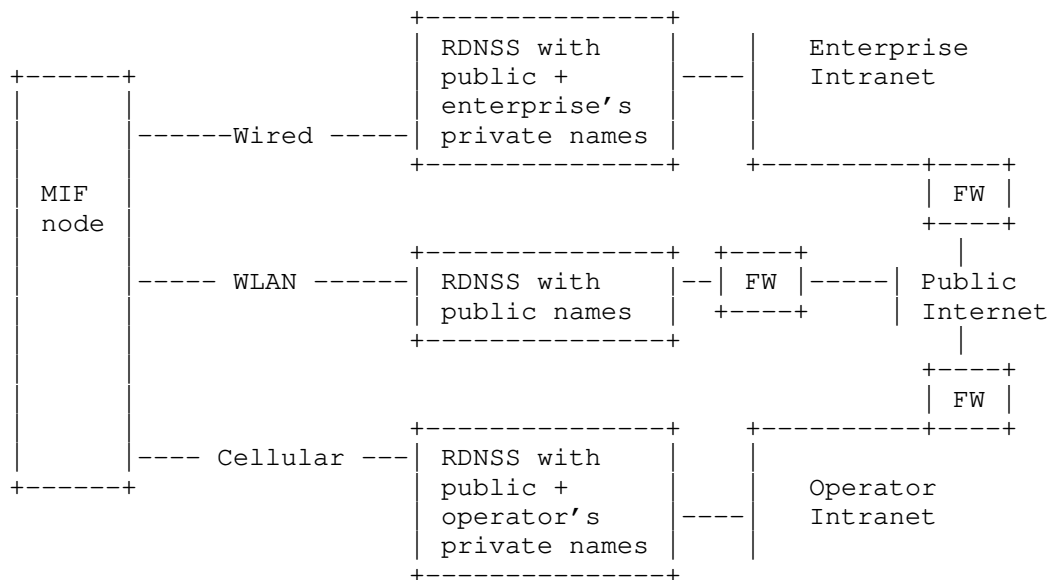


Figure 4: Scenario of Multiple Interface DNS Service Discovery

CoAP [I-D.ietf-core-coap] is an IETF designed RESTful protocol for constrained environment. CoAP defines a link-format for service discovery of the particular CoAP server, i.e., `"/.well-known/core"`. If the CoAP client has multiple access networks as shown in Figure 5, the situation turns to be more complex. For instance, if the MIF client wants to find a humidity sensing resource, but does not know which domain contains the information, it basically needs to send multiple CoAP GET requests with the well-known URL. Once it gets a response for the required resource, it can send the corresponding request to get the information. However this way is sub-optimal especially for constrained devices. MIF service discovery SHOULD consider the efficiency of the service discovery process.

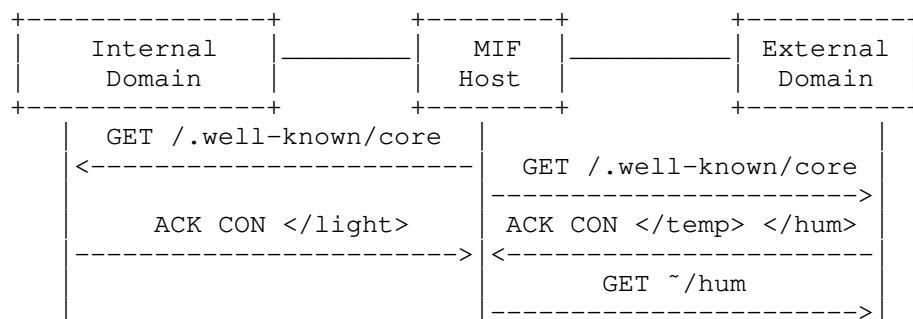


Figure 5: CoAP Service Discovery for a MIF Host

As a summary of the above analysis, the general problems and requirements of service discovery in a MIF environment can be summarized as follows:

**Service Directory Service Configuration:** Service directory is the entity that stores or can get the stored relationship between service names and service pointers. Different interfaces or provisioning domains have their different service directories. How to configure them on the MIF host and how the MIF host utilizes the configured information are important for the service discovery process to behave correctly.

**Service Directory Selection:** After the service directory information is configured on the host, the host is able to select the correct directory to send the query. The host can utilize auxiliary information available or send the query to all the directories that have been configured. The behavior of MIF host to select a correct directory is also important for a stable system.



Service Pointer/Address Resolution: The same service may have different available addresses and pointers, and some service has limited connectivity. So the resolution process should be able to return to the MIF host a record that is accessible from at least one of the interfaces. Efficiency SHOULD be taken into consideration in this phase.

Service Route Selection: With the pointers returned, the host should route the service level data to the service instance identified by the returned pointers.

## 5. IANA Considerations

This document has no IANA requests.

## 6. Security Considerations

The query response exchanges should be protected by security mechanisms. If the response contains invalid information, e.g. a pointer to a worm website, it harms. As a consequence, the service discovery should protect bogus information injected by attackers or intruders. The security consideration ought to be made by the underlining protocols, and it is out the scope of this problem statement document.

## 7. References

### 7.1. Normative References

- [I-D.ietf-mif-happy-eyeballs-extension]  
Chen, G., Williams, C., Wing, D., and A. Yourtchenko,  
"Happy Eyeballs Extension for Multiple Interfaces", draft-  
ietf-mif-happy-eyeballs-extension-02 (work in progress),  
February 2013.
- [RFC6731] Savolainen, T., Kato, J., and T. Lemon, "Improved  
Recursive DNS Server Selection for Multi-Interfaced  
Nodes", RFC 6731, December 2012.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762,  
February 2013.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service  
Discovery", RFC 6763, February 2013.

### 7.2. Informative References

[I-D.ietf-core-coap]

Shelby, Z., Hartke, K., and C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-18 (work in progress), June 2013.

[I-D.ietf-homenet-arch]

Chown, T., Arkko, J., Brandt, A., Troan, O., and J. Weil, "Home Networking Architecture for IPv6", draft-ietf-homenet-arch-10 (work in progress), August 2013.

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

[RFC6418] Blanchet, M. and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement", RFC 6418, November 2011.

[RFC6419] Wasserman, M. and P. Seite, "Current Practices for Multiple-Interface Hosts", RFC 6419, November 2011.

#### Authors' Addresses

Zhen Cao  
China Mobile  
Xuanwumenxi Ave. No. 32  
Beijing 100871  
China

Phone: +86-10-52686688  
Email: zehn.cao@gmail.com, caozhen@chinamobile.com

Aaron Yi Ding  
Cambridge University / Helsinki University  
William Gates Building, 15 JJ Thomson Ave  
CB3 0FD Cambridge  
United Kingdom

Phone: +44-7934034801; +358-9-19151296  
Email: Aaron.Ding@cl.cam.ac.uk

MIF WG  
Internet-Draft  
Intended status: Informational  
Expires: January 2, 2015

H. Deng  
China Mobile  
S. Krishnan  
Ericsson  
T. Lemon  
Nominum  
M. Wasserman  
Painless Security, LLC  
July 1, 2014

Guide for application developers on session continuity by using MIF API  
draft-deng-mif-api-session-continuity-guide-04

## Abstract

Today most smart terminals are equipped with multiple interfaces such as 3G/LTE and WiFi, and users experience some loss of connectivity while switching interfaces. The MIF API draft [I-D.ietf-mif-api-extension] has specified an API to announce interface status information to the applications. Once the application receives such information, it can use this information to reconnect to its peer(s), and this could significantly improve the user experience.

## 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 RFC 2119 [RFC2119].

## Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 January 2, 2015.

## Copyright Notice

Copyright (c) 2014 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

1. Introduction . . . . .	2
2. Related MIF API information . . . . .	3
3. Using different source address to reconnect the server . . . . .	3
4. Generic guidelines for writing applications to handle new interfaces becoming available . . . . .	4
5. Generic guidelines for writing applications to handle interfaces becoming unavailable . . . . .	5
6. IANA Considerations . . . . .	5
7. Security Considerations . . . . .	5
8. Acknowledgements . . . . .	5
9. Normative References . . . . .	6
Authors' Addresses . . . . .	6

## 1. Introduction

A significant and increasing number of smart mobile terminals have multiple interfaces for connectivity (e.g. Wifi and 3G/LTE). These interfaces may have very characteristics in terms of reliability, available bandwidth, delay/jitter as well as cost per bit. There is some form of connection manager on the end device that picks an interface for communication based on some pre-configured policy and/or based on dynamic conditions. The initially selected interface may become deprioritized (e.g. due to a lower cost interface becoming available) or may become unavailable (e.g. due to loss of coverage when moving out of a WiFi hotspot). New interfaces may become available due to administrative action (e.g. manual activation of a specific connectivity technology) or due to dynamic conditions (e.g. entering coverage area of a wireless network or plugging in an ethernet cable). In order to handle such changes in connectivity, applications need to be aware of network status changes and react to them. This document provides a guide to writing such applications.

The MIF API [I-D.ietf-mif-api-extension] document specifies an API that is capable of providing information regarding changes in network and interface connectivity status. By using this information, application developers can develop applications that can survive changes in connectivity and even benefit from them.

The MIF MPVD Architecture [I-D.ietf-mif-mpvd-arch] document defines the notion of a PVD (set of network configuration information), and PVDs somehow must be exposed, in case applications are not PVD-aware, or indirectly participating the selection of PVD, or knowing of the PVDs based on PVD APIs. The MIF API [I-D.ietf-mif-api-extension] document specifies "Connect to PVD" message, application developers may develop application that can changes between different PVD connectivity.

## 2. Related MIF API information

MIF API draft [I-D.ietf-mif-api-extension] defines a few messages that are related to notifying whether an interface is available or not. The messages are defined in Section 3.5.1 (Announce Interfaces) and Section 3.5.4 (No Interface). Similar functionality is available for addresses using the messages defined in Section 3.5.12 (Announce Address) and Section 3.5.14 (No Address Announcement). The API also specifies interface change information in section 3.5.23.5 (Interface is going up) and 3.5.23.4 (Interface is going away). Both interface and address information could be used by the application to infer the availability of a new endpoint for communication or the loss of an existing endpoint for communication.

## 3. Using different source address to reconnect the server

The applications deployed on mobile hosts usually setup the connection with the server, then trying to keep the connection up as long as they can. This works reasonable well when the host has only one communication interface. Once the host has more than one communication interface, such as 3G/LTE and WLAN, such applications cease to work well. e.g. The per bit cost and the connection speed are different on these two interfaces, and the user would always prefer to change another cheaper and faster connection. e.g. While connecting to a WLAN interface after being connected to LTE, the mobile terminal would get a different set of configuration parameters including the IP address, DNS server and default gateway. Application would normally break after such change in connectivity if the original interface (3G/LTE) is turned off and manual intervention is usually required to reinitiate connectivity.

If the application is designed with changing network connectivity in mind, then the application could be carefully designed reconnect to

its peer based on MIF API notification about new interface(s) and/or new address(es). The application needs to start testing the usability of the new interface(s)/address(es) immediately and determine whether they are usable and, if so, decide what traffic to switch over. Please note that there are other solutions for handling address changes in the lower layers (network and transport) like MIPv6, shim6, and MPTCP that can shield the application from address changes. The guidelines provided in this document are also applicable when these techniques are being used. Also, there might be load balancers present on the server side and it may become very difficult to preserve sessions after an address change has occurred.

In most cases even when a mobile terminal gets WLAN connectivity and gets an IP address assigned, but it could still be disconnected from the Internet due to lack of authentication. As a consequence, the interface needs to be tested for internet connectivity before switching communication from an existing interface to a newly available interface.

4. Generic guidelines for writing applications to handle new interfaces becoming available

The recommended steps for the application developer to keep the session continuity based on MIF API are listed below:

Step 1: Application subscribes to the MIF API for interface and address change notifications;

Step 2: Application connects to the server based on interface 1 (either 3G/LTE or WLAN);

Step 3: When a new interface comes up or a new address is configured, the MIF API notifies the application.

Step 4: The application tries to re-connect to its peer from the newly available interface. If the connectivity check succeeds, then the application can successfully switch the communication over to the new interface based on policy or user initiated selection. Otherwise communication stays on the existing interface. The decision process on how a preferred interface is selected is out of scope of this document and might be the topic for a separate high level API document.

Step 5: The interface initially used for communication may now be turned off without disrupting communications if no other applications are using it.

## 5. Generic guidelines for writing applications to handle interfaces becoming unavailable

The recommended steps for the application developer to keep the session continuity based on MIF API are listed below:

Step 1: Application subscribes to the MIF API for interface and address change notifications;

Step 2: Application connects to the server based on interface 1 (either 3G/LTE or WLAN);

Step 3: When an interface or address, that is currently being used for communication, becomes unavailable the MIF API notifies the application.

Step 4: The application requests the MIF API to acquire a list of interfaces that are currently available. Based on locally configured preferences, the application tries to re-connect to its peer from one of the available interfaces. If the connectivity check succeeds, then the application can successfully switch the communication over to this interface.

Step 5: If the connectivity check fails, the application needs to redo the check for each of the available interfaces in order of preference until it can successfully connect to its peer.

Step 6: If at least one available interface is still able to connect to the peer, the application can switch over to this interface without disrupting communications.

## 6. IANA Considerations

This document does not require any IANA actions.

## 7. Security Considerations

Some applications may associate the the source address of the communication with the credentials used, it they may require refreshing the credentials after the application switches to using a new source address.

## 8. Acknowledgements

The authors would like to thank Pete McCann, Julien Laganier, Dapeng Liu, Dave Thaler, Brian Carpenter and Pierrick Seite for their comments and suggestions for improving this document.

## 9. Normative References

[I-D.ietf-mif-api-extension]

Liu, D., Lemon, T., Ismailov, Y., and Z. Cao, "MIF API consideration", draft-ietf-mif-api-extension-05 (work in progress), February 2014.

[I-D.ietf-mif-mpvd-arch]

Anipko, D., "MIF MPVD Architecture", draft-ietf-mif-mpvd-arch-01 (work in progress), May 2014.

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

## Authors' Addresses

Hui Deng  
China Mobile  
No.32 Xuanwumen West Street  
Xicheng District,  
Beijing 100053  
China

Email: denghui02@gmail.com

Suresh Krishnan  
Ericsson  
8400 Blvd Decarie  
Town of Mount Royal, Quebec  
Canada

Email: suresh.krishnan@ericsson.com

Ted Lemon  
Nominum  
Redwood City,  
94063  
USA

Email: Ted.Lemon@nominum.com



Margaret Wasserman  
Painless Security, LLC  
356 Abbott Street,  
North Andover 01845  
USA

Email: [mrw@painless-security.com](mailto:mrw@painless-security.com)

MIF  
Internet-Draft  
Intended status: Informational  
Expires: August 19, 2014

D. Liu  
China Mobile  
Ted. Lemon  
Nominum  
Yuri. Ismailov  
Ericsson  
Z. Cao  
China Mobile  
February 15, 2014

MIF API consideration  
draft-ietf-mif-api-extension-05

Abstract

Hosts may connect to the internet using more than one network API at a time, or to a single network on which service is provided by more than one provider. Existing APIs are inadequate to allow applications to successfully use the network in this environment. This document presents a new abstract API that provides the minimal set of messages required to enable an application to communicate successfully in this environment.

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 August 19, 2014.

Copyright Notice

Copyright (c) 2014 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

1. Introduction . . . . .	2
2. Conventions used in this document . . . . .	3
3. MIF API Concept . . . . .	3
3.1. Provisioning Domains . . . . .	4
3.2. MIF API Elements . . . . .	4
3.2.1. Application Element . . . . .	5
3.2.2. High Level API . . . . .	5
3.2.3. MIF API . . . . .	6
3.2.4. Communications API . . . . .	6
3.2.5. Network Link API . . . . .	6
3.2.6. MIF API communication model . . . . .	7
3.2.7. MIF Messages . . . . .	7
3.3. Example Usage . . . . .	14
4. Security Considerations . . . . .	16
5. IANA Considerations . . . . .	16
6. Acknowledgments . . . . .	16
7. References . . . . .	16
7.1. Normative References . . . . .	16
7.2. Informative References . . . . .	16
Authors' Addresses . . . . .	17

## 1. Introduction

Traditionally, applications that communicate on the network have done so over a single network link, which is provided by a single service provider. However, this operating environment is now the exception rather than the rule. Most devices now have multiple wireless interfaces that are, in practice, connected to networks operated by different providers. These networks may or may not have different reachability characteristics with respect to any given service an application may wish to connect to.

For example, consider a typical modern host with two wireless interfaces: a wireless interface connected to a broadband network, and another connected to some kind of cellular network. The same host may also have a wired interface which is sometimes connected to a third broadband link. It is also quite common for hosts to have

VPN links that are configured, for example, for access to corporate networks, or for access to network privacy services.

As a result, it is now quite typical that a program attempting to communicate in such an environment will be presented with conflicting configuration information from more than one provider. In addition, the cost of bandwidth on different links and the power required by those links may require consideration.

The API specified in this document is intended to describe the minimal complete set of API calls required to implement higher level APIs that solve these problems. It is not expected that applications will be implemented to this API, although it should be possible to do so. Rather, we expect this API to be used as a basis for building higher-level APIs that provide domain-specific solutions to these problems. The reason for specifying a lower-level API is to enable any arbitrary domain-specific API to be implemented, since no single higher-level API is likely to satisfy the needs of every application.

The API specified here is an abstract API. This means that we specify the functionality that is required to implement the API, but we do not provide specific bindings for any programming language: these are left up to the implementation. The API is described in terms of messages sent and messages received, rather than in terms of procedure calls, because it is necessary to be able to interleave these messages; a procedure call API necessarily precludes interleaving.

This document is intended to be read and used as a checklist by operating system vendors who are interested in providing adequate functionality to applications that must run on hosts in environments like the ones described here. It should also be useful to purchasers of devices that must operate in such environments, so that they can tell if they are getting a device that can actually succeed in these environments.

## 2. Conventions used in this document

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].

## 3. MIF API Concept

The MIF API is intended to deal with situations where more than one interface may be active at a time. It must also deal with situations where a single interface is connected to a link that provides more

than one type of network service. The most common example of this that we expect is a dual-stack network configuration.

### 3.1. Provisioning Domains

Document [I-D.ietf-mif-mpvd-arch] defines Provisioning Domain (PvD) architecture and its associated mechanism, such as PvD identity/naming concept, conveying mechanism etc. According to [I-D.ietf-mif-mpvd-arch], a provisioning domain is a consistent set of network configuration information. Classically, the entire set available on a single interface is provided by a single source, such as network administrator, and can therefore be treated as a single provisioning domain. In modern IPv6 networks, multihoming can result in more than one provisioning domain being present on a single link.

To properly handle these multiple-service interfaces, we specify the API not in terms of interfaces, but in terms of provisioning domains. From the perspective of the MIF API, a provisioning domain consists of a link, plus all the configuration information received on that link for that provisioning domain. So for an IPv4 provisioning domain, that would be whatever information is received from the DHCP server. For an IPv6 provisioning domain, the information received through router advertisements would be combined with the information received via DHCPv6.

### 3.2. MIF API Elements

There are a number of different, essentially independent, pieces of software that need to be connected together in order to fully support a successful MIF communication strategy. These elements are shown in figure 3.1.

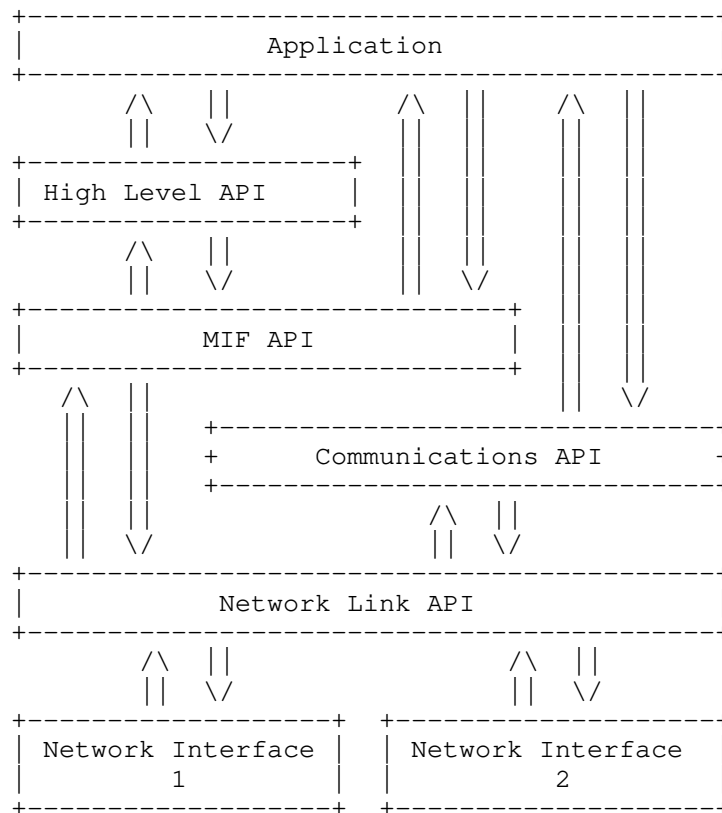


Figure 1: MIF API Elements

### 3.2.1. Application Element

This is an actual application. Applications fall into a variety of broad categories, including network servers, web browsers, peer-to-peer programs, and so on. Although we are focusing here on the mechanisms required to allow these applications to originate connections to remote nodes, it is worth noting that applications must also be able to receive connections from remote nodes.

### 3.2.2. High Level API

Applications are generally expected to originate connections using some general-purpose high-level API suited to their particular function. It is likely that different applications may use different high-level APIs to communicate, depending on their particular needs. We do not describe the functioning of such high-level APIs; however,

one such API under current consideration is the Happy Eyeballs for MIF [reference]. These APIs are expected to be able to be implemented using functionality like that described in the MIF API.

### 3.2.3. MIF API

This is the API being described in this document. Generally speaking, this API is used by higher-level APIs. However, it is permissible for applications to use the MIF API when it is deemed necessary. Currently, several modern web browsers take this approach to establishing network connections, rather than relying on vendor-provided connection mechanisms.

### 3.2.4. Communications API

Once an application has originated a connection with a remote node using either a high-level API or the MIF API, it must communicate. Similarly, when an application receives a connection from a remote node, it must communicate with that remote node. The communications API is used for this communication. Popular examples of such APIs include the POSIX socket API and a variety of other related APIs.

It is likely that in some instances, implementations of the MIF API will be done as extensions to the Communications API provided by a particular operating system; the functional separation we show here is intended to allow us to illustrate only those features required in a MIF environment, while relying on existing communications APIs to provide the rest.

### 3.2.5. Network Link API

This is the software that is responsible for actually managing whatever network links are present on a node, whether these are physical links or tunnels. What precisely this functional box contains may vary greatly from device to device. On a typical modern computer workstation, this functionality would almost certainly reside entirely in the system kernel; however, on an embedded device everything from the Application down to the Network Link API could easily be running together on the bare metal as a single program.

The Network Link API can completely be concealed from the Application, so we don't show a connection between them on the functional diagram, and indeed we do not talk about the functionality provided by this API. The reason for showing it on the functional diagram is simply to show that there likely is an API in common between MIF and the Communications API.

### 3.2.6. MIF API communication model

MIF API requests are made in the form of messages posted to the MIF API, and messages received from it. To accomplish this, several API calls are available. These calls mediate communication between the MIF API and the High Level API, or between the MIF API and the Application. In addition, the CHECK MESSAGE call allows the application to probe for or wait for messages from any of the APIs.

#### 3.2.6.1. POST MESSAGE call

This call causes a message to be posted to the MIF API. The call posts the message, and then returns.

#### 3.2.6.2. CHECK MESSAGE call

This call checks to see if there is a message waiting either from the High Level API, the MIF API, or the Communications API. Ideally it should be able to report the availability of any message or event that the application might anticipate receiving, so that the application can simply block waiting for such an event using this call. The application should be able to do a non-blocking probe, wait for some limited period of time, or wait indefinitely.

An example of a function of this type in existing practice is the POSIX poll() system call.

#### 3.2.6.3. GET MESSAGE call

This call checks to see if there is a message waiting. If there is no message, it returns a status code indicating that there is no message waiting. If there is a message, it returns the message.

### 3.2.7. MIF Messages

MIF messages always go in one direction or the other: from the subscriber to the MIF API, or to the subscriber from the MIF API. We use the term "subscriber" here to mean either the Application or the High Level API, since either is permitted to communicate with the MIF API.

Messages described here are grouped according to function.

#### 3.2.7.1. Announce Interfaces

This message is sent to the MIF API to ask it to send a message announcing the existence of any interface. When the MIF API receives this message from a subscriber, it iterates across the list of all



known interfaces; for each known interface, it sends an Interface Announcement message to the subscriber.

In addition, the MIF API sets a flag indicating that the subscriber is interested in learning about new interfaces. When the MIF API detects the presence of a new interface, it sends an Interface Announcement message for that interface to the subscriber. This would happen, for instance, when a new tunnel is configured, or when a USB device that is a network interface is discovered by the Network API.

Also, if a network interface goes away, either because the physical network device is disconnected, or because a tunnel is disabled, the MIF API will send a No Interface Announcement message to the subscriber.

#### 3.2.7.2. Stop Announcing Interfaces

This message is sent to the MIF API when a subscriber is no longer interested in receiving announcements about new interfaces. Subsequently, the MIF API will no longer send Interface Announcement or No Interface Announcement messages to the subscriber.

#### 3.2.7.3. Interface Announcement

This message announces the existence of an interface. The announcement includes an interface display name and interface identifier.

#### 3.2.7.4. No Interface Announcement

This message announces that an interface that had been previously announced is no longer present. The announcement includes the interface identifier.

#### 3.2.7.5. Announce Provisioning Domain

This message requests the MIF API to announce the availability of any provisioning domains configured on a particular interface. The interface identifier must be specified.

Upon receipt, the MIF API will iterate across the list of Provisioning Domains present for a particular interface, and will send a Provisioning Domain Announcement for each such Provisioning Domain.

In addition, the MIF API will set a flag indicating that the subscriber wishes to know about new provisioning domains as they

appear. Subsequently, when a new Provisioning Domain appears, the MIF API will send a Provisioning Domain Announcement message to the subscriber.

Finally, if a Provisioning Domain expires or is invalidated, the MIF API will send the subscriber a No Provisioning Domain Announcement message for that Provisioning Domain.

In the event that an interface on which provisioning domains has been announced goes away, a No Provisioning Domain Announcement message will be sent for each provisioning domain that had previously been announced on that interface before the No Interface Announcement message is sent.

Once a No Interface Announcement message has been sent, any subscriber that had subscribed to Provisioning Domain announcements for that interface will be automatically unsubscribed.

#### 3.2.7.6. Stop Announcing Provisioning Domains

This message requests that the MIF API stop sending the subscriber Provisioning Domain Announcement and No Provisioning Domain Announcement messages. The subscriber must indicate the interface for which it no longer wishes to receive Provisioning Domain announcements.

#### 3.2.7.7. Provisioning Domain Announcement

This message is sent by the MIF API to the subscriber to indicate that a new Provisioning Domain has successfully been configured on an interface. The announcement includes the interface identifier and the provisioning domain identifier.

#### 3.2.7.8. No Provisioning Domain Announcement

This message is sent by the MIF API to the subscriber to indicate that an existing, previously announced provisioning domain has expired or otherwise become invalid, and can no longer be used.

#### 3.2.7.9. Announce Configuration Element

This message is sent by the subscriber to request a specific configuration element from a specific provisioning domain. A provisioning domain identifier must be specified.

The MIF API will respond by iterating across the complete list of configuration elements for a provisioning domain, sending a

Configuration Element Announcement message to the subscriber for each one.

Additionally, if any Configuration Elements subsequently complete for a particular provisioning domain, the MIF API will send a Configuration Element Announcement message to the subscriber for each such element. If a Configuration Element becomes invalidated after it has been announced, the MIF API will send a No Configuration Element message.

If a provisioning domain expires or becomes invalid, the MIF API will iterate across the list of remaining configuration elements for that provisioning domain and send a No Configuration Element Announcement message for each such configuration element.

#### 3.2.7.10. Configuration Element Announcement

The Configuration Element Announcement message includes a Provisioning Domain ID and a Configuration Element Type, which can be one of the following: Config Element RA Config Element DHCPv6 Config Element DHCPv4 etc.

#### 3.2.7.11. No Configuration Element Announcement

The No Configuration Element Announcement message indicates that a previously valid configuration element for a provisioning domain is no longer valid. The message includes a provisioning domain identifier and a configuration element type.

#### 3.2.7.12. Stop Announce Configuration Element

The Stop Announce Configuration Element message requests that MIF API stop announce configuration element.

#### 3.2.7.13. Announce Address

This message is sent by the subscriber to request announcements of valid IP addresses for a specific provisioning domain. A provisioning domain identifier must be specified.

The MIF API will respond by iterating across the complete list of configuration elements for a provisioning domain, sending a Address Announcement message to the subscriber.

Additionally, if any new Address is subsequently configured on a particular provisioning domain, the MIF API will send an Address Announcement message to the subscriber for each such element. If an

address becomes invalidated after it has been announced, the MIF API will send a No Address Announcement message.

If a provisioning domain expires or becomes invalid, the MIF API will iterate across the list of remaining configuration elements for that provisioning domain and send a No Address Announcement message for each such address.

#### 3.2.7.14. Address Announcement

The Address Announcement message includes single IPv4 or IPv6 address and a Provisioning Domain identifier, as well as the valid and preferred lifetimes for that IP address (IPv6 only).

#### 3.2.7.15. Stop Announcing Address

The Stop Announcing Address message requests the MIF API to stop announcing address.

#### 3.2.7.16. No Address Announcement

The No Address Announcement message indicates that a previously valid address for a provisioning domain is no longer valid. The message includes a provisioning domain identifier and an IPv4 or IPv6 address.

#### 3.2.7.17. Get Configuration Data

The Get Configuration Data message is sent to the MIF API, and includes a Provisioning Domain ID, a Configuration Element Type, and a Configuration Information Identifier.

Configuration Information Identifiers: DNS Server List etc.

The MIF API searches the configuration database for the specific type of Configuration Element on the specified Provisioning Domain to see if there is any configuration data of the specified type. If so, the MIF API sends a Configuration Data message to the subscriber; otherwise it sends a No Configuration Data message to the subscriber.

#### 3.2.7.18. Translate Name

The Translate Name message is sent to the MIF API. It includes a provisioning domain and a name, which is a UTF8 string naming a network node. The message also includes a Translation Identifier, which the subscriber must ensure is unique across all outstanding name service requests.

The MIF API begins a name resolution process. As results come in from the name resolution process, the MIF API sends Name Translation messages to the subscriber for each such result.

Name resolution can be handled by one or more translations systems such as local host table lookup, Domain Name System, NIS, LLNMR, and is implementation-dependent. \*\*need to think about this

#### 3.2.7.19. Stop Translating Name

This message is sent to the MIF API to indicate that the subscriber is no longer interested in additional results from a particular name translation process. The message includes the Translation Identifier.

#### 3.2.7.20. Name Translation

The MIF API sends a Name Translation message to subscribers whenever results come in from a name translation process being performed on behalf of the subscriber. The Name Translation message includes the Translation ID generated by the subscriber, and an IP address returned by the translation process. If a single translation result contains more than one IP address, or IP addresses of different types, the MIF API sends a single Name Translation message for each such IP address.

#### 3.2.7.21. Connect to PvD

The Connect to PvD message is used for the advanced application to select the PvD. Advanced application can use this message to select a specific PvD by providing the PvD identifier as parameter. This is the advanced case that discussed in section 6.3 of [I-D.ietf-mif-mpvd-arch].

#### 3.2.7.22. Connect to Address

The Connect to Address message contains an IP address, a provisioning domain identifier, and a connection identifier which the subscriber must ensure is unique. The MIF API attempts to initiate a TCP connection to the specified IP address using one or more source addresses that are valid for the specified provisioning domain, according to the source address selection policy for that provisioning domain.

If the connection subsequently succeeds, the MIF API will send a Connected message to the subscriber. If it subsequently fails, the MIF API will send a Not Connected message to the subscriber.

### 3.2.7.23. Connect to Address From Address

The Connect to Address From Address message contains a source IP address, a destination IP address, a provisioning domain identifier, and a connection identifier which the subscriber must ensure is unique. The MIF API attempts to initiate a TCP connection to the specified IP address using the specified source address.

If the connection subsequently succeeds, the MIF API will send a Connected message to the subscriber. If it subsequently fails, the MIF API will send a Connection Failed message to the subscriber.

### 3.2.7.24. Connected

The Connected message contains the connection identifier that was provided in a previous Connect to Address or Connect to Address From Address message sent by the subscriber. It also contains an token, suitable for use with the connection API, for communicating with the end node to which the connection was established.

### 3.2.7.25. Not Connected

The Not Connected message contains the connection identifier that was provided in a previous Connect to Address or Connect to Address From Address message sent by the subscriber. It also contains an indication as to what went wrong with the connection.

### 3.2.7.26. Application Connectivity Management

The following APIs are used for application connectivity management.

#### 3.2.7.26.1. Application: Wants to connect

This message is sent by the application to the MIF API that indicates the application wants to connect to the network. The purpose of this call is to trigger the MIF API to engage in any work that is required to configure the network. If all interfaces are already operational, this message is a no-op. An application would typically send this message either because it has no provisioning domains on which it can attempt to connect, or because it has failed to connect on any existing provisioning domain.

#### 3.2.7.26.2. Application: Connection is idle

This message is sent by the applicaiton to the MIF API to indicate that the application is not expecting to receive any data or send any data. This is a signal to the MIF API that, for example a radio that consumes a lot of power can be put into a temporary idle state, but

that the application expects to resume communication in the future using the existing connection.

#### 3.2.7.26.3. Application: Connection can be broken

This message is sent by the application to the MIF API to indicate that the application can tolerate the connection being broken. This is a signal that the application could use the connection in the future if it were not broken, but can re-establish the connection if it is broken without any loss of functionality. A MIF API implementation on a power-conservative device might take this as a signal to shut down radios to conserve power.

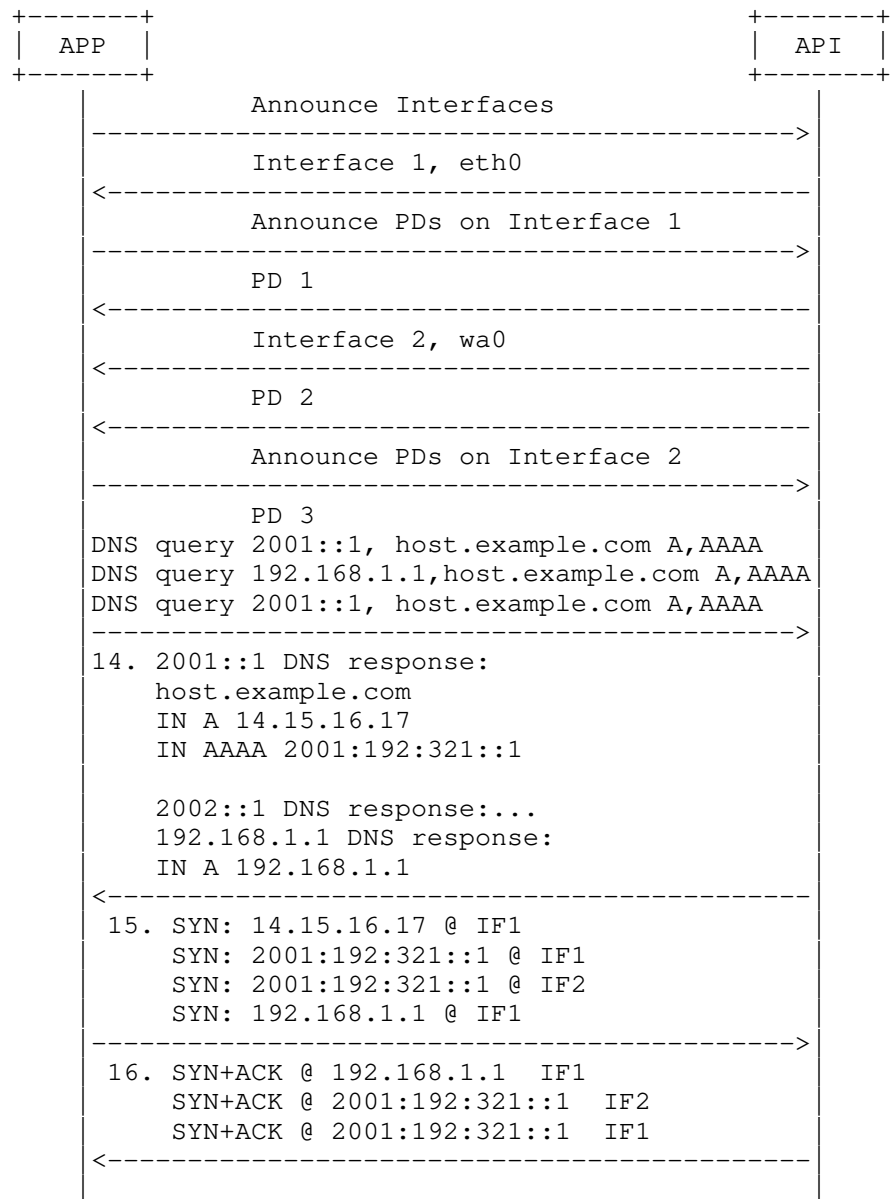
#### 3.2.7.26.4. Interface is going away

This message is sent by the MIF API to the application to indicate that an interface is going away. This can happen when the interface is still up but the system intends to take it down.

#### 3.2.7.26.5. Interface is going up

This message is sent by the MIF API to the application to indicate that an interface is going up. This can happen when the interface is still down but the system intends to take it up.

### 3.3. Example Usage



## MIF API communication model

As shown in the preceding example, the application first invokes the MIF API to get a list of all the network interfaces in the host. As



soon as each interface has been identified, the application invokes the MIF API to get a list of provisioning domains that are attached to that interface.

The application then invokes the MIF API to look up a name in the context of each provisioning domain. The name lookup may return more than one IP address for each queried host name.

The application then tries to connect to each such IP addresses by sending tcp SYN packet to each destination IP addresses through the provisioning domain on which it received that name. Some of the destination IP addresses may return an ACK packet; others may not.

The application then chooses a connection based on its preferred criteria. For example, the criteria may based on the quality of the link, who answered first, or whether, for example, a TLS authentication succeeds on that connection.

#### 4. Security Considerations

This document specifies an abstract API and will not affect any existing protocols. It does not introduce any new security risk.

#### 5. IANA Considerations

None

#### 6. Acknowledgments

The authors want to thank Teemu Savolainen from Nokia, Dayi Zhao from Bitway, Dave Thaler from Microsoft and others for their useful suggestions and discussions. We would also like to acknowledge Yuri Ismailov's work as the author of the initial version of this document, but was drawn away by other work and let us continue.

#### 7. References

##### 7.1. Normative References

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

##### 7.2. Informative References

[I-D.ietf-mif-mpvd-arch]  
Anipko, D., "Multiple Provisioning Domain Architecture", draft-ietf-mif-mpvd-arch-00 (work in progress), February 2014.

[I-D.scharf-mptcp-api]

Scharf, M. and A. Ford, "MPTCP Application Interface Considerations", draft-scharf-mptcp-api-02 (work in progress), July 2010.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

#### Authors' Addresses

Dapeng Liu  
China Mobile  
Unit2, 28 Xuanwumenxi Ave, Xuanwu District  
Beijing 100053  
China

Email: liudapeng@chinamobile.com

Ted Lemon  
Nominum  
Redwood City  
CA 94063  
USA

Email: Ted.Lemon@nominum.com

Yuri Ismailov  
Ericsson  
Stockholm  
Sweden  
USA

Email: yuri@ismailov.eu

Zhen Cao  
China Mobile  
Unit2, 28 Xuanwumenxi Ave, Xuanwu District  
Beijing 100053  
China

Email: caozhen@chinamobile.com

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 27, 2012

W. Dec  
Cisco Systems  
T. Mrugalski, Ed.  
ISC  
T. Sun  
China Mobile  
B. Sarikaya  
Huawei USA  
February 24, 2012

DHCPv6 Route Options  
draft-ietf-mif-dhcpv6-route-option-04

Abstract

This document describes DHCPv6 Route Options for provisioning IPv6 routes on DHCPv6 client nodes. This is expected to improve the ability of an operator to configure and influence a nodes' ability to pick an appropriate route to a destination when this node is multi-homed and where other means of route configuration may be impractical.

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 RFC 2119 [RFC2119].

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 August 27, 2012.

Copyright Notice

Copyright (c) 2012 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

1. Introduction . . . . .	3
2. Problem overview . . . . .	3
3. Motivation . . . . .	4
3.1. Use cases . . . . .	4
3.2. Raised concerns . . . . .	9
3.2.1. Vendor-specific option . . . . .	9
3.2.2. Unicast RA . . . . .	9
3.2.3. DHCPv6 requires client to use one server . . . . .	10
3.2.4. Use VLANs . . . . .	10
4. DHCPv6 Based Solution . . . . .	11
4.1. Default route configuration . . . . .	11
4.2. Configuring on-link routes . . . . .	11
4.3. Deleting obsolete route . . . . .	11
4.4. Applicability to routers . . . . .	12
4.5. Updating Routing Information . . . . .	12
4.6. Limitations . . . . .	13
5. DHCPv6 Route Options . . . . .	13
5.1. Next Hop Option Format . . . . .	14
5.2. Route Prefix Option Format . . . . .	15
6. DHCPv6 Server Behavior . . . . .	16
7. DHCPv6 Client Behavior . . . . .	17
7.1. Conflict resolution . . . . .	18
8. IANA Considerations . . . . .	18
9. Security Considerations . . . . .	19
10. Contributors and Acknowledgements . . . . .	19
11. References . . . . .	20
11.1. Normative References . . . . .	20
11.2. Informative References . . . . .	20
Authors' Addresses . . . . .	21

## 1. Introduction

The Neighbor Discovery (ND) protocol [RFC4861] provides a mechanism for hosts to discover one or more default routers on a directly connected network segment. Extensions to the Router Advertisement (RA) protocol defined in [RFC4191] allow hosts to discover the preferences for multiple default routers on a given link, as well as any specific routes advertised by these routers. This provides network administrators with a new set of tools handle multi-homed host topologies and influence the route selection by the host. This ND based mechanism however is sub optimal or impractical in some multi-homing scenarios, where DHCPv6 [RFC3315] is seen to be more viable.

This draft defines the DHCPv6 Route Options for provisioning IPv6 routes on DHCPv6 clients. The proposed option is primarily envisaged for use by DHCPv6 client nodes that are capable of making basic IP routing decisions and maintaining an IPv6 routing table, broadly in line with the capabilities of a generic host as described in [RFC4191].

Throughout the document the words node and client are used as a reference to the device with such routing capabilities, hosting the DHCPv6 client software. The route information is taken to be equivalent to static routing, and limited in the number of required routes to a handful.

## 2. Problem overview

The solution described in this document applies to multi-homed scenarios including ones where the client is simultaneously connected to multiple access network (e.g. WiFi and 3G). The following scenario is used to illustrate the problem as found in typical multi-homed residential access networks. It is duly noted that the problem is not specific to IPv6, occurring also with IPv4, where it is today solved by means of DHCPv4 classless route information option [RFC3442], or alternative configuration mechanisms.

In multi-homed networks, a given user's node may be connected to more than one gateway. Such connectivity may be realized by means of dedicated physical or logical links that may also be shared with other users nodes. In such multi-homed networks it is quite common for the network operator to offer the delivery of a particular type of IP service via a particular gateway, where the service can be characterised by means of specific destination IP network prefixes. Thus, from an IP routing perspective in order for the user node to select the appropriate gateway for a given destination IP prefix,

recourse needs to be made to classic longest destination match IP routing, with the node acquiring such prefixes into its routing table. This is typically the remit of dynamic Internal Gateway Protocols (IGPs), which however are rarely used by operators in residential access networks. This is primarily due to operational costs and a desire to contain the complexity of user nodes and IP Edge devices to a minimum. While, IP Route configuration may be achieved using the ICMPv6 extensions defined in [RFC4191], this mechanism does not lend itself to other operational constraints such as the desire to control the route information on a per node basis, the ability to determine whether a given node is actually capable of receiving/processing such route information. A preferred mechanism, and one that additionally also lends itself to centralized management independent of the management of the gateways, is that of using the DHCP protocol for conveying route information to the nodes.

### 3. Motivation

The following section enumerates use cases, both in existing networks and as well as in envisaged future deployments. Usage scenarios are specified here in no particular order. As those use cases are described by various network operators, their scenarios may partially overlap.

Discussion: this section is rather long. Nevertheless, there were concerns raised that such option is not needed. Such extensive list can possibly solve those concerns. Number of use cases should be limited in future revisions. Alternatively, they can be moved to a separate motivation draft, if needed.

#### 3.1. Use cases

Use case 1: In Broadband network environment where the CPE is multi-homed to two upstream edge routers and each router provides connectivity for different types of services for example internet access and Video on Demand (restricted inside a walled garden) and the Service Provider would like to avoid routing on the CPE, there is a need to provision static route entries on RGs/CPEs. Service Provider requires a centralized control/management point for storing the customer's related information (IPv6 prefix, IPv6 routes and other provisioned information) and DHCPv6 is a good place for that. Using RA's would require to manually provision the edge router and this operation is not always possible, for example when router is operated by 3rd party. Broadband Forum document WT-124 issue 3 [BBF-WT-124] calls for this draft to solve the problem.

Use case 2: Operators want (approximate) feature parity so that they

can have (approximate) alignment between their operational procedures for v4 and v6, especially in a dual stack network. Having similar mechanisms for both protocols is desired due to lower operational expenses (OPEX).

Use case 3: In cellular networks, it is efficient for the network to configure routing information in central DHCPv6 server to do unified routing policy information. The gateways (GGSN in cellular network) only need to perform DHCPv6 relay. The Option code sent by clients can be used as an indication that host is MIF capable, so that network need not to do such configuration to host without MIF capabilities.

Use case 4: In cellular network, DHCPv6 is used for IPv6 parameter configuration and RA is used for SLACC of handset. This behavior was introduced in 3GPP Release 8 (or earlier). The network gateway in cellular network (e.g., GGSN) can naturally support DHCPv6 extension since the gateway acts as a DHCPv6 relay. However, it is very hard to update those gateways to use RA announcing the route information. The handsets with MIF feature need to visit subscribed/operator provided service. Some traffic is routed to the operator's network through 3G interface instead of to Internet through WiFi. DHCPv6 will be used to configure these specific routes. This use case is described in [THREEGPP-23.853].

Use case 5: PMIPv6 use case in LTE network. In LTE cellular network, both GTP and PMIPv6 are used for mobility management. In GTP, it is a point-to-point link between mobile host and PGW (PDN Gateway). However, in PMIPv6 case, the point-to-point link is between mobile host and SGW(Serving Gateway). The PGW sends /64 prefix to SGW through PBA. The SGW sends RA to mobile host. Route option may be needed when the host is multi-homed if it is simultaneously connected to the cellular network and WiFi or it simultaneously connects to multiple APNs in the cellular network. If RA is used for route configuration, both PGW and SGW(whose number is larger than PGW) need to be updated. Moreover, since a host can only connect to one SGW at a time, the SGW have to keep multiple route information received from different PGWs for one host and send them by RA to the host separately. This makes RA is not favorable in this use case.

Use case 6: WiFi networks. Some WiFi hotspots provide local services ("walled garden"). The route configuration on hosts or RGs is needed to direct some traffic to local network, while other traffic to the Internet. While this can be achieved using Route Information Option (RIO) in RA for all nodes that support [RFC4191], it does not allow doing so on a per-host basis.

Use case 7: VPN network. When a user connects to enterprise VPN

network, the routing of VPN traffic need to be configured. Due to the large number of such VPN networks, we cannot assume all the VPN network only use RA. DHCPv6 provides another choice which may be preferred by the VPN network. This situation is described in [RFC4191], Section 5.2. Hosts that do not support RFC4191 will not operate properly.

Use case 8: Selective walled garden. Figure 1 illustrates the case of two clients connected to a shared link. Both clients are assumed to have global IPv6 addresses and obtain their Internet connectivity via Router2 by means of a configured or a discovered default route. Client 1 however, unlike Client 2, is intended to run a specific application, e.g. VoIP, that is meant to access ServerA by means of Router1 with Server A being otherwise not reachable from the Internet. In addition to the global IP address Client1 may be assigned with another IP address of a more restricted scope for the purpose of communicating with Server A.

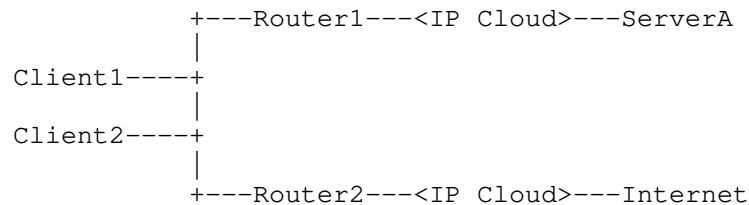


Figure 1: Walled garden scenario

The problem in the above scenario comes down to the fact that in order to reach Server A, Client1 requires to use a more specific route whose next-hop address is Router1. An ICMPv6 based mechanism for disseminating more specific route information, as defined in [RFC4191], disseminates this information via the shared link also to Client2. Often the operator wants to avoid this redundant dissemination to passing to Client2. In addition many operators prefer to be able to manage specific client route information from a centralized repository instead of managing directly such configuration on a router, as is required with the ICMPv6 based scheme. The former requirement is driven by the desire to provide to each client only the information required for their intended role which may be tied to a specific service, as well as to allow the possibility to introduce other routers into the scenario for purposes of load sharing. The requirement for more centralized configuration management is often due to administrative boundaries within an operator's organization as well as an existing operational practice that are in place for IPv4, all of which make router based configuration difficult.



Use case 9: Multihoming problem. A multihomed IPv6 host or gateway needs to solve at least 3 problems to operate properly when more than one link is operational:

1. Source address selection
2. Next-hop selection
3. DNS server selection

Problems one and three are solved by [I-D.ietf-6man-addr-select-opt] and [I-D.ietf-mif-dns-server-selection], respectively. It should be noted that both mechanisms use DHCPv6 as well. This draft attempts to solve problem two. Below is a brief explanation of the problem. See draft [I-D.ietf-v6ops-ipv6-multihoming-without-ipv6nat] for detailed problem analysis, background information and additional discussion regarding the need for a DHCPv6 solution to route information problem and IPv6 multihoming in general (with focus on aforementioned 3 problems).

In multihoming environment, server can restrict assignment of additional prefixes only to hosts that support more advanced next-hop and address selection requirements. (See Section 5.2 of [I-D.ietf-v6ops-ipv6-multihoming-without-ipv6nat]). Obviously this MUST be done on a per-host basis. Information about node capability is obtained via Option Request Option (ORO) in Solicit message, so support for Route Options is also used as means to report node capabilities to a network.

Use case 10: In static networks (i.e. networks that have static routers that are not changing over time, like home network with), such as some enterprise, hosting provider networks or even home network with a single router, it may be possible to stop using RA mechanism and deliver all configuration parameters to hosts using DHCPv6 only. This approach solves the rogue RA problem (i.e. a node that is not an approved router starts announcing RA in a network may hijack traffic from other hosts). This approach may be appealing in some cases, but not in all. For example if there is security association shared between clients and a DHCPv6 server, it may be useful to trust DHCP and disable RA mechanism. Also, environments that need DHCP for extended information, including but not limited to communicating information like DNS servers, hostnames, NTP servers, TFTP boot information and so on are forced to run two protocols increasing complexity and troubleshooting, where we have proof of concept in IPv4 that only one protocol (DHCP) should be needed.

Use case 11: It also has been proposed that route information option may be used as tie breaker in networks that deploy both DHCPv6 route

option and RA. DHCPv6 server could announce routing information along with RA. Legitimate router is also announced over DHCPv6. Host that receives conflicting information over RA may use additional information received from DHCPv6 as a tie breaker. This proposal [nanog-beijnum] was not investigated further.

Use case 12: DHCP-based configuration provides different failure mode than RA. While RA-based configuration works better in networks that offer redundant uplink using separate routers (second router can quickly take over upstream traffic), there are many deployments that cannot use that advantage, because of a single uplink. Current home networks with a single uplink as most obvious example. On the other hand, RA is more severely impacted by rogue entity problem. New rogue RA device may instantly break all other devices on the network. New rogue DHCP server will cause no immediate harm, may cause slow breakage over time, and may in fact never cause any breakage. This is due to the fundamental design choices of each protocol and it is hard to make either work the other way.

Use case 13: DHCP-based configuration may use mostly unicast traffic, while RA-based configuration mostly uses multicast. In some environments implementing multicast traffic may be cumbersome, e.g. in WiMAX environment not every subscriber station (SS) supports multicast channels and multicast capability must be emulated by base station (BS) using redundant transmissions. Classic, stateless, multicasted RA is in disadvantage compared to DHCP with standard unicast option enabled. While it is possible to selectively send unicasted RAs to selected subscribers, such architecture is essentially a stateful RA, thus forfeiting major benefit of RA being stateless.

Use case 14: Separated networks. In networks that do not have any routers, two DHCPv6 clients get a global address from DHCPv6 server. They cannot ping each other due to the fact that they do not know prefix that is available on-link. While it is tempting to suggest that separated networks should use link-local addressing, other factors should be taken into consideration. A stateful DHCPv6 may be used as a node monitoring tool, thus having advantage over link-local address usage. The also may be sensor networks that have outside connectivity only sporadically, e.g. uplink is established periodically to gather readings, but most of the time router is powered down for power reasons. Route Option in DHCPv6 could be used to configure on-link routes, while router could announce itself using short-lived RA.

Those requirements and use cases can be summarized as following:

1. In view of the DHCPv6 requirements in several fields, vendor-specific options lead to several segmented definitions. An IETF defined general option is a better choice.
2. Per user/host configuration makes DHCPv6 be used for the on-demand configuration.
3. As there is no well-defined central management system for prefix delegation and routing options via RA, it seems that DHCPv6 is the only available solution. It is better to have a generic option than a bunch of competing vendor options.
4. While this work was initially started with multihoming in mind, it is useful for single interface devices as well.

In a sense this route configuration mechanism makes DHCPv6 complete. Without it, this protocol cannot fully provision all configuration parameters to a host on its own.

### 3.2. Raised concerns

Opponents of this option proposed several alternative approaches. This section attempts to address raised issues.

#### 3.2.1. Vendor-specific option

Claim: During discussion about route configuration, some opponents say that routing information should be defined as vendor specific option.

Response: There are many ISPs, cellular and BBF network operators, CPE vendors, hardware vendors, DHCP implementors that want to implement and deploy this mechanism. Using vendor-specific option would severely limit interoperability and would make adoption and deployment much more complicated.

This solution is not a technology-specific requirement, it is requested by wide variety of companies, so it is not a vendor specific.

#### 3.2.2. Unicast RA

Claim: Some proponents insist that instead of using DHCPv6 solution, RA should be used instead. Some propose to send unicast RA with RIO option on a per-host basis.

Response: While this approach technically does not violate existing specs, it uses RA in a stateful way, thus the benefit of RA being

stateless is lost. Furthermore, it would require deploying additional mechanism, like RADIUS to deliver necessary information about hosts to routers. Authors consider deploying such stateful RA server with RADIUS support more complicated to deploy than the solution it tries to avoid (DHCPv6).

As there is no well-defined central management system for prefix delegation and routing options via RA, it seems that DHCPv6 is the only available solution. It is better to have a generic option than a bunch of competing vendor options.

Another concern raised is that RIO is not mandatory nor optional in 3GPP system and there is currently not support in 29.061 RADIUS or Diameter profile, so use of that alternative is somewhat limited in some cases.

### 3.2.3. DHCPv6 requires client to use one server

Claim: DHCPv6 has less rich semantics as client has to pick one out of all available server.

Response: While that is how currently most clients are implemented, there is nothing in [RFC3315] that mandates that. It is true that DHCPv6 was not designed with several provisioning domains. On the contrary, section 17.1.3 states that "Upon receipt of one or more valid Advertise messages, the client selects one or more Advertise messages based upon the following criteria.". This means that DHCPv6 client can obtain parameters from all available DHCPv6 servers, not just selected one. As such, DHCPv6 may work with overlapping provisioning domains. Authors acknowledge that this possibility is currently rather theoretical, as most known implementations do not take advantage of that possibility.

### 3.2.4. Use VLANs

Claim: There was a proposal to use VLANs as a solution to lack of per-host capability in RA mechanism.

Response: Deploying VLANs complicates network topology much more than adding a single DHCPv6 option. Furthermore in many cases it is not possible to deploy VLANs in any reasonable way, e.g. in multihost environment. Also, low cost devices (e.g. CPE) often do not offer VLAN capabilities, but they are very much capable of supporting DHCPv6. Another objection of esthetic nature. Using layer 2 mechanisms to work around limitations in layer 3 is not elegant.

#### 4. DHCPv6 Based Solution

A DHCPv6 based solution allows an operator an on demand and node specific means of configuring static routing information. Such a solution also fits into network environments where the operator prefers to manage Residential Gateway (RG) configuration information from a centralized DHCP server.

[I-D.ietf-v6ops-ipv6-multihoming-without-ipv6nat] provides additional background to the need for a DHCPv6 solution to the problem.

In terms of the high level operation of the solution defined in this draft, a DHCPv6 client interested in obtaining routing information request the route options using the DHCPv6 Option Request Option (ORO) sent to a server. A Server, when configured to do so, provides the requested route information as part of a nested options structure covering; the next-hop address; the destination prefix; the route metric; any additional options applicable to the destination or next-hop.

##### 4.1. Default route configuration

Defined mechanism may be used to configure default route. Default route is configured using RT\_PREFIX option that specifies ::/0 route, included as suboption in NEXT\_HOP.

Server MUST NOT define more than one default route.

##### 4.2. Configuring on-link routes

Server may also configure on-link routes, i.e. routes that are available directly over the link, not via routers. To specify on-link routes, server MAY include RTPREFIX option directly in Advertise and Reply messages.

##### 4.3. Deleting obsolete route

There are two mechanisms that allow removing a route. Each defined route has a route lifetime. If specific route is not refreshed and its timer reaches 0, client MUST remove corresponding entry from routing table.

In cases, where faster route removal is needed, server SHOULD return RT\_PREFIX option with route lifetime set to 0. Client that receives RT\_PREFIX with route lifetime set to 0 MUST remove specified route immediately, even if its previous lifetime did not expire yet.

#### 4.4. Applicability to routers

Contrary to Router Advertisement mechanism, defined in [RFC4861] that explicitly limits configuration to hosts, routing configuration over DHCPv6 defined in this document may be used by both hosts and routers. (This limitation of RA mechanism was partially lifted by W-1 requirement formulated in [RFC6204].)

One of the envisaged usages for this solution are residential gateways (RG) or Customer Premises Equipment (CPE). Those devices very often perform routing. It may be useful to configure routing on such devices over DHCPv6. One example of such use may be a class of premium users that are allowed to use dedicated router that is not available to regular users.

#### 4.5. Updating Routing Information

Network configuration occasionally changes, due to failure of existing hardware, migration to newer equipment or many other reasons. Therefore there a way to inform clients that routing information have changed is required.

There are several ways to inform clients about new routing information. Every client SHOULD periodically refresh its configuration, according to Information Refresh Time Option, so server may send updated information the next time client refreshes its information. New routes may be configured at that time. As every route has associated lifetime, client is required to remove its routes when this timer expires. This method is particularly useful, when migrating to new router is undergoing, but old router is still available.

Server MAY also announce routes via soon to be removed router with lifetimes set to 0. This will cause the client to remove its routes, despite the fact that previously received lifetime may not yet expire.

Aforementioned methods are useful, when there is no urgent need to update routing information. Bound by timer set by value of Information Refresh Time Option, clients may use outdated routing information until next scheduled renewal. Depending on configured value this delay may be not acceptable in some cases. In such scenarios, administrators are advised to use RECONFIGURE mechanism, defined in [RFC3315]. Server transmits RECONFIGURE message to each client, thus forcing it to immediately start renewal process.

See also Section 4.6 about limitations regarding dynamic routing.

#### 4.6. Limitations

Defined mechanism is not intended to be used as a dynamic routing protocol. It should be noted that proposed mechanism cannot automatically detect routing changes. In networks that use dynamic routing and also employ this mechanism, clients may attempt using routes configured over DHCPv6 even though routers or specific routes ceased to be available. This may cause black hole routing problem. Therefore it is not recommended to use this mechanism in networks that use dynamic routing protocols. This mechanism SHOULD NOT be used in such networks, unless network operator can provide a way to update DHCP server information in case of router availability changes.

Discussion: It should be noted that DHCPv6 server is not able to monitor health of existing routers. As there are currently more than 60 options defined for DHCPv6, it is infeasible to implement mechanism that would monitor huge set of services and stop announcing its availability in case of service outage. Therefore in case of prolonged unavailability human intervention is required to change DHCPv6 server configuration. If that is considered a problem, network administrators should consider using other alternatives, like RA and ND mechanisms (see [RFC4861]).

User is also encouraged to read Section 3.2.

#### 5. DHCPv6 Route Options

A DHCPv6 client interested in obtaining routing information includes the NEXT\_HOP and RT\_PREFIX options as part of its Option Request Option (ORO) in messages directed to a server (as allowed by [RFC3315], i.e. Solicit, Request, Renew, Rebind or Information-request messages). A Server, when configured to do so, provides the requested route information using zero, one or more NEXT\_HOP options in messages sent in response (Advertise, and Reply). So as to allow the route options to be both extensible, as well as conveying detailed info for routes, use is made of a nested options structure. Server sends one or more NEXT\_HOP options that specify the IPv6 next hop addresses. Each NEXT\_HOP option conveys in turn zero, one or more RT\_PREFIX options that represents the IPv6 destination prefixes reachable via the given next hop. Server includes RT\_PREFIX directly in message to indicate that given prefix is available directly on-link. Server MAY send a single NEXT\_HOP without any RT\_PREFIX suboptions or with RT\_PREFIX that contains ::/0 to indicate available default route. The Formats of the NEXT\_HOP and RT\_PREFIX options are defined in the following sub-sections.

The DHCPv6 Route Options format borrows from the principles of the Route Information Option defined in [RFC4191].

### 5.1. Next Hop Option Format

Each IPv6 route consists of an IPv6 next hop address, an IPv6 destination prefix (a.k.a. the destination subnet), and a host preference value for the route. Elements of such route (e.g. Next hops and prefixes associated with them) are conveyed in NEXT\_HOP option that contains RT\_PREFIX suboptions.

The Next Hop Option defines the IPv6 address of the next hop, usually corresponding to a specific next-hop router. For each next hop address there can be zero, one or more prefixes reachable via that next hop.

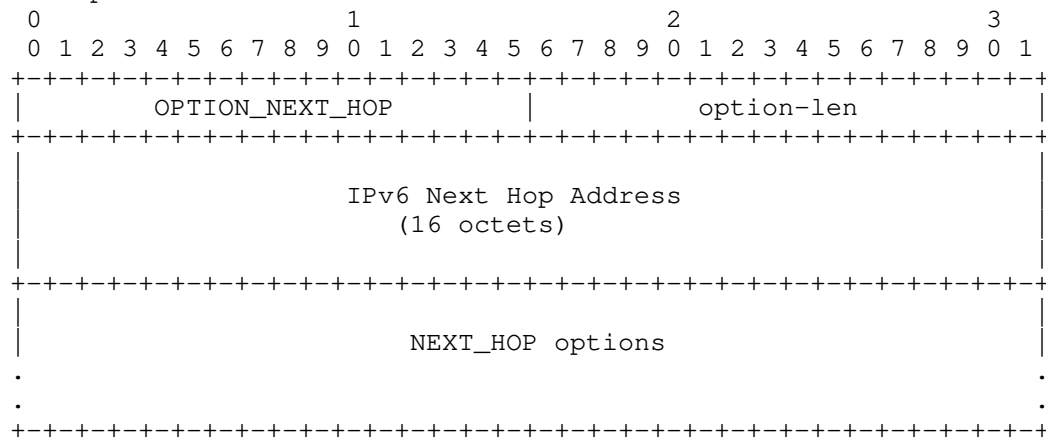


Figure 2: IPv6 Next Hop Option Format

option-code: OPTION\_NEXT\_HOP (TBD1).

option-len: 16 + Length of NEXT\_HOP options field.

IPv6 Next Hop Address: 16 octet long field that specified IPv6 address of the next hop.

NEXT\_HOP options: Options associated with this Next Hop. This includes, but is not limited to, zero, one or more RT\_PREFIX options that specify prefixes reachable through the given next hop.



## 5.2. Route Prefix Option Format

The Route Prefix Option is used to convey information about a single prefix that represents the destination network. The Route Prefix Option is used as a sub-option in the previously defined Next Hop Option. It may also be sent directly in message to indicate that route is available directly on-link.

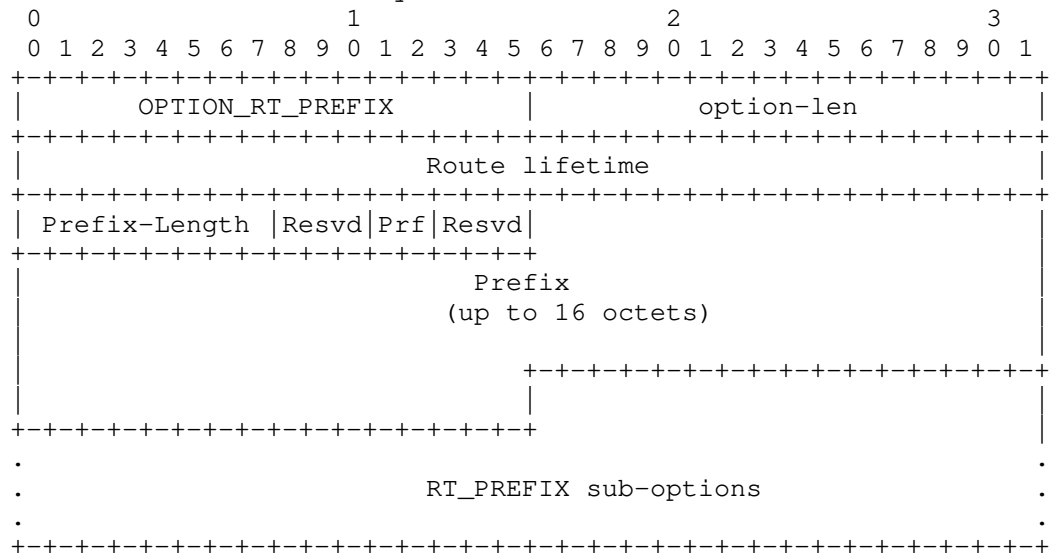


Figure 3: Route Prefix Option Format

option-code: OPTION\_RT\_PREFIX (TBD2).

option-len: Length of the Route Prefix option including all its sub-options.

Route lifetime 32-bit unsigned integer. Specifies lifetime of the route information, expressed in seconds (relative to the time the packet is sent). There are 2 special values defined. 0 means that route is no longer valid and must be removed by clients. A value of all one bits (0xffffffff) represents infinity.

Prefix Length: 8-bit unsigned integer. The length in bits of the IP Prefix. The value ranges from 0 to 128. This field represents the number of valid leading bits in the prefix.

- Resvd: Reserved field. Server MUST set this value to zero and client MUST ignore its content.
- Prf(Route Preference): 2-bit signed integer. The Route Preference indicates whether to prefer the router associated with this prefix over others, when multiple identical prefixes (for different routers) have been received. If the Reserved (10) value is received, the Route Information Option MUST be ignored.
- Metric: Route Metric. 8-bit signed integer. The Route Metric indicates whether to prefer the next hop associated with this prefix over others, when multiple identical prefixes (for different next hops) have been received.
- Prefix: a variable size field that specifies Rule IPv6 prefix. Length of the field is defined by prefix6-len field and is rounded up to the nearest octet boundary (if case when prefix6-len is not divisible by 8). In such case additional padding bits must be zeroed.

RT\_PREFIX options: Options specific to this particular prefix.

Values for preference field have meaning identical to Route Information Option, defined in [RFC4191], Section 2.1:

01 High

00 Medium (default)

11 Low

10 Reserved - MUST NOT be sent

## 6. DHCPv6 Server Behavior

When configured to do so, a DHCPv6 server shall provide the Next Hop and Route Prefix Options in ADVERTISE and REPLY messages sent to a client that requested the route option. Each Next Hop Option sent by the server must convey at least one Route Prefix Option.

Server includes NEXT\_HOP option with possible RT\_PREFIX suboptions to designate that specific routes are available via routers. Server includes RT\_PREFIX options directly in Advertise and Reply messages to inform that specific routes are available directly on-link.

If there is more than one route available via specific next hop,

server MUST send only one NEXT\_HOP for that next hop, which contains multiple RT\_PREFIX options. Server MUST NOT send more than one identical (i.e. with equal next hop address field) NEXT\_HOP option.

Servers SHOULD NOT send Route Option to clients that did not explicitly requested it, using the ORO.

Servers MUST NOT send Route Option in messages other than ADVERTISE or REPLY.

Servers MAY also include Status Code Option, defined in Section 22.13 of the [RFC3315] to indicate the status of the operation.

Servers MUST include the Status Code Option, if the requested routing configuration was not successful and SHOULD use status codes as defined in [RFC3315] and [RFC3633].

The maximum number of routing information in one DHCPv6 message depend on the maximum DHCPv6 message size defined in [RFC3315]

## 7. DHCPv6 Client Behavior

A DHCPv6 client compliant with this specification MUST request the NEXT\_HOP and RT\_PREFIX Options in an Option Request Option (ORO) in the following messages: Solicit, Request, Renew, Rebind, and Information-Request. The messages are to be sent as and when specified by [RFC3315].

When processing a received Route Options a client MUST substitute a received 0::0 value in the Next Hop Option with the source IPv6 address of the received DHCPv6 message. It MUST also associate a received Link Local next hop addresses with the interface on which the client received the DHCPv6 message containing the route option. Such a substitution and/or association is useful in cases where the DHCPv6 server operator does not directly know the IPv6 next-hop address, other than knowing it is that of a DHCPv6 relay agent on the client LAN segment. DHCPv6 Packets relayed to the client are sourced by the relay using this relay's IPv6 address, which could be a link local address.

The Client SHOULD refresh assigned route information periodically. The generic DHCPv6 Information Refresh Time Option, as specified in [RFC4242], can be used when it is desired for the client to periodically refresh of route information.

The routes conveyed by the Route Option should be considered as complimentary to any other static route learning and maintenance

mechanism used by, or on the client with one modification: The client MUST flush DHCPv6 installed routes following a link flap event on the DHCPv6 client interface over which the routes were installed. This requirement is necessary to automate the flushing of routes for clients that may move to a different network.

Client MUST confirm that routers announced over DHCPv6 are reachable, using one of methods suitable for specific network type. The most common mechanism is Neighbor Unreachability Detection (NUD), specified in [RFC4861]. Client SHOULD use NUD to verify that received routers are reachable before adjusting its routing tables. Client MAY use other reachability verification mechanisms specific to used network technology. To avoid potential long-lived routing black holes, client MAY periodically confirm that router is still reachable.

#### 7.1. Conflict resolution

Information received via Route Options over DHCPv6 MUST be treated equally to routing information obtained via other sources. In particular, from the RA perspective, DHCPv6 provisioning should be treated as if yet another RA was received. Preference field should be taken into consideration during route information processing. In particular, administrators are encouraged to read [RFC4191], Section 4.1 for guidance.

To facilitate information merge between DHCPv6 and RA, DHCPv6 option conveys the same information as RIO, specified in [RFC4191], albeit on-wire format is slightly different. The differences are:

Metric field (available in previous version of this draft) has been replaced with 2-bit preference field that is in line with RIO information.

RIO uses 128-length prefix field, while DHCPv6 option uses variable prefix length. That difference is used to minimize packet size as it avoid transmitting zeroed octets. Despite slightly different encoding, delivered information is exactly the same.

If prefix is available directly on-link, Route Prefix option is conveyed directly in DHCPv6 message, not withing Next Hop option. That feature is considered a superset, compared to RIO.

#### 8. IANA Considerations

IANA is kindly requested to allocate DHCPv6 option code TBD1 to the OPTION\_NEXT\_HOP and TBD2 to OPTION\_RT\_PREFIX. Both values should be

added to the DHCPv6 option code space defined in Section 24.3 of [RFC3315].

## 9. Security Considerations

The overall security considerations discussed in [RFC3315] apply also to this document. The Route option could be used by malicious parties to misdirect traffic sent by the client either as part of a denial of service or man-in-the-middle attack. An alternative denial of service attack could also be realized by means of using the route option to overflowing any known memory limitations of the client, or to exceed the client's ability to handle the number of next hop addresses.

Neither of the above considerations are new and specific to the proposed route option. The mechanisms identified for securing DHCPv6 as well as reasonable checks performed by client implementations are deemed sufficient in addressing these problems.

It is essential that clients verify that announced routers are indeed reachable, as specified in Section 7. Failing to do so may create black hole routing problem.

This mechanism may introduce severe problems if deployed in networks that use dynamic routing protocols. See Section 4.6 for details.

DHCPv6 becomes a complete provisioning protocol with this mechanism, i.e. all necessary configuration parameters may be delivered using DHCPv6 only. It was suggested that in some cases this may lead to decision of disabling RA. While RA-less networks could offer lower operational expenses and protection against rogue RAs, they would not work with nodes that do not support this feature. Therefore such decision is not recommended, unless all effects are carefully analyzed. It is worth noting that disabling RA support in hosts would solve rogue RA problem, it would in fact only change the issue into rogue DHCPv6 problem. That is somewhat beneficial, however, as rogue RA may affect all nodes immediately while rogue DHCPv6 server will affect only new nodes, that boot up after rogue server manifests itself.

Reader is also encouraged to read DHCPv6 security considerations document [I-D.ietf-dhc-secure-dhcpv6].

## 10. Contributors and Acknowledgements

This document would not have been possible without the significant

contribution provided by: Arifumi Matsumoto, Hui Deng, Richard Johnson, and Zhen Cao.

The authors would also like to thank Alfred Hines, Ralph Droms, Ted Lemon, Ole Troan, Dave Oran, Dave Ward, Joel Halpern, Marcin Siodelski, Alexandru Petrescu, Roberta Maglione, Tim Chown, Brian Carpenter, Dave Thaler, Lorenzo Colitti and Leo Bicknell for their comments and useful suggestions.

This work has been partially supported by Department of Computer Communications (a division of Gdansk University of Technology) and the Polish Ministry of Science and Higher Education under the European Regional Development Fund, Grant No. POIG.01.01.02-00-045/09-00 (Future Internet Engineering Project).

## 11. References

### 11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.

### 11.2. Informative References

- [BBF-WT-124]  
Broadband Forum, "BBF WT-124 issue 3", BBF WT-124i3, 2011.
- [I-D.ietf-6man-addr-select-opt]  
Matsumoto, A., Fujisaki, T., Kato, J., and T. Chown,  
"Distributing Address Selection Policy using DHCPv6",  
draft-ietf-6man-addr-select-opt-03 (work in progress),  
February 2012.
- [I-D.ietf-dhc-secure-dhcpv6]  
Jiang, S. and S. Shen, "Secure DHCPv6 Using CGAs",  
draft-ietf-dhc-secure-dhcpv6-04 (work in progress),  
December 2011.
- [I-D.ietf-mif-dns-server-selection]

Savolainen, T., Kato, J., and T. Lemon, "Improved DNS Server Selection for Multi-Interfaced Nodes", draft-ietf-mif-dns-server-selection-07 (work in progress), October 2011.

[I-D.ietf-v6ops-ipv6-multihoming-without-ipv6nat]

Troan, O., Miles, D., Matsushima, S., Okimoto, T., and D. Wing, "IPv6 Multihoming without Network Address Translation", draft-ietf-v6ops-ipv6-multihoming-without-ipv6nat-04 (work in progress), February 2012.

[RFC3442] Lemon, T., Cheshire, S., and B. Volz, "The Classless Static Route Option for Dynamic Host Configuration Protocol (DHCP) version 4", RFC 3442, December 2002.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

[RFC4242] Venaas, S., Chown, T., and B. Volz, "Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 4242, November 2005.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC6204] Singh, H., Beebe, W., Donley, C., Stark, B., and O. Troan, "Basic Requirements for IPv6 Customer Edge Routers", RFC 6204, April 2011.

[THREEGPP-23.853]

Stojanovski, S., "3GPP TR 23.853: Operator Policies for IP Interface Selection (OPIIS)", 3GPP TR 23.853, August 2011, <<http://www.3gpp.org/ftp/Specs/html-info/23853.htm>>.

[nanog-beijnum]

van Beijnum, I., "", , June 2011, <<http://mailman.nanog.org/pipermail/nanog/2011-June/037242.html>>.

## Authors' Addresses

Wojciech Dec  
Cisco Systems  
Haarlerbergweg 13-19  
1101 CH Amsterdam  
The Netherlands

Email: wdec@cisco.com

Tomasz Mrugalski (editor)  
Internet Systems Consortium, Inc.  
950 Charter Street  
Redwood City, CA 94063  
USA

Phone: +1 650 423 1345  
Email: tomasz.mrugalski@gmail.com

Tao Sun  
China Mobile  
Unit2, 28 Xuanwumenxi Ave  
Beijing, Xuanwu District 100053  
China

Phone:  
Email: suntao@chinamobile.com

Behcet Sarikaya  
Huawei USA  
1700 Alma Dr. Suite 500  
Plano, TX 75075  
United States

Phone: +1 972-509-5599  
Fax:  
Email: sarikaya@ieee.org  
URI:





Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: May 17, 2017

G. Chen  
China Mobile  
C. Williams  
Consultant  
D. Wing  
A. Yourtchenko  
Cisco Systems, Inc.  
November 13, 2016

Happy Eyeballs Extension for Multiple Interfaces  
draft-ietf-mif-happy-eyeballs-extension-11

Abstract

This memo proposes extensions to the Happy Eyeball's algorithm requirements defined in RFC6555 for use with the multiple provisioning domain architecture. The Happy Eyeballs in MIF would make the selection process smoother by using connectivity tests over pre-filtered interfaces according to defined policy. This would choose the best interface with an automatic fallback mechanism.

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 May 17, 2017.

Copyright Notice

Copyright (c) 2016 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

1. Introduction . . . . .	3
2. Terminology . . . . .	3
3. Use Cases . . . . .	3
3.1. WiFi is broken . . . . .	3
3.2. Policy Conflict . . . . .	4
4. Happiness Parameters . . . . .	4
4.1. Hard Set . . . . .	5
4.1.1. Operator Policy . . . . .	5
4.1.2. User Preference . . . . .	5
4.2. Soft Set . . . . .	6
4.2.1. Provisioning Domain Identity . . . . .	6
4.2.2. DNS Selection . . . . .	6
4.2.3. Next Hop . . . . .	6
4.2.4. Source Address Selection . . . . .	6
4.2.5. Common Practice . . . . .	6
5. HE-MIF Process Requirements . . . . .	7
5.1. First Step, Filter . . . . .	7
5.2. Second Step, Sort . . . . .	8
5.2.1. Interface Validation . . . . .	8
5.2.2. Name Resolution . . . . .	8
5.2.3. Connection Establishment . . . . .	8
6. Implementation Framework . . . . .	9
7. Additional Considerations . . . . .	9
7.1. Usage Scope . . . . .	9
7.2. Fallback Timeout . . . . .	9
7.3. DNS Selections . . . . .	10
7.4. Flow Continuity . . . . .	11
7.5. Interworking with Happy Eyeball . . . . .	11
7.6. Multipath Applicability . . . . .	11
8. IANA Considerations . . . . .	11
9. Security Considerations . . . . .	12
10. Acknowledgements . . . . .	12
11. References . . . . .	12
11.1. Normative References . . . . .	12
11.2. Informative References . . . . .	13
Authors' Addresses . . . . .	14

## 1. Introduction

The MIF problem statement [RFC6418] describes problems specific for nodes attached to multiple provisioning domains. Specifically, there is a issue description that a node has selected an interface and obtained a valid IP address from the network, but Internet connectivity is not available. This memo intends to address the issue and elaborate more in Section 3.1.

[RFC7556] describes the multiple provisioning domain architecture. It refers to using connectivity tests to validate a Provisioning Domain (PvD). Given a number of implicit/explicit PvDs, plus preferences/policy, what is the process to follow to select the best PvD to use for any given connection. In the event that two or more are deemed to be best, how are the Happy Eyeballs (HE) techniques applied to find the best and deal with resilience. This memo also proposes process requirements using Happy Eyeballs (HE) extensions.

There are a variety of algorithms that can be envisioned. This document describes additional parameters and processes that need to be considered in addition to the HE algorithm requirements defined in [RFC6555] necessary to support multiple interfaces, so that a node with multiple interfaces can select the best path for a particular connection-oriented flow (e.g., TCP, SCTP).

## 2. Terminology

This document makes use of following terms:

- o Happy Eyeballs (HE): specifies requirements for an algorithm that reduces the user-visible connection delay for dual-stack hosts with a single interface per-protocol.
- o Happy Eyeballs - Multi-Interface (HE-MIF): Extends the Happy Eyeballs concept to the multiple provisioning domain architecture. It describes additional requirements for algorithms that offer connectivity tests on PVD-aware or non-PVD-aware nodes [RFC7556] to select the best interface for a specific connection request.

## 3. Use Cases

The section describes scenarios the HE-MIF targeted to use.

### 3.1. WiFi is broken

Assuming a MIF node has both a 3GPP mobile network interface and a WiFi interface, a common practice would be to always prefer the WiFi connection when the node enters an area with WiFi available. In this

situation, a node might assume that because a valid IP address has been allocated, the WiFi link provides connectivity to destinations through the Internet. However, this might not be the case for several reasons:

- o WiFi access-point authentication requirements
- o WiFi has no global Internet connectivity
- o Instability at layer 2

In order to resolve this problem, the user would need to disable the device's interface preferences, e.g. by disabling the WiFi interface. HE-MIF offers users the possibility of configuring their preferences for the choice of the most suitable network interface to use, such as via setting on their mobile phone.

In this case, users may prefer to wait an appropriate time period for connections to be established over a WiFi path. If no connection can be made it will fall back to attempting the connection over a 3GPP mobile network path.

### 3.2. Policy Conflict

A node has network access via both WiFi and 3GPP networks. In a mobile network, IPv6-only may be preferable since IPv6 has the potential to be simpler than dual-stack. The WiFi access offers IPv4 only. In this scenario, the combination of source address selection [RFC6724] and preferring the WiFi interface may cause a problem. The transition to IPv6 may mean that IPv6 is the preferred protocol, so the 3GPP interface should be chosen even though it could be considered a suboptimal selection e.g. the WiFi interface likely is less expensive.

## 4. Happiness Parameters

This section provides input parameter proposal that HE-MIF should catch. Two sets of "Happiness" parameters have been defined. It serves applications and initiates HE-MIF connection tests subsequently. By following the process described below, MIF nodes can select an appropriate interface that best meets the configuration parameters defined by the user. The two sets of "Happiness" parameters are called Hard Set and Soft Set respectively.

#### 4.1. Hard Set

Hard set contains parameters which should be complied with. It helps to select candidate interfaces through which a particular flow should be directed. These should be seen as constraints on the choice, such as provider policies, support for IPv4 or IPv6, and other parameters which would prevent a particular interface and transport from being used by a particular flow. Parameters in the hard set should be easy to use and understand. When several parameters in the hard set are in conflict, the user's preference should be prioritized.

##### 4.1.1. Operator Policy

Operators may deliver the customized policies for a particular network environment because of geo-location or service regulation considerations. One example relevant for 3GPP networks is an operator delivering policies from an Access Network Discovery and Selection function (ANDSF) [TS23.402].

The ANDSF provides a node with policies and network selection information to influence the selection between different access technologies, such as 3GPP mobile networks, WiFi access. The ANDSF can provide the node with three types of information[TS24.302].

- o Access network discovery and selection information: it includes a list of access networks available in the vicinity of the node. The information may include the access technology types (e.g. WiFi), network identifiers (e.g. SSID in the case of WiFi) as well as validity conditions (e.g. where and when).
- o Inter-System Mobility Policies (ISMPs): they are a set of operator-defined rules and preferences that affect the inter-system mobility decisions, e.g. decisions about whether to use 3GPP mobile network or a WiFi network.
- o Inter-System Routing Policies (ISRPs): the node uses ISRPs when it can route IP traffic simultaneously over multiple radio access networks. It could provide routing policies in an IP flow granularity.

##### 4.1.2. User Preference

User's preference: users may express preferences which likely not have a formally technical language, like "No 3/4G while roaming", "Only download applications larger than 20Mb over WiFi", etc. Those information are normally input from User Interface (UI).

#### 4.2. Soft Set

Soft set contains factors which impact the selection of the path across which a particular flow should be transmitted among the available interfaces and transports which meet the hard set requirements described above.

##### 4.2.1. Provisioning Domain Identity

A PVD-aware node uses PVD Identity (PvD-ID) to select a PvD with a matching ID for special-purpose connection requests. The PvD-ID may be generated by the node implicitly or received from the network explicitly. For explicit PvDs, the node could take the parameter from PvD ID Option [I-D.ietf-mif-mpvd-id] via the configuration protocols ([I-D.ietf-mif-mpvd-dhcp-support] or [I-D.ietf-mif-mpvd-ndp-support]). A PVD-aware node may decide to use one preferred PVD or allow the use of multiple PVDs simultaneously for applications. The node behavior should be consistent with MPVD architecture [RFC7556].

##### 4.2.2. DNS Selection

At the name service lookup step, the node has to choose a recursive DNS server to use. A HE-MIF node should take the parameter of RDNSS Selection DHCP Option [RFC6731] to select an interface for a particular namespace.

##### 4.2.3. Next Hop

[RFC4191] allows the configuration of specific routes to a destination. A HE-MIF node should take the parameters of router preference and route information to identify the next hop.

##### 4.2.4. Source Address Selection

For each destination, once the best next hop is found, the node should consider IP prefix and precedence parameter in policy table to select the best source address according to the rule defined in [RFC6724].

##### 4.2.5. Common Practice

There is relevant common practice related to interface selection, e.g. Prefer WiFi over a 3GPP interface, if available. Such conventions should also be considered.

## 5. HE-MIF Process Requirements

An HE-MIF node may use the two sets of parameters as two steps in the interface selection process. The first step is to use the Hard Set to synthesize policies from different actors (e.g., users or network operators). These hard set parameters will provide a filter which will exclude not qualifying interfaces from any further consideration.

The second step is to influence how a node makes a connection when multiple interfaces still remain in the candidate list after first step. This is essentially sorting behavior. In the multiple provisioning domain architecture, a PVD aware node makes connectivity tests as described in Section 5.3 of [RFC7556]. A PVD agnostic node take other parameters apart from PVD-ID in the Soft Set to proceed the sort process.

The two steps are described in more details in the following sub-sections. It should be noted that HE-MIF does not prescribe such two-step model. It will be very specific to particular cases and implementations. The two step model mainly describes requirements for how to use the hard/soft set.

### 5.1. First Step, Filter

One goal of the filter is to reconcile multiple selection policies from users or operators. Afterwards, merged demands would be mapped to a set of candidate interfaces, which are judged as qualified.

Decision on the reconciliation of different policies will depend very much on the deployment scenario. An implementation may not be able to determine priority for each policies without explicit configuration provided by users or administrator. For example, an implementation may by default always prefer the WiFi because of cost saving consideration. Whereas, other users may turn off a device's WiFi interface to guarantee use of a 3GPP network interface to assure higher reliability or security.

The decision on mergence of policies may be made by implementations, or by node administrators. However, it's worth to note that a demand from users should be normally considered higher priority than from other actors.

The merged policies serve as a filter which is iterated across the list of available interfaces. Qualified interfaces are selected and the proceed to the second step.



## 5.2. Second Step, Sort

### 5.2.1. Interface Validation

The Sort process aims to select the best interface and provide fallback capacities. As stated in [RFC7556], a PVD-aware node shall perform connectivity tests and, only after validation of the PVD, consider using it to serve application connections requests. In current implementations, some nodes already implement this, e.g., by trying to reach a dedicated web server (see Section 3.1.2 [RFC6419]). If anything is abnormal, it assumes there is a proxy on the path. This status detection is recommended to be used in HE-MIF to detect DNS interception or an HTTP proxy that forces a login or a click-through. Unexamined PVDs or interfaces should be accounted as "unconnected". It should not join the sort process.

### 5.2.2. Name Resolution

Name resolution is executed on the validated interfaces. Before the requests are initiated, it should check if there is a matching PVD ID for the destination name. A PVD agnostic node may request DNS server selection DHCP option [RFC6731] for interface selection guidance. Those information may weight a particular interface to be preferred to others sending resolving requests. If the node can't find useful information in the Soft Set, DNS queries would be sent out on multiple interfaces in parallel to maximize chances for connectivity. Some additional discussions of DNS selection consideration of HE-MIF are described in Section 7.3.

### 5.2.3. Connection Establishment

Once a destination address was resolved, a connection is to be setup. For the given destination address, a PVD-aware node selects a next-hop and source address associated with that PVD in the name resolution process. A PVD agnostic node may receive certain next hop in a RA message [RFC4191], the node selects best source address according to the rules [RFC6724].

The interface identified by the source address should be treated to initiate the connection prior to others. This could avoid thrashing the network, by not making simultaneous connection attempts on multiple interfaces. After making a connection attempt on the preferred pairs and failing to establish a connection within a certain time period (see Section 7.2), a HE-MIF implementation will decide to initiate connection attempt using rest of interfaces in parallel. This fallback consideration will make subsequent connection attempts successful on non-preferable interfaces.

The node would cache information regarding the outcome of each connection attempt. Cache entries would be flushed periodically. A system-defined timeout may take place to age the state. Maximum on the order of 10 minutes defined in [RFC6555] is recommended to keep the interface state changes synchronizing with IP family states.

If there is no specific Soft Set provided, all selected interfaces should be treated equally. For a node implementing multipath transports (for example, Multipath TCP (MPTCP) [RFC6182]), the interfaces could be treated as valid to perform subsequent multipath process, such as starting subflow. A node only supporting single physical transport would initiate on several interface simultaneously. The goal here is to provide the most fast connection for users, by quickly attempting to connect using each candidate interface. Afterwards, the node would do the same caching and flushing process as described above.

## 6. Implementation Framework

The simplest way to implement the processes described in this document is within the application itself. This would not require any specific support from the operating system beyond the commonly available APIs that provide transport service. It could also be implemented using a high-level API approach, linking to the MIF-API [I-D.ietf-mif-api-extension].

## 7. Additional Considerations

### 7.1. Usage Scope

Connection-oriented transports (e.g., TCP, SCTP) are directly applied as scoped in [RFC6555]. For connectionless transport protocols (e.g., UDP), a similar mechanism can be used if the application has request/response semantics. Further investigations are out of the document scope.

### 7.2. Fallback Timeout

When the preferred interface was failed, HE-MIF would trigger a fallback process to start connection initiation on several candidate interfaces. A period of time should be set to invalidate the interface and fallback to others. Aggressive timeouts may achieve quick interface handover, but at the cost of traffic that may be chargeable on certain networks, e.g. the handover from WiFi to 3GPP networks brings a charge to customers. Considering the reasons, it is recommended to prioritize the input from users (e.g., real customers or applications) through user interface. For default-setting on a system, a hard error [RFC1122] in replied ICMP could

serve as a trigger for the fallback process. When the ICMP soft error is present or non-response was received, it's recommended that the timeout should be large enough to allow connection retransmission. [RFC1122] states that such timer must be at least 3 minutes to provide TCP retransmission. However, several minutes delay may not be inappropriate for user experiences. A widespread practice [RFC5461] sets 75 seconds to optimize connection process.

More optimal timer may be expected. The particular setting will be very specific to implementations and cases. The memo didn't try to provide a concrete value because of following concerns.

- o RTT (Round-Trip Time) on different interfaces may vary quite a lot. A particular value of timeout may not accurately help to make a decision that this interface doesn't work at all. On the contrary, it may cause a misjudgment on a interface, which is not very fast. In order to compensate the issues, the timeout setting based on past experiences of a particular interface may help to make a fair decision. Whereas, it's going beyond the capability of Happy Eyeballs [RFC6555]. Therefore, it leaves a particular implementation.
- o In some cases, fast interface may not be treated as "best". For example, a interface could be evaluated in the principle of bandwidth-delay, termed "Bandwidth-Delay-Product ". Happy Eyeballs measures only connection speed. That is, how quickly a TCP connection is established . It does not measure bandwidth. If the fallback has to take various factors into account and make balanced decision, it's better to resort to a specific context and implementation.

### 7.3. DNS Selections

During the Sort process, HE-MIF prioritizes PVD-ID match or [RFC6731] inputs to select a proper server. It could help to address following two cases.

- o A DNS answer may be only valid for a specific provisioning domain, but the DNS resolver may not be aware of that because the DNS reply is not kept with the provisioning from which the answer comes. The situation may become worse if asking internal name with public address response or asking public name with private address answers.
- o Some FQDNs can be resolvable only by sending queries to the right server (e.g., intranet services). Otherwise, a response with NXDOMAIN is replied. Fast response is treated as optimal only if

the record is valid. That may cause messy for data connections, since NXDOMAIN doesn't provide useful information.

HE-MIF can help to solve the issues of DNS interception with captive portal. The DNS server modified and replied the answer with the IP address of captive portal rather than the intended destination address. In those cases, TCP connection may succeed, but Internet connectivity is not available. It results in lack of service unless user has authenticated. HE-MIF recommended using network connectivity status probes to examine a pre-configured URL for detecting DNS interception on the path (see more in Section 5.2). The node will be able to automatically rely upon other interfaces to select right DNS servers by excluding the unexamined interfaces.

#### 7.4. Flow Continuity

[I-D.deng-mif-api-session-continuity-guide] describes session continuity guidance for application developers. The flow continuity topic is beyond this document scope.

#### 7.5. Interworking with Happy Eyeball

HE-MIF process could cooperate with HE [RFC6555]. HE is executed on an interface which is selected to make connection establishment (see Section 5.2.3). for example, a node following PvD policy to pick a interface and make both IPv4/IPv6 connection attempts in consistent with HE requirements. The interface state management in HE-MIF is designed to synchronize with IP family states. It could facilitate the HE executions.

#### 7.6. Multipath Applicability

Some nodes may support transports that provide an abstraction of a single connection, aggregating multiple underlying connections. Multipath TCP (MPTCP) [RFC6182] is an example of such a transport protocol. For connections provided by such transports, a node may leverage the "happiness" parameters and process on the underlying connections. Following the HE-MIF requirements, each connection could be performed consistently with user/operator's preference and corresponding provisioning domain information.

### 8. IANA Considerations

This memo does not include any IANA requests.

## 9. Security Considerations

The security consideration is following the statement in [RFC6555] and [RFC6418].

## 10. Acknowledgements

The authors would like to thank Margaret Wasserman, Hui Deng, Erik Kline, Stuart Cheshire, Teemu Savolainen, Jonne Soininen, Simon Perreault, Zhen Cao, Dmitry Anipko, Ted Lemon, Daniel Migault, Russ White and Bing Liu for their helpful comments.

Many thanks to Ralph Droms, Ian Farrer, Jouni Korhonen, Mirja Khlewind and Suresh Krishnan for their detailed reviews.

## 11. References

### 11.1. Normative References

- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<http://www.rfc-editor.org/info/rfc1122>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<http://www.rfc-editor.org/info/rfc4191>>.
- [RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <<http://www.rfc-editor.org/info/rfc6555>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<http://www.rfc-editor.org/info/rfc6724>>.
- [RFC6731] Savolainen, T., Kato, J., and T. Lemon, "Improved Recursive DNS Server Selection for Multi-Interfaced Nodes", RFC 6731, DOI 10.17487/RFC6731, December 2012, <<http://www.rfc-editor.org/info/rfc6731>>.
- [TS23.402] 3rd Generation Partnership Project, 3GPP., "Architecture enhancements for non-3GPP accesses v8.8.0", December 2009.

[TS24.302]

3rd Generation Partnership Project, 3GPP., "Access to the 3GPP Evolved Packet Core (EPC) via non-3GPP access networks v14.0.0", June 2016.

## 11.2. Informative References

[I-D.deng-mif-api-session-continuity-guide]

Deng, H., Krishnan, S., Lemon, T., and M. Wasserman, "Guide for application developers on session continuity by using MIF API", draft-deng-mif-api-session-continuity-guide-04 (work in progress), July 2014.

[I-D.ietf-mif-api-extension]

Liu, D., Lemon, T., Ismailov, Y., and Z. Cao, "MIF API consideration", draft-ietf-mif-api-extension-05 (work in progress), February 2014.

[I-D.ietf-mif-mpvd-dhcp-support]

Krishnan, S., Korhonen, J., and S. Bhandari, "Support for multiple provisioning domains in DHCPv6", draft-ietf-mif-mpvd-dhcp-support-02 (work in progress), October 2015.

[I-D.ietf-mif-mpvd-id]

Krishnan, S., Korhonen, J., Bhandari, S., and S. Gundavelli, "Identification of provisioning domains", draft-ietf-mif-mpvd-id-02 (work in progress), October 2015.

[I-D.ietf-mif-mpvd-ndp-support]

Korhonen, J., Krishnan, S., and S. Gundavelli, "Support for multiple provisioning domains in IPv6 Neighbor Discovery Protocol", draft-ietf-mif-mpvd-ndp-support-03 (work in progress), February 2016.

[RFC5461] Gont, F., "TCP's Reaction to Soft Errors", RFC 5461, DOI 10.17487/RFC5461, February 2009, <<http://www.rfc-editor.org/info/rfc5461>>.

[RFC6182] Ford, A., Raiciu, C., Handley, M., Barre, S., and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC 6182, DOI 10.17487/RFC6182, March 2011, <<http://www.rfc-editor.org/info/rfc6182>>.

[RFC6418] Blanchet, M. and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement", RFC 6418, DOI 10.17487/RFC6418, November 2011, <<http://www.rfc-editor.org/info/rfc6418>>.

[RFC6419] Wasserman, M. and P. Seite, "Current Practices for Multiple-Interface Hosts", RFC 6419, DOI 10.17487/RFC6419, November 2011, <<http://www.rfc-editor.org/info/rfc6419>>.

[RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<http://www.rfc-editor.org/info/rfc7556>>.

#### Authors' Addresses

Gang Chen  
China Mobile  
29, Jinrong Avenue  
Xicheng District,  
Beijing 100033  
China

Email: [phdgang@gmail.com](mailto:phdgang@gmail.com), [chengang@chinamobile.com](mailto:chengang@chinamobile.com)

Carl Williams  
Consultant  
El Camino Real  
Palo Alto, CA 94306  
USA

Email: [carlw@mcsr-labs.org](mailto:carlw@mcsr-labs.org)

Dan Wing  
Cisco Systems, Inc.  
170 West Tasman Drive  
San Jose, CA 95134  
USA

Email: [dwing@cisco.com](mailto:dwing@cisco.com)

Andrew Yourtchenko  
Cisco Systems, Inc.  
De Kleetlaan, 7  
Diegem B-1831  
Belgium

Email: [ayourtch@cisco.com](mailto:ayourtch@cisco.com)

MIF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: May 7, 2013

D. Migault  
Francetelecom - Orange  
C. Williams  
MCSR Labs  
November 3, 2012

IPsec Multiple Interfaces Problem Statement  
draft-mglt-mif-security-requirements-03.txt

Abstract

IKEv2 is the protocol used to set up and negotiate Security Associations between nodes. IKEv2 has not been designed for nodes with multiple interfaces.

This document is focused on IKEv2 ability to set up IPsec protected communications between nodes with multiple interfaces. This document states the problems and provides requirements for IKEv2 to ease IPsec for multiple interface communication.

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 May 7, 2013.

Copyright Notice

Copyright (c) 2012 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

1. Requirements notation . . . . .	4
2. Introduction . . . . .	4
3. Use Case 1: VPN with Multiple Interfaces . . . . .	5
3.1. Initial MIF IPsec Configuration . . . . .	6
3.1.1. Description . . . . .	6
3.1.2. Problem Statement . . . . .	6
3.1.3. Requirements . . . . .	8
3.2. Mobility . . . . .	8
3.2.1. Description . . . . .	8
3.2.2. Problem Statement . . . . .	9
3.2.3. Requirements . . . . .	11
3.3. Multihoming . . . . .	12
3.3.1. Description . . . . .	12
3.3.2. Problem Statement . . . . .	13
3.3.3. Requirements . . . . .	13
3.4. Adding an Interface . . . . .	14
3.4.1. Description . . . . .	14
3.4.2. Problem Statement . . . . .	17
3.4.3. Requirements . . . . .	18
3.5. Deleting an Interface . . . . .	18
3.5.1. Description . . . . .	18
3.5.2. Problem Statement . . . . .	18
3.5.3. Requirements . . . . .	18
4. Use Case 2: MIF applications and IPsec Tunnel mode . . . . .	19
5. Use Case 3: MIF aware applications with Transport mode . . . . .	20
5.1. Initial MIF IPsec Configuration . . . . .	21
5.1.1. Description . . . . .	21
5.1.2. Problem Statement . . . . .	21
5.1.3. Requirements . . . . .	21
5.2. Mobility . . . . .	22
5.2.1. Description . . . . .	22
5.2.2. Problem Statement . . . . .	23
5.2.3. Requirements . . . . .	24
5.3. Multihoming . . . . .	24
5.3.1. Description . . . . .	24
5.3.2. Problem Statement . . . . .	24
5.3.3. Requirements . . . . .	25
5.4. Adding an Interface . . . . .	25
5.5. Delete an Interface . . . . .	25
6. Security Considerations . . . . .	25

7. IANA Considerations . . . . .	25
8. Acknowledgment . . . . .	25
9. References . . . . .	26
9.1. Normative References . . . . .	26
9.2. Informational References . . . . .	26
Authors' Addresses . . . . .	26

## 1. Requirements notation

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].

## 2. Introduction

IPsec protocol suite [RFC4301],[RFC5996] is mainly used to:

- Extend a trusted domain over an untrusted network: This typically corresponds to the Virtual Private Network (VPN) use case. A Security Gateway is a trusted entry point to a trusted network. The end user is connected to an untrusted network and tunnels its traffic to the Security Gateway in a encrypted tunnel using the IPsec tunnel mode. The Security Gateway decapsulates the traffic and forwards it on the trusted network. Once the traffic is in the trusted network it is usually not encrypted anymore. In other words, the traffic is protected from the end user terminal to the Security Gateway, that it to say over the untrusted network.
- Provide end-to-end security: With end-to-end security, the traffic is protected from the source - or the end user in our case - to the destination. The traffic does not require to be tunneled, and any segments of the network between the end user and the destination is considered as untrusted. With end-to-end security, one does not require encapsulation, and the IPsec transport mode can be used.

Currently most devices have multiple interfaces. Mobile phones have most of the time a Wireless LAN (WLAN) and a Radio Access Network (RAN) interface. Laptop can easily have Ethernet / WLAN / RAN with WiMAX interfaces. Furthermore, USB dongle can be plugged to provide additional RAN and WLAN interfaces. Regular PCs, Servers, or CPEs have multiple Ethernet interfaces, with additional WLAN interfaces.

Protocols like SCTP [RFC4960] or MOBIKE [RFC4555] have been designed to use these multiple interfaces for multihoming. Only a single interface is used at a time. The interface used to carry the IP datagrams is called the Primary interface and other interfaces are called Secondary or Alternate interfaces. Alternate interfaces are only expected to be used in case the Primary interface fails.

However, multihoming does not enable the simultaneous use of multiple interfaces which can provide a better use of the available bandwidth. MPTCP [RFC6182] has been designed for that purpose, and SCTP [RFC4960] can also be used for it. Raiciu and al. [Raiciu] showed how using multiple paths improve the performances and robustness of

data centers compared to TCP. Furthermore, a communication may be connected simultaneously to different networks with different technologies and takes advantage of their different characteristics. This is typically the case of Offload when ISPs are offloading their RAN communications to a WLAN network. Motivations for offloading is that RAN cannot support all mobile traffic [Cisco]. As a result, with a RAN and a WLAN interface, Mobile phones and ISPs may balance the communications between an unreliable WLAN with economical bandwidth and always connected RAN with expensive bandwidth.

The document focuses on how applications and services protected with IPsec can also take advantage of multiple interfaces. The traditional VPN application with multiple interfaces is the first use case we consider. However, with the offload usage, ISPs are offloading unprotected communications, services from a trusted network - like the RAN - to an untrusted and unreliable network - like the WLAN. This means that the ISP must protect the communications related to these services and applications while being offloaded. IPsec appears to be one way to secure communications transparently to the application.

They are two ways to secure the communications with IPsec. One way is to tunnel the communication to a Security Gateway. The other is to provide end to end security. The document will consider both ways.

Section 3 considers the specific case of VPN with multiple interfaces. Section 4 extends the previous use case by considering the general case of IPsec protected communications using the Tunnel mode. Finally Section 5 considers the case of IPsec protected communications with the Transport mode. For each case, the document details different scenarios that take advantage of multiple interfaces. Then it positions IKEv2 toward each of these scenarios and points out requirements

### 3. Use Case 1: VPN with Multiple Interfaces

This section describes the VPN scenario with connectivity described in figure 1, the End User (EU) has multiple interfaces and figure 1 represents 3 interfaces bound to 3 IP addresses EU @IP\_outer(i), (i in {1, 2, 3}).

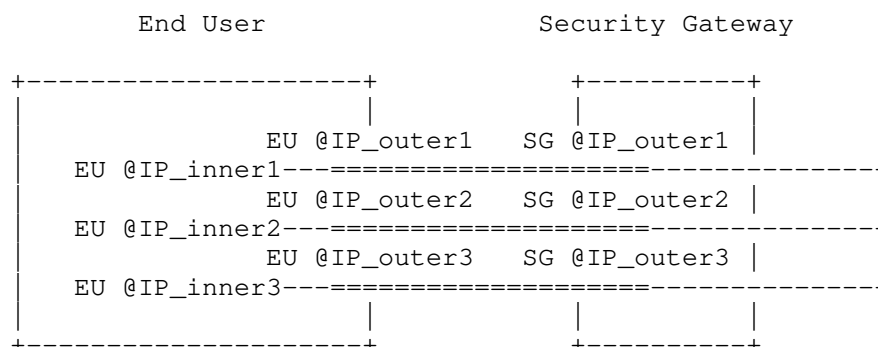


Figure 1: VPN with Multiple Interfaces

### 3.1. Initial MIF IPsec Configuration

#### 3.1.1. Description

This section details how the End User with its three interfaces set (EU @IP\_outer(i), i in {1, 2, 3}) can set an IPsec configuration as represented in figure 1. We consider the IPsec configuration is set using IKEv2, and that the End User uses only a single IKEv2 channel. In other words, each interface MUST NOT be considered independently from each other with its own IKEv2 channel and its own Security Associations.

One of these End User IP addresses is used to set the IKEv2 channel. This IP address is used to set the IKE\_SA as well as for all IKEv2 exchanges. Suppose EU @IP\_outer1 is used for the IKE\_SA.

Using the IKEv2 channel, the End User requests the inner IP addresses EU @IP\_inner(i), i in {1, 2, 3}. If the Security Gateway has multiple interfaces, it advertises the End User, what are the available interfaces.

Once the End User has inner and outer IP addresses, it starts negotiating via the IKEv2 channel the different Security Associations. For each Security Association, the End User and the Security Gateway SHOULD be able to agree on the Traffic Selectors (i.e. the inner IP addresses) as well as the outer IP addresses used for the Tunnel.

#### 3.1.2. Problem Statement

This section positions the current IKEv2 specifications toward the scenario described in Section 3.1.1

To request multiple inner IP addresses, the End User can use the IKEv2 with multiple INTERNAL\_IP\*\_ADDRESS Configuration Attributes in the CFG Payload (Section 3.15 [RFC5996]).

Currently IKEv2 does not provide ways for the Security Gateway to announce the End User the available outer IP addresses - SG @IP\_outer1, SG @IP\_outer2 and SG @IP\_outer3. [I-D.arora-ipsecme-ikev2-alt-tunnel-addresses] details how this could be mitigated. Note that in the VPN use case, the initiator - that is to say the End User - is more likely to request the Security Gateway outer IP addresses, then the reverse. In other words, there seems very few interest for the responder to know the different outer IP addresses of the End User. However, as detailed in Section 5 the more general case SHOULD consider that both initiator and responder can advertise the available interface when the IKEv2 negotiation is initiated.

IKEv2 makes possible the negotiation of the Security Associations associated to each of the EU @IP\_inner(i) IP addresses using a Traffic Selector Payload with one or multiple Traffic Selectors (section 3.13 [RFC5996]). IKEv2 even enables the simultaneous negotiation of Security Associations. However, currently the Security Association negotiation does not specify the outer IP addresses. The outer IP addresses are those used for the IKEv2 channel. In other words, current IKEv2 only considers a single working IP address for both the End User and the Security Gateway. Figure 2 illustrates current IKEv2 capabilities in the VPN use case with different Traffic Selectors associated to a single outer IP address. While negotiating a Security Association, IKEv2 SHOULD be able to specify the source and destination IP addresses.

Note that the benefits of specifying the outer IP addresses provides the End User or Initiator the ability to use simultaneously multiple interfaces. In the specific case of figure 1, the Security Gateway will most likely have a single IP outer IP address. We considered multiple IP addresses on the Security Gateway for the more general case.

Currently, IKEv2 does not provide the ability to negotiate the outer IP addresses of the Tunnel. By default, the outer IP addresses of the Child Security Associations are those used for the IKEv2 channel. This results in the configuration as represented in figure 2. The configuration of figure 1 does not result from an IKEv2 negotiation.

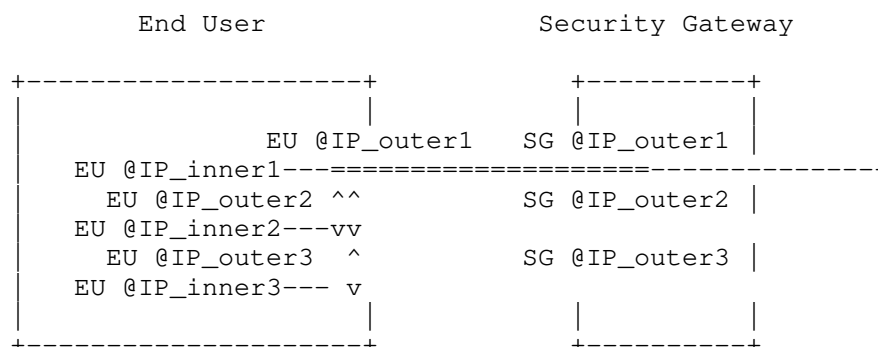


Figure 2: VPN with Multiple Interfaces  
Current IKEv2 negotiation

### 3.1.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 1, IKEv2 SHOULD make possible:

- 1. To specify the different outer IP addresses for the tunnel mode in the Security Association negotiation.
- 2. Make possible the Responder and Initiator to announce its interfaces.

## 3.2. Mobility

### 3.2.1. Description

This section considers how a node with multiple interfaces can modify the value of the outer IP address. In the Tunnel mode, changing the outer IP address results in a mobility, however this should be seen as updating a parameter of the Security Association. Figure 3 illustrates a mobility where EU @IP\_outer3 in Figure 1 is updated by EU @IP\_outer4.

In fact, the Security Association associated to EU @IP\_inner3 includes the outer IP address of the tunnel. The End User and the Security Gateway MUST change this outer IP address from EU @IP\_outer3. The End User MUST modify its Security Association so that packets sent to the Security Gateway are using a valid IP address. Similarly, the Security Gateway MUST update its Security Association so that it can send packets to a reachable destination IP address. The notification of the update is performed using the IKEv2 channel, that is to say in our case EU @IP\_outer1.

Figure 3 illustrates the case where EU @IP\_outer4 is using the same network hardware interface as EU @IP\_outer3. This corresponds to the

case where, for example, the End User decides to use EU @IP\_outer4 instead of EU @IP\_outer3 on the same hardware network interface. Other mobility use cases may also consider the EU @IP\_outer4 may be associated to a different network hardware, including the one associated to EU @IP\_outer(i), i in {1, 2}. Then, EU @IP\_outer4 is different from EU @IP\_outer3 but may be one of the EU @IP\_outer(i), i in {1, 2}.

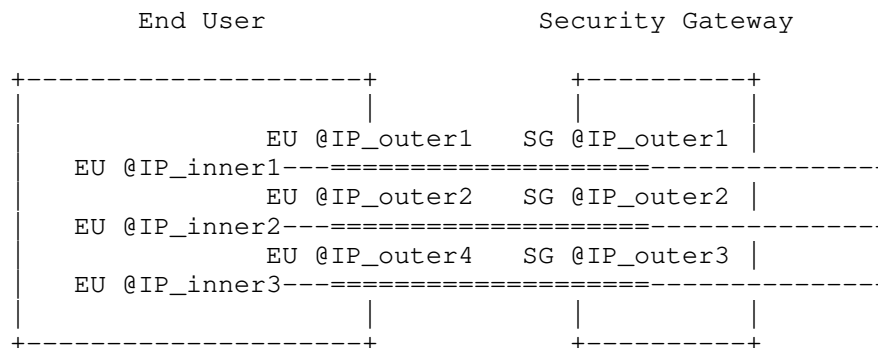


Figure 3: VPN Mobility

### 3.2.2. Problem Statement

Currently IKEv2 proposes different alternative to update a Security Association, and modify the outer IP address of the Tunnel. However none of them really address the description provided in Section 3.2.1

#### 3.2.2.1. MOBIKE

MOBIKE [RFC4555] provides an UPDATE\_SA\_ADDRESSES exchange that updates the outer IP address of the tunnel. As explained in this section MOBIKE cannot be used in the general case described in figure 3 because the updated IP address is necessarily the one associated to the IKEv2 channel. This limitation is due to the fact that MOBIKE has been designed for a single interface.

MOBIKE does not explicitly specify in its message the IP address that has to be updated and the new value for this IP address. The IP address to be updated is the one used by the IKEv2 channel, and the new IP address to consider is the IP address used in the IP header of the UPDATE\_SA\_ADDRESSES message.

If EU @IP\_outer1 is equal to EU @IP\_outer3, then sending an UPDATE\_SA\_ADDRESSES would update the outer tunnel IP address of the Security Associations using the IP address of the IKEv2 channel, that is at least EU @IP\_outer1 and EU @IP\_outer3, with EU @IP\_outer4.



This case is only a specific case and is not applicable when the outer IP address to update is different from the IP address used for the IKEv2 channel.

If EU @IP\_outer3 is different from EU @IP\_outer1, then, the only way to use MOBIKE is to move the IKEv2 channel to EU @IP\_outer3, that is updating EU @IP\_outer1 by EU @IP\_outer3, and then updating EU @IP\_outer3 by EU @IP\_outer4. This is not convenient because all traffic on EU @IP\_outer1 has been transferred to EU @IP\_outer3, and then to EU @IP\_outer4. Furthermore, it is only possible for managed mobility, because we need EU @IP\_outer3 to be a valid interface until IKEv2 uses EU @IP\_outer3. In other words, if EU @IP\_outer3 fails suddenly, moving the IKEv2 channel to EU @IP\_outer3 is not possible anymore.

As a result MOBIKE cannot be used to handle the mobility described in Section 3.2.1.

#### 3.2.2.2. CREATE\_CHILD\_SA

A second alternative is to renegotiates a new Security Association between the End User and the Security Gateway. IKEv2 provides the CREATE\_CHILD\_SA Exchange (Section 1.3 [RFC5996]) to create a new Security Association. Similarly Section 3.1.2 this exchange does not specify the outer IP address of the Tunnel. By default, the outer IP address of the Tunnel is the IP address used for the IKEv2 channel. This does not address the use case described in Section 3.2.1.

If requirements of Section 3.1.3 were fulfilled, that is to say even if the CREATE\_CHILD\_SA would enable to negotiate the outer IP addresses of the Tunnel, then, using the CREATE\_CHILD\_SA exchange would be an alternative. However, this alternative would still suffer from several drawbacks:

- Not Mandatory: The CREATE\_CHILD\_SA is not a mandatory IKEv2 feature, especially for light implementations. For these implementation, an non reachable interface would require re-negotiating both the IKE\_SA and the new Security Association. Furthermore, there is currently no way to advertise whether the implementation supports or not this exchange.
- Resource Consuming Exchange: The CREATE\_CHILD exchange creates a Security Association from scratch and requires all parameters of the Security Association to be specified. This results in a quite complex exchange, which does not take advantage of the already negotiated parameters, like nonces, Keys, Traffic Selectors, Nonces, SPIs. Instead it requires all these parameters to be renegotiated, generation of nonces, keys, as well as multiple interactions with IPsec databases which requires more resources than updating a single parameter within

- a Security Association.
- Two-Successive Exchange: The CREATE\_CHILD exchange creates a new Security Association, however, the previously used Security Association has not been removed from the IPsec databases. As a result, once the new Security Association has been created, a new exchange SHOULD be performed to delete the previous Security Association with the Delete Payload (Section 3.11 [RFC5996]). The Delete Payload specifies the Security Associations to Delete.
- Per Security Association Exchange: The CREATE\_CHILD\_SA exchange creates a specific Security Association, which means that there are as many CREATE\_CHILD\_SA exchanges as Security Association to update. In our case, multiple Security Associations may be bound to a single interface, so the Security Association granularity is not convenient for interface management. Updating an interface implies that all Security Association bound to this interface MUST be updated. In the use case illustrated by figure 3, the End User a single Security Association per interface, so interface and Security Association management have similar granularity. On the other end, for the Security Gateway with a single interface, i.e. (all SG @IP\_outer(i), i in{1, 2, 3} are the same), interface and Security Association do not have the same granularity. Note that with a single interface the Security Gateway would be able to use MOBIKE, but not with two interface (i.e. is SG @IP\_outer2 and SG @IP\_outer3 would be the same).

### 3.2.2.3. One IKE channel per Interface

A fourth alternative consists renegotiating an complete independent IKEv2 channel and a new Security Association. This is out of the scope of this document. This may result as having a IKEv2 channel per interface. Furthermore, independent IKEv2 channels may not simplify IPsec configuration and may result in multiple Security Associations matching a given Traffic Selector, which may cause trouble at least for outbound traffic. Furthermore, in this case, the End User and the Security Gateway must proceed to an authentication.

### 3.2.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 3, IKEv2 SHOULD make possible:

- 1. To update the outer IP address of the tunnel with a IP address that differs from those used for the IKEv2 channel. The Update is not a per security Association negotiation but SHOULD replace all Security Association associated to the old IP address. For all these Security Associations, the old IP

address is replaced by the new IP address. This consists in extending MOBIKE UPDATE\_SA\_ADDRESSES exchange.

### 3.3. Multihoming

#### 3.3.1. Description

This section considers how a node can take advantage of multiple interfaces with multihoming. In case one of these interface fails, then another interface can be used instead. Moving the traffic from one interface to the other is called mobility. This section deals with multihoming, that is the two peers agree that in case an interface fails, a mobility should be triggered on the agreed interface.

Suppose, as represented in figure 4, EU @IP\_outer3 is not reachable anymore. Applications that are multiple interfaces aware, and also bound to the others EU @IP\_inner(i) (i in {1, 2}) IP addresses may handle EU @IP\_outer3 non reachability. On the other hand non multiple interfaces aware applications (like regular TCP connections) bound to EU @IP\_outer3 are stalled and cannot use the other interfaces.

One way to recover the EU @IP\_inner3 unreachability is to reconfigure the Security Association and replace EU @IP\_outer3 by EU @IP\_outer(i) (i in {1, 2}). Figure 5 shows that EU @IP\_outer3 is replaced by EU @IP\_outer2. EU @IP\_outer2 has been provided as an Alternate IP address of EU @IP\_outer3. This means that when one or the other peer notice EU @IP\_outer3 is down, it can trigger a mobility with the appropriated outer IP address. More specifically, the Security Gateway can overcome the failure of EU @IP\_outer3, if it detects the failure before the End User. The End User and the Security Gateway can also agree on an ordered list of Alternate IP addresses.

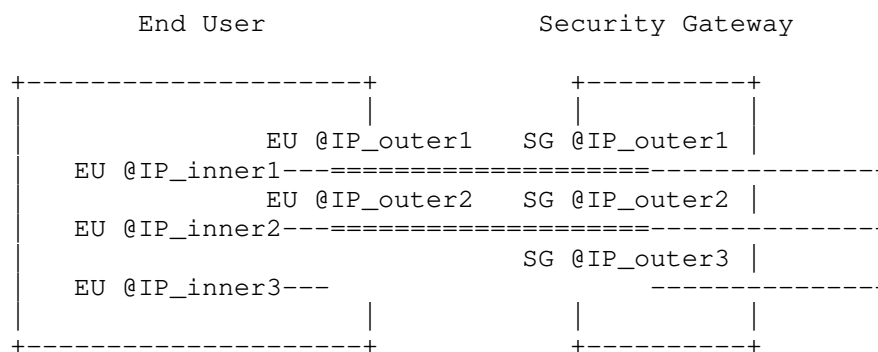


Figure 4: VPN with Mobility/Multihoming between

Multiple Interfaces: EU @IP\_outer3 unreachable

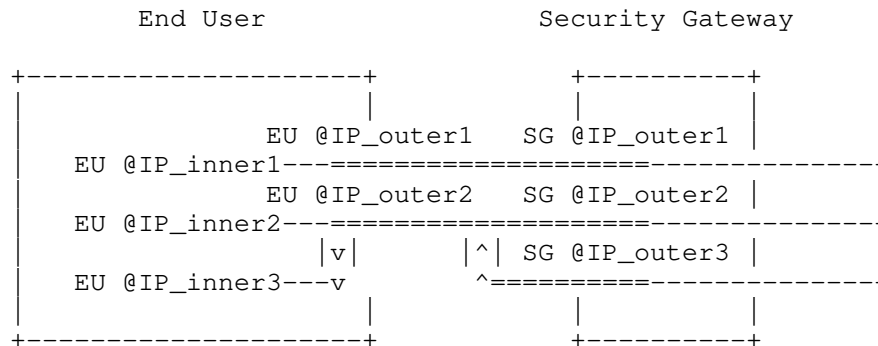


Figure 5: VPN with Mobility/Multihoming between Multiple Interfaces: EU @IP\_outer2 replaces EU @IP\_outer3

### 3.3.2. Problem Statement

Currently Multihoming is handled by MOBIKE with the ADDITIONAL\_IP\*\_ADDRESS Notify Payloads. As with mobility, these payloads are only provided for the interface used by the IKEv2 channel. The main reason is that MOBIKE has been designed for a single interface. In our case, MOBIKE would only make possible to provide Alternate IP addresses to EU @IP\_outer1.

What happens to packets when the Security Gateway performs Multihoming and the End User has not updated its Security Association? Both End User and Security Gateway Security Associations are configured to use the EU @IP\_outer3 IP address. When the Security Gateway notices EU @IP\_outer3 is not reachable it updates its Security Association, triggers a mobility exchange and may start sending packets to EU @IP\_outer2 before the End User has proceeded to the update of its Security Associations. The End User receives this packet and performs a Security Association match. Outer IP addresses will not be performed a match, and the match occurs with the Security Policy Index (SPI). The packet is checked against the Security Policy Databases Selectors. These selectors are based on the inner IP addresses and have not been modified. As a result, packets will not be discarded.

### 3.3.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 3, IKEv2 SHOULD make possible:

- 1. To provide Alternate IP addresses for IP addresses that are different from the one used by the IKEv2 channel. This extends the Multihoming features of MOBIKE to multiple interfaces.
- 2. Reduce the complexity of Multihoming. Although a node **MUST** be able to provide Alternate IP address for a given IP address, it should also be able to provide all its interfaces, and if multihoming is supported on both side, a multihoming rule should be derived by default from this list.

### 3.4. Adding an Interface

#### 3.4.1. Description

Nodes with multiple interfaces may have some interfaces supporting the VPN whereas other interfaces have not been assigned an IP address. When this interface has been assigned an IP address, the current VPN communication may take advantage of this newly available interface. This section is concerned on how a given communication can take advantage of a newly available interface and set its IPsec settings in an optimal way.

Figure 6 represents the End User with multiple interfaces connected to the Security Gateway. We only represented a single interface for the Security Gateway but more interfaces may be also considered. In figure 7, the Security Gateway has an additional interface that becomes active, it advertises the End User this interface is available. The End User may perform some latency and Round Trip Time measurements and decide to use it. In the figure 7, the End User moves the traffic associated to its interface EU @IP\_outer3 to the newly available interface SG @IP\_outer2 of the Security Gateway. Moving the traffic is performed through a mobility operation as described in Section 3.2.

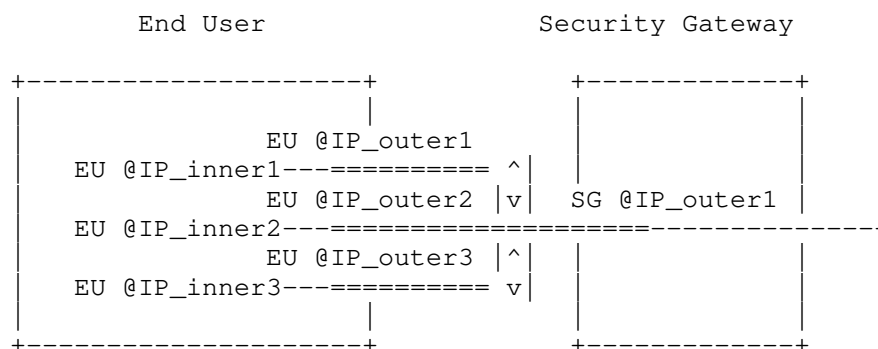


Figure 6: Security Gateway with a single Interface

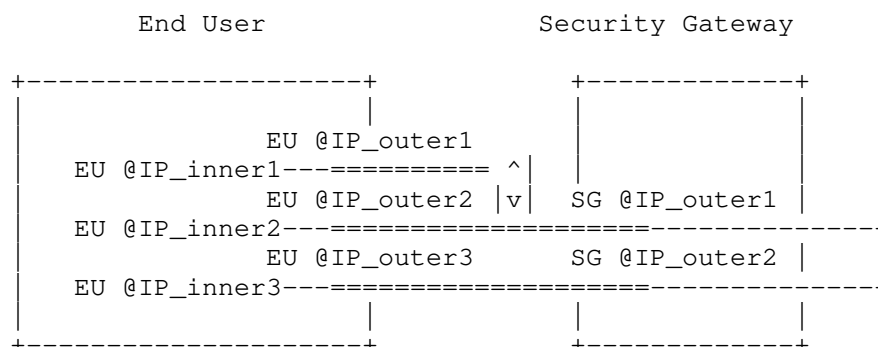


Figure 7: Security Gateway adding an Interface  
New Interface used by the EU @IP\_outer3

Figure 6 and 7 illustrated the case, where the Security Gateway has an additional active interface. In this case, the Security Gateway let the End User decide which interface they prefer to use. By announcing the newly available interfaces no new Security Associations are created. On the other hand, the End User may also want that any service using the other interfaces can use this newly available interface. This requires to derive the Security Associations associated to the new interface from those associated to the already established interfaces. The Security Associations derived for the newly active interface are not created from scratch with a complete negotiation. This case is illustrated by figure 8 and 9.

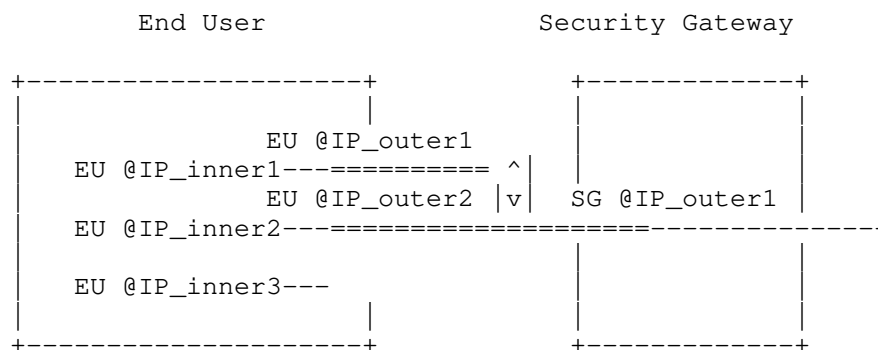


Figure 7: End User with an inactive interface

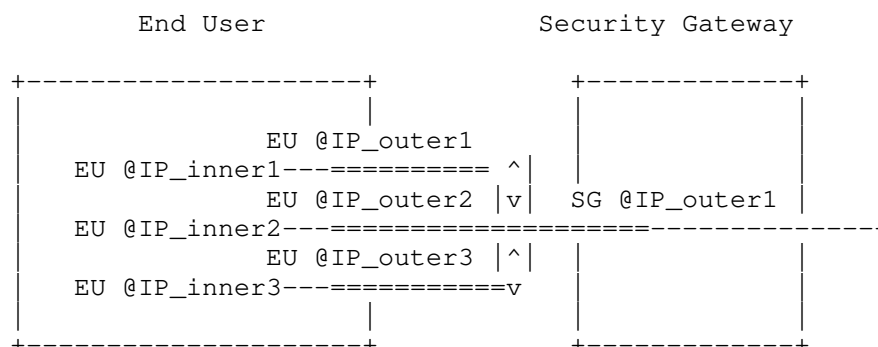


Figure 8: End User with a newly active interface EU @IP\_outer3. All traffic associated to EU @IP\_outer1 and EU @IP\_outer2 is able to use EU @IP\_outer3

In our case, the End User already had a specific inner IP address associated to the newly available interface EU @IP\_outer3. This makes possible the End User to generate the new IPsec Security Associations and new Security Policies associated to EU @IP\_outer3. When the Security Gateway receives the request to add the newly available interface, it may set the newly Security Policies and Security Associations. However, the End User may not have an inner IP address EU @IP\_inner3, and may combine the request to the Security Gateway to add the new interface, with a request for a EU @IP\_inner3 address. In that case, the Security Gateway first sets the IPsec databases, and the End User sets the IPsec databases when it receives the inner IP address.

When an interface is added, unless otherwise specified, the End User wants that all services, except IKEv2 using the available outer IP addresses (EU @IP\_outer1 and @IP\_outer2 addresses) may also be configured to use the newly available IP address EU @IP\_outer3. By adding an interface the End User is not using a finer granularity than the interface granularity. In other words, it does not want to specify how Security Associations are derived. They should be derived in an automatic way. In return, deriving Security Associations and Security Policies is expect to optimize their creation as opposed to using CREATE\_CHILD\_SA.

In the example of figure 7 and 8, the End User is likely to create Security Associations derived from those established with the interfaces EU @IP\_outer1 and EU @IP\_outer2. All services using EU @IP\_outer1 or EU @IP\_outer2 will be able to use EU @IP\_outer3 with the inner IP address EU @IP\_inner3.

The idea is to copy the Security Association associated with EU @IP\_outer1 replace EU @IP\_outer1 by EU @IP\_outer3 and EU @IP\_inner1 by EU @IP\_inner3. SPIs MUST also be changed since there are unique for the Security Association. Then we perform the same with EU @IP\_outer2.

Note that it is important to specify an ordered list of EU @IP\_outer address from which the new SAs are derived, so to guarantee that these new Security Associations are derived the same way on both peers. Then the new Security Association MUST be created only if there are no already existing matching SPD selectors.

In the most basic case of VPN, we only have one Security Association per interface. All services using EU @IP\_inner(i) are tunneled to EU @IP\_outer(i) i in {1,2}. Adding EU @IP\_outer3 only requires to derive Security Association from one interface EU @IP\_outer1 and EU @IP\_outer2. Then, the End User needs to specify the inner and outer IP addresses EU @IP\_inner3, EU @IP\_outer3 and in the specific case represented on figure 7 the outer IP address of the Security Gateway SG @IP\_outer3. The resulting exchange may look something like the exchange represented in figure 10. The mandatory parameters are the IP address used for the traffic selectors, and the outer IP address for the Tunnel on the End User. The destination outer IP address of the Tunnel is optional and, if not specified may be the one used by the IKEv2 channel. The list of interfaces from which are derived the Security Associations and the Security Policies may also be optional. A default value for this list may be the ordered list of associated outer IP addresses of the End User. The nonce may be used to create SPIs.

	End User	Security Gateway
request	Add Interface (EU @IP_inner3, ---> EU @IP_outer3, [outer-destination] [interface-list] [nonce])	
normal case		<--- N()
error case		<--- N(error)

Figure 10: Principle of the Adding Interface exchange

### 3.4.2. Problem Statement

Currently IPsec does not provide any means for a peer to advertise a new interface is available. MOBIKE makes possible to advertise a Alternate IP address is available. However Alternate IP addresses



are only intended to be use in case the Primary Interface is down. In our case, the interface is ready for use. This issue is similar to the one detailed in Section 3.1.2. However, here the announcement corresponds to a dynamic changes, and the list of available IP address does not occurs during the IKE\_INIT exchange, but in a regular information exchange.

Currently the only way IKEv2 provides to create new Security Associations is the CREATE\_CHILD\_SA exchange. Disadvantages of this exchange have been described in Section 3.2.2. The key advantage of adding an interface is to provide an optimized interface management exchange instead of a Security Association management exchange.

#### 3.4.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 1, IKEv2 SHOULD make possible:

- 1. Make possible the Responder and Initiator to announce its interfaces outside the IKE\_INIT exchange. This requirements is similar to the one of Section 3.1.3
- 1. Make possible the Responder and Initiator to automatically derive Security Associations and Security Policies from the existing interface.

#### 3.5. Deleting an Interface

##### 3.5.1. Description

Nodes with multiple interfaces in dynamic environment may have interfaces that are not reachable anymore. This may trigger mobility or multihoming actions. However, the node may also want to delete the Security Associations bound to this interface either as a Tunnel outer IP address or as a Traffic Selector.

##### 3.5.2. Problem Statement

Currently IKEv2 does not make possible to delete an interface from multiple Security Associations. IKEv2 provides a Delete Payload (Section 3.11 [RFC5996] that deletes one or multiple specific Security Associations, identified by their SPI.

##### 3.5.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 3, IKEv2 SHOULD make possible:

- 1. Delete an interface, that is to say all Security Associations associated to that interface.

#### 4. Use Case 2: MIF applications and IPsec Tunnel mode

This section considers applications that can deal with multiple interfaces. This ability can be done with transport layer protocols like MPTCP or SCTP or with applications using one or multiple UDP / TCP connections over the various interfaces, and that manages how to send the data.

The difference between multiple interfaces aware applications and the VPN use case is that the tunnels are established per services, whereas the VPN tunnel all traffic is tunneled to a unique Security Gateway. This may increase the number of Security Associations between the End User and the Security Gateway. This section details motivation for using the IPsec Tunnel mode with multiple interfaces aware applications and position it to the VPN use case of Section 3.

Applications may use the tunnel mode for end-to-end security and to benefit from the Mobility features provided by the Tunnel mode. More specifically, using the Tunnel mode provides Mobility without breaking the connectivity, if upper layer is not mobility aware.

Other motivations for using the Security Gateway is that the End User chose not to tunnel all its traffic to the Security Gateway, but only the traffic that worth being protected. For example, an End User may chose not to tunnel its "youtube" traffic, as well as some of its "https" traffic (as well as its application layer protected traffic). On the other hand, it may want to tunnel all non-protected "http" (as well as other non protected communications).

If each service proposes different Security Gateways, the use case is very similar to the VPN use case, for each service. The main difference is that Security Association are established with different Traffic Selectors.

If multiple services are using the same Security Gateway, this will result for each interface, in multiple Security Associations established with the same Security Gateway - one per service. This case is very similar to the VPN use case but with multiple Security Associations. If "s" is the number of Services connected on the Security Gateway the number of Security Associations is at least "s" (5services are considered independent). If some applications are using multiple flows, then this number may be even larger. In that case, adding an interface results in at least negotiating "s" new Security Associations. Using the CREATE\_CHILD\_SA exchange may

require "s" exchanges whereas using the Adding interface exchange requires only one exchange. This use case is represented in figure 11.

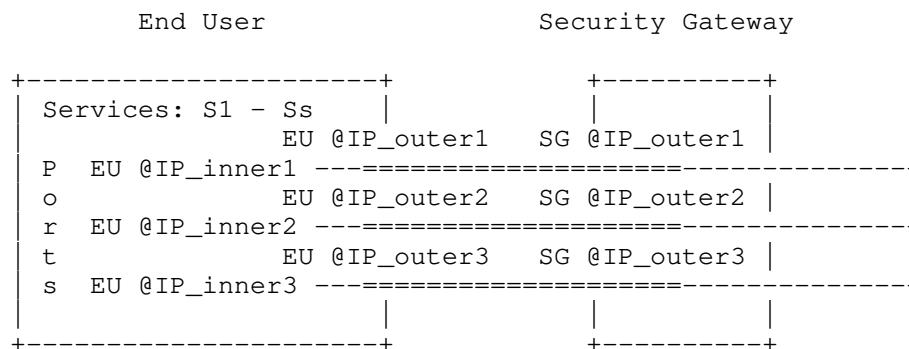


Figure 11: MIF aware applications

Requirements of this use case have already been mentioned in the VPN use case.

#### 5. Use Case 3: MIF aware applications with Transport mode

This Use Case is very similar to the Use Case 2 except that the Transport mode is used instead of the Tunnel mode. The Use Case is illustrated with figure 12.

Unlike in the VPN use case in Section 3 or for multiple interfaces aware applications described in Section 4 using IPsec tunnel mode, the IPsec Transport mode does not involve inner IP addresses.

With Transport mode, we may consider two types of applications. The applications that can handle multiple interfaces. This can be done with transport protocols like MPTCP or SCTP or with a connection manager at the application layer. These applications may have Security Associations on all interfaces. Other Applications with a single using TCP/UDP and without specific connection managers may only deal with a single interface and may only have an Security Association associated to this interface.

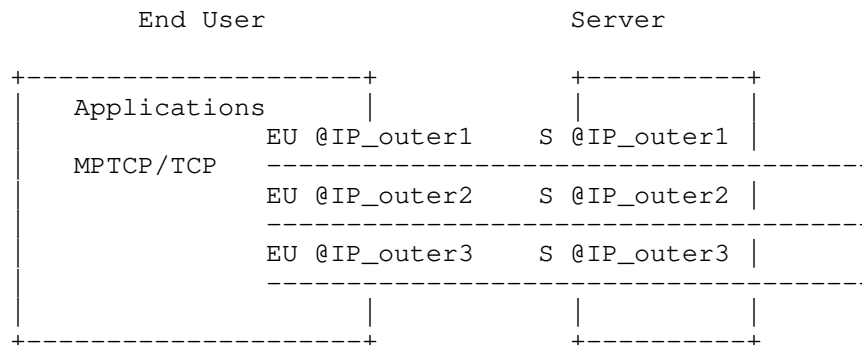


Figure 12: MIF aware applications with the Transport mode

## 5.1. Initial MIF IPsec Configuration

### 5.1.1. Description

In Figure 12, the End User initiates an IKEv2 negotiation using EU @IP\_outer1 and S @IP\_outer1. The Server provides the End User the available interfaces (S @IP\_outer1 i in {1, 2, 3}). Then the End User negotiates Security Associations between the EU @IP\_outer(i) and S @IP\_outer(i) i in {1,2,3} using different Traffic Selectors.

### 5.1.2. Problem Statement

Currently IKEv2 does not make possible a node to announce its available interfaces.

The Transport mode, does not involve tunnel outer IP addresses. Current Security Association exchange enables Traffic Selectors negotiation. These Traffic Selectors are used both for the Security Policy Index (Traffic Selectors) for outgoing traffic and for the Security Association Index for incoming traffic. Current IKEv2 specification enables to set IPsec as described in figure 11.

### 5.1.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 1, IKEv2 SHOULD make possible

- 1. Make possible the Responder and Initiator to announce its interfaces. This requirement is similar to the requirements for VPNs.

## 5.2. Mobility

With regular TCP connection a change of the IP address breaks the connection. Applications may use mobility with the Transport mode with transport protocols that handles with multiple interfaces (like MPTCP or SCTP for example), with multiple independent TCP/UDP connections on the different interfaces. The application manages its connections at the application layer.

Mobility with Transport mode MUST be understood as updating an existing Security Association. The purpose of the IPsec Mobility and the Transport mode is to avoid to create a new Security Association when the IP address of an interface is changing. IPsec configures the layer so that the application can securely go on with its communications. TCP connections are restarted, since changing the IP address will most likely break the existing connection. UDP will start sending on the other interface. Mobility is intended to reduce the time IPsec requires to configure its Security Associations.

With the Tunnel mode, IPsec was in charge of securing and transporting IP datagrams. With the Transport mode, IPsec only secures the communication. Transport of the IP datagrams is shared between the application and the transport layer. Application and IPsec layers are independent and have their own way to handle with mobility. Synchronization between these two layers MUST be performed to avoid that the application moves the traffic on an interface whereas IPsec DISCARD this traffic. Although we do not intend to provide a complete list of how to synchronize these two layers, the list below provides some example where these two layers are synchronized:

- 1. For End Users with two interfaces. In that case, the interface the application may use is determined.
- 2. For applications that are configured with two interfaces.
- 3. For applications that we know the interface they will choose. Like those setting priority to interfaces. This could be set by using Multihoming and ordering the Alternate IP addresses.
- 4. If the Mobility exchange is triggered by the new socket, new packet sent. This case reduces the latency over a CREATE\_CHILD\_SA exchange, but does not anticipate the decision of the application.

### 5.2.1. Description

The mobility scenario we consider in this section is an application using a single interface EU @IP\_outer3 for example. As represented in figure 13, this interface is down. Then the End User get assigned a new IP address EU @IP\_outer4 and uses this interface as represented in figure 14. Both End User and Server MUST update Security Policies

and Security Associations that used EU @IP\_outer3 and replace the value with EU @IP\_outer4. Unlike the Tunnel mode, Traffic Selectors also need to be updated.

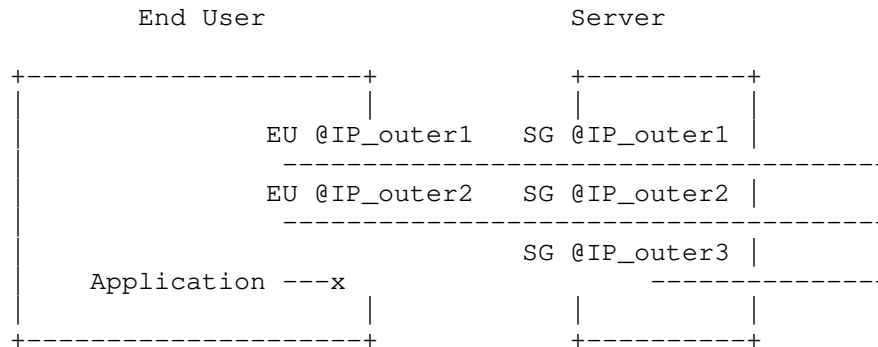


Figure 13: Mobility with Transport mode and Multiple Interfaces: EU @IP\_outer3 unreachable.

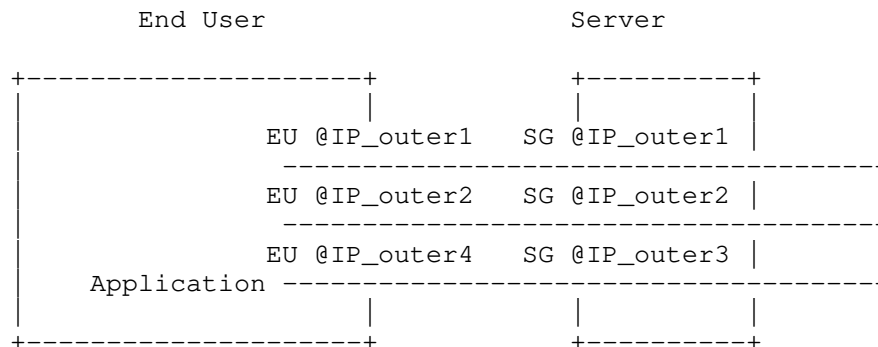


Figure 14: Mobility with Transport mode and Multiple Interfaces: EU @IP\_outer4 replaces EU @IP\_outer3.

### 5.2.2. Problem Statement

Currently IKEv2 does not provide extension that perform any mobility operation.

MOBIKE has only been designed for the Tunnel mode.

The CREATE\_CHILD\_SA suffers for limitations exposed in Section 3.2.2: It is not mandatory in IKEv2 implementation, the exchange requires

much resources as updating the Security associations. Most of the time, it requires an addition Delete exchange and is a per Security Association exchange. However, because no tunnel IP address requires to be negotiated, the CREATE\_CHILD\_SA can set the Security Associations and Policies as described in figure 14.

### 5.2.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 1, IKEv2 SHOULD make possible

- 1. Extend MOBIKE to the Transport mode
- 2. Extend MOBIKE with Transport mode to multiple interfaces requirements described in Section 3.2.3.

### 5.3. Multihoming

Multihoming consists in providing Alternate Interfaces in case a running interface is down, so peers are aware of the parameters to update. Multihoming can be seen as pre-configuring an mobility operation.

#### 5.3.1. Description

With Multihoming, when the End User sets its IPsec configuration as illustrated in figure 12, the End User also specifies for each interface the corresponding Alternate IP address. Although this can be done on a per interface value, we suggest that when multiple interfaces are provided, Alternate IP addresses can be derived automatically and assigned to each interface without being explicitly mentioned. Suppose that in the case of figure 13, for example EU @IP\_outer2 is provisioned as the Alternate IP address of EU @IP\_outer3.

When EU @IP\_outer3 is down, then the End User or the Server triggers a mobility exchange as described in section Section 5.2.1.

#### 5.3.2. Problem Statement

Currently IKEv2 does not make possible to provision Alternate IP addresses for the Transport mode. MOBIKE has only been designed for the Tunnel mode, then as mentioned in Section 3.3.2, MOBIKE only assigns the Alternate IP address for the IP address used by the IKEv2 channel. This is because MOBIKE has been designed for a single interface.

Note that with the Transport mode, the Alternate Address is provided to the outer IP address that is also used as a Traffic Selector, whereas in the Tunnel mode, the Alternate IP address is provided for

the tunnel outer IP address.

Note also that the IKEv2 channel is a special case where Alternate Address is associated to the Transport mode. In fact the IKEv2 channel uses Transport mode, not the Tunnel mode.

### 5.3.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 1, IKEv2 SHOULD make possible to:

- 1. Extend MOBIKE Multihoming to the Transport mode
- 2. Extend MOBIKE with Transport mode to multiple interfaces requirements described in Section 3.3.3. Alternate IP address should be assigned to any interface and can be automatically be derived. Alternate IP address concerns Traffic Selectors and Security Association Indexes.

### 5.4. Adding an Interface

Adding an interface works exactly as described in Section 3.4. The only difference is that when an interface is added with the Transport mode, Traffic Selectors will automatically be associated to this newly added interface, which was not necessarily the case with the Tunnel mode.

### 5.5. Delete an Interface

Similarly to the addition of a new interface, Deleting an interface works exactly as described in Section 3.5. The only difference is that with the Transport mode, Security Associations and Security Policies to delete are these where the specified interface appears as a Traffic Selector rather than as an outer tunnel IP address.

## 6. Security Considerations

The whole document sets MIF requirements for a security protocol.

## 7. IANA Considerations

There is no IANA consideration here.

## 8. Acknowledgment

We would like to thank Daniel Palomares, Pierrick Seite, Brian Carpenter, Hui Deng, Jong-Hyouk Lee, Juan Carlos Zuniga and



Konstantinos Pentikousis for their useful comments.

## 9. References

### 9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, June 2006.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

### 9.2. Informational References

- [Cisco] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-2015", February 2011.
- [I-D.arora-ipsecme-ikev2-alt-tunnel-addresses] Arora, J. and P. Kumar, "Alternate Tunnel Addresses for IKEv2", draft-arora-ipsecme-ikev2-alt-tunnel-addresses-00 (work in progress), April 2010.
- [RFC6182] Ford, A., Raiciu, C., Handley, M., Barre, S., and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC 6182, March 2011.
- [Raiciu] Arora, C., Barre, S., Plunkte, C., Greenhalgh, A., Wischik, D., and M. Handley, "Improving datacenter performance and robustness with multipath TCP", SIGCOMM 2011 Toronto, Canada, August 2011.

Authors' Addresses

Daniel Migault  
Francetelecom - Orange  
38 rue du General Leclerc  
92794 Issy-les-Moulineaux Cedex 9  
France

Phone: +33 1 45 29 60 52  
Email: mglt.ietf@gmail.com

Carl Williams  
MCSR Labs  
Philadelphia, PA 19103  
USA

Phone: 650-279-5903  
Email: carlw@mcsr-labs.org



MIF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 18, 2013

T. Reddy  
P. Patil  
D. Wing  
Cisco  
October 15, 2012

Relay-Supplied DHCPv6 Precedence Options  
draft-reddy-mif-dhcpv6-precedence-ops-02

Abstract

Network configuration of hosts is currently relatively static with little consideration of dynamic network characteristics. The network infrastructure is aware of dynamic network characteristics. This specification extends DHCPv6 so that the DHCPv6 relay agent can influence a host's configuration.

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 April 18, 2013.

Copyright Notice

Copyright (c) 2012 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

1. Introduction . . . . .	3
2. Terminology . . . . .	3
3. Usage Scenarios . . . . .	3
3.1. IPv6 Multihoming . . . . .	3
3.2. Disabling IPv6 Temporary Addresses . . . . .	4
3.2.1. Avoiding Excessive IP-Based Authentication . . . . .	4
3.2.2. Reducing Management Impact . . . . .	5
4. Options . . . . .	5
4.1. Address Selection option . . . . .	6
4.2. Relay-Supplied Prefix Option . . . . .	7
5. Relay Agent Behaviour . . . . .	8
6. DHCPv6 Server Behaviour . . . . .	8
6.1. Address Selection option . . . . .	8
6.2. Relay-Supplied Prefix Option . . . . .	9
7. Security Considerations . . . . .	9
8. IANA Considerations . . . . .	9
9. Change History . . . . .	9
9.1. Changes from draft-reddy-mif-dhcpv6-precedence-ops-00 to -01 . . . . .	10
9.2. Changes from draft-reddy-mif-dhcpv6-precedence-ops-01 to -02 . . . . .	10
10. References . . . . .	10
10.1. Normative References . . . . .	10
10.2. Informative References . . . . .	11
Authors' Addresses . . . . .	11

## 1. Introduction

DHCPv6 allows relatively static information to be configured in hosts, which is somewhat limiting. On a dynamic network, the DHCPv6 relay agent can observe characteristics of a network -- such as IPv6 multihoming which might be temporarily unavailable or need load balancing of traffic towards each upstream ISPs. By including additional information in relayed DHCPv6 messages, the DHCPv6 relay agent can influence the DHCPv6 server to provide answers that are better suited to the host's configuration on the network.

In this document we propose new DHCPv6 options to be added by the DHCPv6 relay agent when it generates a Relay-Forwarded message. [RFC6724] defines default address selection mechanisms for IPv6 that allow nodes to select appropriate address when faced with multiple source and/or destination addresses to choose between. An initial desire is to influence the DHCPv6 server's responses that modify the host's address policy table [I-D.ietf-6man-addr-select-opt] based on observed network characteristics.

## 2. 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].

## 3. Usage Scenarios

The DHCPv6 extension described in this document is useful with IPv6 multihoming and with IP address-based authentication.

### 3.1. IPv6 Multihoming

- o In Proxy Mobile IPv6 [RFC5213] where Mobile Node is assigned prefixes from both local access network and home network. This will allow selected traffic to go through the Mobile Packet Core and the rest through the Local access Network. When DHCPv6 Relay Agent is co-located with the mobile access gateway, the proposal is for the relay agent to influence the DHCPv6 Server in the home network by adding the Address Selection option. The relay agent can add an Address Selection option to the DHCPv6 request suggesting the local access network address selection policy table overriding the default address selection parameters and policy table. The DHCPv6 server in the home network will merge the policy received in Address Selection option with it's own policy table as explained in section 4.3 of

[I-D.ietf-6man-addr-select-opt]. This updated policy table will be provided to the DHCPv6 client (MN) in Address Selection option (OPTION\_ADDRSEL\_TABLE). When the DHCPv6 Server is co-located with the mobile access gateway, the DHCPv6 Server in the local access network will receive the policy table from the DHCPv6 server in the home network using DHCPv6 INFORMATION-REQUEST. The DHCPv6 server in local access network will merge the received policy table with it's local policy table. The following figure depicts this scenario.

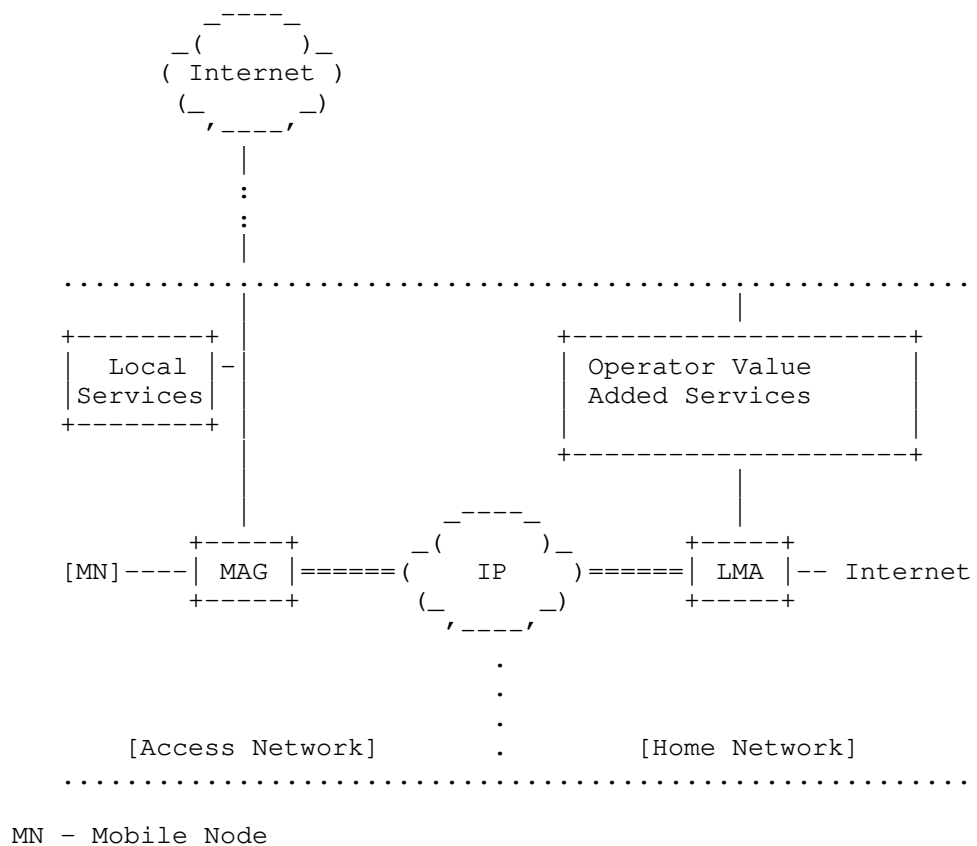


Figure 1: Proxy Mobile IPv6

### 3.2. Disabling IPv6 Temporary Addresses

#### 3.2.1. Avoiding Excessive IP-Based Authentication

Some managed networks authenticate hosts with an authentication supplicant or for hosts lacking the supplicant perform address-based authentication. When Address-based authentication is used, re-

authentication occurs for each address obtained by the host, which can create a lot of authentication transactions. To reduce this chatter, it can be useful to disable IPv6 Privacy Addresses [RFC4941] on those hosts using address-based authentication. In a managed network, this option will ensure that temporary addresses are disabled for hosts without authentication supplicant. This way managed networks can conditionally disable temporary addresses for only a set of hosts.

The relay agent may be configured with the external prefixes that will be assigned to the host. In that case, the relay agent would use the Address Selection option. In the case where the relay agent is unaware of the external prefixes that will be assigned to the host, the relay agent uses the Relative Precedence option. Details for processing those options are described later in the document.

Whenever either of those options is used, a DHCPv6 server that understands those options will ignore the IA\_TA options in the DHCPv6 request, effectively disabling the use of temporary addresses for that host.

### 3.2.2. Reducing Management Impact

In addition, there are known issues in managing privacy extensions in certain scenarios. These are described in managing privacy extensions [I-D.gont-6man-managing-privacy-extensions]. In such scenarios, conditionally disabling temporary addresses allows administrators to better manage deployments.

## 4. Options

To realize the functions described above, this document defines new DHCPv6 option Relay-Supplied Prefix and updates the Address Selection option defined in [I-D.ietf-6man-addr-select-opt]. These DHCPv6 options are added by the DHCPv6 relay agent when it relays a DHCPv6 message, and both MAY appear together in the same DHCPv6 message.



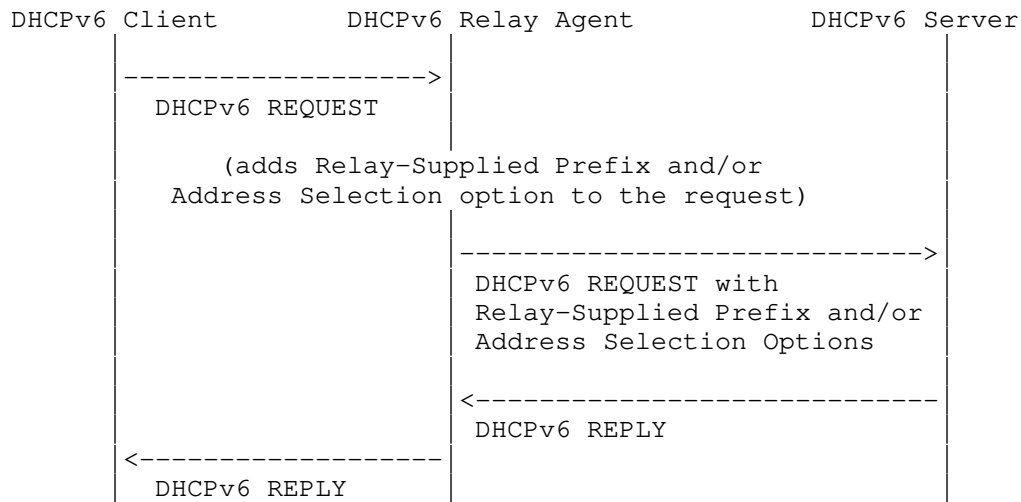


Figure 2: Message Flow, Relay Agent adding Option

Relay-Supplied Prefix option carries host and network information observed by the DHCPv6 relay agent such as host does not support 802.1x supplicant and will be subjected to web-authentication. The Address Selection option allows prioritizing among a list of prefixes the DHCPv6 relay agent expects the DHCPv6 server to provide to the host.

#### 4.1. Address Selection option

The layout of the Address Selection option is below:

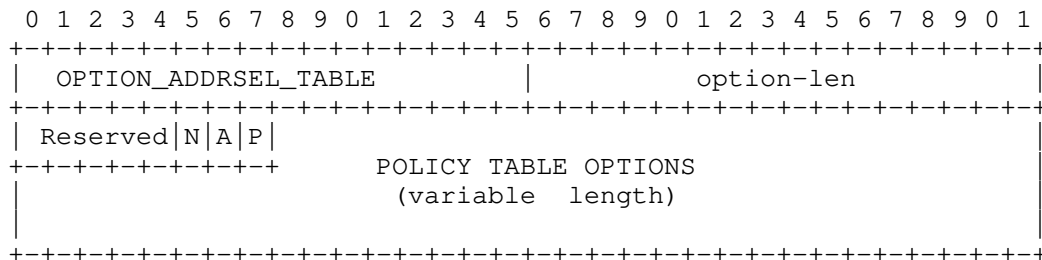


Figure 3: Option Type 1 message format

The fields are described below:

option-code : OPTION\_ADDRSEL\_TABLE defined in  
[I-D.ietf-6man-addr-select-opt]

option-len: Option Length

Reserved: Must be 0 and ignored by the server.

N: A value of 1 indicates that the relay agent wants the DHCPv6 server to ignore any IA\_TA options in the DHCPv6 request, as if the IA\_TA options were not present. This effectively disables privacy extensions [RFC4941]. A value of 0 indicates the IA\_TA options, if present in the DHCPv6 request, are processed normally by the DHCPv6 server. This value has no impact on destination prefixes.

A: This flag MUST be set to 0 and ignored by the DHCPv6 server

P: This flag MUST be set to 0 and ignored by the DHCPv6 server.

Prefix Table Options: Zero or more Address Selection Policy Table options defined in [I-D.ietf-6man-addr-select-opt].

#### 4.2. Relay-Supplied Prefix Option

The Relay-Supplied Prefix option is defined below:

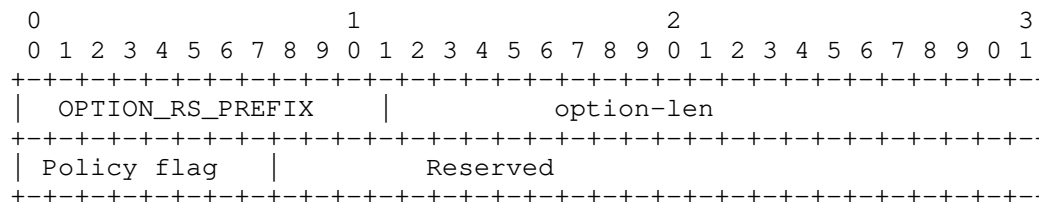


Figure 4: Option Type 2 message format

option-len: Length of the option.

Policy flag: 8-bit unsigned integer.

Reserved: Must be 0 and ignored by the server.

The Policy Flag is defined below, and the actions taken by the DHCPv6 server based on this flag are described in Section 6.

Value	Name	Description
0x01	IPV6_DIS_TEMP_ADDR	Disable IPv6 Temporary Address

Figure 5: Policy flag Values

## 5. Relay Agent Behaviour

DHCPv6 relay agents that implement this specification MUST be configurable for sending the Address Selection option and the Relay-Supplied Prefix option. Relay agents SHOULD have separate configuration for each option to determine if it is to be added to DHCPv6 request. A relay agent will include these options in the option payload of a Request message. DHCPv6 relay agent should set Address Selection option when there is a need to change the label/precedence value for prefixes in scenario's discussed in Section 3.1 and/or disable IPv6 temporary addresses for the host.

Discussion: To reduce end-user configuration of the DHCPv6 relay agent, the DHCPv6 relay agent can use the mechanism specified in [RFC3633] to automatically learn the IPv6 prefixes that will be delegated to DHCPv6 clients. DHCPv6 relay agent in future can use leasequery-like capability discussed in section 3.2 of RFC [RFC5007] to learn the prefix information from DHCPv6 server.

DHCPv6 relay agent should set Relay-Supplied Prefix option when it receives DHCPv6 request from a host with specific characteristics like authenticated using address based mechanism. Relative Precedence option is used when the relay agent is unaware of the external prefixes to be assigned to the host.

## 6. DHCPv6 Server Behaviour

Upon receiving a DHCPv6 request containing the Address Selection option or the Relay-Supplied Prefix Option, the DHCPv6 server processing is described below:

### 6.1. Address Selection option

Address Selection option - The DHCPv6 server should send a reply to the host with the prefixes received from DHCPv6 relay agent along with Precedence. The DHCPv6 server will merge the policy received in Address Selection option with it's own policy table as explained in section 4.3 of [I-D.ietf-6man-addr-select-opt].

If the option has "N" bit set to 1, the server SHOULD ignore the IA\_TA options in the DHCPv6 request, effectively disabling the use of temporary addresses for that prefix. The DHCPv6 server will ignore the "N" bit for destination prefixes.

Note : If DHCPv6 servers receives both options with conflicting flags IPV6\_DIS\_TEMP\_ADDR and "N" bit then it SHOULD treat it as mis-configuration on the relay agent and discard these options.

## 6.2. Relay-Supplied Prefix Option

The Relay-Supplied Prefix Option contains flags that defines the characteristics of the host.

1. IPV6\_DIS\_TEMP\_ADDR - This flag indicates that Temporary IPv6 address allocation is to be disabled for the host. The DHCPv6 server should ignore any IA\_TA options in the DHCPv6 request.

## 7. Security Considerations

Relay-Supplied Prefix is exchanged only between the DHCPv6 relay agent and DHCPv6 server and Address Selection option can originate either from the server or the relay agent, section 21.1 of [RFC3315] provides details on securing DHCPv6 messages sent between servers and relay agents. And, section 23 of [RFC3315] provides general DHCPv6 security considerations.

It is possible for a DHCPv6 client to include the Relay-Supplied Prefix option or the Address Selection options, which would be received by a DHCPv6 server. This would cause the DHCPv6 client to receive a different DHCPv6 response than it would have otherwise received. .

## 8. IANA Considerations

IANA is requested to assign option code to OPTION\_RS\_PREFIX from the option-code space as defined in section "DHCPv6 Options" of [RFC3315].

## 9. Change History

[Note to RFC Editor: Please remove this section prior to publication.]

9.1. Changes from draft-reddy-mif-dhcpv6-precedence-ops-00 to -01

- o Added Proxy Mobile IPv6 with traffic offload use-case in Section 3.1.
- o Updated Section 3.2.1 to highlight the ability to disable temporary addresses selectively.

9.2. Changes from draft-reddy-mif-dhcpv6-precedence-ops-01 to -02

- o Updated usecase in section 3.1
- o Changed Absolute Precedence Option

10. References

10.1. Normative References

- [I-D.gont-6man-managing-privacy-extensions]  
Gont, F. and R. Broersma, "Managing the Use of Privacy Extensions for Stateless Address Autoconfiguration in IPv6", draft-gont-6man-managing-privacy-extensions-01 (work in progress), March 2011.
- [I-D.ietf-6man-addr-select-opt]  
Matsumoto, A., Fujisaki, T., and T. Chown, "Distributing Address Selection Policy using DHCPv6", draft-ietf-6man-addr-select-opt-06 (work in progress), September 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.
- [RFC5007] Brzozowski, J., Kinnear, K., Volz, B., and S. Zeng, "DHCPv6 Leasequery", RFC 5007, September 2007.

- [RFC5213]   Gundavelli, S., Leung, K., Devarapalli, V., Chowdhury, K.,  
              and B. Patil, "Proxy Mobile IPv6", RFC 5213, August 2008.
- [RFC6724]   Thaler, D., Draves, R., Matsumoto, A., and T. Chown,  
              "Default Address Selection for Internet Protocol Version 6  
              (IPv6)", RFC 6724, September 2012.

#### 10.2.   Informative References

- [RFC3633]   Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic  
              Host Configuration Protocol (DHCP) version 6", RFC 3633,  
              December 2003.

#### Authors' Addresses

Tirumaleswar Reddy  
Cisco Systems, Inc.  
Cessna Business Park, Varthur Hobli  
Sarjapur Marathalli Outer Ring Road  
Bangalore, Karnataka   560103  
India

Email: [tiredy@cisco.com](mailto:tiredy@cisco.com)

Prashanth Patil  
Cisco Systems, Inc.  
Cessna Business Park, Varthur Hobli  
Sarjapur Marthalli Outer Ring Road  
Bangalore, Karnataka   560103  
India

Email: [praspati@cisco.com](mailto:praspati@cisco.com)

Dan Wing  
Cisco Systems, Inc.  
170 West Tasman Drive  
San Jose, California   95134  
USA

Email: [dwing@cisco.com](mailto:dwing@cisco.com)



MIF WG  
Internet-Draft  
Intended status: Informational  
Expires: February 15, 2014

P. Seite  
Orange  
JC. Zuniga  
InterDigital Communications, LLC  
August 14, 2013

MIF API for Connection Management  
draft-seite-mif-cm-02.txt

Abstract

There is currently a need to present a coherent connection management behaviour for different terminal platforms (e.g. mobile phones, PCs, tablets, etc.). This document discusses how a connection manager can use the MIF API to provide this coherent behaviour and enhance the end user's experience when a terminal is able to connect to multiple interfaces. The goal of this document is not to define a connection manager specification, but to focus on the interaction with the MIF API and suggest relevant generic messages for the interface.

This document is for discussion and its intention is to help clarifying the utilization of the MIF API in a connection management context and propose some relevant considerations.

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 February 15, 2014.

Copyright Notice

Copyright (c) 2013 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

1. Introduction	2
2. Architecture of a MIF terminal	2
3. Use-case	3
4. Functions of the connection manager	5
5. Security Considerations	9
6. IANA Considerations	9
7. Acknowledgements	9
8. References	9
8.1. Normative References	9
8.2. Informative References	9
Authors' Addresses	10

## 1. Introduction

[I-D.ietf-mif-api-extension] describes an abstract API that provides commands and services for applications and higher layer APIs running on a terminal with more than one interface. There is currently a need to present a coherent connection management behaviour for different terminal platforms (e.g. mobile phones, PCs, tablets, etc.), as users often experience a very different behaviour when connecting with various platforms to the same networks and for the same purposes (e.g. web browsing, email access, dedicated applications, etc.). This document builds on top of the MIF API and aims to discuss how connection managers can use the MIF API to provide a coherent and constant behaviour to the users. The goal of this document is not to define a connection manager specification, but to focus on the interaction with the MIF API and suggest relevant generic messages for the interface.

This document is only for discussion; its intention is to help clarifying the utilization of the MIF API in a connection management context.

## 2. Architecture of a MIF terminal

The terminal's MIF API based architecture is an instantiation of the MIF API model described in [I-D.ietf-mif-api-extension]; main functions and APIs are described below:

- o MIF API: it provides information as per [I-D.ietf-mif-api-extension].
- o Connection Manager: it is an application relying on the MIF API; this application acts on behalf of other running applications. The connection manager decides about the mapping between IP flows and interfaces, then configures the IP stack, and lower layers, accordingly, e.g. configuration of the routing table. The connection manager relies on information provided either by the MIF API or the OS API. Only interface with the MIF API is in the scope of this document.
- o OS API: Provides the interface to manipulate IP object configuration, e.g. routing table.

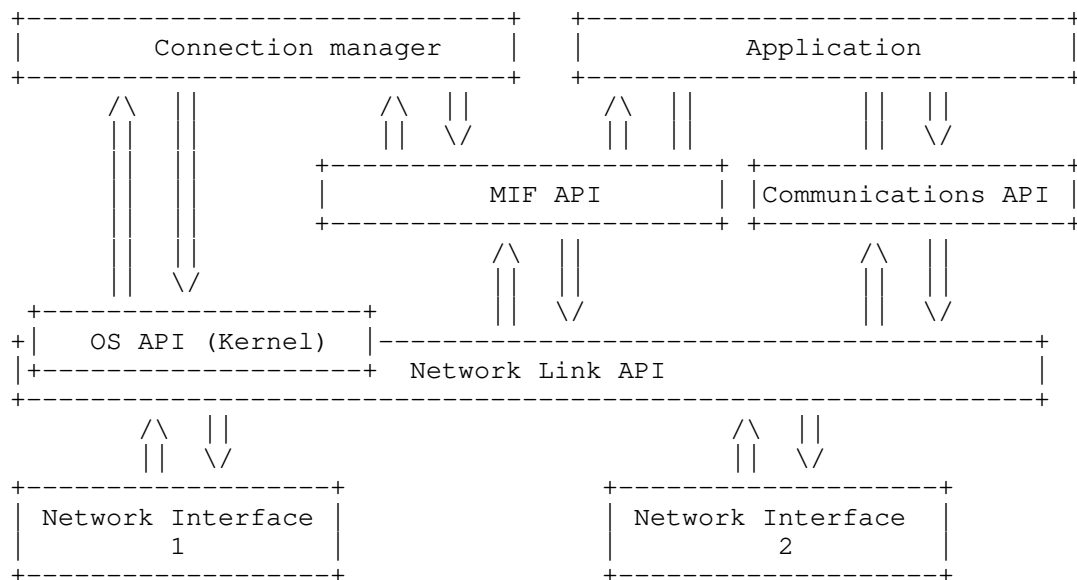


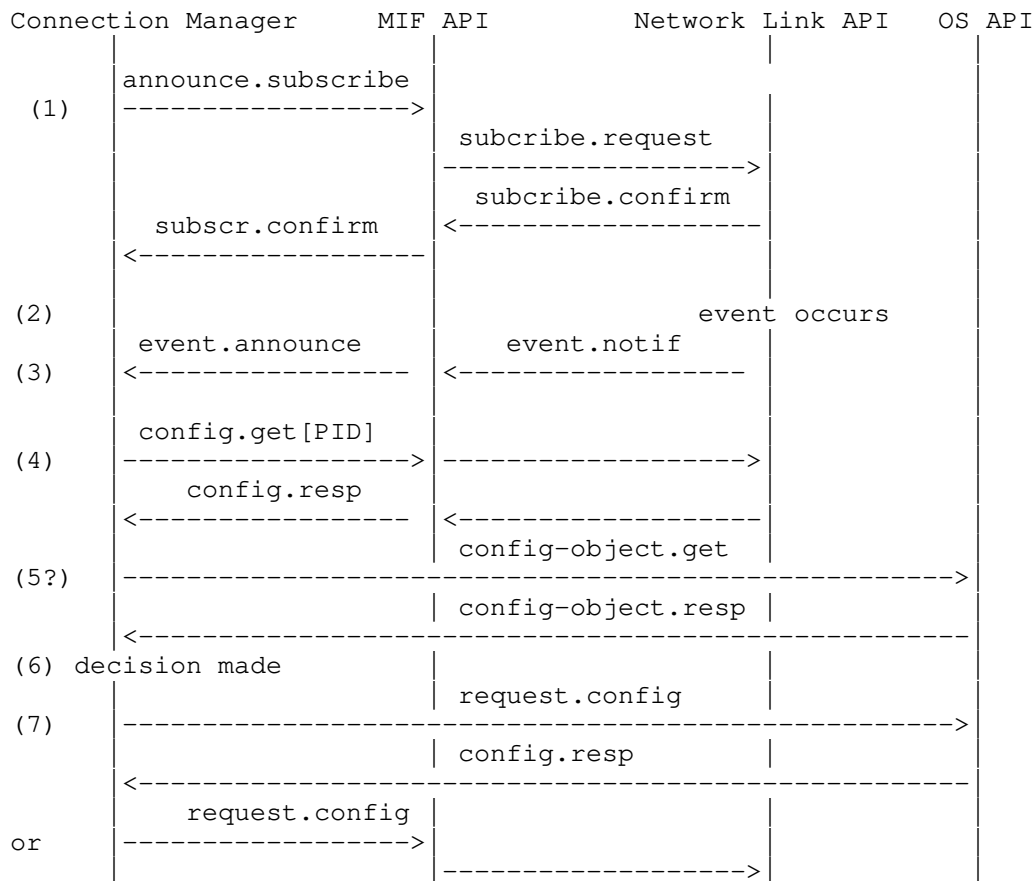
Figure 1: MIF API framework

### 3. Use-case

The presented use-case aims to illustrate the behaviour of the MIF API in a concrete situation. The use-case is as follows:

1. Multiple IP communications are running simultaneously; each can be mapped to different interfaces/provisioning domains.
2. The connection manager selects the appropriate interface/provisioning domain for the application; making a decision according to various criteria (information provided by the MIF and lower layers APIs, user preferences, and so on).

The interaction between the different APIs is depicted in Figure 2. It is assumed that at least one IP communication is running. Then, an interface event occurs and the connection manager decides to move the communication to a different interface.



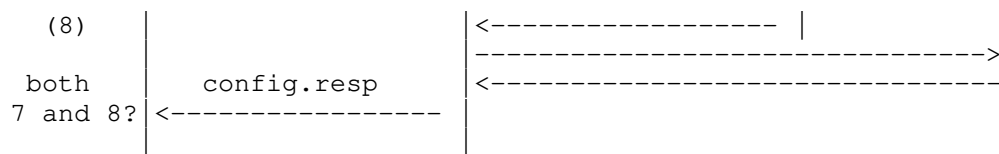


Figure 2: APIs interaction

Operations are as follows:

1. The connection manager subscribes to the MIF API notifications [I-D.ietf-mif-api-extension].
2. An event, to which the connection manager has subscribed, occurs; e.g. a new interface becomes available or a low radio signal level is crossed.
3. The connection manager is notified about the event.
4. In order to take its decision, the connection manager gets some configuration information from the MIF API.
5. The connection manager fetches additional information from the OS API
6. The connection manager decides to move the ongoing IP communication to another interface.
7. The connection manager requests the OS API to reconfigure one or multiple interfaces according to the decision; for example, the connection manager could request reconfiguration of the routing table or trigger a MIP operation.

#### 4. Functions of the connection manager

This section focuses on the interactions between the connection manager and the MIF API and OS API. The interactions between the connection manager and other complementary APIs, like user preferences and/or ANDSF network operator policies are out of the scope of this document.

A connection manager may also rely on different abstraction layers together with the MIF API. The IEEE 802.21 MIH SAP [IEEE802.21] is an example of such an abstraction layer, which can be seen as a partial instantiation of the MIF API. A companion document

[I-D.zuniga-mif-802-21-overview] addresses interaction between IEEE 802.21 and MIF API.

Generic connection manager functions and their relation to the MIF API are described below. The following assumes the MIF API is the unique API for any manipulation of IP objects. However, current MIF API allows to gather information from the IP stack but not to configure it. As a consequence three functions are added to the MIF API: `GetIPtype()`, `ConfigFlowRouting()` and `SetSourceAddress()`.

`Subscribe(eventID)`

Description: register for a MIF API event notification, e.g. WLAN scan results ready, WLAN connected, WLAN disconnected, interface is going to be disconnected detected (e.g. because of low radio signal level detected), Cellular connected, Cellular disconnected, etc.

Input: identifier of the event to be notified. Some events are defined in [I-D.ietf-mif-api-extension]

API: MIF API

`UnSubscribe(eventID)`

Description: unregister to a MIF API notification.

Input: identifier of event. Some events are defined in [I-D.ietf-mif-api-extension]

API: MIF API

`ListInterfaces()`

Description: return the list of available interfaces with their characteristics. Interfaces may have different access technologies.

Input: n/a

API: MIF API

`ListProvisioningDomains()`

Description: return the list of available provisioning domains with their characteristics.

Input: n/a

API: MIF API

GetStatus(IID)

Description: provide the status of the interface, e.g. enabled/disabled, active, idle, connection failed, connecting, disconnecting, scanning, unknown state, etc.

Input: Interface Identifier

API: MIF API

IPconnectivityCheck(PID, IP[])

Description: check IP connectivity to the intranet/Internet: the interface may have a valid IP address but no IP connectivity to data networks (e.g. web based authentication through a captive portal).

Input: Provisioning domain Identifier, IPaddresses to be tested

API: MIF API

GetConfiguration(IID)

Description: retrieve layer 2 configuration information for a given interface.

Input: Interface Identifier

API: OS API

SetConfiguration(IID)

Description: configures an interface, e.g. enable/disable, scan, etc.

Input: Interface Identifier

API: OS API

GetConfiguration(PID)

Description: retrieve configuration information for a given provisioning domain(IP address(es), DNS, default gateway, authentication method, associated interface(s))

Input: Provisioning domain Identifier

API: MIF API

SetConfiguration(PID)

Description: configure provisioning domain information (IP address(es), default gateway, authentication method, associated interface, routing table, etc.)

Input: Provisioning domain Identifier

API: MIF API

GetTheoreticalQoS(IID)

Description: provide information on the theoretical interface capabilities (e.g. upload/download speed)

Input: Interface Identifier

API: MIF API

GetAvailableQoS(IID)

Description: provide information on the quality of communication (Jitter, delay, average upload data rate, average Download data rate, signal strength, etc.)

Input: Interface Identifier

API: MIF API

GetIPType(IP address)

Description: return the type of address and properties (e.g. local, remote, mobile IP anchored, etc.). This function is to be added to the MIF API as per [I-D.ietf-mif-api-extension]. [I-D.korhonen-dmm-prefix-properties].

Input: IP address

API: updated MIF API

ConfigFlowRouting(ROUTE, FlowID)

Description: associate a route, ROUTE, to the IP flow identified by FlowID, e.g. as defined in [RFC6088]. This function is to be added to the MIF API as per [I-D.ietf-mif-api-extension].

Input: routing table identifier, flow identifier

API: updated MIF API

SetSourceAddress(IP, FlowID)

Description: influence source address selection for a given IP flow. This function is to be added to the MIF API as per [I-D.ietf-mif-api-extension].

Input: IP source address, flow identifier

API: updated MIF API

## 5. Security Considerations

TBD.

## 6. IANA Considerations

This document has no actions for IANA.

## 7. Acknowledgements

The authors would like to express their gratitude to Ralph Droms, Ted Lemon and Dave Thaler for the fruitful discussions regarding MIF API and connection managers.

## 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.
- [RFC6088] Tsirtsis, G., Giarreta, G., Soliman, H., and N. Montavont, "Traffic Selectors for Flow Bindings", RFC 6088, January 2011.
- [RFC6089] Tsirtsis, G., Soliman, H., Montavont, N., Giaretta, G., and K. Kuladinithi, "Flow Bindings in Mobile IPv6 and Network Mobility (NEMO) Basic Support", RFC 6089, January 2011.

### 8.2. Informative References

[I-D.deng-mif-api-session-continuity-guide]



Deng, H., Krishnan, S., Lemon, T., and M. Wasserman,  
"Guide for application developers on session continuity by  
using MIF API", draft-deng-mif-api-session-continuity-  
guide-03 (work in progress), October 2012.

[I-D.ietf-mif-api-extension]

Liu, D., Lemon, T., and Z. Cao, "MIF API consideration",  
draft-ietf-mif-api-extension-03 (work in progress),  
November 2012.

[I-D.ietf-netext-logical-interface-support]

Melia, T. and S. Gundavelli, "Logical Interface Support  
for multi-mode IP Hosts", draft-ietf-netext-logical-  
interface-support-07 (work in progress), April 2013.

[I-D.korhonen-dmm-prefix-properties]

Korhonen, J., Patil, B., Gundavelli, S., Seite, P., and D.  
Liu, "IPv6 Prefix Mobility Management Properties", draft-  
korhonen-dmm-prefix-properties-03 (work in progress),  
October 2012.

[I-D.zuniga-mif-802-21-overview]

Zuniga, J. and P. Seite, "IEEE 802.21 Overview", draft-  
zuniga-mif-802-21-overview-00 (work in progress), February  
2013.

[IEEE802.21]

IEEE, "IEEE Standard for Local and Metropolitan Area  
Networks - Part 21: Media Independent Handover Services",  
IEEE LAN/MAN Std 802.21-2008, January 2009.", 2009, <  
[http://www.ieee802.org/21/private/Published%20Spec/  
802.21-2008.pdf](http://www.ieee802.org/21/private/Published%20Spec/802.21-2008.pdf)>.

#### Authors' Addresses

Pierrick Seite  
Orange  
4, rue du Clos Courtel, BP 91226  
Cesson-Sevigne 35512  
France

Email: [pierrick.seite@orange.com](mailto:pierrick.seite@orange.com)

Juan Carlos Zuniga  
InterDigital Communications, LLC  
1000 Sherbrooke Street West, 10th floor  
Montreal, Quebec H3A 3G4  
Canada

Email: [JuanCarlos.Zuniga@InterDigital.com](mailto:JuanCarlos.Zuniga@InterDigital.com)