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 equiped 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 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

Table of Contents

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-awared, 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 appliation 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 Inteface). Similar functionaility 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 availablility 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 assgined, 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 inteface. 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, Quebecy
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

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

Copyright Notice

Table of Contents

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 ny 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 recieved 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.

```
                        +-------------------------------------------+
                        |               Application                 |
                        +-------------------------------------------+
                               /\   ||          /\  ||    /\  ||
                               ||   \/          ||  ||    ||  ||
                        +--------------------+  ||  ||    ||  ||
                        | High Level API     |  ||  ||    ||  ||
                        +--------------------+  ||  ||    ||  ||
                               /\   ||          ||  ||    ||  ||
                               ||   \/          ||  \/    ||  ||
                        +------------------------------+  ||  ||
                        |           MIF API            |  ||  ||
                        +------------------------------+  ||  ||
                           /\  ||                          ||  \/
                           ||  ||   +------------------------------+
                           ||  ||   +      Communications API      +
                           ||  ||   +------------------------------+
                           ||  ||                /\  ||
                           ||  \/                ||  \/
                        +-------------------------------------------+
                        |            Network Link API               |
                        +-------------------------------------------+
                                /\  ||                /\  ||
                                ||  \/                ||  \/
                        +-------------------+  +--------------------+
                        | Network Interface |  | Network Interface  |
                        |         1         |  |          2         |
                        +-------------------+  +--------------------+
```

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 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 amd 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 amd 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, LLMNR, 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

```
                +-------+                                  +-------+
                |  APP  |                                  |  API  |
                +-------+                                  +-------+
                    |          Announce Interfaces                |
                    |--------------------------------------------->|
                    |          Interface 1, eth0                  |
                    |<---------------------------------------------|
                    |          Announce PDs on Interface 1        |
                    |--------------------------------------------->|
                    |          PD 1                               |
                    |<---------------------------------------------|
                    |          Interface 2, wa0                   |
                    |<---------------------------------------------|
                    |          PD 2                               |
                    |<---------------------------------------------|
                    |          Announce PDs on Interface 2        |
                    |--------------------------------------------->|
                    |          PD 3                               |
                    |DNS query 2001::1, host.example.com A,AAAA   |
                    |DNS query 192.168.1.1,host.example.com A,AAAA|
                    |DNS query 2001::1, host.example.com A,AAAA   |
                    |--------------------------------------------->|
                    |14. 2001::1 DNS response:                    |
                    |    host.example.com                         |
                    |    IN A 14.15.16.17                         |
                    |    IN AAAA 2001:192:321::1                  |
                    |                                             |
                    |    2002::1 DNS response:...                 |
                    |    192.168.1.1 DNS response:                |
                    |    IN A 192.168.1.1                         |
                    |<---------------------------------------------|
                    | 15. SYN: 14.15.16.17 @ IF1                  |
                    |     SYN: 2001:192:321::1 @ IF1              |
                    |     SYN: 2001:192:321::1 @ IF2              |
                    |     SYN: 192.168.1.1 @ IF1                  |
                    |--------------------------------------------->|
                    | 16. SYN+ACK @ 192.168.1.1  IF1              |
                    |     SYN+ACK @ 2001:192:321::1  IF2          |
                    |     SYN+ACK @ 2001:192:321::1  IF1          |
                    |<---------------------------------------------|
                    |                                             |
```

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 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

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: December 25, 2014

G. Chen
China Mobile
C. Williams
Consultant
D. Wing
A. Yourtchenko
Cisco Systems, Inc.
June 23, 2014

Happy Eyeballs Extension for Multiple Interfaces
draft-ietf-mif-happy-eyeballs-extension-05

Abstract

Currently the interface selection in multi-interface environment is exclusive - only one interface can be used at the time, frequently needing manual intervention. Happy Eyeballs in MIF would make the selection process smoother by using the connectivity checks over a pre-filtered interfaces according to defined policy. This would choose the fastest interface with an automatic fallback.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 25, 2014.

Table of Contents

1. Introduction

In multiple interface context, the problems raised by hosts with multiple interfaces have been discussed. The MIF problem statement[RFC6418] described the various issues when using a wrong domain selection on a MIF node. Happy Eyeballs (HE) [RFC6555] described how a dual-stack client can determine the functioning path to a dual-stack server. It's using stateful algorithm to help applications quickly determine if IPv6 or IPv4 is the most fast path to connect a server. That is a good method to achieve smart path selection. However, the assumption is a single-homed context. The interaction with multiple interfaces is deferred for further study.

[I-D.ietf-mif-mpvd-arch] has proposed a multiple provisioning domain architecture. This memo has been proposed to extend happy eyeballs algorithm to fit into the multiple interfaces architecture. Several additional considerations have been elaborated to analyze the user demands and initiate HE-MIF connections. It allows a node with multiple interfaces picking a fast flow path.

2. Use Cases

The section enumerates several concrete use cases in existing networks.

Case 1: WiFi is broken

- o [Scenario] A MIF node has both 3/4G and WiFi interface. When the node enters a WiFi area, a common practice would always prefer WiFi because it’ cheap and fast-speed normally.

- o [Problem] User assumes the WiFi is working, because the node already got IP address from WiFi. However, he can’t run applications due to Internet connectivity being unavailable. This may be an authentication required coming into play, or unstable Layer 2 conditions. In order to figure out the problems, users have to turn off the WiFi manually.

- o [Workaround] Users can indicate their desire with some setting on the phone. For instance, they may prefer to wait a little bit of time but not forever. After the timer is expired, users would finally give up the WiFi path and try to establish connection over 3G path. Users may won’t want the wait time too short, because the 3G path for most people is more expensive than WiFi path.

Case 2: VPN (Virtual Private Network) scenario

- o [Scenario] In some cases, a node has multiple interface because of VPN. Users would only have interests to connect a corporate network inside VPN. While, connecting to Internet would work outside the VPN.

- o [Problem] That is normally a implementation consideration that unmanaged interface may be considered less trustworthy than managed. It results in trusted interfaces having the highest priority. This setting may steer all traffic to VPN interface. When this is a traffic heading to a corporate site, everything is fine. But sometimes, the connections out to Internet sites may suffer from long-distance path delays.

- o [Workaround] It’s desirable if routing could be bound to each interface. However, a node following weak host model[RFC1122] takes routing tables as node-scoped. Some sophisticated VPN clients may enable split tunneling to access a public network (e.g. the Internet) and a local LAN simultaneously. VPN may adopt different security policies to applications, for example disable split tunneling to enforce security tunnel only to a particular application. It may be useful to facilitate the implementation by

performing parallel IP connectivity checks before selecting an interface.

Case 3: 3G/LTE tethering scenario

- o [Scenario] Many mobile phones are equipped with software to offer tethered Internet access. It shares their Internet connection with another Internet-capable mobile phone or other devices over Wi-Fi.

- o [Problem] The tethering WiFi link is not free , i.e., it might be 3/4G backhaul. The policy of "always WiFi" leads to all traffic being sent over the tethering WiFi. Usually, such tethering WiFi link puts sharing limitation to access nodes. It could cause contention on both that WiFi link and the backhaul 3G link, while it be higher cost than going on the 3/4G that is built in the handset.

- o [Workaround] To solve that, it is necessary for the node to be aware of not only the link layer information, but also services information, like billable or free. That could help to facilitate the execution of the algorithm. Same concern has been documented in Section 4.4 of [RFC6418])

Case 4: Policy Conflict

- o [Scenario] A node has WiFi and 3/4G access simultaneously. In mobile network, IPv6-only may be preferable since IPv6 has the potential to be simpler than dual-stack. WiFi access still remains on IPv4.

- o [Problem] The problem is caused by policy confliction. The transition to IPv6 is likely to encourage IPv6 and prefer IPv6[RFC6724]. If the 3/4G path has IPv6 on it and the WiFi does not, a suboptimal interface might be chosen from the cost saving perspective.

- o [Workaround] Users interests should be well understood and considered before interface selection. The different preconditions may impact subsequent behaviors. Users concern about high-reliability or high-speed or less-cost should make different choice. A flexible mechanism should be provided allow to make smart decision.

3. Happiness Parameters

This section provides the design proposal for HE-MIF. Two sets of "Happiness" parameters have been defined. It serves upper applications and initiates HE-MIF connections to below level API subsequently. Going through the process, MIF nodes could pick an appropriate interface which would correspond to user demands. The two sets of "Happiness" parameters are called Hard set and Soft set respectively.

- o Hard Set: It contains parameters which have mandatory indications that interface behavior should comply with. This might provide an interface for applications constraints or delivering operator’s policies. Basically, parameters in Hard set should be easy-to-use and easy-to-understand. The users would directly use those. When several hard parameters were conflicted, user’s preference should be prioritized.

  - * User’s preference: users would express preferences which may not have a formally technical language , like "No 3G while roaming", "Only use free WiFi", etc.

  - * Operator policy: operators would deliver the customized policies in particular network environments due to geo-location or services regulation considerations. One example in 3GPP network is that operator could deliver policies from access network discovery and selection function (ANDSF).

- o Soft Set: It contains factors involving in the selection of the fastest path. The following is considered as for the justification.

  - * PVD-ID (Provisioning Domain Identity): PVD-aware node may decide to use one preferred PVD or allow use mulitple PVDs simultanenously for applications. The node behavior should be consistent with MPVD architecture.

  - * Next hop: [RFC4191] allows configuration of specific routes to a destination.

  - * DNS selection: [RFC6731] could configure nodes with information to indicate DNS server address for a particular namespace.

  - * Source address selection: the information provided by [RFC6724] would be considered.

* Other factors: There is a common practice may impact interface selection, e.g. WiFi is preferable. Such conventional experiences should also be considered.

4. HE Behavior in MIF

Corresponding to the two sets of parameters, a HE-MIF node may take a two-steps approach. One is to do "Hard" decision to synthesize policies from different actors (e.g., users and network operator). In a nutshell, that is a filter which will exclude the interfaces from any further consideration. The second is to adjust how we make a connection on multiple interfaces after the filter. It’s sorting behavior. In the multiple provisioning domain architecture, a PVD aware node takes connectivity tests as described in Section 5.3 of [I-D.ietf-mif-mpvd-arch]. A PVD agnostic node take other parameters in the Soft Set to proceed the sort process.

Those two steps are described as following sub-sections. It should be noted that HE-MIF doesn’t prescribe such two-step model. It will be very specific to particular cases and implementations. For example, if one interface on a particular PVD is left after the first step, the process would be ceased.

4.1. First Step, Filter

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

Decision on 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 due to cost saving consideration. Whereas, users may dedicatedly prefer 3/4G interface to seek high-reliability or security benefits even to manually turn off WiFi interface. The decision on mergence of policies may be made by implementations, by node administrators, even by other standards investigating customer behavior. However, it’s worth to note that a demand from users should be normally considered higher priority than from other actors.

The merged policies would serve as a filter principle doing iterate across the list of all known interfaces. Qualified interface would be selected to sort processing at next step.

4.2. Second Step, Sort

Sort process guarantees fast interface selection with fallback capacities. As stated in [I-D.ietf-mif-mpvd-arch], a PVD-aware node shall perform connectivity test and, only after validation of the PVD, consider using it to serve application connections requests. A common practise is to probe a pre-configured URL to check network connectivity status as soon as a node access a network at bootstrap or changes to different networks, e.g. Windows Vista, Windows 7, Windows Server 2008 and iOS. 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 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.

Afterwards, two phases normally are involved in a sort process, i.e. name resolving and connection establishment. Parameters in a soft set should considered at this stage.

When a node initiates name resolution requests, it should check if there is a matched 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 one prior 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 on relevant PVDs in parallel to maximize chances for connectivity. Some additional discussions of DNS selection consideration of HE-MIF are described in Section 6.3.

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 RA message[RFC4191] , the node selects best source address according to the[RFC6724] rules. When destination and source pairs are identified, it should be treated with higher priority compared to others and choose to initiate the connection in advance. 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 6.2), a HE-MIF implementation will decide to initiate connection attempt using rest of interfaces in parallel. This fallback consideration may 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 are no specific Soft Set provided, all selected interfaces should be equally treated. The connections would initiate on several interface simultaneously. The goal here is to provide fast connection for users, by quickly attempting to connect using one of interfaces. Afterwards, the node would do the same caching and flushing process as described above.

5. Implementation Framework

The simplest way for the implementation is within the application itself. The mechanism described in the document would not require any specific support from the operating system beyond the commonly available APIs that provide transport service. It could also be implemented as high-level API approach, linking to MIF-API [I-D.ietf-mif-api-extension]. A number of enhancements could be added, making the use of the high-level APIs much more productive in building applications.

6. Additional Considerations

6.1. Usage Scope

Connection-oriented transports (e.g., TCP, SCTP) could be 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 (e.g., as done by Interactive Connectivity Establishment (ICE) to select a working IPv6 or IPv4 media path[RFC6157])."

6.2. Fallback Timeout

When the preferred interface was failed, HE-MIF would trigger fallback process to start connection initiation on several candidate interfaces. It should set a reasonable wait time to comfort user experience. 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 3/4G would bring a bill 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. Several minutes delay may not 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 due to 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.

6.3. DNS Selections

In 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 on a specific provisioning domain, but DNS resolver may not be aware of that because 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.

By doing HE-MIF, it 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 4.2). The node will be able to automatically rely upon other interfaces to select right DNS servers by excluding the unexamined interfaces.

6.4. Flow Continuity

Interface changing should only happen at the beginning of new session in order to keep flow continuity for ongoing TCP session. Dynamic movement of traffic flows are beyond the scope of this document.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

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

9. 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 and Ted Lemon for their helpful comments.

10. References

10.1. Normative References

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

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

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, April 2012.

[RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

[RFC6731] Savolainen, T., Kato, J., and T. Lemon, "Improved Recursive DNS Server Selection for Multi-Interfaced Nodes", RFC 6731, December 2012.

10.2. Informative 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., "Multiple Provisioning Domain Architecture", draft-ietf-mif-mpvd-arch-01 (work in progress), May 2014.

[RFC5461] Gont, F., "TCP’s Reaction to Soft Errors", RFC 5461, February 2009.

[RFC6157] Camarillo, G., El Malki, K., and V. Gurbani, "IPv6 Transition in the Session Initiation Protocol (SIP)", RFC 6157, April 2011.

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

Authors’ Addresses

Gang Chen
China Mobile
53A,Xibianmennei Ave.,
Xuanwu District,
Beijing 100053
China

Email: phdgang@gmail.com

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

Email: carlw@mcsr-labs.org

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

Email: dwing@cisco.com

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

Email: ayourtch@cisco.com

MIF Working Group
Internet-Draft
Intended status: Informational
Expires: January 02, 2015

D. Anipko, Ed.
Microsoft Corporation

July 03, 2014

Multiple Provisioning Domain Architecture
draft-ietf-mif-mpvd-arch-02

Abstract

This document is a product of the work of MIF architecture design team. It outlines a solution framework for some of the issues, experienced by nodes that can be attached to multiple networks. The framework defines the notion of a Provisioning Domain (PVD) - a consistent set of network configuration information, and PVD-aware nodes - nodes which learn PVDs from the attached network(s) and/or other sources and manage and use multiple PVDs for connectivity separately and consistently.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Nodes attached to multiple networks may encounter problems due to conflict of the networks configuration and/or simultaneous use of the multiple available networks. While existing implementations apply various techniques ([RFC6419]) to tackle such problems, in many cases the issues may still appear. The MIF problem statement document [RFC6418] describes the general landscape as well as discusses many specific issues and scenarios details.

Problems, enumerated in [RFC6418], can be grouped into 3 categories:

1. Lack of consistent and distinctive management of configuration elements, associated with different networks.

2. Inappropriate mixed use of configuration elements, associated with different networks, in the course of a particular network activity / connection.

3. Use of a particular network, not consistent with the intent of the scenario / involved parties, leading to connectivity failure and / or other undesired consequences.

An example of (1) is a single node-scoped list of DNS server IP addresses, learned from different networks, leading to failures or delays in resolution of names from particular namespaces; an example of (2) is use of an attempt to resolve a name of a HTTP proxy server, learned from a network A, with a DNS server, learned from a network B, that is likely to fail; an example of (3) is use of an employer-sponsored VPN connection for peer-to-peer connectivity, unrelated to employment activities.

This architecture describes a solution to these categories of problems, respectively, by:

1. Introducing a formal notion of the PVD, including PVD identity, and ways for nodes to learn the intended associations among acquired network configuration information elements.

2. Introducing a reference model for a PVD-aware node, preventing inadvertent mixed use of the configuration information, which may belong to different PVDs.

3. Providing recommendations on PVD selection based on PVD identity and connectivity tests for common scenarios.

1.1. 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].

2. Definitions and types of PVDs

Provisioning Domain: 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. In some scenarios, it is also possible for elements of the same domain to be present on multiple links.

Typical examples of information in a provisioning domain, learned from the network, are: source address prefixes that can be used by connections within the provisioning domain, IP address of DNS server, name of HTTP proxy server if available, DNS suffixes associated with the network, default gateway address etc.

PVD-aware node: a node that supports association of network configuration information into PVDs, and using these PVDs to serve requests for network connections in ways, consistent with recommendations of this architecture.

2.1. Explicit PVDs

A node may receive explicit information from the network and/or other sources, about presence of PVDs and association of particular network information with a particular PVD. PVDs, constructed based on such information, are referred to in this document as "explicit".

Protocol changes/extensions will likely be required to support the explicit PVDs by IETF-defined mechanisms. As an example, one could think of one or several DHCP options, carrying PVD identity and / or its elements. A different approach could be to introduce a DHCP option, which only carries identity of a PVD, while the association of network information elements with that identity, is implemented by the respective protocols - such as e.g., with a Router Discovery [RFC4861] option associating an address range with a PVD.

Specific, existing or new, features of networking protocols to enable delivery of PVD identity and association with various network information elements will be defined in companion design documents.

Link-specific and/or vendor-proprietary mechanisms for discovery of PVD information, different from the IETF-defined mechanisms, can be used by the nodes separately from or together with IETF-defined mechanisms, as long as they allow to discover necessary elements of the PVD(s). Another example of a delivery mechanism for PVDs are key exchange or tunneling protocols, such as IKEv2 [RFC5996] that allow transporting host configuration information. In all cases, by default nodes must ensure that the lifetime of all dynamically discovered PVD configuration is appropriately limited by the relevant events - for example, if an interface media state change was indicated, the previously discovered information may no longer be valid and needs to be re-discovered or confirmed.

It is expected, that how the node makes use of the PVD information, generally is independent of the specific particular mechanism/ protocol that was used to receive the information.

It shall be possible for sources of PVD information to communicate that some of their configuration elements could be used within a context of other networks/PVDs. PVD-aware nodes, based on such declaration and their policies, may choose to inject such elements into some or all other PVDs they connect to.

In some network topologies, the network infrastructure elements may need to advertise multiple PVDs. The details of how this is done generally will be defined in the individual companion design documents. However, where different design choices are possible, the choice that requires smaller number of packets shall be preferred for efficiency.

2.2. Implicit PVDs and incremental adoption of the explicit PVDs

It is likely that for a long time there may be networks which do not advertise explicit PVD information, since deployment of any new features in networking protocols is a relatively slow process.

When connected to networks, which don’t advertise explicit PVD information, PVD-aware node shall automatically create separate PVDs for received configuration. Such PVDs are referred to in this document as "implicit".

With implicit PVDs, PVD-aware nodes may still provide benefits to their users as compared to non-PVD aware nodes, by using network information from different interfaces separately and consistently to serve network connection requests, following best practices described in Section 5.

In the mixed mode, where e.g., multiple networks are available on the link the interface is attached to, and only some of the networks advertise PVD information, the PVD-aware node shall create explicit PVDs based on explicitly learned PVD information, and associate the rest of the configuration with an implicit PVD created for that interface.

2.3. Relationship between PVDs and interfaces

By default, implicit PVDs are limited to network configuration information received on a single interface. If additional information is available to the host through mechanisms out of scope of this document, the host may form implicit PVDs with different granularity, such as e.g. spanning multiple interfaces (an example scenario, where this may be useful, is a Homenet with a router that has multiple internal interfaces).

Explicit PVDs, in practice will often also be scoped to a configuration related to a particular interface, however per this architecture there is no such requirement or limitation and as defined in this architecture, explicit PVDs may include information related to more than one interfaces, if the node learns presence of the same PVD on those interfaces and the authentication of the PVD ID meets the level required by the node policy (generally, authentication of a PVD ID may be also required in scenarios, involving only one connected interface and/or PVD).

It is an intent of this architecture to support such scenarios among others. Hence, it shall be noted that no hierarchical relationship exists between interfaces and PVDs: it is possible for multiple PVDs to be simultaneously accessible over one interface, as well as single PVD to be simultaneously accessible over multiple interfaces.

2.4. PVD identity/naming

For explicit PVDs, PVD ID (globally unique ID, that possibly is human-readable) is received as part of that information. For implicit PVDs, the node assigns a locally generated with a high probability of being globally unique ID to each implicit PVD.

PVD-aware node may use these IDs to choose a PVD with matching ID for special-purpose connection requests, in accordance with node policy or choice by advanced applications, and/or to present human-readable representation of the IDs to the end-user for selection of Internet-connected PVDs.

A single network provider may operate multiple networks, including networks at different locations. In such cases, the provider may chose whether to advertise single or multiple PVD identities at all

or some of those networks, as it suits their business needs. This architecture doesn’t impose specific requirements in this regard.

When multiple nodes are connected to the same link, where one or more explicit PVDs are available, this architecture assumes that the information about all available PVDs is advertized by the networks to all the connected nodes. At the same time, the connected nodes may have different heuristics, policies and/or other settings, including configured set of their trusted PVDs, which may lead to different PVDs actually being used by different nodes for their connections.

Possible extensions, where different sets of PVDs may be advertised by the networks to different connected nodes, are out of scope of this document.

2.5. Relationship to dual-stack networks

When applied to dual-stack networks, the PVD definition allows for multiple PVDs to be created, where each PVD contain information for only one address family, or for a single PVD that contains information about multiple address families. This architecture requires that accompanying design documents for the PVD-related protocol changes must support PVDs containing information from multiple address families. PVD-aware nodes must be capable of dealing with both single-family and multi-family PVDs.

For explicit PVDs, the choice of either of the approaches is a policy decision of a network administrator and/or node user/administrator. Since some of the IP configuration information that can be learned from the network can be applicable to multiple address families (for instance DHCP address selection option [RFC7078]), it is likely that dual-stack networks will deploy single PVDs for both address families.

For implicit PVDs, by default PVD-aware nodes shall including multiple IP families into single implicit PVD created for an interface. At the time of writing of this document in dual-stack networks it appears to be a common practice for configuration of both address families to be provided by a single source.

A PVD-aware node that provides API to use / enumerate / inspect PVDs and/or their properties shall provide ability to filter PVDs and/or their properties by address family.

2.6. Elements of PVD

3. Conveying PVD information using DHCPv6 and Router Advertisements

DHCPv6 and Router Advertisements are the two most common methods of configuring hosts and they would need to be extended to convey explicit PVD information. There are several things that need to be considered before finalizing a mechanism to augment DHCPv6 and RAs with PvD information.

3.1. Separate messages or one message

When information from several PVDs is available at the same configuration source, there are two possibilities regarding how to send these out. One way is to send information from different provisioning domains in separate messages. The other is to combine information from several PVDs onto one message. The latter method has the advantage of being more efficient but could have issues due to authentication and authorization issues as well as potential issues with accommodating common information and information not tagged with any PVD information.

3.2. Securing the PVD information

DHCPv6 and RAs both provide some form of authentication that ensures the identity of the source as well as the integrity of the contents that have been secured. While this is useful, the authenticity of the information provides no information whether the configuration source is actually allowed to provide information from a given PVD. In order to do be able to do this, there must be a mechanism for the owner of the PVD to attach some form of authorization token to the configuration information that is delivered.

3.3. Backward compatibility

The extensions to RAs and DHCPv6 should be defined in such a manner than unmodified hosts (i.e. hosts not aware of PvDs) will continue to function as well as they did before the PvD information got added. This could imply that some information may need to be duplicated in order to be conveyed to legacy hosts. Similarly PvD aware hosts need to be able to handle legacy configuration sources which do not provide PvD information. There are also several initiatives ongoing that are aimed at adding some form of additional information to prefixes [refs to draft-bhandari and draft-korhonen] and any new mechanism should try to consider co-existence with these existing mechanisms.

3.4. Selective propagation

When a configuration source has information regarding several PvDs it is not clear whether it should provide information about all of them to any host that requests info from it. While it may be reasonable in some cases, this might become an unreasonable burden once the number of PvDs starts increasing. One way to restrict the propagation of useless information is for the host to select the PvD information they desire in their request to the configuration source. One way this could be accomplished is by using an ORO with the PvDs that are of interest. The configuration source can then respond with only the requested information.

By default, a configuration source SHOULD provide information related to all provisioning domains without expecting the client to request the PvD(s) it requires. This is necessary to ensure that hosts that do not support requesting selective PvD information will continue to work. Also note that IPv6 neighbor discovery does not provide any functionality analogous to the DHCPv6 ORO.

In this case, when a host receives PvD information it does not require, the information can simply be discarded. Also, in constrained networks such as LLNs, the amount of configuration information needs to be restricted to ensure that the load on the hosts is bearable while keeping the information identical across all the hosts.

In case selective propogation is required, some form of PvD discovery mechanism needs to be specified so that hosts/applications can be pre-provisioned to request a specific PvD. Alternately, the set of PvDs that the network can provide to the host can be propogated to the host using RAs or stateless DHCPv6. The discovery mechanism may potentially support the discovery of available PvDs on a per-host basis.

3.5. Retracting/updating PvD information

After the PvD information is provided to the host it may be outdated or updated with newer information before the hosts would normally request updates. Thos would require the mchanism to be able to update and/or withdraw all (or some subset) of information related to a given PvD. For efficiency reasons, there should be a way to specify that all the information from the PvD needs to be reconfigured instead of individually updating each item associated with the PvD.

3.6. Conveying configuration information using IKEv2

Internet Key Exchange protocol version 2 (IKEv2) [RFC5996] [RFC5739] is another widely used and a popular method of configuring IP information in a host. In the case of IKEv2 the provisioning domain could actually be implicitly learnt from the Identification - Responder (IDr) payloads the IKEv2 initiator and the responder inject during the IKEv2 exchange. The IP configuration may depend on the named IDr. Another possibility could be adding specific provisioning domain identifying payload extensions to IKEv2. All of the considerations listed above for DHCPv6 and RAs potentialy apply to IKEv2 as well.

4. Example network configurations and number of PVDs

4.1. A mobile node

As an example, consider a mobile node that has two network interfaces: one is a mobile network interface, the other is a Wi-Fi network interface. When the mobile node connects only to the mobile network, it will typically have one PvD, implicit or explicit. Then when the mobile node discovers and connects to a Wi-Fi network, it will have zero or more (typically - one) additional PvD(s).

Some of the existing OS implementations only allow one active network connection. In that case, only the PVD(s) associated with that interface will be connected PvD at any given time.

As an example, the mobile network can explicitly deliver the PvD information through the PDP context activation process. Then the PvD aware mobile node will treat the mobile network as an explicit PvD. Conversely, the legacy Wi-Fi network may not explicitly communicate the PvD information to the mobile node. The PvD aware mobile node will treat the network configuration of the Wi-Fi network as associated with an implicit PvD in this case.

The following diagram illustrates the use of different PvDs in this scenario:

```
                 <------      Wi-Fi 'Internet' PvD  ------>
        +--------+
        | +----+ |    +----+         _   __             _ _
        | |WiFi| |    |    |        ( '    )            ( '   )_
        | |-IF +-|----+    |-------------------------(         ')
        | |    | |    |WiFi|      (         )        (_ Internet  _)
        | +----+ |    | AP |     (           )         '- __  _) -
        |        |    |    |    (  Service    )
        |        |    +----+    (  Provider's   )
        |        |              (   Networks    -
        | +----+ |               '_            )
        | |CELL| |                (          )           ( '   )_
        | |-IF +-|-------------------------------------(         ')
        | |    | |                (_     __)          (_ Internet  _)
        | +----+ |                 '- --               '- __  _) -
        +--------+
                 <------- Mobile 'Internet' PvD ----------->
```

An example PvD use with Wi-Fi and mobile interfaces.

4.2. A node with a VPN connection

If the node has established a VPN connection, zero or more (typically - one) additional PvD(s) will be created. These may be implicit or explicit PvD(s). The routing to the IP addresses within this PvD will be set up via the VPN connection, and the routing to addresses outside the scope of this PvD will remain unaffected. If there were already N connected PvDs on the node prior to establishing VPN

connection, once a VPN session is connected typically the number of PvDs will become N+1.

The following diagram illustrates the use of different PvDs in this scenario:

```
                  <------      'Internet' PvD       ------>
           +--------+
           | +----+ |    +----+         _   __              _ _
           | |Phy | |    |    |        ( '    )            ( '   )_
           | |-IF +-|----+    |-------------------------(          ')
           | |    | |    |    |      (         )        (_ Internet  _)
           | +----+ |    |    |     (           )         '- __  _) -
           |        |    |Home|    (  Service    )            ||
           |        |    |gate|    (  Provider's   )           ||
           |        |    |-way|    (   Network     -           ||
           | +----+ |    |    |    '_            )        +---------+  +--------
----+
           | |VPN | |    |    |      (          )         |   VPN   |  |
    |
           | |-IF +-|----+    |---------------------------+ Gateway |--+  Privat
e   |
           | |    | |    |    |       (_     __)          |         |  |  Servic
es  |
           | +----+ |    +----+         '- --             +---------+  +--------
----+
           +--------+
                  <-------------  Explicit 'VPN' PvD ----------->
```

An example PvD use with VPN.

4.3. A home network and a network operator with multiple PvDs

An operator may use separate PvDs for individual services which they offer to their customers. This may be used so that services can be designed and provisioned to be completely independent of each other allowing for complete flexibility in combinations of services which are offered to customers.

From the perspective of the home network and the node, this model is functionally very similar to being multihomed to multiple upstream operators: Each of the different services offered by the service provider is its own PvD with associated PvD information. In this case, the operator may provide a generic/default PvD (explicit or implicit), which provides Internet access to the customer. Additional services would then be provisioned as explicit PvDs for subscribing customers.

The following diagram illustrates this, using video-on-demand as a service-specific PvD:

```
                     <------ Implicit 'Internet' PvD ------>
                  +----+     +----+         _,  __             _, _
                  |    |     |    |        ( '    )           ( '   )_
                  | PC +-----+    |-------------------------(         ')
                  |    |     |    |      (         )        (_ Internet  _)
                  +----+     |    |     (           )         '- __  _) -
                             |Home|    (  Service    )
                             |gate|    (  Provider's   )
                             |-way|    (   Network     -
                  +-----+    |    |    '_            )        +---------+
                  | Set |    |    |      (          )         |ISP Video|
                  | Top +----+    |---------------------------+on Demand|
                  | Box |    |    |       (_     __)          | Service |
                  +-----+    +----+         '- --             +---------+
                        <-- Explicit 'Video-on-Demand' PvD -->
```

An example PvD use with a homet network.

In this case, the number of PVDs that a single operator could provision is based on the number of independently provisioned services which they offer. Some examples may include:

- o Real-time packet voice
- o Streaming video
- o Interactive video (n-way video conferencing)
- o Interactive gaming
- o Best effort / Internet access

5. Reference model of PVD-aware node

5.1. Constructions and maintenance of separate PVDs

It is assumed that normally, configuration information contained in a single PVD, shall be sufficient for a node to fulfill a network connection request by an application, and hence there should be no need to attempt to merge information across different PVDs.

Nevertheless, even when a PVD lack some parts of the configuration, merging of information from different PVD(s) shall not be done automatically, since typically it would lead to issues described in [RFC6418].

A node may use other sources, such as e.g., node local policy, user input or other mechanisms, not defined by IETF, to either construct a PVD entirely (analogously to static IP configuration of an interface), or supplement with particular elements all or some PVDs learned from the network, or potentially merge information from

different PVDs, if such merge is known to the node to be safe, based on explicit policies.

As an example, node administrator could inject a not ISP-specific DNS server into PVDs for any of the networks the node could become attached to. Such creation / augmentation of PVD(s) could be static or dynamic. The particular implementation mechanisms are outside of the scope of this document.

5.2. Consistent use of PVDs for network connections

PVDs enable PVD-aware nodes to use consistently a correct set of configuration elements to serve the specific network requests from beginning to end. This section describes specific examples of such consistent use.

5.2.1. Name resolution

When PVD-aware node needs to resolve a name of the destination used by a connection request, the node could decide to use one, or multiple PVDs for a given name lookup.

The node shall chose one PVD, if e.g., the node policy required to use a particular PVD for a particular purpose (e.g. to download an MMS using a specific APN over a cellular connection). To make the choice, the node could use a match of the PVD DNS suffix or other form of PVD ID, as determined by the node policy.

The node may pick multiple PVDs, if e.g., they are general purpose PVDs providing connectivity to the Internet, and the node desires to maximize chances for connectivity in Happy Eyeballs style. In this case, the node could do the lookups in parallel, or in sequence. Alternatively, the node may use for the lookup only one PVD, based on the PVD connectivity properties, user choice of the preferred Internet PVD, etc.

In either case, by default the node uses information obtained in a name service lookup to establish connections only within the same PVD from which the lookup results were obtained.

For simplicity, when we say that name service lookup results were obtained from a PVD, what we mean is that the name service query was issued against a name service the configuration of which is present in a particular PVD. In that sense, the results are "from" that particular PVD.

Some nodes may support transports and/or APIs, which provide an abstraction of a single connection, aggregating multiple underlying connections. MPTCP [RFC6182] is an example of such transport protocol. For the connections provided by such transports/APIs, a PVD-aware node may use different PVDs for servicing of that logical

connection, provided that all operations on the underlying connections are done consistently within their corresponding PVD(s).

5.2.2. Next-hop and source address selection

For the purpose of this discussion, let’s assume the preceding name lookup succeeded in a particular PVD. For each obtained destination address, the node shall perform a next-hop lookup among routers, associated with that PVD. As an example, such association could be determined by the node via matching the source address prefixes/ specific routes advertized by the router against known PVDs, or receiving explicit PVD affiliation advertized through a new Router Discovery [RFC4861] option.

For each destination, once the best next-hop is found, the node selects best source address according to the [RFC6724] rules, but with a constraint that the source address must belong to a range associated with the used PVD. If needed, the node would use the prefix policy from the same PVD for the best source address selection among multiple candidates.

When destination/source pairs are identified, then they are sorted using the [RFC6724] destination sorting rules and the prefix policy table from the used PVD.

5.2.3. Listening applications

Consider a host, connected to several PVDs and running an application that opens a listening socket/transport API object, where the application authorized by the host policy to use a subset connected PVDs, that may or may not be equal to the complete set of the connected PVDs. For example, in case there are different PVDs on a Wi-Fi and a cellular interfaces, for general internet traffic the host could decide to use only one preferred PVD at a time (and accordingly, advertise to remote peers the host name and addresses associated with that PVD), or it could decide to use one PVD as a preferred one by default for outgoing connections, while still allowing use of the other PVDs simultaneously. Another example is where a host established a VPN connection. Depending on the security policies provisioned on the host, all or some applications may or may not be allowed to use the VPN PVD and/or other PVDs.

For non-PVD aware applications, the OS policies determine the authorized set of PVDs and the preferred outgoing PVD. For PVD-aware applications, both the authorized set of PVDs and the default outgoing PVD can be determined as a meet of the subset produced by the OS policies and the set of PVD IDs or characteristics, provided

by the application. The application input could be provided on per-application, per-transport-API-object or per-transport-API-call basis. The API for application input may have an option for to specify whether the input should be treated as a preference instead of a requirement.

5.2.3.1. Processing of incoming traffic

Unicast IP packets are received on a specific IP address, associated with a PVD. For multicast packets, the host can derive the association with a PVD from other configuration information, such as an explicit PVD property or local policy.

The node OS or middleware may apply more advanced techniques for determination of the resultant PVD and/or authorization of the incoming traffic. Those techniques are outside of scope of this document.

If the determined receiving PVD of the packet is not in the allowed subset of PVDs for the particular app/transport API object, the packet should be handled in the same way as if there were no listener.

5.2.3.1.1. Connection-oriented APIs

For connection-oriented APIs, when the initial incoming packet is received, the packet PVD is remembered for the established connection, and used for handling of the outgoing traffic for that connection. While typically the connection-oriented APIs use connection-oriented transport protocol, such as TCP, it is possible to have a connection-oriented API, which uses generally connection-less transport protocol, such as UDP. For APIs/protocols, which support multiple IP traffic flows associated with a single transport API connection object (such as e.g. multi path TCP), the processing rules may be adjusted accordingly.

5.2.3.1.2. Connection-less APIs

For connection-less APIs, the host should provide an API, which PVD-aware applications could use to query the PVD associated with the packet. For outgoing traffic on this transport API object, the OS should use the selected outgoing PVDs, determined as described above.

5.2.4. Enforcement of security policies

PVDs by themselves don’t define and can’t be used for communication of security policies. When implemented in a network, this architecture provides the host with information about the connected networks. The actual behavior of the host then depends on the host policies (provisioned through mechanisms out of scope of this document), applied taking received PVD information into account. In some scenarios, such as e.g . VPN, such policies could require the host to use only a particular VPN PVD for some/all of the applications’ traffic (VPN ’disable split tunneling’ also known as ’force tunneling’ behavior), or apply such restrictions only to select applications and allow simultaneous use of the VPN PVD togetheer with the other connected PVDs by the other or all applications (VPN ’split tunneling’ behavior).

5.3. Connectivity tests

Although some PVDs may appear as valid candidates for PVD selection (e.g. good link quality, consistent connection parameters, etc.), they may provide limited or no connectivity to the desired network or the Internet. For example, some PVDs provide limited IP connectivity (e.g., scoped to the link or to the access network), but require the node to authenticate through a web portal to get full access to the Internet. This may be more likely to happen for PVDs, which are not trusted by the given PVD-aware node.

An attempt to use such PVD may lead to limited network connectivity or connection failures for applications. To prevent the latter, a PVD-aware node may perform connectivity test for the PVD, before using it to serve network connection requests of the applications. In current implementations, some nodes do that, for instance, by trying to reach a dedicated web server (e.g., see [RFC6419]).

Per Section 5.2, a PVD-aware node shall maintain and use multiple PVDs separately. The PVD-aware node shall perform connectivity test and, only after validation of the PVD, consider using it to serve application connections requests. Ongoing connectivity tests are also required, since during the IP session, the end-to-end connectivity could be disrupted for various reasons (e.g. poor L2, IP QoS issues); hence a connectivity monitoring function is needed to check the connectivity status and remove the PVD from the set of usable PVDs if necessary.

There may be cases where a connectivity test for PVD selection may be not appropriate and should be complemented, or replaced, by PVD selection based on other factors. This could be realized e.g., by leveraging some 3GPP and IEEE mechanisms, which would allow to expose some PVD characteristics to the node (e.g. 3GPP Access Network Discovery and Selection Function (ANDSF) [TS23.402], IEEE 802.11u [IEEE802.11u]/ANQP).

5.4. Relationship to interface management and connection managers

Current devices such as mobile handsets make use of proprietary mechanisms and custom applications to manage connectivity in environments with multiple interfaces and multiple sets of network configurations. These mechanisms or applications are commonly known as connection managers [RFC6419].

Connection managers sometimes rely on policy servers to allow the node, connected to multiple networks, perform the network selection. They can also make use of routing guidance from the network (e.g. 3GPP ANDSF [TS23.402]). Although connection managers solve some connectivity problems, they rarely address the network selection problems in a comprehensive manner. With proprietary solutions, it is challenging to present a coherent behaviour to the end user of the device, as different platforms present different behaviours even when connected to the same network, with the same type of interface, and for the same purpose.

6. PVD support in APIs

In all cases changes in available PVDs must be somehow exposed, appropriately for each of the approaches.

6.1. Basic

Applications are not PVD-aware in any manner, and only submit connection requests. The node performs PVD selection implicitly, without any otherwise applications participation, and based purely on node-specific administrative policies and/or choices made by the user in a user interface provided by the operating environment, not by the application.

As an example, such PVD selection can be done at the name service lookup step, by using the relevant configuration elements, such as e.g., those described in [RFC6731]. As another example, the PVD selection could be done based on application identity or type (i.e., a node could always use a particular PVD for a VOIP application).

6.2. Intermediate

Applications indirectly participate in selection of PVD by specifying hard requirements and soft preferences. As an example, a real time communication application, intending to use the connection for exchange of real time audio/video data, may indicate a preference or a requirement for connection quality, which could affect PVD selection (different PVDs could correspond to Internet connections with different loss rates and latencies). Another example is a connection of an infrequently executed background activity, which checks for availability of applications updates and performs large downloads - for such connections, a cheaper or zero cost PVD may be preferrable, even if such connection will have a higher relative loss rate or lower bandwidth. The node performs PVD selection, based on applications inputs and policies and/or user preferences. Some / all properties of the resultant PVD may be exposed to applications.

6.3. Advanced

PVDs are directly exposed to applications, for enumeration and selection. Node polices and/or user choices, may still override the application preferences and limit which PVD(s) can be enumerated and/or used by the application, irrespectively of any preferences which application may have specified. Depending on the implementation, such restrictions, imposed per node policy and/or user choice, may or may not be visible to the application.

7. PVD-aware nodes trust to PVDs

7.1. Untrusted PVDs

Implicit and explicit PVDs for which no trust relationship exists are considered untrusted. Only PVDs, which meet the requirements in Section 7.2, are trusted; any other PVD is untrusted.

In order to avoid various forms of misinformation that can be asserted when PVDs are untrusted, nodes that implement PVD separation cannot assume that two explicit PVDs with the same identifier are actually the same PVD. A node that did make this assumption would be vulnerable to attacks where for example an open Wifi hotspot might assert that it was part of another PVD, and thereby might draw traffic intended for that PVD onto its own network.

Since implicit PVD identifiers are synthesized by the node, this issue cannot arise with implicit PVDs.

Mechanisms exist (for example, [RFC6731]) whereby a PVD can provide configuration information that asserts special knowledge about the reachability of resources through that PVD. Such assertions cannot be validated unless the node has a trust relationship with the PVD; assertions of this type therefore must be ignored by nodes that receive them from untrusted PVDs. Failure to ignore such assertions could result in traffic being diverted from legitimate destinations to spoofed destinations.

7.2. Trusted PVDs

Trusted PVDs are PVDs for which two conditions apply. First, a trust relationship must exist between the node that is using the PVD configuration and the source that provided that configuration; this is the authorization portion of the trust relationship. Second, there must be some way to validate the trust relationship. This is the authentication portion of the trust relationship. Two mechanisms for validating the trust relationship are defined.

It shall be possible to validate the trust relationship for all advertised elements of a trusted PVD, irrespectively of whether the PVD elements are communicated as a whole, e.g. in a single DHCP option, or separately, e.g. in supplementary RA options. Whether or not this is feasible to provide mechanisms to implement trust relationship for all PVD elements, will be determined in the respective companion design documents.

7.2.1. Authenticated PVDs

One way to validate the trust relationship between a node and the source of a PVD is through the combination of cryptographic authentication and an identifier configured on the node. In some cases, the two could be the same; for example, if authentication is done with a shared secret, the secret would have to be associated with the PVD identifier. Without a (PVD Identifier, shared key) tuple, authentication would be impossible, and hence authentication and authorization are combined.

However, if authentication is done using some public key mechanism such as a TLS cert or DANE, authentication by itself isn’t enough, since theoretically any PVD could be authenticated in this way. In addition to authentication, the node would need to be configured to trust the identifier being authenticated. Validating the authenticated PVD name against a list of PVD names configured as trusted on the node would constitute the authorization step in this case.

7.2.2. PVDs trusted by attachment

In some cases a trust relationship may be validated by some means other than described in Section 7.2.1, simply by virtue of the connection through which the PVD was obtained. For instance, a handset connected to a mobile network may know through the mobile network infrastructure that it is connected to a trusted PVD, and whatever mechanism was used to validate that connection constitutes the authentication portion of the PVD trust relationship. Presumably such a handset would be configured from the factory, or else through mobile operator or user preference settings, to trust the PVD, and this would constitute the authorization portion of this type of trust relationship.

8. Acknowledgements

This document was created as a product of a MIF architecture design team and includes contributions from the MIF working group participants.

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

There are at least three different form of attacks that can be performed using configuration sources that use multiple provisioning domains.

Tampering with configuration information provided An attacker may attempt to modify the information provided inside the PVD container option. These attacks can easily be prevented by using the message integrity features provided by the underlying protocol used to carry the configuration information. e.g. SEND [RFC3971] would detect any form of tampering with the RA contents and the DHCPv6 [RFC3315] AUTH option that would detect any form of tampering with the DHCPv6 message contents. This attack can also be performed by a compromised configuration source by modifying information inside a specific , in which case the mitigations proposed in the next subsection may be helpful.

Rogue configuration source A compromised configuration source such as a router or a DHCPv6 server may advertise information about PvDs that it is not authorized to advertise. e.g. A coffee shop may advertise configuration information purporting to be from an enterprise and may try to attract enterprise related traffic. The only real way to avoid this is that the PvD related configuration container contains embedded authentication and authorization information from the owner of the PvD. Then, this attack can be detected by the client by verifying the authentication and authorization information provided inside the PVD container option after verifying its trust towards the PvD owner (e.g. a certificate with a well-known/common trust anchor).

Replay attacks A compromised configuration source or an on-link attacker may try to capture advertised configuration information and replay it on a different link or at a future point in time. This can be avoided by including some replay protection mechanism such as a timestamp or a nonce inside the PvD container to ensure freshness of the provided information.

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.

11.2. Informative References

[IEEE802.11u] IEEE, "IEEE Standard 802.11u-2011 (Amendment 9: Interworking with External Networks)", 2011.

[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.

[RFC3971] Arkko, J., Kempf, J., Zill, B. and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

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

[RFC5739] Eronen, P., Laganier, J. and C. Madson, "IPv6 Configuration in Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5739, February 2010.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y. and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

[RFC6182] Ford, A., Raiciu, C., Handley, M., Barre, S. and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC 6182, March 2011.

[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.

[RFC6724] Thaler, D., Draves, R., Matsumoto, A. and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

[RFC6731] Savolainen, T., Kato, J. and T. Lemon, "Improved Recursive DNS Server Selection for Multi-Interfaced Nodes", RFC 6731, December 2012.

[RFC7078] Matsumoto, A., Fujisaki, T. and T. Chown, "Distributing Address Selection Policy Using DHCPv6", RFC 7078, January 2014.

[TS23.402] 3GPP, "3GPP TS 23.402; Architecture enhancements for non-3GPP accesses; release 12", .

Author’s Address

Dmitry Anipko, editor
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
USA

Phone: +1 425 703 7070
Email: dmitry.anipko@microsoft.com

IPv6 Maintenance
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2015

J. Korhonen
Broadcom
S. Krishnan
Ericsson
S. Gundavelli
Cisco Systems
July 4, 2014

Support for multiple provisioning domains in IPv6 Neighbor Discovery Protocol
draft-kk-mpvd-ndp-support-02

Abstract

The MIF working group is producing a solution to solve the issues that are associated with nodes that can be attached to multiple networks. One part of the solution requires associating configuration information with provisioning domains. This document details how configuration information provided through IPv6 Neighbor Discovery Protocol can be associated with provisioning domains.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The MIF working group is producing a solution to solve the issues that are associated with nodes that can be attached to multiple networks based on the Multiple Provisioning Domains (MPVD) architecture work [I-D.ietf-mif-mpvd-arch]. One part of the solution requires associating configuration information with Provisioning Domains (PVD). This document describes an IPv6 Neighbor Discovery Protocol (NDP) [RFC4861] mechanism for explicitly indicating provisioning domain information along with any configuration that will be provided. The proposed mechanism uses an NDP option that indicates the identity of the provisioning domain and encapsulates the options that contain the configuration information as well as any accompanying authentication/authorization information. The solution defined in this document aligns as much as possible with the existing IPv6 Neighbor Discovery security, namely with Secure Neighbor Discovery (SeND) [RFC3971].

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. PVD Container option

The PVD container option (PVD_CO) is used to mark the start of the configuration options that belong to the explicitly identified provisioning domain. The PVD container option MUST encapsulate exactly one PVD identifier option (PVD_ID, see Section 4). The PVD container option MAY occur multiple times in the same NDP message but each of these PVD container options MUST have a different PVD identity specified under its PVD identity option. The PVD container options MUST NOT be nested.

A PVD container is intended to be used in IPv6 Router Advertisement (RA) NDP messages. However, including a PVD container or identity options inside a Router Solicitation (RS) NDP messages is also possible (actually, in this way a host can solicit for information from a specific provisioning domain). The PVD container option MUST NOT be included in a NDP message without accompanying PVD identity option (see Section 4). If, for some reason, the NDP message does not include the accompanying PVD identity option, then the implementation MUST ignore the PVD container option and SHOULD log the event. The PVD container MUST NOT be fragmented i.e., should the IPv6 packet be fragmented, the PVD container and the accompanying PVD

identity MUST both be inside the same fragment.

Since implementations are required to ignore any unrecognized options [RFC4861], the backward compatibility and the reuse of existing NDP options is implicitly enabled. Implementations that do not recognize the PVD container option plain ignore it and also skip PVD container option "encapsulated" NDP options normally without associating them into any provisioning domain (since the implementation has no notion of provisioning domains). For example, the PVD container could "encapsulate" a Prefix Information Option (PIO), which would mark that this certain advertised IPv6 prefix belongs and originates from a specific provisioning domain. However, if the implementation does not understand provisioning domains, then this specific PIO is also skipped and not configured to the interface.

The optional security for the PVD container is based on X.509 certificates [RFC6487] and reuses mechanisms already defined for SeND [RFC3971] [RFC6495]. However, the use of PVD containers does not assume or depend on SeND being deployed or even implemented. The PVD containers SHOULD be signed per PVD certificates, which provides both integrity protection and proves that the configuration information source is authorized for advertising the given information. See [RFC6494] for discussion how to enable deployments where the certificates needed to sign PVD containers) belong to different administrative domains i.e. to different provisioning domains.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type=PVD_CO  |    Length     |S|  Reserved   |   Name Type   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
:                      Key Hash (optional)                      :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
:                 Digital Signature (optional)                  :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Possible zero padding to ensure 8 octets alignment      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: PVD Container Option

Type

PVD Container; Set to TBD1.

Length

Length of the PVD_CO. The actual length depends on the number of "encapsulated" NDP options, the length of the PVD identifier option and the optional Key Hash/Digital Signature/Padding.

S

Security enabled/disabled flag. If S=0 then security (signing) of the PVD_CO is disabled. If S=1 then security (signing) is enabled.

Name Type

Names the algorithm used to identify a specific X.509 certificate using the method defined for the Subject Key Identifier (SKI) extension for the X.509 certificates. The usage and the Name Type registry aligns with the mechanism defined for SeND [RFC6495]. Name Type values starting from 3 are supported and an implementation MUST at least support SHA-1 (value 3). Note that if S=0 the Name field serves no use.

Key Hash

This field is only present when S=1. A hash of the public key using the algorithm identified by the Name Type. The procedure how the Key Hash is calculated is defined in [RFC3971] and [RFC6495].

Digital Signature

This field is only present when S=1. A signature calculated over the PVD_CO option including all option data from the beginning of the option until to the end of the container. The procedure of calculating the signature is identical to the one defined for SeND [RFC3971]. During the signature calculation the contents of the Digital Signature option MUST be treated as all zero.

Implementations MUST ensure that the PVD container option meets the 8 octets NDP option alignment requirement. This MAY imply adding padding zero octets to the tail of the PVD container option until the alignment requirement has been met. The padding is independent of the ’S’ flag setting.

If the PVD_CO does not contain a digital signature, then other means to secure the integrity of the NDP message SHOULD be provided, such as utilizing SeND. However, the security provided by SeND is for the entire NDP message and does not allow verifying whether the sender of the NDP message is actually authorized for the information for the provisioning domain.

If the PVD_CO contains a signature and the verification fails, then the whole PVD_CO, PVD_ID and other NDP options MUST be silently ignored and the event SHOULD be logged.

4. PVD Identity option

The PVD identity option (PVD_ID) is used to explicitly indicate the identity of the provisioning domain that is associated with the configuration information encapsulated by the PVD container option. A PVD container option MUST have exactly one PVD identity option. However, the PVD identity option MAY also be included in a NDP message without the PVD container option. In this case it merely serves as a hint of provisioning domain and could, for example, be used in an RS message to solicit information from specific provisioning domains.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type=PVD_ID  |    Length     | Identity                      ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: PVD_ID Option

Type

PVD identifier; Set to TBD2.

Length

Length of the PVD_ID.

Identity

The provisioning domain identity. The contents of this field is defined in a separate document [I-D.kkbg-mpvd-id]. Note that the Identity field may need to be zero padded at the tail to meets the natural NDP options' alignment.

If the receiver of the PVD identity option does not understand any of

the ID-Types, then anything belonging to this provisioning domain MUST be silently discarded. This would mean the PVD identity option, the PVD container option and all other options.

5. Set of allowable options

The PVD container option MAY be used to encapsulate any allocated IPv6 NDP options, which may appear more than once in a NDP message. The PVD container option MUST NOT be used to encapsulate other PVD_CO option(s).

6. Security Considerations

An attacker may attempt to modify the information provided inside the PVD container option. These attacks can easily be prevented by using SeND [RFC3971] or per PVD container signature that would detect any form of tampering with the IPv6 NDP message contents.

A compromised router may advertise configuration information related to provisioning domains it is not authorized to advertise. e.g. A coffee shop router may provide configuration information purporting to be from an enterprise and may try to attract enterprise related traffic. The only real way to avoid this is that the provisioning domain container contains embedded authentication and authorization information from the owner of the provisioning domain. Then, this attack can be detected by the client by verifying the authentication and authorization information provided inside the PVD container option after verifying its trust towards the provisioning domain owner (e.g. a certificate with a well-known/common trust anchor).

A compromised configuration source or an on-link attacker may try to capture advertised configuration information and replay it on a different link or at a future point in time. This can be avoided by including some replay protection mechanism such as a timestamp or a nonce inside the PVD container to ensure freshness of the provided information. This specification does not define a replay protection solution. Rather it is assumed that if replay protection is required, the access network and hosts also deploy existing security solutions such as SeND [RFC3971].

7. IANA Considerations

This document defines two new IPv6 NDP options into the "IPv6 Neighbor Discovery Option Formats" registry. The options TBD1 and TBD2 are described in Section 3 and Section 4.

8. Acknowledgements

The authors would like to thank the members of the MIF architecture design team for their comments that led to the creation of this draft.

9. References

9.1. Normative References

[I-D.kkbg-mpvd-id]
Krishnan, S., Korhonen, J., Bhandari, S., and S. Gundavelli, "Identification of provisioning domains", draft-kkbg-mpvd-id-00 (work in progress), February 2014.

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

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

[RFC6494] Gagliano, R., Krishnan, S., and A. Kukec, "Certificate Profile and Certificate Management for SEcure Neighbor Discovery (SEND)", RFC 6494, February 2012.

[RFC6495] Gagliano, R., Krishnan, S., and A. Kukec, "Subject Key Identifier (SKI) SEcure Neighbor Discovery (SEND) Name Type Fields", RFC 6495, February 2012.

9.2. Informative References

[I-D.ietf-mif-mpvd-arch]
Anipko, D., "Multiple Provisioning Domain Architecture", draft-ietf-mif-mpvd-arch-01 (work in progress), May 2014.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, February 2012.

Appendix A. Examples

A.1. One implicit PVD and one explicit PVD

Figure 3 shows how the NDP options are laid out in an RA for one implicit provisioning domain and one explicit provisioning domain. The example does not include security (and signing of the PVD container). The assumption is the PVD identity consumes 14 octets.

The explicit provisioning domain ("starducks.example.com" in a NAI Realm format) contains a specific PIO for 2001:db8:abad:cafe::/64 and the MTU of 1337 octets. The implicit provisioning domain configures a prefix 2001:db8:cafe:babe::/64 and the link MTU of 1500 octets. There are two cases: 1) the host receiving the RA implements provisioning domains and 2) the host does not understand provisioning domains.

1. The host recognizes the PVD_CO and "starts" a provisioning domain specific configuration. Security is disabled, thus there are no Key Hash or Digital Signature fields to process. The prefix 2001:db8:abad:cafe::/64 is found and configured on the interface. Once the PVD_ID option is located the interface prefix configuration for 2001:db8:abad:cafe::/64 and the MTU of 1337 octets can be associated to the provisioning domain found in the PVD_ID option.

   The rest of the options are parsed and configured into the implicit provisioning domain since there is no encapsulating provisioning domain. The interface is configured with prefix 2001:db8:cafe:babe::/64. The implicit provisioning domain uses the link MTU of 1500 octets, whereas the "starducks.example.com" provisioning domain uses the MTU of 1337 octets (this means when packets are sourced using 2001:db8:abad:cafe::/64 prefix the link MTU is different than when sourcing packets using 2001:db8:cafe: babe::/64 prefix).

2. The host ignores the PVD_CO (including the PVD_ID and other options) and ends up configuring one prefix on its interface (2001:db8:cafe:babe::/64) with a link MTU of 1500 octets.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      134      |       0       |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Cur Hop Limit |0|1|  Reserved |       Router Lifetime         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Reachable Time                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Retrans Timer                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ <+
|  Type=PVD_CO  |      10       |0|  Reserved   |      0        |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                               0                               |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|       3       |       4       |      64       |1|1| Reserved1 |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Valid Lifetime                        |  P
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  V
|                       Preferred Lifetime                      |  D
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Reserved2                           |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                      2001:db8:abad:cafe::                     ~  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|  Type=PVD_ID  |       4       |   id-type=4   |       21      |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
~  "starducks.example.com",'\0','\0','\0','\0','\0','\0','\0'   |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|        5      |      1        |           Reserved            |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                            1337                               |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ <+
|       3       |       4       | Prefix Length |1|1| Reserved1 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Valid Lifetime                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Preferred Lifetime                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Reserved2                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      2001:db8:cafe:babe::                     ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        5      |      1        |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            1500                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: An RA with one implicit PVD and one explicit PVD

Authors' Addresses

Jouni Korhonen
Broadcom
Porkkalankatu 24
FIN-00180 Helsinki
Finland

Email: jouni.nospam@gmail.com

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

Phone: +1 514 345 7900 x42871
Email: suresh.krishnan@ericsson.com

Sri Gundavelli
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sgundave@cisco.com

DHC Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 17, 2014

S. Krishnan
Ericsson
J. Korhonen
Broadcom
S. Bhandari
Cisco Systems

February 13, 2014

Support for multiple provisioning domains in DHCPv6
draft-kkb-mpvd-dhcp-support-01

Abstract

The MIF working group is producing a solution to solve the issues that are associated with nodes that can be attached to multiple networks. One part of the solution requires associating configuration information with provisioning domains. This document details how configuration information provided through DHCPv6 can be associated with provisioning domains.

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 17, 2014.

Table of Contents

1. Introduction

The MIF working group is producing a solution to solve the issues that are associated with nodes that can be attached to multiple networks based on the Multiple Provisioning Domains (MPVD) architecture work [I-D.anipko-mif-mpvd-arch]. One part of the solution requires associating configuration information with provisioning domains. This document describes a DHCPv6 mechanism for explicitly indicating provisioning domain information along with any configuration that will be provided. The proposed mechanism uses a DHCPv6 option that indicates the identity of the provisioning domain and encapsulates the options that contain the configuration information as well as any accompanying authentication/authorization information.

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. PVD Container option

The PVD container option is used to encapsulate and group together all the configuration options that belong to the explicitly identified provisioning domain. The PVD container option MUST encapsulate exactly one OPTION_PVD_ID. The PVD container option MAY occur multiple times in the same message, but each of these PVD container options MUST have a different PVD identity specified under its PVD identity option. The PVD container option SHOULD contain exactly one OPTION_PVD_AUTH.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        OPTION_PVD             |         option-length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+            encapsulated-options (variable length)             .
.                                                               .
+---------------------------------------------------------------+
```

Figure 1: PVD Container Option

- o option-code: OPTION_PVD (TBA1)

- o option-length: Length of encapsulated options

- o encapsulated-options: options associated with this provisioning domain.

4. PVD Identity option

The PVD identity option is used to explicitly indicate the identity of the provisioning domain that is associated with the configuration information encapsulated by the PVD container option.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       OPTION_PVD_ID           |         option-length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    PVD identity information                   |
+                       (variable length)                       +
+                                                               +
.                                                               .
+---------------------------------------------------------------+
```

Figure 2: PVD ID Option

- o option-code: OPTION_PVD_ID (TBA2)

- o option-length: Length of PVD identity information

- o PVD identity information: The provisioning domain identity. The contents of this field is defined in a separate document [PVDIDS].

5. PVD Authentication and Authorization option

The PVD authentication and authorization option contains information that could be used by the DHCPv6 client to verify whether the configuration information provided was not tampered with by the DHCPv6 server as well as establishing that the DHCPv6 server was authorized to advertise the information on behalf of the PVD per OPTION_PVD basis. The contents of the authentication/authorization information is provided by the owner of the provisioning domain and is completely opaque to the DHCPv6 server that passes along the information unmodified. Every OPTION_PVD option SHOULD contain at

most one OPTION_PVD_AUTH option. The OPTION_PVD_AUTH option MUST be the last option inside the OPTION_PVD option.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      OPTION_PVD_AUTH          |         option-length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ <+
|   name-type   |               key-hash                        :  |
+-+-+-+-+-+-+-+-+                                               :  |
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Auth
:                                                               : info
:                      digital-signature                        :  |
:                                                               :  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ <+
```

Figure 3: PVD Auth Option

- o option-code: OPTION_PVD_AUTH (TBA3)

- o option-length: Length of the Auth info

- o name-type: Names the algorithm used to identify a specific X.509 certificate using the method defined for the Subject Key Identifier (SKI) extension for the X.509 certificates. The usage and the Name Type registry aligns with the mechanism defined for SeND [RFC6494][RFC6495]. Name Type values starting from 3 are supported and an implementation MUST at least support SHA-1 (value 3).

- o key-hash: A hash of the public key using the algorithm identified by the Name Type. The procedure how the Key Hash is calculated is defined in [RFC3971] and [RFC6495]

- o digital-signature: A signature calculated over the encapsulating OPTION_PVD including all option data from the beginning of the option while setting the digital-signature field to zero. The procedure of calculating the signature is identical to the one defined for SeND [RFC3971].

  [TODO: There may be some alignment considerations here for some implementations as DHCPv6 options are not aligned.]

6. Set of allowable options

The PVD container option MAY be used to encapsulate any allocated DHCPv6 options but MUST NOT be used to encapsulate another OPTION_PVD option. [TODO: Should we add any other exclusions?]

7. Behaviour of DHCPv6 entities

This section describes role of DHCPv6 entities involved in requesting and receiving DHCPv6 configuration or prefix and address allocation.

7.1. Client and Requesting Router Behavior

DHCPv6 client or requesting router can request for configuration from provisioning domain in the following ways:

- o In the SOLICIT message it MAY include OPTION_PVD_ID requesting configuration for the specific PVD ID indicated in the OPTION_PVD_ID option. It can include multiple OPTION_PVD_ID options to indicate its preference for more than one provisioning domain. The PVD ID it requests is learnt via configuration or any other out of band mechanism not defined in this document.
- o In the SOLICIT message include an OPTION_ORO option with the OPTION_PVD option code to request configuration from all the PVDs that the DHCPv6 server can provide.

The client or requesting router parses OPTION_PVD options in the response message. The Client or Requesting router MUST then include all or subset of the received OPTION_PVD options in the REQUEST message so that it will be responsible for the configuration information selected.

If DHCPv6 client or requesting router receives OPTION_PVD options but does not support PVD, it SHOULD ignore the received option(s).

7.2. Server and Delegating Router Behavior

If the Server or Delegating router supports PVD and it is configured to provide configuration data in one or more provisioning domains, it selects configuration for the PVD based allocation in the following way:

- o If OPTION_PVD option code within OPTION_ORO is not present in the request, it MUST NOT include provisioning domain based configuration. It MAY select configuration and prefix allocation from a default PVD defined.

- o If OPTION_PVD_ID is included, it selects information to be offered from that specific PVD if available.
- o If OPTION_PVD option code within OPTION_ORO is included, then based on its configuration and policy it MAY offer configuration from the available PVD(s).

When PVD information and configuration are selected for address and prefix allocation the server or delegating router responds with an ADVERTISE message after populating OPTION_PVD.

If OPTION_PVD is not included, then the server or delegating router MAY allocate the prefix and provide configuration as specified in [RFC3315] and[RFC3633] and MUST NOT include OPTION_PVD option in the response.

If OPTION_ORO option includes the OPTION_PVD option code but the server or delegating router does not support PVD, then it SHOULD ignore the OPTION_PVD and OPTION_PVD_ID options received.

If both client/requesting router and server/delegating router support PVD but cannot offer configuration with PVD for any other reason, it MUST respond to client/requesting router with appropriate status code as specified in [RFC3315] and [RFC3633].

8. Security Considerations

An attacker may attempt to modify the information provided inside the PVD container option. These attacks can easily be prevented by using the DHCPv6 AUTH option [RFC3315] that would detect any form of tampering with the DHCPv6 message contents.

A compromised DHCPv6 server or relay agent may insert configuration information related to PvDs it is not authorized to advertise. e.g. A coffee shop DHCPv6 server may provide configuration information purporting to be from an enterprise and may try to attract enterprise related traffic. The only real way to avoid this is that the PvD container contains embedded authentication and authorization information from the owner of the PvD. Then, this attack can be detected by the client by verifying the authentication and authorization information provided inside the PVD container option after verifying its trust towards the PvD owner (e.g. a certificate with a well-known/common trust anchor).

A compromised configuration source or an on-link attacker may try to capture advertised configuration information and replay it on a different link or at a future point in time. This can be avoided by including some replay protection mechanism such as a timestamp or a nonce inside the PvD container to ensure freshness of the provided

information.

9. IANA Considerations

This document defines three new DHCPv6 options to be allocated out of the registry at http://www.iana.org/assignments/dhcpv6-parameters/

OPTION_PVD (TBA1)
OPTION_PVD_ID (TBA2)
OPTION_PVD_AUTH (TBA3)

10. Acknowledgements

The authors would like to thank the members of the MIF architecture design team for their comments that led to the creation of this draft.

11. Normative References

[I-D.anipko-mif-mpvd-arch] Anipko, D., "Multiple Provisioning Domain Architecture", draft-anipko-mif-mpvd-arch-05 (work in progress), November 2013.

[PVDIDS] Krishnan, S., Korhonen, J., Bhandari, S., and S. Gundavelli, "Identification of provisioning domains", draft-kkbg-mpvd-id-00 (work in progress), February 2014.

[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.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

[RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.

[RFC6494] Gagliano, R., Krishnan, S., and A. Kukec, "Certificate Profile and Certificate Management for SEcure Neighbor Discovery (SEND)", RFC 6494, February 2012.

[RFC6495] Gagliano, R., Krishnan, S., and A. Kukec, "Subject Key Identifier (SKI) SEcure Neighbor Discovery (SEND) Name Type Fields", RFC 6495, February 2012.

Authors' Addresses

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

Phone: +1 514 345 7900 x42871
Email: suresh.krishnan@ericsson.com

Jouni Korhonen
Broadcom Communications
Porkkalankatu 24
FIN-00180 Helsinki
Finland

Email: jouni.nospam@gmail.com

Shwetha Bhandari
Cisco Systems
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Phone: +91 80 4426 0474
Email: shwethab@cisco.com

mif Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2014

S. Krishnan
Ericsson
J. Korhonen
Broadcom
S. Bhandari
Cisco Systems
S. Gundavelli
Cisco

February 14, 2014

Identification of provisioning domains
draft-kkbg-mpvd-id-00

Abstract

The MIF working group is producing a solution to solve the issues that are associated with nodes that can be attached to multiple networks. This document describes several methods of generating identification information for provisioning them and a format for carrying such identification in configuration protocols.

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 18, 2014.

Table of Contents

1. Introduction

The MIF working group is producing a solution to solve the issues that are associated with nodes that can be attached to multiple networks based on the Multiple Provisioning Domains (MPVD) architecture work [I-D.ietf-mif-mpvd-arch]. This document describes a format for carrying identification information along with a few alternatives for reasonable sources for PVD identification. Since the PVD IDs are expected to be unique, the identification sources provide some level of uniqueness using either a hierarchical structure (e.g. FQDNs and OIDs) or some form of randomness (e.g. UUID and ULAs). Any source that does not provide either guaranteed or probabilistic uniqueness is probably not a good candidate for identifying provisioning domains.

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. Provisioning domain identity format

The identity of the PVD is independent of the configuration protocol used to communicate it and is formatted as follows.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    id-type    |   id-length   |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
+                     PVD identity information                  +
.                        (variable length)                      .
+---------------------------------------------------------------+
```

Figure 1: PVD ID Option

- o id-type: Describes the type of identification information. This document defines six types of PVD identity information
  - 0x01: UUID [RFC4122]
  - 0x02: UTF-8 string
  - 0x03: OID [OID]
  - 0x04: NAI Realm [RFC4282]
  - 0x05: FQDN
  - 0x06: ULA Prefix [RFC4193]
  - Further types can be added by IANA action.
- o id-length: Length of the PVD identification in octets not including the id-type and id-length fields.
- o PVD identity information: The PVD identification that is based on the id-type.

4. Security Considerations

An attacker may attempt to modify the PVD identity provided in a configuration protocol. These attacks can be prevented by using the configuration protocol mechanisms such as SEND [RFC3971] and DHCPv6 AUTH option [RFC3315] that detect any form of tampering with the configuration.

A compromised configuration source, on the other hand, cannot easily be detected by a configuration client. The only real way to avoid this is that the PvD identification is directly associable to some form of authentication and authorization information from the owner of the PvD (e.g. an FQDN can be associated with a DANE cert). Then, this attack can be detected by the client by verifying the authentication and authorization information provided inside the PVD container option after verifying its trust towards the PvD owner (e.g. a certificate with a well-known/common trust anchor that).

5. IANA Considerations

This document creates a new registry for PVD id types. The initial values are listed below

- 0x01: UUID [RFC4122]
- 0x02: UTF-8 string
- 0x03: OID [OID]
- 0x04: NAI Realm [RFC4282]
- 0x05: FQDN
- 0x06: ULA Prefix [RFC4193]

6. Acknowledgements

The authors would like to thank the members of the MIF architecture design team, Ted Lemon, Brian Carpenter, Bernie Volz and Alper Yegin for their contributions to this draft.

7. Normative References

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

[OID] IANA, "PRIVATE ENTERPRISE NUMBERS", SMI Network Management Private Enterprise Codes, http://www.iana.org/assignments/enterprise-numbers/enterprise-numbers.

[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.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

[RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.

Authors' Addresses

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

Phone: +1 514 345 7900 x42871
Email: suresh.krishnan@ericsson.com

Jouni Korhonen
Broadcom Communications
Porkkalankatu 24
FIN-00180 Helsinki
Finland

Email: jouni.nospam@gmail.com

Shwetha Bhandari
Cisco Systems
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Phone: +91 80 4426 0474
Email: shwethab@cisco.com

Sri Gundavelli
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sgundave@cisco.com

MIF
Internet Draft
Intended status: Informational
Expires: January 3, 2015

Hongke Zhang
Boxin Du
Mi Zhang
Fei Song
Beijing Jiaotong University
July 3, 2014

MIF API extension for combining IEEE 802.21
draft-zhang-mif-api-extension-802-21-00.txt

Abstract

The Application Program Interface (API) of MIF, specified in the MIF API consideration, must rely on lower layer functionalities when handover between homogeneous or heterogeneous networks is necessary. To improve the connectivity experience, the existing MIF API needs to be extended for solving such problem. IEEE is also aimed at the similar issue from different way. A kind of logical entities over the link layer protocol for handling the seamless handover has been defined in IEEE 802.21. This document proposes a mechanism via integrating MIF API and IEEE 802.21 to support application service better.

Status of this Memo

Table of Contents

1. Introduction

In MIF context, improved connectivity experiences SHOULD be created. Enhancing the performance of horizontal and vertical switches between networks is one of main targets. Such situation is quite similar with Media Independent Handover (MIH) described in [IEEE 802.21]. The MIF Application Program Interface (API) specified in the MIF API consideration [I-D.ietf-mif-api-extension] describes a set of message calls to implement higher level APIs that solve the problems in multiple interface scenarios. However, this draft only provides a minimal set of message calls REQUIRED to implement the API. New functions could be added.

According to [IEEE 802.21], the Media Independent Handover Function (MIHF) is a logical entity that facilitates MIH decision making based on inputs from the MIHF. It provides abstracted services to higher layers. Communications with the lower layer of the mobility-management protocol stack can be achieved through technology-specific interfaces in MIHF.

Although two mechanisms, MIF API and MIHF, are working in different layers and defined by different organizations, the requirements of compatibility are obvious. Some of the functions of MIF API SHOULD be supported by a connection manager (i.e. the MIHF), and vice versa. Owing to the advantages of both MIHF and its Service Access Points (SAPs), the functions of MIHF could be utilized in MIF API for handover issues. This document extends message calls of MIF API to support the MIH. Like [I-D.ietf-mif-api-extension], no bindings for programming languages are provided because they are left up to the implementation. This document only describes the messages sent and received. It can be read as a checklist for operating system vendors.

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 RFC-2119 [RFC2119].

3. The Relationship between IEEE MIHF and MIF API

The purpose of IEEE 802.21 is to improve the user experience of mobile devices by facilitating handover between IEEE 802 networks whether or not they are of different media types, including both wired and wireless. It also aims at making it possible for mobile devices to perform seamless handover between IEEE 802 and non IEEE networks. This standard defines:

1) A framework that enables service continuity while a Mobile Node (MN) transitions between heterogeneous link-layer technologies;

2) MIHF;

3) MIH_SAP and associated primitives for users to get services of MIHF;

4) The definitions of new link-layer SAPs and associated primitives for each link-layer technology. They help MIHF to collect link information and control link behaviors during handovers.

The concept of MIHF is a functional entity to realize the high-performance handovers. The advantage of MIHF is that it provides media-independent services to higher layers no matter which media-specific technology of lower layer is being used, such as IEEE Std.802.3, IEEE Std.802.11, IEEE Std.802.16, 3GPP and 3GPP2. It also defines three kinds of SAPs (will be detailed later) of MIHF and their primitives that interact between different layers. The MIHF can also act as a filter: the messages received from link layer SHOULD be processed and submitted to higher layer for meeting the subscribers' need. Therefore, MIHF should work under MIF API. In fact, MIF API SHOULD be served as a user of MIHF, which is shown in Figure 1. The subscribers can then only interact with the MIHF via one kind of SAPs (i.e. MIH_SAP) without knowing lower things. The MIH protocol of this standard is not discussed in this document.

Three kinds of MIHF services are defined in the standard. They are Media Independent Event Service (MIES), Media Independent Command Service (MICS) and Media Independent Information Service (MIIS). MIES provides event classification, event filtering, event reporting corresponding to dynamic changes in link characteristics, link status and link quality. It originates from lower layers and can be passed to MIHF or upper users for the detection of handover requirement. MICS enables MIH users to manage and control link behavior relevant to handover and mobility. It is invoked by users or MIHF and makes effect on MIHF or lower layers.

For example, in MN-initiated handover scenario, MICS is adopted for MN switching between different links. MIIS allows MN and network entities to discover information that influences the selection of appropriate networks during handovers. Figure 2 [IEEE 802.21] shows MIH services and their initiation.

```
+-------------------------------------+
|             Application             |
+-------------------------------------+
   /\     ||     /\   ||    /\   ||
   ||     \/     ||   ||    ||   ||
+--------------+ ||   ||    ||   ||
|High Level API| ||   ||    ||   ||
+--------------+ ||   ||    ||   ||
   /\     ||     ||   ||    ||   ||
   ||     \/     ||   \/    ||   ||
+------------------------+  ||   ||
|        MIF API         |  ||   ||
+------------------------+  ||   ||
    /\      ||              ||   \/
    ||      ||   +--------------------+
    ||      ||   | Communication API  |
    ||      ||   +--------------------+
    ||      ||         /\  /\
    ||      ||         ||  ||
    ||      \/         \/  \/
+-------------------------------------+
|                   MIHF              |
+-------------------------------------+
         /\              ||
         ||              \/
+-------------------------------------+
|         Network Link API            |
+-------------------------------------+
    /\    ||              /\    ||
    ||    \/              ||    \/
+---------------+     +---------------+
|    Network    |     |    Network    |
|  Interface 1  |     |  Interface 2  |
+---------------+     +---------------+
```

Table 1 The relationship of MIHF & MIF API

The letters a, b, c in Figure 2 are:

a. MIH_SAP

b. MIH_LINK_SAP

c. LLC_SAP

The SAPs are divided into two categories:

1) Media dependent SAP (including MIH_LINK_SAP and LLC_SAP)

2) Media independent SAP (MIH_SAP)

```
+-------------+     Command      +-----------------------------+
|             +     Service      |                             |
|            / \ <---------------+                             |
|           +   +                |       MIH User              |
|           |   |    Event       |                             |
| MIHF      |   |   Service      |    Layer 3 or Higher        |
|           | a |--------------->|    Mobility Protocol        |
| Event     |   |                |                             |
| Service   |   |  Information   |                             |
|           +   +   Service      |                             |
| Command    \ /<--------------->|  +-------+        +-------+ |
| Service     +                  +--+   c   +-+---+--+   c   +-+
|             |     Command      +  +-------+ |   +  +-------+ |
| Information |     Service     / \           |  / \           |
| Service     |--------------->+   +          | +   +          |
|             |                |   | Network1 | |   | Network2 |
|             |      Event     |   | (e.g.IEEE| |   | (e.g.IEEE|
|             |     Service    | b |  802.16) | | b |  802.11) |
|             |<---------------|   |          | |   |          |
|             |                |   |          | |   |          |
|             |   Information  |   |          | |   |          |
|             |     Service    +   +          | +   +          |
|             |<--------------->\ /           |  \ /           |
|             |                  +            |   +            |
|             |                  |            |   |            |
+-------------+                  +------------+   +------------+
```

Table 2 MIH services and their initiation

SAP of the MIHF (i.e. MIH_SAP) is media independent. The MIH_SAP defines the interface between the MIHF and MIH users such as an upper layer mobility protocol or a handover function (e.g., MIF API) that might reside at higher layer transport entity as well. MIH_SAP allows the MIHF to provide services to the upper layers, the network management plane and the data bearer plane. Upper layers need to subscribe with the MIHF as users to receive MIHF events. In MIF case, the MIF API can directly send commands to the local MIHF via messages which use the service primitives of the MIH_SAP.

All the messages REQUIRED for communicating successfully in MIF environment that described in the [I-D.ietf-mif-api-extension] MUST also be used here. Those messages define the way that MIF API interacts with the higher layers or applications. New messages need

to be added into this set for handover process. These new messages SHOULD be exchanged between MIHF and MIF API. Some of them may use the services of MIHF.

4. The Extension of MIF API for Handover: A Case Study

This section introduces the extension of message calls of MIF API in two parts based on the classification of handovers (MN-initiated handover and the network-initiated handover). To handle these two kinds of handovers successfully, MIF API SHOULD be extended respectively based on the characteristics of process.

4.1. The MN-initiated Handover

7 steps are included in the MN-initiated handover:

1) Information query. The MN collects network information from the MIIS server that the MN is connected to.

2) Resource availability check. The MIF API sends request to find candidates and then receives a list of candidate networks in response message.

3) Resource preparation. The MN SHOULD determine which target network is suitable and request it for resource preparation.

4) Establish new L2 connection. The MIF API initiates a new link connection.

5) Link up indication. The MIHF of MN notifies the MIF API that the link is up.

6) Higher layer handover execution.

7) Resource release. The original serving network resources must be released in the end.

The following messages, which are only the interactions between a MIHF and MIF APIs, need to be added.

4.1.1. Get Information

This message is sent by the MIF API for the inquiry of the neighbouring networks information. In MIH, we can use the MIH_Get_Information.request for the same purpose. After receiving this message, the MIHF inquires the MIIS server for the information, and then returns a list of network information needed for MIF API.

4.1.2. Information Post

This message is sent to the MIF API by the MIIS server as a result of the Get Information message. MIH_Get_Information.confirm can be used to convey such information.

4.1.3. Parameter Report

When attaching to a specific network, the MIF API needs to receive the link status from the lower layers to better control the whole connection. The MIHF can receive reports from the link layers and submit to the higher layers. If the link is going down, the MIF API must notify its subscribers using message "Interface is going away". The application or higher API could try to establish new connections by sending "Wants to connect" to MIHF and the connection process will begin from step 2 (i.e. Resource availability check).

Another situation is that once the MIF API receives a "Wants to connect" message from its subscriber, the MIF API SHOULD accordingly trigger a whole connection process to a new network. This can also begin from step 2.

4.1.4. Check Resources MN

Before the connection starts, the MN SHOULD check the resource availability at the candidate networks. This message is sent to the MIHF by the MIF API. The serving network SHOULD request each candidate. The final result SHOULD be returned to the higher layer. The MIH_MN_HO_Candidate_Query.request can be used in the MIH case.

4.1.5. Resource Availability

When receiving the resource availability of the candidates from the serving network, the MIHF SHOULD provide them to the MIF API. The MIH_MN_HO_Candidate_Query.confirm can be used in the MIH case.

4.1.6. Connect to Interface

This message is sent to the MIF API by the upper applications. When the MIF API receives the Resource Availability, it could post the message to the higher layer. The upper application can use "Connect to Interface" to choose a preferred network interface. More details about the choosing methods need further discussion.

4.1.7. Resource Preparation Messages

The MIF API can use the MIH_MN_HO_Commit.request, including the target network information, to request the chosen network for resource preparation. When the preparation is done, the MIHF receives the response from the target network. It sends a MIH_MN_HO_Commit.confirm message to the MIF API for informing the status of the previously issued target notification request.

4.1.8. Establish Link Messages

MIF API can use the MIH_Link_Action.request to solve the connection establishment problem. This primitive defined by the [IEEE 802.21] is to control the local or remote lower link layers. It includes a MIHF ID and a Link Actions List, which can realize many functions of controlling. After the action has been executed, the MIF API should receive a MIH_Link_Action.confirm for result indication.

4.1.9. Link Up

After the new link is established, the MAC layers MUST deliver a Link_Up.indication to the MIHF. The MIHF then passes the MIH_Link_Up.indication message to the MIF API. The MIF API can notify the upper applications by delivering the "Link is going up" message. Then the higher layer handover execution might be triggered and the traffic flow can be re-established.

4.1.10. Handover Completed

This message is sent to the local MIHF by the MIH API for releasing the resources of the previous serving network which was originally allocated to the MN. After identifying that the release is successfully done, the target network Point of Service (PoS) sends the confirm message to the MIHF and the MIHF SHOULD also inform the MIF API with the same information.

4.1.11. Handover Completion Confirmation

This message is sent to the MIF API by the MIHF indicating that the resource of the previous network is successfully released.

4.2. The Network-initiated Handover

There are also 7 steps in an intact network-initiated handover, like the MN-initiated handover:

1) Information query.

2) Resource availability check.

3) Resource preparation.

4) Establish a L2 connection using MIH_Link_Action.request.

5) Sent link indications to the MIF API.

6) Higher layer handover execution.

7) Resource release.

The differences between Network-initiated case and MN-initiated case are in step 1 and step 2: the Get Information request and Information Query are respectively initiated by the MIH user of the serving network. When such MIH user obtains the information from the MIIS server, it sends requests to the MN for response message containing the MN’s handover acknowledgement, MN’s preferred link and PoS lists.

In step 3, the commit of target network is also initiated by the MIH user of a serving network. After the resource is prepared, the PoS of serving network SHOULD notify the MN for the establishment of L2 connection in step 4.

The following messages should be added in MIF API:

4.2.1. Candidate Query Notification

This message is sent to MN’s MIHF from the PoS of the serving network with a list of PoAs of each candidate network link. Such message means the MN SHOULD consider new access network. This message can use the MIH_Net_HO_Candidate_Query.indication.

4.2.2. Candidate Query Result

This message is sent to the local serving network’s MIHF from MN’s MIF API, specifying whether the request of handover is permitted or not. MIH_Net_HO_Candidate_Query.response can be used here. If the handover is permitted, a new access network SHOULD be considered at handover initiation stage.

4.2.3. Check Resources Net

This message is sent to the MN’s MIF API by the PoS of serving network with a list of target network information and a set of resource parameters assigned to the MN for the handover. MIH_Net_HO_Commit.indication can be used here. Then the MIF API can

trigger the establishment of L2 connection by using Link_Action.request. After the link connection is done, MIF API needs to inform the serving network. Link_Action.request might also have a list of actions for handover control during the link connection period.

4.2.4. Confirm Chosen Target

This message is sent by the MN’s MIF API as the response of the MIH_HO_Commit.indication, showing that the indication is received. MIH_Net_HO_Commit.response can be used here. Also such message might include a list of results of previous actions.

4.2.5. Establish Link Messages

This message is exactly the same as that of the MN-initiated process, for establishing a new L2 link connection.

4.2.6. Link Up

This message is exactly the same as that of the MN-initiated process, for informing the MIF API that the L2 link is completed.

4.2.7. Handover Completed

This message is exactly the same as that of the MN-initiated process, for releasing the resources that have already attained by the MN.

4.2.8. Handover Completion Confirmation

This message is exactly the same as that of the MN-initiated process, for confirming that the resources of the previous network are successfully released.

5. Discussions for New Messages from MIF Perspective

New messages described in this document are critical for information exchanging and function achievement between MIHF and MIF API. Since both "upper layer requirements gathering" and "lower layer command delivering" (or reverse) can be achieved via messages, the logical relationship should be discussed in-depth. The messages below are only the initial examples. Further updating is still needed.

5.1. Get Information

According to [IEEE 802.21], this information query is related to a specific interface. It has the flexibility to query either a specific

data within a network interface or an extended schema of a given network. When the advanced application or upper APIs send "Announce Interface", "Announce PVD" and "Announce IE (Information Elements)", the MIF API could obtain the information by using the Get_information.request in MIH.

5.2. Release the Ongoing Connection

The [I-D.ietf-mif-api-extension] defines a message called "Connection can be broken", which means that the MN can tolerate the connection being broken, e.g. for power conservation. When the subscribers of MIF API delivers this message, the MIF API SHOULD send "Release the ongoing connection" to the MIHF so that directly releasing the current resources of network to cut off this connection. But this action will not weaken the function of the application.

5.3. Establish New L2 Connection

According to [IEEE 802.21], the MIH_Link_Actions can trigger a L2 link connection. When MN wants to set up a TCP connection with an IP host, the MIF API will receive "Connect to Address" or "Connect to Address from Address" messages. Then the MIF API can use the MIH_Link_Actions.request to ask MIHF for a new L2 link connection.

6. Discussions for New Messages from IEEE Perspective

The following tables, table1, 2, 3 and 4, are from the [IEEE 802.21]. These messages might be used in the MIF API because they are exchanged between the MIHF and its MIH users. Some of them have been discussed above, but the specific usage of the rest is not represented. This section is only a direct listing of all these messages, their category and brief descriptions. Further discussion is still needed.

Messages (Information)	Description
MIH_Get_Information	Request to get information from repository
MIH_Push_Information	Notify the MN of operator policies or other information

Table 1 Information Messages of MIHF

Messages (Event)	Description
MIH_Link_Detected	Link of a new access network has been detected. This event is typically on the MN when the first PoA of an access network is detected
Track_timeout	This event is not generated when subsequent PoAs of the same access network are discovered
MIH_Link_Up	L2 connection is established and link is available for use
MIH_Link_Down	L2 connection is broken and link is not available for use
MIH_Link_Paremeters _Report	Link parameters have crossed a specified threshold and need to be reported
MIH_Link_Going_Down	Link conditions are degrading and connection loss is imminent
MIH_Link_Handover _Imminent	L2 handover is imminent based on either the changes in the link conditions or additional information available in the network
MIH_Link_Handover _Complete	L2 handover to a new PoA has been completed
MIH_Link_PDU _Transmit_Status	Indicate transmission status of a PDU

Table 2 Event Messages of MIHF

Messages (Command)	Description
MIH_Link_Get _Parameters	Get the status of a link
MIH_Net_HO _Candidate _Query	Network initiates handover and sends a list of suggested networks and associated points of attachment
MIH_Link_Configure Thresholds	Configure link parameter thresholds
MIH_Link_Actions	Control the behavior of a set of links
MIH_MN_HO_Candidate _Query	Command used by MN to query and obtain handover related information about possible candidate networks
MIH_N2N_HO_Query _Resources	command sent by the serving MIHF entity to the target MIHF entity for resource query
MIH_MN_HO_Commit	Command used by MN to notify the serving network of the decided target network information
MIH_Net_HO_Commit	Command used by the network to notify the MN of the decided target network information
MIH_N2N_HO_Commit	Command used by a serving network to inform a target network that an MN is about to move toward that network, initiate context transfer and perform handover preparation
MIH_MN_HO_Complete	Notification from MIHF of the MN to the target or source MIHF indicating the status of handover completion
MIH_N2N_HO_Complete	Notification from MIHF of the MN to the target or source MIHF indicating the status of handover completion
MIH_N2N_HO_Complete	Notification from either source or target MIHF to the peer MIHF indicating the status of the handover completion

Table 3 Command Messages of MIHF

Messages (Service management)	Description
MIH_Capability _Discover	Discover list of Events and Commands supported by MIHF
MIH_Register	Register with a remote MIHF
MIH_DeRegister	Deregister with a remote MIHF
MIH_Event_Subscribe	Subscribe for MIH event notification
MIH_Event_Unsubscibe	Unsubscribe from MIH event notification

Table 4 Service Management of MIHF

7. Security Considerations

This document does not contain any security considerations.

8. IANA Considerations

There are presently no IANA considerations with this document.

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.

9.2. Informative References

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

[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), Feb. 2014.

10. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Hongke Zhang
Beijing Jiaotong University (BJTU)
Beijing, 100044, P.R.China

Email: hkzhang@bjtu.edu.cn

Boxin Du
Beijing Jiaotong University (BJTU)
Beijing, 100044, P.R.China

Email: 10291070@bjtu.edu.cn

Mi Zhang
Beijing Jiaotong University (BJTU)
Beijing, 100044, P.R.China

Email: 13120174@bjtu.edu.cn

Fei Song
Beijing Jiaotong University (BJTU)
Beijing, 100044, P.R.China

Email: fsong@bjtu.edu.cn