

IPSECME Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: March 5, 2010

D. Migault  
Orange Labs R&D  
September 2009

MOBIKE eXtension (MOBIKE-X) for Transport Mobility and Multihomed IKE\_SA  
[draft-mglt-ipsec-mm-mobikex-00.txt](#)

## Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on March 5, 2010.

## Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (<http://trustee.ietf.org/license-info>). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

## Abstract

This draft provides a description of MOBIKE mobility and multihoming extension (MOBIKE-X). The extension is elaborated from the MOBIKE [[RFC4555](#)] protocol as well as mobility and multihoming requirements

---

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009

described in [[I-D.mglt-ipsec-mm-requirements](#)]. Extension concerns the Security Association facilities in a multihoming and mobile environment. More specifically this draft considers the use of multiple interfaces and the transport mode of IPsec. One of the goal of this draft was to make this extension compatible with MOBIKE [[RFC4555](#)]. We especially point out the differences from MOBIKE [[RFC4555](#)], as well as how its interacts with MOBIKE.

## Table of Contents

<a href="#">1.</a>	Requirements notation . . . . .	<a href="#">4</a>
<a href="#">2.</a>	Terminology . . . . .	<a href="#">4</a>
<a href="#">3.</a>	Introduction . . . . .	<a href="#">4</a>
<a href="#">4.</a>	Requirements . . . . .	<a href="#">7</a>
<a href="#">5.</a>	MOBIKE-X Protocol overview . . . . .	<a href="#">8</a>
<a href="#">5.1.</a>	UPDATE action . . . . .	<a href="#">8</a>
<a href="#">5.2.</a>	ADD and REMOVE actions . . . . .	<a href="#">9</a>
<a href="#">5.3.</a>	Response to a CHECK Request . . . . .	<a href="#">10</a>
<a href="#">5.4.</a>	SELECTing the SA . . . . .	<a href="#">10</a>
<a href="#">5.5.</a>	Alternate IP Address . . . . .	<a href="#">10</a>
<a href="#">5.6.</a>	MOBIKE Version . . . . .	<a href="#">11</a>
<a href="#">5.7.</a>	Other Considerations . . . . .	<a href="#">11</a>
<a href="#">6.</a>	Notify Payload Description . . . . .	<a href="#">12</a>
<a href="#">6.1.</a>	MOBIKE_SUPPORTED Notify Payload . . . . .	<a href="#">12</a>
<a href="#">6.2.</a>	MOBIKE_UNSUPPORTED_VERSION Notify Payload . . . . .	<a href="#">12</a>
<a href="#">6.3.</a>	SELECTOR Notify Payload . . . . .	<a href="#">13</a>
<a href="#">6.4.</a>	END_OF_SELECTOR Notify Payload . . . . .	<a href="#">13</a>
<a href="#">6.5.</a>	ADDITIONAL_IP_ADDRESS Notify Payload . . . . .	<a href="#">13</a>
<a href="#">6.6.</a>	UPDATE_SA_ADDRESSES Notify Payload . . . . .	<a href="#">14</a>
<a href="#">6.7.</a>	ADD_SA_ADDRESS Notify Payload . . . . .	<a href="#">15</a>
<a href="#">6.8.</a>	REMOVE_SA_ADDRESS Notify Payload . . . . .	<a href="#">16</a>
<a href="#">7.</a>	Protocol Exchanges . . . . .	<a href="#">17</a>
<a href="#">7.1.</a>	Initial Exchange . . . . .	<a href="#">17</a>
<a href="#">7.1.1.</a>	MOBIKE_SUPPORTED Notify Payload . . . . .	<a href="#">17</a>
<a href="#">7.1.2.</a>	MOBIKE_UNSUPPORTED_VERSION Notify Payload . . . . .	<a href="#">21</a>
<a href="#">7.1.3.</a>	Initial exchange state diagrams . . . . .	<a href="#">21</a>
<a href="#">7.2.</a>	Alternate IP Address Exchange . . . . .	<a href="#">26</a>
<a href="#">7.3.</a>	Alternate IP Address Exchange State Diagram . . . . .	<a href="#">27</a>
<a href="#">7.4.</a>	UPDATE Exchange . . . . .	<a href="#">29</a>
<a href="#">7.5.</a>	UPDATE Exchange State Diagram . . . . .	<a href="#">32</a>
<a href="#">7.6.</a>	Multihoming Exchanges . . . . .	<a href="#">36</a>
<a href="#">7.7.</a>	Multihoming Exchanges State Diagrams . . . . .	<a href="#">37</a>

<a href="#">8.</a>	<a href="#">SELECTOR Notify Payload</a>	<a href="#">39</a>
<a href="#">9.</a>	<a href="#">SELECTOR State diagrams</a>	<a href="#">42</a>
<a href="#">10.</a>	<a href="#">ID Notify Payload Parameter</a>	<a href="#">45</a>
<a href="#">11.</a>	<a href="#">Packet Format</a>	<a href="#">45</a>
<a href="#">11.1.</a>	<a href="#">Notify Payload</a>	<a href="#">45</a>

<a href="#">11.2.</a>	<a href="#">Notify Message -- status type</a>	<a href="#">46</a>
<a href="#">11.2.1.</a>	<a href="#">MOBIKE_SUPPORTED</a>	<a href="#">46</a>
<a href="#">11.2.2.</a>	<a href="#">ADDITIONAL_IP_ADDRESS</a>	<a href="#">46</a>
<a href="#">11.2.3.</a>	<a href="#">NO_ADDITIONAL_ADDRESS</a>	<a href="#">47</a>
<a href="#">11.2.4.</a>	<a href="#">SELECTOR</a>	<a href="#">47</a>
<a href="#">11.2.5.</a>	<a href="#">END_OF_SELECTOR</a>	<a href="#">47</a>
<a href="#">11.2.6.</a>	<a href="#">UPDATE_SA_ADDRESSES</a>	<a href="#">47</a>
<a href="#">11.2.7.</a>	<a href="#">ADD_SA_ADDRESS Notify Payload</a>	<a href="#">47</a>
<a href="#">11.2.8.</a>	<a href="#">REMOVE_SA_ADDRESS Notify Payload</a>	<a href="#">47</a>
<a href="#">11.2.9.</a>	<a href="#">Notify Message -- status type table</a>	<a href="#">47</a>
<a href="#">11.3.</a>	<a href="#">Notify Message -- error type</a>	<a href="#">48</a>
<a href="#">11.3.1.</a>	<a href="#">MOBIKE_UNSUPPORTED_VERSION</a>	<a href="#">48</a>
<a href="#">11.3.2.</a>	<a href="#">UNACCEPTABLE_ADDRESS</a>	<a href="#">48</a>
<a href="#">11.3.3.</a>	<a href="#">DOES_NOT_FIT_SPD</a>	<a href="#">48</a>
<a href="#">11.3.4.</a>	<a href="#">UNACCEPTABLE_PARAMETERS</a>	<a href="#">48</a>
<a href="#">11.3.5.</a>	<a href="#">UNEXPECTED_PAYLOAD_AFTER_SELECTOR</a>	<a href="#">49</a>
<a href="#">11.3.6.</a>	<a href="#">UNMATCHED_SELECTOR_SET</a>	<a href="#">49</a>
<a href="#">11.3.7.</a>	<a href="#">Notify Message -- error type table</a>	<a href="#">49</a>
<a href="#">11.4.</a>	<a href="#">Notify Parameters</a>	<a href="#">49</a>
<a href="#">11.4.1.</a>	<a href="#">VERSION</a>	<a href="#">49</a>
<a href="#">11.4.2.</a>	<a href="#">SPI</a>	<a href="#">50</a>
<a href="#">11.4.3.</a>	<a href="#">IP</a>	<a href="#">51</a>
<a href="#">11.4.4.</a>	<a href="#">Multihoming Information Payload (MIP)</a>	<a href="#">52</a>
<a href="#">11.4.5.</a>	<a href="#">IPSEC_MODE</a>	<a href="#">54</a>
<a href="#">11.4.6.</a>	<a href="#">IPSEC_PROTO</a>	<a href="#">54</a>
<a href="#">11.4.7.</a>	<a href="#">SELECTOR_SPECIFIC_VALUE_PARAMETER</a>	<a href="#">55</a>
<a href="#">11.4.8.</a>	<a href="#">ID</a>	<a href="#">56</a>
<a href="#">11.4.9.</a>	<a href="#">Parameter Code Type</a>	<a href="#">56</a>
<a href="#">12.</a>	<a href="#">Security Considerations</a>	<a href="#">57</a>
<a href="#">13.</a>	<a href="#">IANA Considerations</a>	<a href="#">57</a>
<a href="#">14.</a>	<a href="#">Acknowledgments</a>	<a href="#">59</a>
<a href="#">15.</a>	<a href="#">References</a>	<a href="#">59</a>
<a href="#">15.1.</a>	<a href="#">Normative References</a>	<a href="#">59</a>
<a href="#">15.2.</a>	<a href="#">Informative References</a>	<a href="#">59</a>
	<a href="#">Author's Address</a>	<a href="#">59</a>

---

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009

## 1. Requirements notation

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

## 2. Terminology

- Mobile Node (MN) : In this draft the Mobile Node is the peer that performs the mobility action. The Mobile Node does not have to be understood as in MIPv6.
- Initiator : The Initiator is the peer that initiates the exchange. It sends a message to the Responder. It is important to note that if two peers are connected, the Initiator of one exchange can be the Responder of another exchange. When a mobility action is performed then the Initiator is also the Mobile Node.
- Responder : The Responder is the peer receiving a message. The message is sent from the Initiator.
- Alternate IP Address : The alternate IP address of a peer is the IP address a peer is not currently using but that might be used latter. An alternate IP address should used only if the current IP address does not work anymore.

## 3. Introduction

This document provides a description of MOBIKE-X. We assume the

reader is familiar with IPsec [[RFC4301](#)], IKEv2 [[RFC4306](#)] and with MOBIKE [[RFC4555](#)].

MOBIKE-X extends MOBIKE facilities to change the IP address of an IPsec Security Associations. More specifically, MOBIKE-X includes an IPsec Security Association using a transport mode. MOBIKE-X also extends the multihoming scope of MOBIKE, and considers that simultaneous IP addresses can be used between two peers. Combination of mobility and multihoming facilities provide IP address management facilities. This draft is mainly focused on the multihoming aspects of the IKEv2 channel. This draft does not provide any use cases. Use cases are described in [[I-D.mglt-ipsec-mm-requirements](#)]. This draft considers scenarios and requirements of [[I-D.mglt-ipsec-mm-requirements](#)]. From these requirements, this document defines extensions to MOBIKE which includes new Notify Payloads, and parameters carried by Notify Payloads.

MOBIKE [[RFC4555](#)] proposes a mobility solution for the tunnel mode. MOBIKE use case is a mobile node connected to an Intranet through a

security gateway. For simplicity we consider that the security gateway is connected on the Internet and on a private network. The mobile node negotiates a connection with the security gateway using IKEv2. After the IKEv2 negotiation, the mobile node has a private IP address, and communicates with nodes on the Intranet by tunneling them to the security gateway. The outer IP addresses of the tunnel are the public IP addresses of the security gateway and of the mobile node. MOBIKE has been designed so that the mobile node can change its public IP address without breaking the Security Association between the inner IP addresses, negotiated with IKEv2. In other words, when the mobile node changes its public IP address MOBIKE avoids that the mobile node restarts a IKEv2 negotiation with the security gateway, proceeding to the IKE\_SA\_INIT, IKE\_AUTH and CREATE\_CHILD\_SA exchanges.

With the tunnel mode the selectors of the SPD and the SAD are the inner IP addresses. When outer addresses are changed, only some parameters of the SA are changed. More specifically, selectors of a given SA are kept unchanged during mobility. No new SA needs to be created and so SPD and PAD and their bindings with the SAD are kept unchanged. On the other hand, with the transport mode, the selectors of the SA are changed. When one host changes its IP address, and the

SA is identified through its selectors, it MUST be changed. As a consequence, the SPD MUST also be changed, and the PAD MUST be checked to see if new SA can be created.

MOBIKE considers that peer only has one interface. This brings simplification over what modification has to be done, and where the updates need to be performed. In fact if a node decides to perform a mobility action and has only one interface, then the other peer knows the new address is one being used by the Mobile Node, and the old IP address is the one that the MN used to use. If a peer is multihomed, the MN should be able to indicate what is the new IP address, and what is the old IP address -- that is to say the IP address to be replaced.

MOBIKE also considers multihoming. The type of multihoming MOBIKE considers is the Alternate IP Addresses Multihoming (AM). This means that one peer can provide multiple IP addresses to the other peer, but those IP addresses MUST NOT be used unless the current IP address is not working anymore. In other words, MOBIKE do not consider the Simultaneous IP Addresses Multihoming (SM) that would enable IKEv2 to use simultaneously multiple IP addresses.

Furthermore, one should also notice that multihoming facilities provided by MOBIKE are ONLY intended for the IKEv2 application. This means that the Alternate IP Addresses provided by on peer are only intended to be used by IKEv2. In other words, other applications

than IKEv2 are not using the Alternate IP Addresses.

We do not see in this draft much reasons to enable Simultaneous IP Addresses Multihoming (SM) for IKEv2 as an application. On the other hand we consider that Simultaneous IP Addresses Multihoming (SM) facilities can be provided by IKEv2 so that other application can benefit from it. More precisely, the IPsec layer of one peer MUST be able to inform the other peer IPsec layer that Simultaneous IP Addresses can be used on a given Security Association. Simultaneous IP Addresses Multihoming consists of ADDing or REMOVing one or more IP addresses to a specific Security Association. Such IPsec modifications are necessary to protect and allow the traffic of an application. At least they can avoid IKEv2 negotiation, and multiple IPsec contexts.

Until now we assume that IKEv2 provides messages that only concerns the IPsec parameters. This of course respects the layered model. On the other hand we can also think of interaction between Upper Layer Protocols (ULP). One way is to consider that ULP exchanges provide the IPsec Security Association configuration. The other way considers that such exchanges SHOULD be protected and IPsec is a good way to protect data. Thus rather than using a specific channel negotiated by IKEv2, we can use directly the IKEv2 channel. Furthermore this way would provide a generic multihomed and mobility related information that could be re-used by different ULP. So this way considers that multihoming and mobility information sent via the IKEv2 channel, are used by the IPsec layer to update / modify the Security Association, but are also forwarded to the ULP so that they could consider their own mobility and multihoming action on their own layer. The advantage of such strategy is to avoid multiple exchanges.

With ULP-IKEv2 interaction considerations, one can consider the use of the Alternate IP Addresses not for the IKEv2 application, but also for other applications. This means that when an Alternate IP Address is associated to a given Security Association, IKEv2 forwards this information to the ULP. That is the purpose of the ULP to decide how and when to use this Alternate IP Address. When the ULP considers that the current IP address is not working anymore and that an Alternate IP address could be used, then it asks IKEv2 to eventually perform Return Routability check and to either perform a mobility or multihoming action. If a multihoming action is decided, then the IP address is ADDED to a given Security Association. If a mobility action is decided, then the Security Association is UPDATED.

In this draft, we do not consider how the ULP SHOULD proceed. We do not either consider how the mobility or multihoming information are transmitted to the ULP if they are. This draft only consider the

IKEv2 exchange between the two peers. We mention here possible interactions between ULP and IKEv2 to justify that Multihoming Information can be associated to the IP addresses.

#### [4.](#) Requirements

From the previous section as well as from the [ID.mglt-ipsec-mm-

requirements], the requirements for MOBIKE-X can be split into three categories : the requirements on the Security Association other than the IKE\_SA, the requirements on the IKE\_SA and the requirements on the protocol.

Requirements on the Security Associations other than the IKE\_SA are listed below :

- 1 : Mobility and Multihoming MUST be done with IPsec in transport and in tunnel mode.
- 2 : The Initiator MUST be able to perform an UPDATE, ADD or REMOVE action on a Security Association, that is to say, change the IP address associated to the SA.
- 3 : For Mobility and Multihoming actions -- i.e. UPDATE, REMOVE or ADD action -- the Initiator MUST be able to check if the action is possible on the Responder side.
- 4 : Mobility and Multihoming actions, -- i.e. UPDATE, REMOVE or ADD -- action can explicitly specify which IP address is to be ADDED, REMOVED, UPADTED, and which IP address MUST replaced.
- 5 : When the Responder is requested to check a Mobility or Multihoming action, it MUST respond relevant information if the action can be performed and if not why the action cannot be performed.
- 6 : The Initiator MUST be able to SELECT the SA on which the action applies.
- 7 : The Initiator MUST be able to provide Alternate IP Addresses associated to specific SA.

Requirements on the IKE\_SA are listed below :

- 8 : The Initiator can send an Alternate IP Address to the IKEv2 application
- 9 : The Initiator MUST be able to explicitly specify on which IKE\_SA the action applies to.

Requirements regarding MOBIKE-X are listed bellow :

- 10 : The extension MUST be as close as possible to MOBIKE. This



means that MOBIKE-X MUST take MOBIKE as a base, and make the coding extension as light as possible. This include re-use of Notify Payloads defined in MOBIKE, as well as the same "philosophy".

- 11 : The exchange messages MUST be as low and short as possible. This means that omitted parameters have default values so that they do not need to be specified.
- 12 : Peers MUST be able to agree if the MOBIKE version used is MOBIKE-X or MOBIKE.

## 5. MOBIKE-X Protocol overview

### 5.1. UPDATE action

MOBIKE considers that only one IP address can be used at a time. This has the advantage of removing the ambiguity about which IP address is to be used. In fact the mobile node has established a communication with the security gateway using OLD\_IP. The MN is identified by its ID, has established an IKE\_SA to secure IKE transactions. When the mobile node has a new IP address NEW\_IP, it sends to the security gateway an UPDATE\_SA\_ADDRESSES Notify Payload. There is no need to specify which IP address is to be changed. When receiving an UPDATE\_SA\_ADDRESSES Notify Payload, the security gateway identifies the concerned IKE\_SA, and the concerned CHILD\_SA(s). The IP address that is to be replaced (OLD\_IP) is the outer tunnel IP address of the CHILD\_SA(s), and the IP address of the IKE\_SA associated to the host. The new value (NEW\_IP) to be placed is the IP address used to carry the UPDATE\_SA\_ADDRESSES Notify Payload and located in the IP header. Update of the SAD might requires some checks, such as checking return routability.

With MOBIKE-X we do not consider the Mobile Node uses only one IP address at a time. We do not make any assumption on how the Mobile Node is managing its IP addresses. This means that we do not use the IP layer to find out values needed for the update. MOBIKE-X provides the ability to fully specify into its UPDATE\_SA\_ADDRESSES Notify Payload the IP address to be replaced (OLD\_IP) and the new value of the IP address (NEW\_IP). MOBIKE-X uses the IP parameters that are put into the data field of the UPDATE\_SA\_ADDRESSES Notify Payload. The difference with MOBIKE is that the data field MAY be not empty. The IP parameter may carries certain information associated to the IP address. More specifically, the Initiator MUST specify if the IP address is an OLD\_IP, a NEW\_IP, and information on whether the Initiator wants the UPDATE to occurs now or if it want only to check the UPDATE would match local policies and IPsec Security Policies (CHECK). As far as UPDATE action is concerned, it fulfills

requirements 1, 2, 3 and 4.

Note that the OLD\_IP and NEW\_IP parameters does not have to be always specified. If none of those parameters are specified, the UPDATE\_SA\_ADDRESSES is performed in a similar manner as in MOBIKE way. In this case, the Responder updates the IP addresses used to route packet to/from the Initiator. More specifically, this includes the outer IP addresses in IPsec tunnel mode, the IP address of transport mode, and the IP address of the IKE\_SA. The replacing IP address is the IP address the Initiator use in the IP header of the IKEv2 message. The replacement occurs on the SA and IKE\_SA specified by the SELECTORs Notify Payload. This is coherent with MOBIKE specification. The default values for the SELECTOR Notify Payload is CHILD\_SA\_AND\_IKE\_SA. If the Initiator has a single interface and uses the tunnel mode the outer IP address of the tunnel being replaced. The difference with MOBIKE is that transport mode IPsec SA are also going to be UPDATED. As far as UPDATE action is concerned, it fulfills requirements 10 and 11.

## [5.2.](#) ADD and REMOVE actions

MOBIKE does not specify any equivalent actions for ADD or REMOVE actions. MOBIKE specifies an ADDITIONAL\_IP\_ADDRESS Notify Payload, but its purpose is to specify Alternate IP Addresses. One option would have been to re-use those Notify Payload at least for the ADD operation, and specify in the IP address parameters whether the IP address is an Alternate IP address or not. We decided to have specific Notify Payloads for Alternate IP Addresses since such IP addresses are not "directly" affecting the IPsec Security Associations. More specifically, Alternate IP addresses are either addressed to the IKEv2 application, or can eventually in the future be forwarded to the Upper Layer Protocols.

ADD and REMOVE actions need new Notify Payloads ADD\_SA\_ADDRESS and REMOVE\_SA\_ADDRESS Notify Payloads. Such Payloads work in a similar manner as the UPDATE\_SA\_ADDRESSES Notify Payload. The IP address that has to be ADDED or REMOVED of the SA is specified in an IP parameter. Only one IP parameter needs to be specified. The IP parameter carries some information such as the CHECK bit. The CHECK bit set to 1 specifies the Initiator only wants to check the action matches local and security policies. The IP parameter can also carry multihoming specific information thanks to the Multihoming Information Payload (MIP). Multihoming specific Information are used so that ULP can make their decision on which IP address to choose. In fact there are not expected to be handled by IKEv2. Such information can be the PREFERENCE which indicates how the Initiator

prefer this IP address, the CLASS which indicates the nature of the IP address (HoA, CoA, ...), or reachability information. As far as

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009

ADD and REMOVE actions are concerned, requirements 1, 2, 3 and 4 are fulfilled.

In a similar way as with the UPDATE\_SA\_ADDRESSES Notify Payload, if no IP parameter is provided in the data field, then the ADD or REMOVE action applies to all CHILD\_SA associated to the IKE\_SA. The IP address to be ADDED or REMOVED is the one placed in the IP header. As far as ADD and REMOVE actions are concerned requirements 10 and 11 are fulfilled.

Note that ADD and REMOVE actions do not apply to the IKE\_SA. In this draft we do not consider that the IKEv2 application implements Simultaneous IP Address Multihoming. This is for future development.

### [5.3.](#) Response to a CHECK Request

When the Responder receives an UPDATE, ADD or REMOVE action request with the CHECK bit set to 1, then the Responder is expected to provide information on whether the action can be performed, and if not why the action cannot be performed. If the action can be performed the Responder returns a response without any error Notify Payload. If the action cannot be performed because the IP address does not fit local policies, then a UNACCEPTABLE\_ADDRESS Notify Payload is sent. If the action cannot be performed because the IP address does not match the SPD, then a DOES\_NOT\_FIT\_SPD Notify Payload.

Such Notify Payloads fulfill requirement 5.

### [5.4.](#) SELECTING the SA

MOBIKE-X provides a mechanism to explicitly specify which SA or which IKE\_SA the action applies to. One or more SELECTOR Notify Payload can be used. If the Initiator does not specify any SELECTOR, then the default value for the SELECTOR is IKE\_SA\_AND\_CHILD\_SA, which means that the action MUST apply to the IKE\_SA and all associated CHILD\_SA(s).

Such mechanism fulfills requirements 9, 10, 11 and 5.

## [5.5.](#) Alternate IP Address

MOBIKE defines an `ADDITIONAL_IP4_ADDRESS` and `ADDITIONAL_IP6_ADDRESS` Notify Payloads so that a peer can advertise the other peer an IP address. This message specifies an Alternate IP Address intended to be used by the IKEv2 application. This means that the implicit selector value is `IKE_SA`. MOBIKE-X uses this Notify Payload, by considering that the specified IP address can also be associated to a

SA. The SAs are specified with the `SELECTORS` Notify Payloads. In that case, the IKEv2 application is not expected to do anything else than forwarding the information to the Upper Layer Protocols. MOBIKE defines `ADDITIONAL_IP*_ADDRESS` with the IP address value inside the data field of the Notify Payload. MOBIKE-X re-uses those messages, but fills the data field of the Notify Payload with the IP parameters.

Note that the use of two distinct `ADDITIONAL_IP*_ADDRESS` Notify Payload is not anymore necessary, since the type of the IP address is specified with the parameter type. Thus we define in this document the `ADDITIONAL_IP_ADDRESS` Notify Payload which is an alias for the `ADDITIONAL_IP6_ADDRESS`. Reasons are that we do not want to have two different Notify Payload for the same action.

Such mechanism fulfills requirements 7, 8 and 10.

Note that this draft does not specify nor provide mechanisms that either enable one peer to know if the alternate IP address is forwarded to the ULP, or that enables communications between the ULP and the IKEv2 application. In other words, it defines that `ADDITIONAL_IP_ADDRESS` Notify Payload can be associated to SA, and so SHOULD NOT be rejected by IKEv2 or generate any error message.

## [5.6.](#) MOBIKE Version

The protocol described in this draft is not fully compatible with MOBIKE. Furthermore, it provides more functionalities. Peers need to agree on which version of the protocol they understand. In this draft, the two possible protocols are MOBIKE or MOBIKE-X, and the choice is exclusive. Negotiation on the MOBIKE version is agreed by using the `MOBIKE_SUPPORT` Notify Payload. MOBIKE-X is designated by

using a version parameter in the data field. The version value considered for MOBIKE-X is 2.

The exchange defined in this draft is expected to support further MOBIKE versions. Thus we defined the MOBIKE\_UNSUPPORTED\_VERSION Notify Payload.

This mechanism fulfill requirements 12.

## [5.7.](#) Other Considerations

As defined in MOBIKE, MOBIKE-X considers that Mobility or Multihoming action can only be triggered by the Initiator.

MOBIKE deals with NAT, and we would like to take advantage of this work, even though we do not consider NAT in this paper.

## [6.](#) Notify Payload Description

### [6.1.](#) MOBIKE\_SUPPORTED Notify Payload

This message is described in MOBIKE [[RFC4555](#)]. MOBIKE-X uses versions parameters to specify which version is supported by the peer. We consider in this paper the version corresponding to MOBIKE as defined in [[RFC4555](#)] as version 1 and the MOBIKE-X version as described in this paper as version 2. A node that implements a MOBIKE-X of version equal or greater than 2 MUST specify the version numbers in its MOBIKE\_SUPPORTED Notify Payload. If no version is specified, then the node is assumed to support only MOBIKE of version 1.

In the IKE\_AUTH exchange, the MOBIKE\_SUPPORTED Notify Payload is sent by the Initiator. The Initiator sends an MOBIKE\_SUPPORTED Notify Payload with the version parameters of the MOBIKE versions it supports. There can be multiple version parameters and the Initiator expects that the Responder will choose one of them. When the Responder receives an MOBIKE\_SUPPORTED Notify Payload, and the Responder supports one of the proposed versions, it chooses one of those and indicates the chosen version with the version parameter in the MOBIKE\_SUPPORTED Notify Payload sent as a response. If none of the version is supported by the Responder, the Responder MUST send an MOBIKE\_UNSUPPORTED\_VERSION Notify Payload.

- Version : The supported version of MOBIKE-X. MOBIKE-X as defined in this draft is considered as version 2, MOBIKE as specified in [\[RFC4555\]](#) is considered as version 1.

At this level of the negotiation both peers have agreed on which MOBIKE version of the protocol they understand.

#### [6.2.](#) MOBIKE\_UNSUPPORTED\_VERSION Notify Payload

This Notify Payload is used to indicate that proposed versions by the Initiator are not supported. This message carries the version the Responder supports. If the Initiator also supports this version and has not already proposed it to the Responder, then it can restart the negotiation.

This Notify Payload is also used to cancel the use of MOBIKE-X, to go back to the initial state where MOBIKE is not supported by any of the peers. This is mainly to avoid ambiguous situation where one peer understands one thing but not the other. For that purpose, we use a specific version value NONE. The Notify Payload can carry the following options :

- Version : The considered version of MOBIKE-X. MOBIKE-X in this draft is considered as version 1. The NONE value is 255.

#### [6.3.](#) SELECTOR Notify Payload

A SELECTOR Notify Payload contains a list of parameters. A SA matches this SELECTOR payload if it matches all parameters within the SELECTOR Notify Payload. A SELECTOR Notify Payload is always followed either by a SELECTOR, an UPDATE\_SA\_ADDRESSES, an ADD\_SA\_ADDRESS, a REMOVE\_SA\_ADDRESS, or an ADDITIONAL\_IP\_ADDRESS Notify Payload. Those Notify Payloads are also called the action. When different SELECTOR Notify Payloads are grouped together before an action, we call this group a SELECTOR\_SET. An SA matches the SELECTOR\_SET when a match occurs with at least one of the SELECTOR payload. The SELECTOR Notify Payload can have the following parameters :

- SPI : By specifying the SPI the action applies to.

- IP : By specifying the IP address the action applies to. The IP parameter can provide information about the location (inner or outer IP address).
- IPSEC\_PROTO : By specifying the IPSEC\_PROTO the action applies to.
- IPSEC\_MODE : By specifying the IPSEC\_MODE, the applies to.
- ID : The Identification Number of the Notify Payload. It is optional. It is recommended to add this option to identify the SELECTOR\_SET. When an error occurs, the SELECTOR Notify Payload causing the error can be identified.

#### [6.4.](#) END\_OF\_SELECTOR Notify Payload

The END\_OF\_SELECTOR Notify Payload indicates the SELECTOR\_SET do not apply any more to the action that follows this Notify Payload. We do not expect any parameters being carried by this payload.

#### [6.5.](#) ADDITIONAL\_IP\_ADDRESS Notify Payload

ADDITIONAL\_IP\_ADDRESS Notify Payloads carry Alternate IP Addresses. ADDITIONAL\_IP\_ADDRESS Notify Payload applies to the IKE\_SA as specified in MOBIKE [[RFC4555](#)]. That is to say the carried IP address is used by the IKEv2 application, only. Alternate IP addresses are used only when the current IP Address is not working anymore.

When ADDITIONAL\_IP\_ADDRESS Notify Payload applies to a regular SA, then IKEv2 SHOULD forward the information to ULP. This draft does not provide any details on how communication is done between ULP and IKEv2. This is not considered in this paper.

The IP address is carried in the Notify Payload in the data field thanks to IP parameter. This is a main difference with MOBIKE [[RFC4555](#)] where the data field directly contains the value of IP address.

Thus the ADDITIONAL\_IP\_ADDRESS Notify Payload can carry the following parameters :

- IP : The IP address to be used as an Alternate IP Address.
- ID : The Identification Number of the Notify Payload. It is optional. It is recommended to add this option so that when an

error occurs, the Notify Payload Notify Payload causing the error can be identified.

IP parameters can carry information about the IP address thanks to the Multihoming Information Payload (MIP). Such information is intended to help the Responder to choose the best Alternate IP Address, in case of failure. Such information includes the REACHABILITY that indicates the REACHABILITY information of the IP address, the CLASS which indicates the nature of the IP address (HoA, CoA...), the PREFERENCE which is a non objective parameter provided by the Initiator.

The CHECK bit indicates if the Initiator expects the action to be performed or if the Initiator is only willing to check the action can be performed. If the Alternate IP Address does not match the local policies then an UNACCEPTABLE\_IP\_ADDRESS Notify Payload is sent back. If the Alternate IP Address does not fit the SPD, then a DOES\_NOT\_FIT\_SPD Notify Payload is sent back. Note that checks do not include Return Routability Check. If the user wants to check the reachability of the Alternate IP address, and if that is possible -- the Initiator is multihomed -- , then the Initiator MUST perform it and initiate the exchange. If NEW\_IP address matches both local policies and the SPD, then a INFORMATIONAL response is sent without error Notify Payloads.

SELECTORs Notify Payload MAY be used to specify which on which Security Association the IP address change MUST occur.

#### [6.6.](#) UPDATE\_SA\_ADDRESSES Notify Payload

UPDATE\_SA\_ADDRESSES Notify Payload indicates the Responder that the Initiator wants to replace OLD\_IP by NEW\_IP, in the selected SAs.

In MOBIKE, the Initiator uses only one IP address. Thus, OLD\_IP is the one associated to the peer in the IKE\_SA and the CHILD\_SA(s). The NEW\_IP is in the IP header of the IP packet carrying IKEv2 message. In MOBIKE-X, the Initiator can use simultaneous IP

addresses, thus we need to be able to explicitly specify in the Notify Payload, OLD\_IP and NEW\_IP.

So the difference with MOBIKE is that the UPDATE\_SA\_ADDRESSES Payload



can carry the following parameters :

- OLD\_IP : The replaced IP address. The parameter is a regular IP parameter, where we set the OLD bit to one. In case of tunnel mode Security Association, the I (inner) and the H (header) bit specifies if the OLD\_IP is the inner or outer IP address.
- NEW\_IP : The replacing IP address. The parameter is a regular IP parameter where we set the NEW bit to one.
- ID : The Identification Number of the Notify Payload. It is optional. It is recommended to add this option so that when an error occurs, the Notify Payload Notify Payload causing the error can be identified.

If NEW\_IP does not match the local policies then an UNACCEPTABLE\_IP\_ADDRESS Notify Payload is sent back. If NEW\_IP does not fit the SPD, then a DOES\_NOT\_FIT\_SPD Notify Payload is sent back. Note that checks do not include Return Routability Check. If the user wants to check the reachability of NEW\_IP, and if that is possible -- the Initiator is multihomed -- , then the Initiator MUST perform it and initiate the exchange.

The CHECK bit indicates if the Initiator expects the action to be performed or if the Initiator is only willing to check the action can be performed.

O, N, bits specify which IP is the OLD\_IP, and which IP is the NEW\_IP. H and I bits specify where in the SA the IP address is located. In SA using tunnel mode, the bit I indicates the IP address is located inside the tunnel (Inner), and H specifies the IP address is in the outer tunnel header (Header). In SA using transport mode, the H bit indicates the IP address is in the outer IP header. H stands for routed IP Header.

The OLD\_IP and NEW\_IP are not required. If there are omitted by the Initiator, then default values are considered.

SELECTORs Notify Payload MAY be used to specify which on which Security Association the IP address change MUST occur.

#### [6.7.](#) ADD\_SA\_ADDRESS Notify Payload

The ADD\_SA\_ADDRESS Notify Payload is very similar as the UPDATE\_SA\_ADDRESSES Notify Payload. The Initiator sends an ADD\_SA\_UPDATE Notify Payload so the Responder ADDs an IP address to

the SA designated by the SELECTORs. The IP address is designated by the IP parameter.

If IP parameter does not match the local policies then an UNACCEPTABLE\_IP\_ADDRESS Notify Payload is sent back. If the IP address does not fit the SPD, then a DOES\_NOT\_FIT\_SPD Notify Payload is sent back. Note that checks do not include Return Routability Check. If the user wants to check the reachability of the IP address, and that is possible -- the Initiator is multihomed -- , then the Initiator MUST perform it and initiates the exchange.

CHECK, H, I have the same signification as with the UPDATE\_SA\_ADDRESSES Notify Payload. The O bit and N bit are not considered, since only one IP address is being ADDED.

The ADD\_SA\_ADDRESS Notify Payload can carry the following parameters :

- IP : The IP address to be used as an alternate IP address.
- ID : The Identification Number of the Notify Payload. It is optional. It is recommended to add this option so that when an error occurs, the Notify Payload Notify Payload causing the error can be identified.

#### [6.8.](#) REMOVE\_SA\_ADDRESS Notify Payload

The REMOVE\_SA\_ADDRESS Notify Payload is very similar as the UPDATE\_SA\_ADDRESSES Notify Payload. The Initiator sends an REMOVE\_SA\_UPDATE Notify Payload so the Responder REMOVES an IP address to the SA designated by the SELECTORs.

If IP parameter does not match the local policies then an UNACCEPTABLE\_IP\_ADDRESS Notify Payload is sent back. If the IP address does not fit the SPD, then a DOES\_NOT\_FIT\_SPD Notify Payload is sent back. Note that checks do not include Return Routability Check. If the user wants to check the reachability of IP address, and if that is possible -- the Initiator is multihomed --, then the Initiator MUST perform it and initiate the exchange. When the Initiator requests a REMOVES on the last IP address of an SA, then the Responder REMOVES the IP address -- or not depending on the CHECK bit value --- and sends a warning Notify Payload CLOSING\_SA.

CHECK, H, I have the same signification as with the UPDATE\_SA\_ADDRESSES Notify Payload. The O bit and N bit are not considered, since only one IP address is being REMOVED.

The REMOVE\_SA\_ADDRESS Notify Payload can carry the following

parameters :

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009

- IP :    The IP address to be used as an alternate IP address.
- ID :    The Identification Number of the Notify Payload. It is optional. It is recommended to add this option so that when an error occurs, the Notify Payload Notify Payload causing the error can be identified.

## 7. Protocol Exchanges

This section provides a description on how Initiators and Responders are expected to behave when receiving or sending the different Notify Payloads involved in this paper. For clarification purpose we illustrate the behavior with state diagrams. State diagrams mostly rely on node's implementation. States and state diagrams are not normative. We also tried to point out different possible implementations.

### 7.1. Initial Exchange

#### 7.1.1. MOBIKE\_SUPPORTED Notify Payload

The initial exchange enables the Initiator to check if the Responder supports MOBIKE-X and negotiates a given version of MOBIKE-X with the Responder. The Initial exchange is derived from [\[RFC4555\]](#) and shows how NAT is considered. As mentioned in the Introduction section NAT is treated in a similar manner as it is in MOBIKE [\[RFC4555\]](#).

---

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009

Initiator	Responder
-----	-----
1) (IP_I1:500 -> IP_R1:500)	
HDR, SAi1, KEi, Ni -->	
N(NAT_DETECTION_SOURCE_IP),	
N(NAT_DETECTION_DESTINATION_IP) -->	
	<-- (IP_R1:500 -> IP_I1:500)
	HDR, SAR1, KER, Nr,
	N(NAT_DETECTION_SOURCE_IP),
	N(NAT_DETECTION_DESTINATION_IP)
2) (IP_I1:4500 -> IP_R1:4500)	
HDR, SK { IDi, CERT, AUTH,	
CP(CFG_REQUEST),	
SAi2, TSi, TSr,	
N(MOBIKE_SUPPORTED)} -->	
	<-- (IP_R1:4500 -> IP_I1:4500)
	HDR, SK { IDr, CERT, AUTH,
	CP(CFG_REPLY),
	SAr2, TSi, TSr,
	N(MOBIKE_SUPPORTED) }

#### Initial exchange

As specified in [[RFC4555](#)], in order to be able to go through NAT, when using MOBIKE, the port used is 4500. The Initiator sends a MOBIKE\_SUPPORTED Notify Payload in which it indicates the proposed versions. The Notify Payload SHOULD have its critical flag on. The version of MOBIKE considered in [[RFC4555](#)] is version 1, and the version of MOBIKE considered in this draft is version 2.

If the Responder does not support MOBIKE, it does not understand the Notify Payload, the exchange is described in [section 2.5 of \[RFC4306\]](#). If the Notify Payload has been set to critical by the Initiator, then the Responder MUST send as specified in [\[RFC4306\]](#) a UNSUPPORTED\_CRITICAL\_PAYLOAD Notify Payload. The Initiator knows that the Responder does not support any version of MOBIKE or MOBIKE-X. If the Notify Payload does not have its critical flag, then the Responder does not send any response and the Initiator SHOULD wait for a given time before assuming the Responder does not support MOBIKE or MOBIKE-X.

If the Responder supports MOBIKE, but does not support MOBIKE-X : When it receives a MOBIKE\_SUPPORTED Notify Payload, it does not understand parameters specified in data field. If the Responder wants to activate MOBIKE, then it sends a MOBIKE\_SUPPORTED Notify

Payload with no version specified in the data field of the Notify Payload. When the Initiator receives this Notify Payload, it understands that with no version specified, the Responder does only understand MOBIKE as specified in [\[RFC4555\]](#). If the Responder does not want to activate MOBIKE, it MUST check the critical flag of the received Notify Payload and either MUST send a UNSUPPORTED\_CRITICAL\_PAYLOAD Notify Payload, or no response at all. When the Initiator receives the UNSUPPORTED\_CRITICAL\_PAYLOAD Notify Payload or no response, it understands that the Responder does not support MOBIKE.

Note that the Initiator cannot make the distinction between a Responder that does not want to activate MOBIKE, or a responder do not support MOBIKE.

If the Responder supports MOBIKE-X and does not want to activate MOBIKE or MOBIKE-X : When it receives the MOBIKE\_SUPPORTED Notify Payload, the Responder MUST check the data field and check if there are any versions proposed. If no version is proposed, then the Initiator only supports MOBIKE. The Responder should check the critical flag of the Notify Payload and either sends a UNSUPPORTED\_CRITICAL\_PAYLOAD Notify Payload, or no response. If the MOBIKE\_SUPPORTED Notify Payload carries version parameters in its data payload, then the Responder knows that the Initiator does support MOBIKE-X. It SHOULD send a MOBIKE\_UNSUPPORTED\_VERSION Notify Payload with a version set to NONE. The Responder knows the

Initiator supports MOBIKE-X since the MOBIKE\_SUPPORTED Notify Payload has version parameters. The Initiator will then understand the response. This case is an alternative to simulate the case the Responder does not even understand MOBIKE and sends only a UNSUPPORTED\_CRITICAL\_PAYLOAD. It clearly states the Responder does support MOBIKE-X, but does not want to activate it.

If the Responder supports MOBIKE-X and does only want to activate MOBIKE as defined in [\[RFC4555\]](#) : When it receives the MOBIKE\_SUPPORTED Notify Payload, it MUST check the MOBIKE\_SUPPORTED Notify Payload carries version parameters in its data field. If no version number is found, the Responder assumes the Initiator does only supports MOBIKE as defined in [\[RFC4555\]](#). It SHOULD send a MOBIKE\_SUPPORTED Notify Payload. If the data field carries version numbers, then the Responder knows the Initiator supports MOBIKE-X. The Responder should check version 1 is proposed. If version 1 is proposed, then the Responder MUST send a MOBIKE\_SUPPORTED Notify Payload with the chosen version, i.e. version 1. If the version 1 is not proposed, then the Responder MUST send a MOBIKE\_UNSUPPORTED\_VERSION Notify Payload indicating the accepted version by the Responder. In that specific case the Notify Payload data field MUST carry version 1. When the Initiator receives the

response it is up to him to decide to restart a MOBIKE\_SUPPORTED exchange with the version 1 proposed or to consider that neither MOBIKE nor MOBIKE-X are activated.

If the Responder supports MOBIKE-X and wants to activate MOBIKE-X : When it receives the MOBIKE\_SUPPORTED Notify Payload, it MUST check the data field of the Notify Payload carries version numbers in the data field. If no version numbers are specified, then the Initiator supports MOBIKE, but does not support MOBIKE-X. The Responder can choose to enable MOBIKE or not. This case is similar as the one when the Responder supports MOBIKE-X but only wants to activate MOBIKE. If the Initiator supports MOBIKE-X, the Responder should check version number 2 is proposed by the Initiator. If version number 2 is not proposed, the Responder MUST send a MOBIKE\_UNSUPPORTED\_VERSION Notify Payload with version set to 2. If version number 2 is proposed then the Responder MUST send a MOBIKE\_SUPPORTED Notify Payload with the chosen version number, that is to say with version number 2. In this case MOBIKE-X is considered activated both on the Initiator side and on the Responder side.

NOTE 1 : The case where the Initiator supports MOBIKE-X and starts a MOBIKE\_SUPPORTED exchange with the Responder which does not supports is specified in [\[RFC4306\] section 2.5](#).

IKEv2 adds a "critical" flag to each payload header for further flexibility for forward compatibility. If the critical flag is set and the payload type is unrecognized, the message MUST be rejected and the response to the IKE request containing that payload MUST include a Notify payload UNSUPPORTED\_CRITICAL\_PAYLOAD, indicating an unsupported critical payload was included. If the critical flag is not set and the payload type is unsupported, that payload MUST be ignored.

NOTE 2 : There is one corner case. Consider the Initiator supports MOBIKE-X, and wants to activate MOBIKE-X but does NOT want MOBIKE, and the Responder does only supports MOBIKE. In that case, the Initiator sends a MOBIKE\_SUPPORTED Notify Payload to the Responder. The Responder supports only MOBIKE, and sends back a MOBIKE\_SUPPORTED Notify Payload. Responder and Initiators have agreed on version 1 of MOBIKE. MOBIKE as described in [\[RFC4555\]](#) does not provides any Notify Payload to remove the MOBIKE option. In other words, there is no equivalent of MOBIKE\_UNSUPPORTED\_VERSION in MOBIKE. So the situation is MOBIKE as been "agreed" between the Initiator and the Responder, the Initiator wants MOBIKE-X and cannot remove MOBIKE. Consequences are that the Initiator cannot use MOBIKE-X functionalities, but can be requested by the Responder some MOBIKE functions. If that bothers the Initiator, then it has to restart an

IKE\_SA\_INT exchange without sending MOBIKE\_SUPPORTED Notify Payload.

#### [7.1.2](#). MOBIKE\_UNSUPPORTED\_VERSION Notify Payload

The MOBIKE\_UNSUPPORTED\_VERSION is a MOBIKE-X specific Notify Payload. It MUST be sent by the Responder when it has checked the Initiator supports MOBIKE-X.

The MOBIKE\_UNSUPPORTED\_VERSION Notify Payload can be received during the MOBIKE\_SUPPORTED negotiation. The Responder as well as the Initiator supports MOBIKE-X but the Responder does not support the version of MOBIKE-X proposed by the Initiator. The other case this

Notify Payload can be used is when the Initiator and the Responder have already agreed on which version of MOBIKE-X to activate, and one of them wants to deactivate the used of MOBIKE or MOBIKE-X.

Suppose the Initiator has started its MOBIKE version negotiation, and both Initiator and Responder support MOBIKE-X. When the MOBIKE\_UNSUPPORTED\_VERSION Notify Payload is received, the Initiator knows the previously proposed versions are not supported by the Responder. The Initiator can read the version carried by the MOBIKE\_UNSUPPORTED\_VERSION. If the version is NONE, it means the Responder supports MOBIKE-X but does not want to activate the MOBIKE or MOBIKE-X extension.

If the Responder supports another version than the one proposed by the Initiator, then it sends its version number into the MOBIKE\_UNSUPPORTED\_VERSION, the Initiator can then check if this version has not been previously been sent and if it supports this version. It can then restart a negotiation and send a MOBIKE\_SUPPORTED Notify Payload with the version supported by the Responder. This gives the opportunity to the Initiator to have a preferred list sent in the first MOBIKE\_SUPPORTED Notify Payload, and a backup of other versions in case the Responder does not support the versions of the "preferred list". The Responder is expected to send a MOBIKE\_SUPPORTED Notify Payload with the proper version. If not the Initiator the Responder is probably badly configured and MOBIKE-X extension is not activated.

The MOBIKE\_UNSUPPORTED\_VERSION Notify Payload can also be received by any of the peer while MOBIKE-X has been negotiated between the Initiator and the Responder. In this situation, the version MUST be set to NONE, and it means that MOBIKE-X is not active anymore.

### 7.1.3. Initial exchange state diagrams

The diagrams and states descriptions provided in this section are not normative. There purpose is to clarify the text description. It

might help for some implementations but implementations are not expected to implement them.

The different states considered in this paper are :



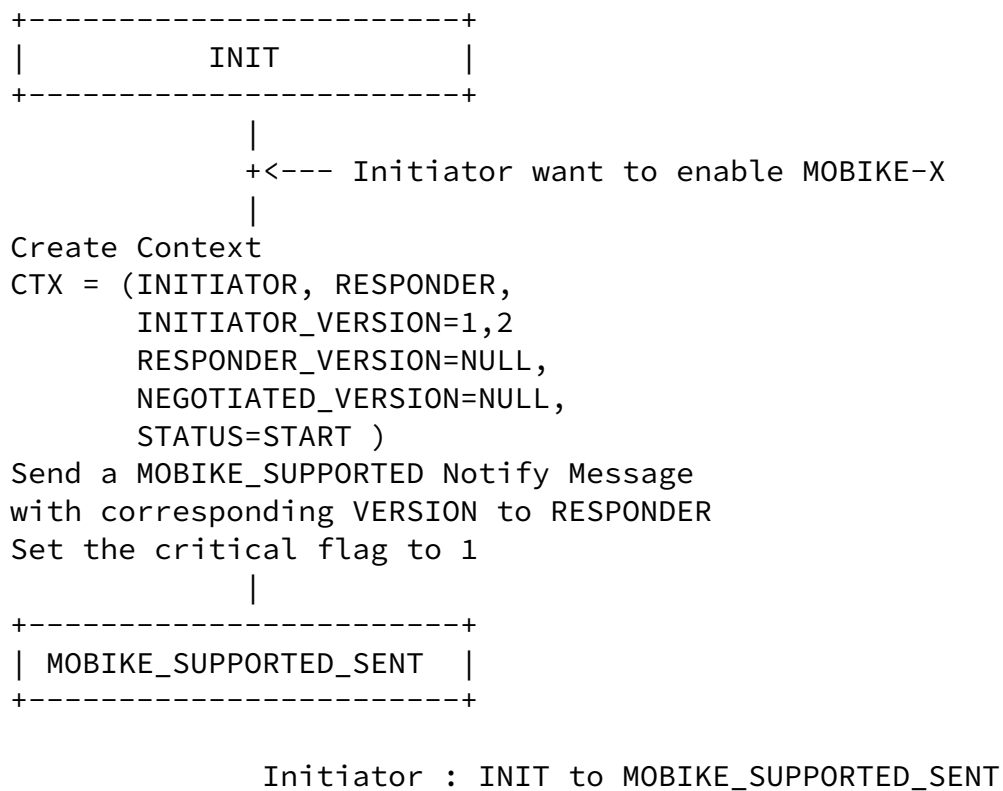
- INIT : This state corresponds to the starting point when no MOBIKE-X negotiation has been initiated.
- MOBIKE\_SUPPORTED\_SENT : This Initiator is in this state when it has sent a MOBIKE\_SUPPORTED Notify Payload. The Initiator is then waiting for a response.
- MOBIKE\_SUPPORTED : This state ends a MOBIKE\_SUPPORTED negotiation. Initiator and the Responder agreed on activating version 1, that is to say MOBIKE as specified in [\[RFC4555\]](#). MOBIKE-X as specified in this paper has not been agreed.
- MOBIKE-X\_SUPPORTED : This state ends a MOBIKE\_SUPPORTED negotiation. Initiator and the Responder agree on activating version greater than 1, that is to say MOBIKE-X as specified in this paper is activated by the Initiator and the Responder. When MOBIKE-X is supported, then MOBIKE is also supported.
- MOBIKE-X\_UNSUPPORTED : This state ends the MOBIKE\_SUPPORTED negotiation. Initiator and Responder did not agree on the activation of MOBIKE-X as described in this paper. This state means the Initiator and the Responder agreed on MOBIKE but not MOBIKE-X.
- MOBIKE\_UNSUPPORTED : This state ends the MOBIKE\_SUPPORTED negotiation. Initiator and Responder did not agree on the activation of MOBIKE-X as described in this paper nor MOBIKE as described in [\[RFC4555\]](#). This state can also be considered as similar to the INIT state, except that one negotiation has been attempted.

In the state diagram, we used a context called CTX. This is also not normative. The Context contains the following information :

- INITIATOR The necessary information to identify the Initiator part of the MOBIKE-X negotiation.
- RESPONDER The necessary information to identify the Responder part of the MOBIKE-X negotiation.
- INITIATOR\_VERSION MOBIKE-X versions proposed by the Initiator to the Responder.
- RESPONDER\_VERSION MOBIKE-X versions proposed by the Responder to the Initiator.
- NEGOTIATED\_VERSION MOBIKE-X versions agreed by the Responder and the Initiator.

- STATUS The status of the negotiation the different values can be START, MOBIKE, MOBIKE-X or NULL.

For simplification, we assume in the Initiator diagram that the Initiator supports MOBIKE-X and proposed to the Responder version 1 (MOBIKE) and version 2 (MOBIKE-X). We also assume in the Responder diagram that if the Responder supports MOBIKE-X, then it choose MOBIKE-X rather than MOBIKE. The Responder state diagram does not have MOBIKE\_UNSUPPORTED or MOBIKE-X\_UNSUPPORTED state, and when a negotiation fails, the Responder does not keep any context. This is an implementation consideration, and we thought that keeping a context to every negotiation could expose the Responder to DoS.



Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009

```

+-----+
| MOBIKE_SUPPORTED_SENT |          INITIATOR, CTX
+-----+

      |          NO
MESSAGE from CTX.RESPONDER ? --TIMEOUT--+-----+
      | YES          YES |          |
MESSAGE.type ==          -----+ CTX.STATUS=NULL
UNSUPPORTED_CRITICAL_PAYLOAD ?          |
      | NO          +-----+
MESSAGE.type = MOBIKE_SUPPORTED && ---+ | MOBIKE_UNSUPPORTED |
MESSAGE has no VERSION ? YES | +-----+
NO          | NO          +-----+
+-- MESSAGE.type == MOBIKE_SUPPORTED &&          v
| MESSAGE.VERSION in          CTX.RESPONDER_VERSION=1
| CTX.INITIATOR_VERSION ?          CTX.STATUS=MOBIKE
|          | YES          CTX.NEGOTIATED_VERSION=1
| CTX.STATUS = MOBIKE-X          |
| CTX.NEGOTIATED_VERSION =          +-----+
| MESSAGE.VERSION          | MOBIKE-X_UNSUPPORTED |
| CTX.RESPONDER_VERSION=MESSAGE.VERSION +-----+
|          |
| +-----+
| | MOBIKE-X_SUPPORTED |
| +-----+
|
+--> MESSAGE.type == MOBIKE_UNSUPPORTED_VERSION
      |          NO
MESSAGE.VERSION == NONE ? -----+
      | YES          NO          v
CTX.RESPONDER_VERSION=1,2 +----- MESSAGE.VERSION == 1 ?
CTX.STATUS=NULL          |          | YES
          |          CTX.RESPONDER_VERSION=1,2
          |          CTX.STATUS=MOBIKE
          |          |
          +-----+          +-----+
          | MOBIKE_UNSUPPORTED |          | MOBIKE-X_UNSUPPORTED |
          +-----+          +-----+
          +-----+          +-----+
          v
CTX.RESPONDER_VERSION=1,2
CTX.STATUS=NULL
          |
+-----+
| MOBIKE_UNSUPPORTED |
+-----+

```

-----+

Initiator : MOBIKE\_SUPPORTED\_SENT to MOBIKE\*\_SUPPORTED

Migault

Expires March 5, 2010

[Page 24]

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009

```

+-----+
|          INIT          |
+-----+

|
|
MESSAGE received && NO +-----+
MESSAGE.type == MOBIKE_SUPPORTED ? --->|          INIT          |
| YES NO +-----+
RESPONDER supports MOBIKE ? -----+
| YES NO v
RESPONDER supports MOBIKE-X (')?-----+ Send UNSUPPORTED_CRITICAL_PAYLOAD
| YES NO|
Does MESSAGE carries VERSION (*)?-----+ +-----+
| YES | |          INIT          |
MESSAGE.VERSION supported ? | +-----+
| YES | NO+-----+
Chose a VERSION +-----+ v
Send MOBIKE_SUPPORTED | Send MOBIKE_SUPPORTED -no version
MESSAGE.VERSION | Create CTX(
Create Context | INITIATOR, RESPONDER
CTX = ( | INITIATOR_VERSION=1
INITIATOR, RESPONDER | RESPONDER_VERSION=[1'] [1, 2*]
INITIATOR_VERSION=MESSAGE.VERSION, | NEGOTIATED_VERSION=1
RESPONDER_VERSION=VERSION, | STATUS=MOBIKE)
NEGOTIATED_VERSION=VERSION, |
STATUS=MOBIKE-X ) | +-----+
| |          MOBIKE_SUPPORTED |
+-----+ | +-----+
|          MOBIKE-X_SUPPORTED | +-----+
+-----+ |
| v
| Send UNSUPPORTED_MOBIKE_VERSION
| with supported VERSIONs
|
+-----+
|          INIT          |
+-----+

```

+-----+

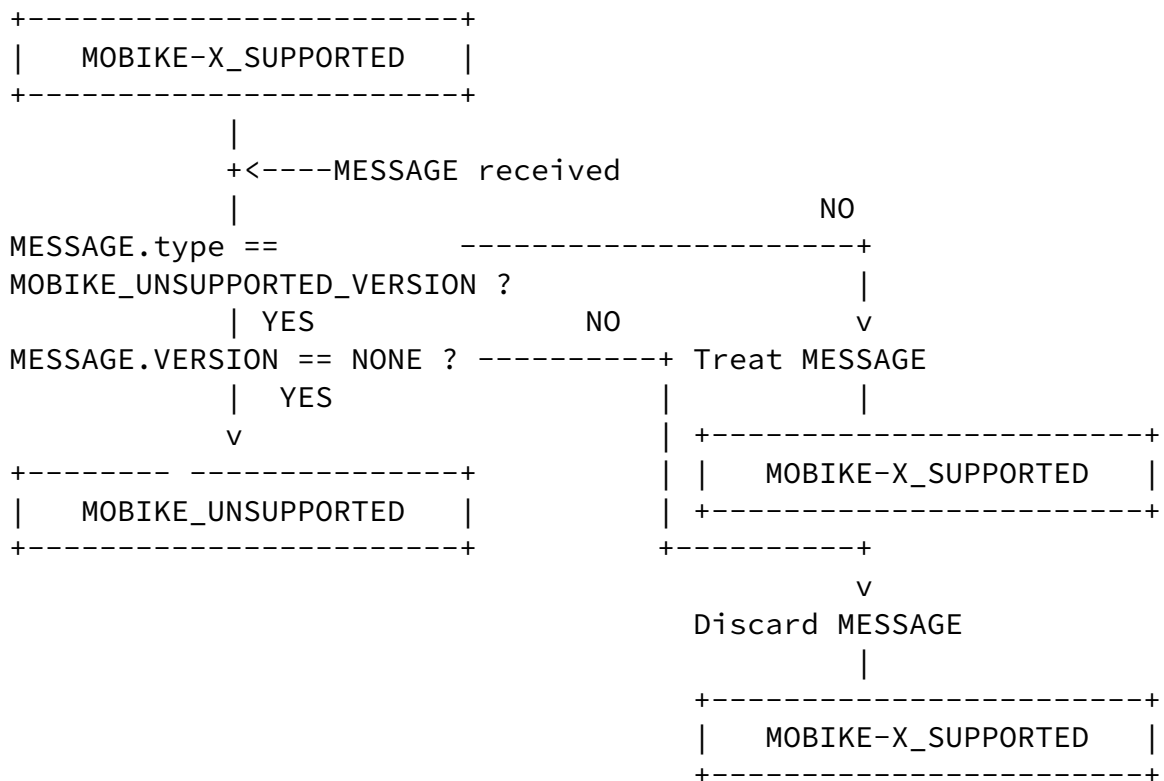
Responder : INIT to MOBIKE\*\_SUPPORTED

Migault

Expires March 5, 2010

[Page 25]

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009



MOBIKE-X\_SUPPORTED to MOBIKE\_UNSUPPORTED

## [7.2.](#) Alternate IP Address Exchange

When the Initiator wants to inform the Responder that it has an Alternate IP address, the Initiator sends an ADDITIONAL\_IP\_ADDRESS

Notify Payload. In MOBIKE [[RFC4555](#)], this case is described in [section 3.6](#). More specifically the data field contains the value of the IP address.

In MOBIKE-X, the main difference is that we use parameter payload that specifies the type of the IP address. Furthermore, the IP parameter provides also some information on the IP addresses. The reason is that with multihoming, the Initiator might have more than one alternate IP address. If the current IP address is not in use, then the Responder has to choose the more adequate IP address. Multihoming Information Payload (MIP) associated to the IP address are intended to provide input for the Responder.

As in the MOBIKE [[RFC4555](#)] document, ADDITIONAL\_IP\_ADDRESS Notify Payload contains only one IP address. More IP addresses could have been placed, but it is better for logs to have elementary actions. One difference with MOBIKE [[RFC4555](#)] is that the type of the expected IP address is specified in the action Notify Payload. In fact, if IPv6 (resp IPv4) Alternate IP Addresses have to be sent, then an ADDITIONAL\_IP6\_ADDRESS (resp. ADDITIONAL\_IP4\_ADDRESS) Notify Payload

is used. MOBIKE-X does not need different Notify Payload since the type of the IP address is defined in the IP address parameter. MOBIKE-X only uses the ADDITIONAL\_IP6\_ADDRESS notify Payload, and it is called ADDITIONAL\_IP\_ADDRESS in this document.

When the Responder receives an ADDITIONAL\_IP\_ADDRESS with a CHECK bit set to 0, the Responder MUST clear the list of Alternate IP address, and fill the list with the IP addresses provided in the multiple ADDITIONAL\_IP\_ADDRESSES Notify Payloads. In other words, Initiator can only send to full list and there is not incremental way to provide / remove Alternate IP Addresses. This mechanism is the one already existing in MOBIKE [[RFC4555](#)].

This draft only cares about Alternate IP Address associated to the IKE\_SA. When an Alternate IP address is associated to an SA, IKEv2 application MUST forward the information to the ULP. How this is performed is beyond the scope of this document. In this document we mainly consider the Initiator sends an Alternate IP address to the IKEv2 application of the Responder.

If the Initiator sends an ADDITIONAL\_IP\_ADDRESS Notify Payload with

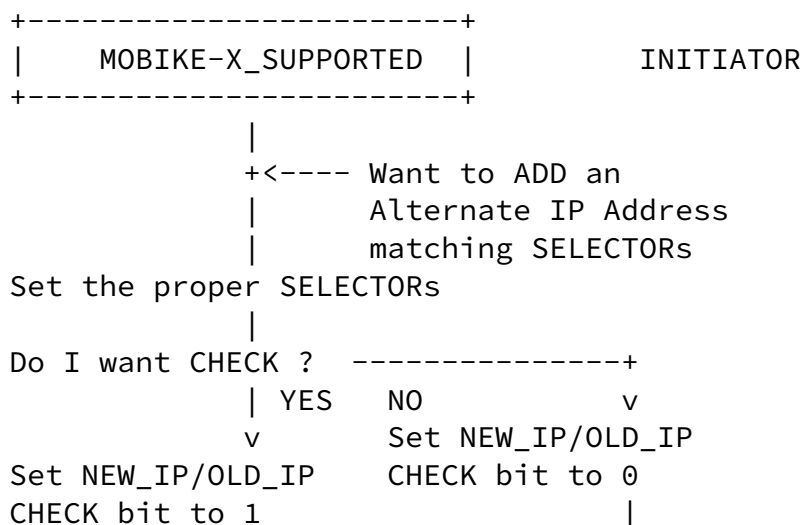
the CHECK bit set to 0, then the Initiator does not request for verifications. The Responder adds the IP address into the list of alternate IP addresses.

If the Initiator wants to CHECK an alternate IP address is valid, then it MUST set the CHECK bit to 1. When the Responder receives an ADDITIONAL\_IP\_ADDRESS Notify Payload with the CHECK bit set to one, the Responder checks the Alternate IP address matches local policies. If the Alternate IP address does not match the local policies, then the Responder MUST send an UNACCEPTABLE\_IP\_ADDRESS Notify Payload. The Responder also checks the Alternate IP address matches with the SPD. If the Alternate IP address does not fit the SPD, then a DOES\_NOT\_FIT\_SPD Notify Payload MUST be sent. Note that with IKE\_SA, any IP address matches the SP policy.

The Initiator indicates the Alternate IP Address thanks to IP parameter. The Initiator MAY set the PREFERENCE, CLASS fields.

In this document Alternate IP addresses have their IP parameters payload with the O, N, I bits values set to 0 and the H bit set to 1.

### [7.3.](#) Alternate IP Address Exchange State Diagram



```

      |
      +<-----+
      |
Set ID parameter
Set the I, O, N to 0
Set the H bit to 1
Send it to Responder
      |
+-----+
| ADDITIONAL_IP_ADDRESS_SENT |
+-----+

```

Initiator : Sending SA\_UPDATE\_ADDRESSES

```

+-----+
| ADDITIONAL_IP_ADDRESS_SENT |          INITIATOR
+-----+
      |
      +<---- RESPONSE is received
      |          from Responder
Does RESPONSE carries ERROR  -----+
      | YES                NO  v
The Alternate IP              The Alternate IP
Address cannot be used        Address can be used.
Eventually forward            Eventually ACKnowledge
ERROR to ULP                  it to ULP
      |
      +<-----+
      |
+-----+
|   MOBIKE-X_SUPPORTED   |
+-----+

```

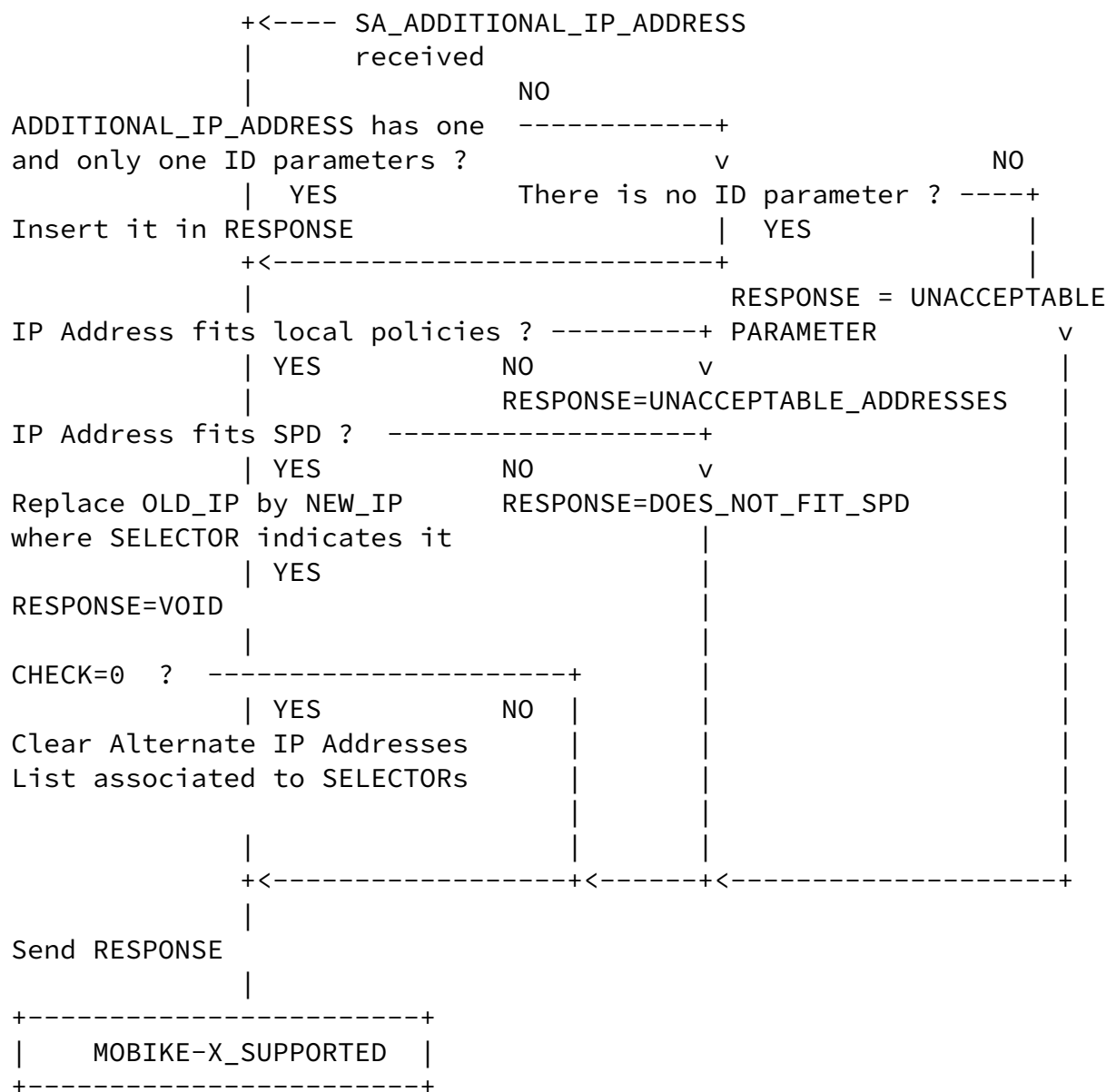
Initiator : Going back to MOBIKE-X\_SUPPORTED State

```

+-----+
|   MOBIKE-X_SUPPORTED   |          RESPONDER
+-----+
      |

```





Responder : Receiving ADDITIONAL\_IP\_ADDRESS

#### [7.4.](#) UPDATE Exchange

The figure below illustrates the different possible IPsec configuration between the responder and the Initiator. The figure represents the different IKE\_SA negotiated by one of the peer. The Initiator can have various IKE\_SA to which are associated different

SAs.

```
:
+-IKE_SA_0
|   iID: INITIATOR (iIP)
|   rID: RESPONDER (rIP_1, ...,IP_n)
|   |
|   +-SA_1 : mode=Transport
|   :       iID: iIP  rID: rIP_1
|   +-SA_j : mode=Transport
|   :       iID: iIP  rID: rIP_k
|   +-SA_m : mode=Tunnel
:           iID: iIP_virtual rID: rIP_1
:           Tunnel header :
:               iID:iIP rID:rIP_1
:
+-IKE_SA_l
|   iID: INITIATOR (iIP)
|   rID: RESPONDER (rIP_1, ...,IP_n)
|   |
|   +-SA_1 : mode=Transport
|   :       iID: iIP  rID: rIP_1
|   +-SA_p : mode=Transport
|   :       iID: iIP  rID: rIP_k
|   +-SA_q : mode=Tunnel
:           iID: iIP_virtual rID: rIP_1
:           Tunnel header :
:               iID:iIP rID:rIP_1
```

#### IKE configuration

The UPDATE\_SA\_ADDRESSES Notify Payload already exists in MOBIKE [[RFC4555](#)]. In MOBIKE OLD\_IP and NEW\_IP are implicit. In MOBIKE-X also they can be implicitly or explicitly specified. OLD\_IP and NEW\_IP are specified with the IP parameter. Distinction between OLD and NEW is done thanks to the O and N bit. According to the IP parameter in the data field we consider the following cases :

- NO IP parameter : is specified. Then the NEW\_IP is the IP address in the IP header carrying the Notify Payload. The OLD\_IP are all IP addresses associated to the Initiator. In SAs using the tunnel mode, only the outer IP address is considered.

---

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009

- A single OLD\_IP parameter : is specified. Then the NEW\_IP address is the one in the IP header of the message carrying the Notify Payload. OLD\_IP is specified, and replaced by NEW\_IP. The H and I bits of the OLD\_IP specifies where in the SA the change occurs.
- A single NEW\_IP parameter : is specified. Then OLD\_IP are all IP address associated to the Initiator. OLD\_IP are replaced by the specified NEW\_IP address. The H and I bits in the OLD\_IP specifies where the update MUST be done.
- One OLD\_IP and one NEW\_IP parameters : are specified. OLD\_IP MUST be replaced by NEW\_IP. The H and I bits MUST be the same on both OLD\_IP and NEW\_IP. Those values specify where the change occurs. If the H and I bits are not the same, then a UNACCEPTABLE\_PARAMETERS Notify Payload MUST be sent as a response.

In any other cases, an UNACCEPTABLE\_PARAMETERS Notify Payload MUST be sent.

[Section 3.5 of \[RFC4555\]](#) provides a description on how to change an IPsec SA in the tunnel mode case. The Initiator decides if a return routability check needs to be performed before updating the IKE\_SA and SAs or not. Then updates the IKE\_SA with the new IP address and set the "pending flag" of the IKE\_SA. If The Initiator does not require the return routability check to be performed before the update, then, the Initiator updates the SAs and the IKE\_SA. When updating the SAs, the Initiator should also consider NAT traversal mechanisms and in some cases enables UDP encapsulation of outgoing ESP packets as well as NAT-Keepalive packets. The Initiator MUST retransmit requests for which it has not received any responses with the updated IKE\_SA, that is to say with the new address, send a UPDATE\_SA\_ADDRESSES Notify Payload and clear the "pending update" flag.

MOBIKE [\[RFC4555\]](#) specifies that when the Responder receives the UPDATE\_SA\_ADDRESSES Notify Payload, it checks if that is the latest one received. The main reasons to that check was that peers were only using one IP address always targeted by the UPDATE action. Such a check SHOULD NOT be performed with MOBIKE-X, since for a given Security Association different IP addresses can be updated independently. Some checks can be performed, but in this paper we believe the easiest way is to performed actions as they are being requested.

The Responder considers the specified NAT options, checks the new IP address matches the local policy. If local policies do not match then the Responder MUST send a UNACCEPTABLE\_ADDRESSES. If policies match, then the Responder updates the IKE\_SA with the IP address

taken from the IP header, send a response. Optionally it can perform a return routability check before updating the SAs. When updating the SAs, NAT options MUST be carefully considered.

When the Initiator receives the response; it checks the considered UPDATE\_SA\_ADDRESSES is still to be considered. If no UNEXPECTED\_NAT\_DETECTED or UNACCEPTABLE\_ADDRESSES Notify Payload are sent, the Initiator updates the SAs if not previously done, and take into account the NAT considerations.

As mentioned in [section 3.5 of \[RFC4555\]](#), the Responder does not change the IPsec SA without receiving a UPDATE\_SA\_ADDRESSES Notify Payload unless the Initiator is unreachable. In that case, the Responder can use an Alternate IP Address.

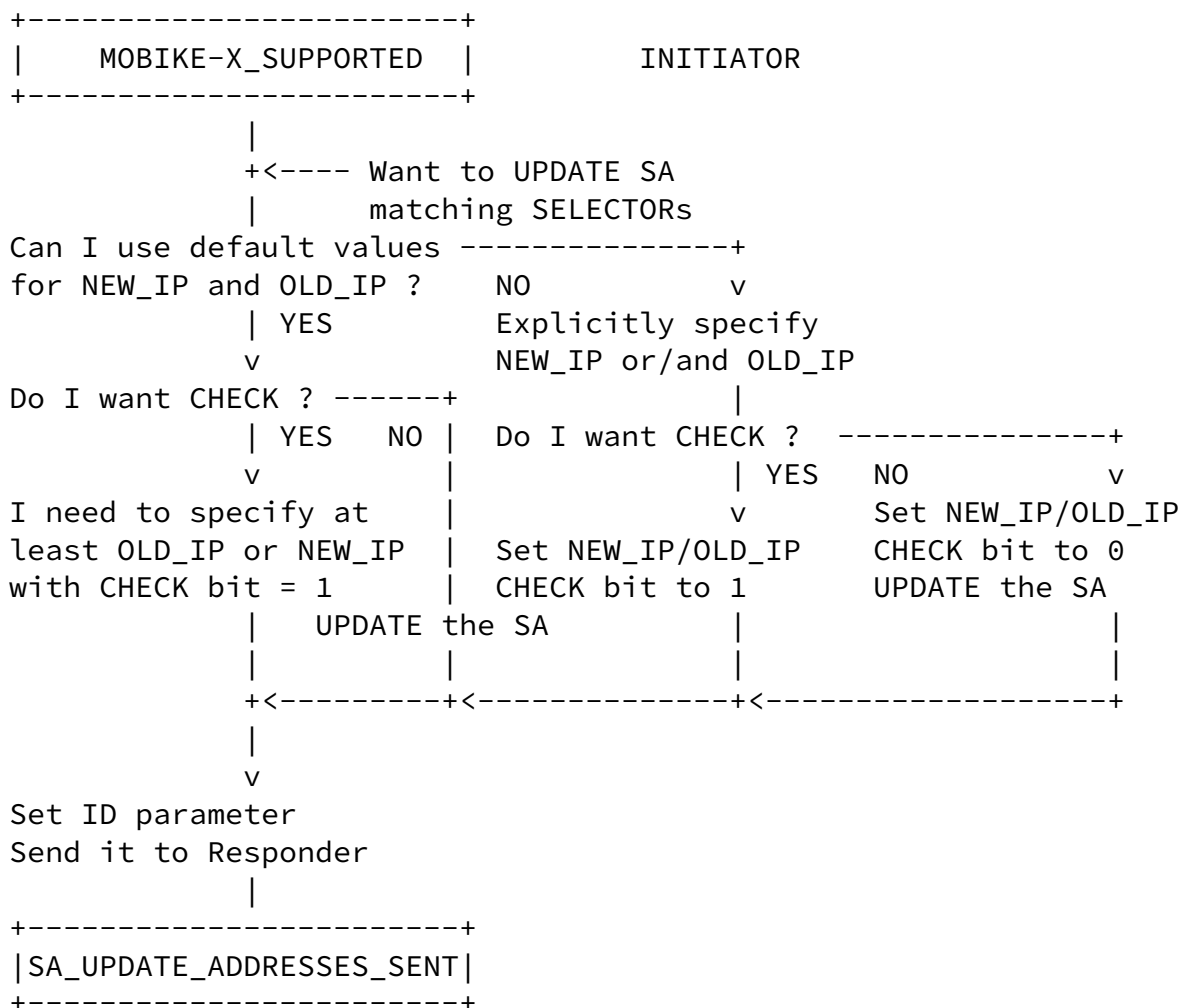
Updating a Security Association in a transport mode is done in a similar manner as it is done for a tunnel mode as described in in [\[RFC4555\]](#). The difference is that updating a tunnel header requires to change one parameter of the SA. This requires to check if that new IP address matches local policies. Changing the IP address of a transport mode SA requires also to check and eventually update the Security Policy Database as well as modifying the binding between the SPD and the SA. If the update cannot be performed because the SPD does not allow it, then the Responder MUST send a DOES\_NOT\_FIT\_SPD Notify Payload. The Initiator MUST try with other IP addresses, so to enable the mobility.

The Initiator can also send an UPDATE\_SA\_ADDRESSES Notify Payload to check wherever the update is possible or not. This is done thanks to the CHECK bit set to one. The CHECK bit is on the IP parameters. When no IP parameters are specified, then the Responder considers that CHECK is set to 0 and that the update MUST occur. When only one IP parameter is specified, then the CHECK bit value is explicitly set by the Initiator. When two IP parameters are specified, then the CHECK value of both parameters MUST be the same. If not, a UNACCEPTABLE\_PARAMETERS Notify Payload MUST be sent. If both CHECK

values are set to 1, then the update MUST NOT occurs. When both of them are set to 0, and the NEW\_IP addresses matches both local and Security Policies, then the update MUST be performed.

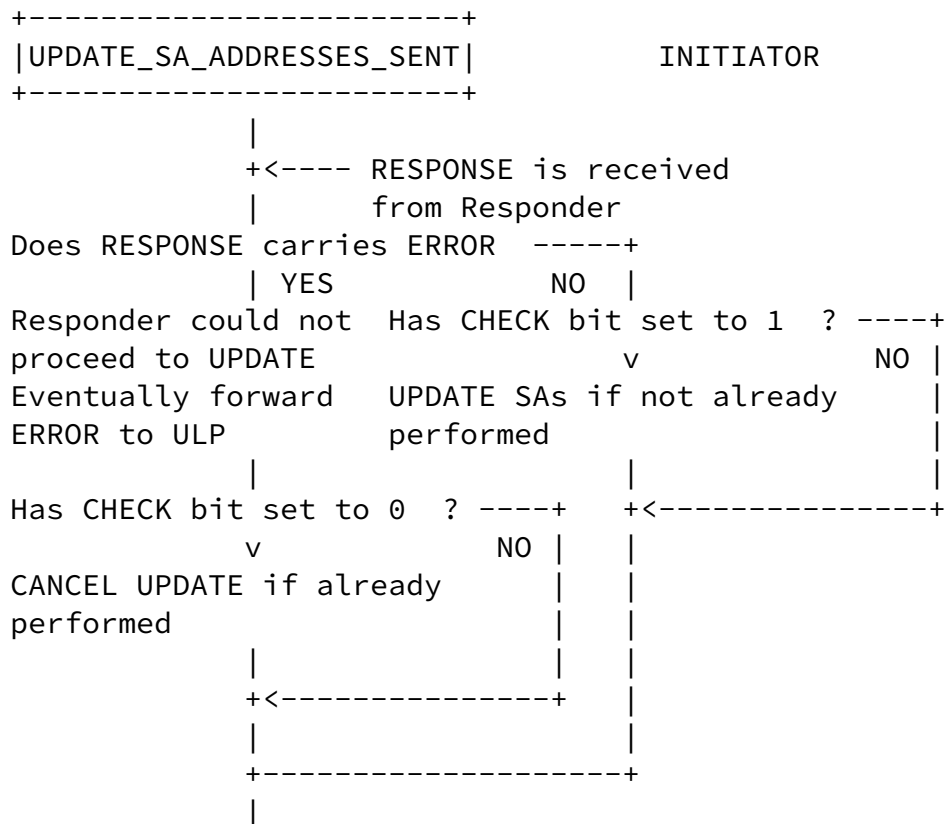
## 7.5. UPDATE Exchange State Diagram

The state diagram is not normative, and is provided only for clarification purposes. In that state diagram we assumed the Initiator specifies the NEW\_IP and OLD\_IP addresses when it is multihomed. Deciding when the OLD\_IP and NEW\_IP or the ID parameters SHOULD be added depends on the local hosts policies.



Initiator : Sending SA\_UPDATE\_ADDRESSES

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009



```
+-----+
|      MOBIKE-X_SUPPORTED      |
+-----+
```

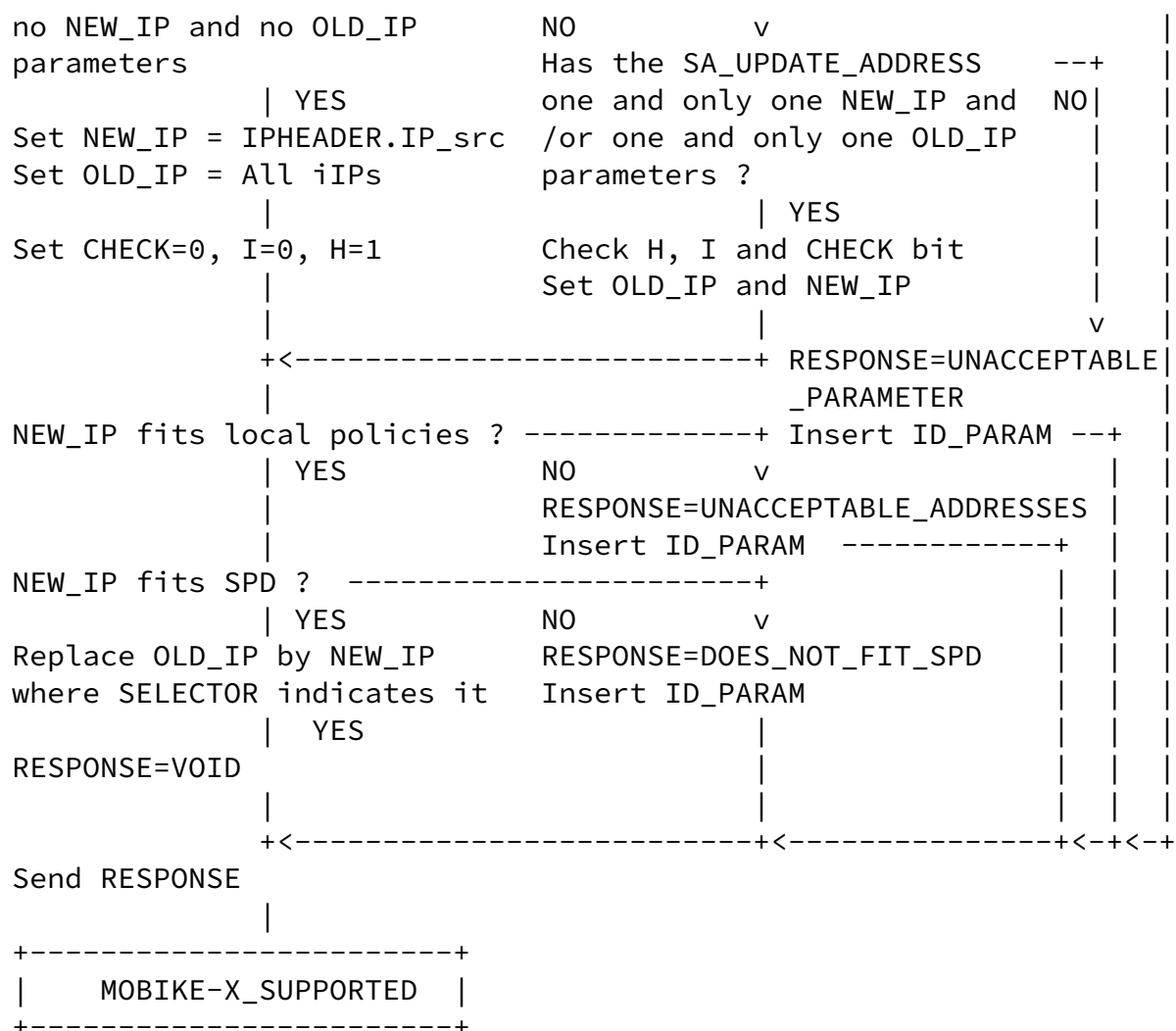
Initiator : Going back to MOBIKE-X\_SUPPORTED State

```

+-----+
|      MOBIKE-X_SUPPORTED      |      RESPONDER      |
+-----+

|
| +<----- UPDATE_SA_ADDRESSES
|           received
|
| NO
|
UPDATE_SA_ADDRESSES has one -----+
and only one ID parameters ?          v          NO
ID_PARAM=ID                          There is no ID parameter ? ----+
| YES                                ID_PARAM=VOID                      v
|                                     YES | RESPONSE = UNACCEPTABLE
| +<-----+
|                                     _PARAMETER
|                                     Insert ID_PARAM -----+
Has the SA_UPDATE_ADDRESSES -----+

```



Responder : Receiving UPDATE\_SA\_ADDRESSES

## 7.6. Multihoming Exchanges

ADD and REMOVE are the Multihoming actions, and the Notify Payloads associated to those actions are the ADD\_SA\_ADDRESS and REMOVE\_SA\_ADDRESS Notify Payloads.

The Initiator sends an ADD\_SA\_ADDRESS or REMOVE\_SA\_ADDRESS Notify Payload, when it wants to ADD or REMOVE one IP address to the SA



specified by the SELECTORs. The draft does not consider the case of the IKE\_SA, since we consider that the IKEv2 application does not support Simultaneous use of multiple IP addresses. In the draft, we consider that if an IKE\_SA is specified by the SELECTORs, then the Multihoming actions are simply skipped.

The Notify Payloads only need to specify one single IP address, i.e. the IP address to be ADDED or REMOVED to or from the SA. This IP address can be explicitly specified using the IP parameter, or implicitly specified. When the IP address is implicitly specified, no IP parameters are specified in the data field. The Responder assume the IP address to be considered is the one in the IP header.

When the IP is specified, then the ADD or REMOVE action can be CHECKED, which means that the Initiator does not expect the change to be performed. Of course, when the IP address is implicitly specified, the Responder considers that the ADD or REMOVE operation MUST occur.

When the IP address is specified, the I and H bits indicates, where the IP address MUST be ADDED or UPDATED -- in IP Header in transport or tunnel mode, or inside the tunnel when the tunnel mode is used. Of course, such bits cannot be specified when the IP address is implicitly specified. In that case, the Responder ADDs or REMOVE any IP address in the IP Header of tunnel and transport mode -- that is to say not the inner IP addresses.

As with the UPDATE\_SA\_ADDRESSES Notify Payload, when the Responder receives it, it does not have to check if that is the last received Notify Payload.

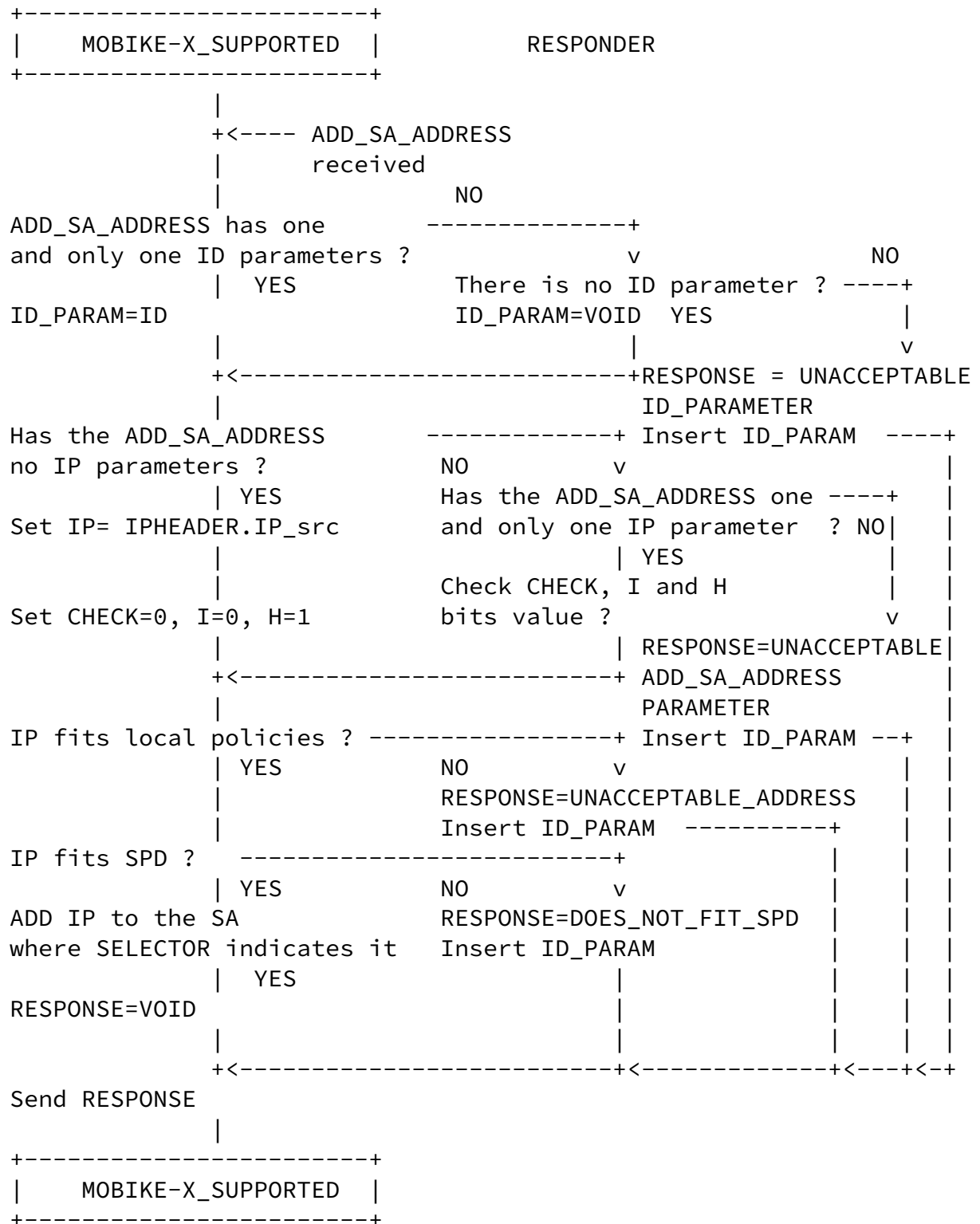
When the Responder receives an ADD\_SA\_ADDRESS or a REMOVE\_SA\_ADDRESS Notify Payload, it checks if the data field is formatted correctly or not. If other parameters then ID or IP parameter is specified or more then one IP parameter is specified, an UNACCEPTABLE\_PARAMETER Notify Payload is sent as a response. If the data field does not carry one IP parameter, the Responder considers the IP address in the IP header of the received message, the operation has to be performed, and I bit is set to zero, H bit is set to 1. If the IP address is specified in the data field, the Responder reads the CHECK bit, the I

operation.

When an ADD action is requested, then the Responder MUST check if the new IP address matches local and Security Policies. If the IP address does not match local policies, then an UNACCEPTABLE\_IP\_ADDRESS Notify Payload is sent as a response. If the IP address does not match the Security Policy, then a DOES\_NOT\_FIT\_SPD Notify Payload is sent as a response. When a REMOVE action is requested, the Responder checks at least one IP address is still associated to the SA, if there is only one IP address associated to the SA, then REMOVing it will close the SA.

#### [7.7.](#) Multihoming Exchanges State Diagrams

Multihoming Actions are very similar as UPDATE actions.



Responder : Receiving ADD\_SA\_ADDRESS

```

+-----+
|      MOBIKE-X_SUPPORTED      |
+-----+

+-----+
|      REMOVE_SA_ADDRESS      |
|      received                |
|      NO                      |
+-----+
REMOVE_SA_ADDRESS has one
and only one ID parameters ?
ID_PARAM=ID
+-----+
There is no ID parameter ?
ID_PARAM=VOID
+-----+
YES
NO
+-----+
RESPONSE = UNACCEPTABLE
ID_PARAMETER
+-----+
Insert ID_PARAM
+-----+
NO
v
Has REMOVE_SA_ADDRESS one
and only one IP parameter ?
YES
NO
+-----+
Check CHECK, I and H
bits value ?
+-----+
v
+-----+
RESPONSE=UNACCEPTABLE
ADD_SA_ADDRESS
PARAMETER
Insert ID_PARAM
+-----+
YES
RESPONSE=VOID
+-----+
+-----+
Send RESPONSE
+-----+
|      MOBIKE-X_SUPPORTED      |
+-----+

```

Responder : Receiving REMOVE\_SA\_ADDRESS

## 8. SELECTOR Notify Payload

The SELECTOR payloads are used to select one, a set of SA or IKE\_SA where an ACTION MUST occur. ACTIONS are represented by the UPDATE\_SA\_ADDRESSES, ADD\_SA\_ADDRESS, REMOVE\_SA\_ADDRESS or ADDITIONAL\_IP\_ADDRESS Notify Payloads. The Initiator might use more than one SELECTOR that constitutes a SELECTOR\_SET. The Initiator has to build the SELECTOR\_SET carefully according to the considered Notify Payload that follows SELECTOR\_SET.

Migault

Expires March 5, 2010

[Page 39]

---

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009

A SELECTOR\_SET is a collection of SELECTOR Notify Payloads. An SA or an IKE\_SA matches the SELECTOR\_SET if a match occurs with at least one SELECTOR of the SELECTOR\_SET. A SELECTOR Notify Payload carries different parameters. An SA or an IKE\_SA matches a SELECTOR if all parameters of the SELECTOR Notify Payload match the SA or the IKE\_SA. By default the SELECTOR value is IKE\_SA\_AND\_CHILD\_SA. In this document, this means that UPDATE\_SA\_ADDRESSES, ADD/REMOVE\_SA\_ADDRESS Notify Payload apply to the IKE\_SA and the CHILD SAs, and the ADDITIONAL\_IP\_ADDRESS applies to the IKE\_SA.

The SELECTOR\_SET precedes the action -- that is to say the UPDATE\_SA\_ADDRESSES, the ADD/REMOVE\_SA\_ADDRESS or the ADDITIONAL\_IP\_ADDRESS Notify Payloads. There can be multiple action Notify Payloads, and all of them apply to the SAs or IKE\_SA that matches the SELECTOR\_SET.

To specify that the SELECTOR\_SET MUST NOT be considered anymore, the Initiator MUST add an END\_OF\_SELECTOR Notify Payload. If an action Notify Payload is placed right after the END\_OF\_SELECTOR Notify Payload, then there is no SELECTOR\_SET specified for this action and the default value is consider -- IKE\_SA\_AND\_CHILD\_SA or IKE\_SA. On the other hand, another SELECTOR\_SET can also be defined after the END\_OF\_SELECTOR Notify Payload.

When a SELECTOR\_SET is defined, any other Notify Payload different from UPDATE\_SA\_ADDRESSES or ADDITIONAL\_IP\_ADDRESS Notify Payload remove the SELECTOR\_SET and set it to its default value.

The SELECTOR\_SET is a regular expression like, and can be modeled as SELECTOR\_SET = SELECTOR\_1 OR ... OR SELECTOR\_n. . A match with the SELECTOR\_SET occurs with an IKE\_SA or an SA, if there is a match with SELECTOR\_1 OR ... OR a match occurs with SELECTOR\_n. If

SELECTOR\_i has k PARAMETERS, then a match occurs with SELECTOR\_i and IKE\_SA or a SA if a match occurs with PARAMETER\_1 AND ... AND a match occurs with PARAMETERS\_k.

The initiator sending a SELECTOR Notify Payload it is recommended the Initiator proceeds as mentioned below :

- 1 : When the Initiator wants to send actions, it SHOULD check to which SA or IKE\_SA the action has to be proceeded. The different SAs or IKE\_SA can be described with different SELECTOR. If no selectors applied, the default value is IKE\_SA\_AND\_CHILD\_SA. The Initiator either has set its SELECTOR\_SET or consider the default values.

- 2 : The Initiator might group the different actions that matches the same SELECTOR\_SET. There are many way the different actions can be bound to SELECTOR\_SET and SELECTOR\_SET adapted to the actions so that to reduce the number of payload or the size of the message sent. This is not specified in this paper.
- 3 : The Initiator appends all the action to the SELECTOR\_SET.
- 4 : If a SELECTOR\_SET is explicitly built, append an END\_OF\_SELECTOR Notify Payload.

The procedure described here is what we thought was the simplest way to implement it and is not normative. There are possible optimizations since without SELECTOR specification the assumed default value is IKE\_SA\_AND\_CHILD\_SA.

When receiving a SELECTOR Notify Payload, the Responder MUST proceed as described below. We define states for the SELECTOR\_SET variable. The SELECTOR\_SET state is NULL when it has not been initiated. In that case the Responder will have to consider the default values of the SELECTOR\_SET. When the SELECTOR\_SET is being built, it is in a START state. This means the SELECTOR\_SET value is not yet fixed, but is on its way to be set. When the SELECTOR\_SET value is set then its state is ESTABLISHED. When an error is encounter, like an unexpected Notify Payload found, an unknown parameter, or badly formed parameter, then the SELECTOR\_SET sets a variable ERROR to 1. When the Responder receives a message, the SELECTOR\_SET value is set to

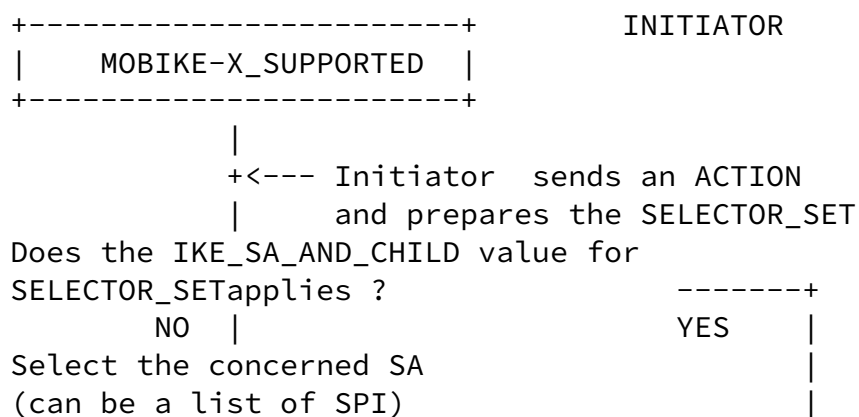
NULL and the ERROR variable is set to 0.

- 1 : If the Notify Payload type are UPDATE\_SA\_ADDRESSES, ADD\_SA\_ADDRESS, REMOVE\_SA\_ADDRESS or ADDITIONAL\_IP\_ADDRESSES, then check the SELECTOR\_SET state value. If the SELECTOR\_SET state value is NULL then perform the action with the default value of the SELECTOR\_SET -- IKE\_SA or IKE\_SA\_AND\_CHILD\_SA. If the SELECTOR\_SET state value is START, then the Responder sets the SELECTOR\_SET state to ESTABLISHED, and applies the Notify Payload to the SA or IKE\_SA specified by the SELECTOR\_SET value. If the SELECTOR\_SET state value is ESTABLISHED, apply the Notify Payload to the IKE\_SA and SA specified by the SELECTOR\_SET value.
- 2 : If the Notify Payload is of type SELECTOR, then check the SELECTOR\_SET state value. If the SELECTOR\_SET state value is NULL, then set the SELECTOR\_SET state value to START and consider the SELECTOR parameters, and add them to the SELECTOR\_SET value. If the SELECTOR\_SET state value is START then add the SELECTOR parameters to the SELECTOR\_SET value. If the SELECTOR\_SET state value is ESTABLISHED then clear the SELECTOR\_SET value, create a new SELECTOR\_SET value, set the SELECTOR\_SET state value to START and add the parameters of the SELECTOR to the SELECTOR\_SET value.

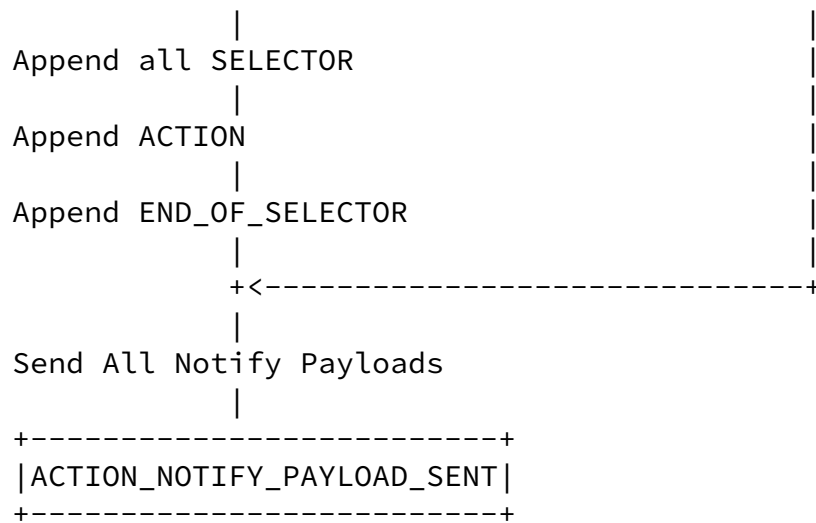
- 3 : If the Notify Payload is of type END\_OF\_SELECTOR clear the SELECTOR\_SET value and set the SELECTOR\_SET state value to NULL.
- 4 : If the Notify Payload is of any other type, or if the Payload is different from the Notify type, then clear the SELECTOR\_SET value and set the SELECTOR\_SET value to NULL. The Responder MUST send a UNEXPECTED\_PAYLOAD\_AFTER\_SELECTOR Notify Payload, set the variable ERROR to 1 and skip the Payloads
- 5 : When the Responder performs the action Notify Payloads and do not find any SA or IKE\_SA it SHOULD send an UNMATCHED\_SELECTOR\_SET.
- 6 : When the Responder finds an Notify Payload, it does not understand, it sends an error message as specified in [[RFC4306](#)] if the critical flag is set to one. No error message otherwise. The Responder MUST set the variable ERROR to 1 and go on analyzing Notify Payload.
- 7 : When the Responder do not understand the parameters of a SELECTOR Notify Payload, it MUST send an error

- UNACCEPTABLE\_SELECTOR\_PARAMETER Notify Payload. It MUST set the ERROR variable to 1.
- 8 : When the Responder performs an action and the ERROR variable is set to 0, it performs it in a regular way. When the ERROR variable is set to 1.
  - 9 : When the Responder performs an action and the ERROR variable is set to 0, it performs it in a regular way. When the ERROR variable is set to 1.

## 9. SELECTOR State diagrams







Initiator : Handling with SELECTORs

```

+-----+
| MOBIKE-X_SUPPORTED |
+-----+
      |

```

NO

RESPONDER

```

SELECTOR_SET_VALUE=
IKE_SA_AND_CHILD_SA
SELECTOR_STATE=NULL
ERROR=0

```



Apply ACTION to SELECTOR\_SET consists of checking the ERROR value of SELECTOR\_SET. If ERROR is set to 1, then the ACTION is not performed and a UNEXPECTED\_PARAMETER Notify Payload is returned. When the ACTION cannot be performed because no SA matches the SELECTORs, then an UNMATCHED\_SELECTOR\_SET Notify Payload is sent. This latest Notify Payload is more a warning message than an error message.

## 10. ID Notify Payload Parameter

The ID parameter purpose is to bind a query Notify Payload to the response Notify Payload. An Initiator can send multiple Notify Payload, some of them might require a response for example if the Notify Payload generating an error. When the Responder receives a Notify Payload with an ID parameter in its data field, which requires a response, it MUST insert the ID parameter in its response Payload.

It is the responsibility of the Initiator to make sure the ID parameters are different, so that it can easily bind the response with the query Notify Payload.

## 11. Packet Format

### 11.1. Notify Payload

The Notify Payload is define in[RFC4306] [section 3.10](#). This Notify Payload is represented as below :

```

 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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
! Next Payload  !C!  RESERVED   !           Payload Length       !
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
! Protocol ID   !   SPI Size    !           Notify Message Type   !
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
!
~                               Security Parameter Index (SPI)      ~
!
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
!
~                               Notification Data                     ~
!
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Notify Payload

In our case, we would fill the different fields as defined below :

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009

- Protocol ID (1 octet) : As mentioned in [RFC4306] "If this notification concerns an existing SA, this field indicates the type of that SA. For IKE\_SA notifications, this field MUST be one (1). For notifications concerning IPsec SAs this field MUST contain either (2) to indicate AH or (3) to indicate ESP. For notifications that do not relate to an existing SA, this field MUST be sent as zero and MUST be ignored on receipt. All other values for this field are reserved to IANA for future assignment." In our case the Protocol is 1, since it is related to the IKE\_SA
- SPI Size (1 octet) : [RFC4306] mentions "h in octets of the SPI as defined by the IPsec protocol ID or zero if no SPI is applicable. For a notification concerning the IKE\_SA, the SPI Size MUST be zero.". In our case the SPI is set to zero.
- Notify Message Type (2 octets) : [RFC4306] mentions "Specifies the type of notification message."
- SPI (variable length) [RFC4306] mentions "Security Parameter Index." In our case this field should not appear.
- Notification Data (variable length) [RFC4306] mentions "Informational or error data transmitted in addition to the Notify Message Type. Values for this field are type specific (see below)."

## 11.2. Notify Message -- status type

In this section we provide assignment numbers for the different Type of Notify Payloads. Such numbers are added to the list provided by the IANA at <http://www.iana.org/assignments/ikev2-parameters>.

### 11.2.1. MOBIKE\_SUPPORTED

The MOBIKE\_SUPPORTED Notify Payload MUST carry the VERSION parameter. They can be one or more VERSION parameter payload. The VERSION value for MOBIKE [RFC4555] is 1. The VERSION value for MOBIKE-X as defined in this draft is 2. The Protocol ID and SPI Size fields are set to zero.

The Notify Message Type for MOBIKE\_SUPPORTED is defined in [RFC4555] and the Type code is 16396.

### [11.2.2.](#) ADDITIONAL\_IP\_ADDRESS

The ADDITIONAL\_IP4\_ADDRESS and ADDITIONAL\_IP6\_ADDRESS are defined in MOBIKE [[RFC4555](#)], and type codes are 16397 and 16398. In this document, we consider only ADDITIONAL\_IP6\_ADDRESS as ADDITIONAL\_IP\_ADDRESS, so its type is 16398.

### [11.2.3.](#) NO\_ADDITIONAL\_ADDRESS

The NO\_ADDITIONAL\_ADDRESS is defined in [[RFC4555](#)] and the type code is 16399.

### [11.2.4.](#) SELECTOR

The SELECTOR Notify Payload is used to define who the ADDITIONAL\_IP\_ADDRESS, the UPDATE\_SA\_ADDRESSES. The Notify Message Type for SELECTOR is 40961 -- We define it as private use number but it should be assigned by the IANA.

The SELECTOR Payload MUST carry selectors parameters in its data payload. The currently supported parameters are IP, SPI, IKE\_AND\_CHILD\_SA and ALL\_IKE\_SA.

### [11.2.5.](#) END\_OF\_SELECTOR

The END\_OF\_SELECTOR defines where the SELECTOR\_SET stop being used. The Notify Message Type for SELECTOR is 40962.

### [11.2.6.](#) UPDATE\_SA\_ADDRESSES

The UPDATE\_SA\_ADDRESSES is described in [[RFC4555](#)] and in this paper. The type code is 16400. The main difference between description in [[RFC4555](#)] and this paper is the UPDATE\_SA\_ADDRESSES Notify Payload can carry the NEW\_IP and the OLD\_IP address as parameters in its data field.

### [11.2.7.](#) ADD\_SA\_ADDRESS Notify Payload

The ADD\_SA\_ADDRESS requests the Responder to ADD an IP address to an associated SA. The type code is 40962. This Notify Payload can

specify the IP address to ADD.

#### [11.2.8.](#) REMOVE\_SA\_ADDRESS Notify Payload

The REMOVE\_SA\_ADDRESS requests the Responder to REMOVE an IP address to an associated SA. The type code is 40963. This Notify Payload can specify the IP address to ADD.

#### [11.2.9.](#) Notify Message -- status type table

Name	Value	Reference
----	-----	-----
MOBIKE_SUPPORTED	16396	<a href="#">[RFC4555]</a>
SELECTOR	40961	
END_OF_SELECTOR	40962	
ADDITIONAL_IP_ADDRESS	16398	<a href="#">[RFC4555]</a>
NO_ADDITIONAL_ADDRESS	16399	<a href="#">[RFC4555]</a>
UPDATE_SA_ADDRESSES	16400	<a href="#">[RFC4555]</a>
ADD_SA_ADDRESS	40963	
REMOVE_SA_ADDRESS	40964	

Notify Message -- status type -- Private values

#### [11.3.](#) Notify Message -- error type

##### [11.3.1.](#) MOBIKE\_UNSUPPORTED\_VERSION

This Notify Payload is used by the Responder to indicate, it does not understand the MOBIKE version proposed by the Initiator. When sending this Notify Payload, the Responder MUST add the supported version of MOBIKE it supports. This Notify Payload can also be used to cancel the use of MOBIKE-X. In that specific case, the version number is NONE.

The Type value associated to this message is the first value of Notify Message error type value assigned for private use, that is to

say : 8192.

#### [11.3.2.](#) UNACCEPTABLE\_ADDRESS

This Notify Payload is defined in [[RFC4555](#)]. This Notify Payload is sent by the Responder when it cannot accept the IP address specified in an UPDATE\_SA\_ADDRESSES. This Notify Payload MUST be sent when the IP address does not match the local policies. More specifically it cannot be used instead of a DOES\_NOT\_FIT\_SPD Notify Payload. The UNACCEPTABLE\_ADDRESS type value is 40 as specified in [[RFC4555](#)].

#### [11.3.3.](#) DOES\_NOT\_FIT\_SPD

This message MUST be sent by the Responder when the proposed IP address is rejected by the SPD. The type value associated to the message is 8193.

#### [11.3.4.](#) UNACCEPTABLE\_PARAMETERS

This message MUST be sent as response when parameters are not well understood by the Responder.

Migault

Expires March 5, 2010

[Page 48]

---

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009

#### [11.3.5.](#) UNEXPECTED\_PAYLOAD\_AFTER\_SELECTOR

The message is sent when the responder does not find an expected payload after the SELECTOR payload. It is more a warning message than an error message. The type value associated to it is 8195.

#### [11.3.6.](#) UNMATCHED\_SELECTOR\_SET

This message MUST be sent when no SA matches the mentioned SELECTOR\_SET. The type value associated to the message is 8196.

#### [11.3.7.](#) Notify Message -- error type table

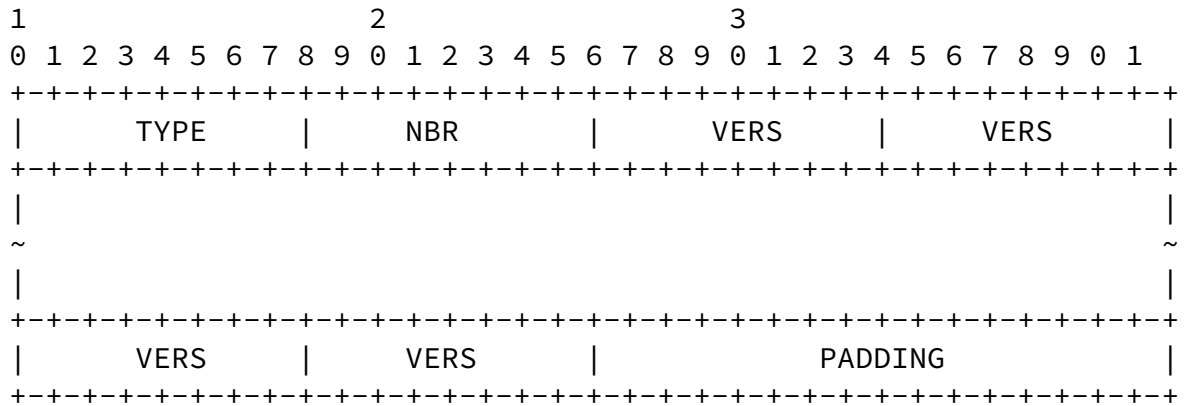
Name	Value	Reference
----	-----	-----
MOBIKE_UNSUPPORTED_VERSION	8192	
UNACCEPTABLE_ADDRESS	40	[ <a href="#">RFC4555</a> ]
DOES_NOT_FIT_SPD	8193	

UNACCEPTABLE_PARAMETER	8194
UNEXPECTED_PAYLOAD_AFTER_SELECTOR	8195
UNMATCHED_SELECTOR_SET	8196

## Notify Message -- error type -- Private values

### 11.4. Notify Parameters

### 11.4.1. VERSION



## VERSION parameter

Where :

TYPE : 16 bits to define the VERSION parameter (1)

NBR : 8 bits to define the number of proposed version. This field defines the where the PADDING bits starts as well as the length of the PADDING field.

VERS : 8 bits to define the version number. MOBIKE is being assigned the version number 1, the current description in this paper is being assigned the version number 2. The NONE value MUST be only carried by the MOBIKE-X\_UNSUPPORTED Notify Payload, and means that MOBIKE-X messages MUST NOT be anymore considered.

PADDING : 8, 16 or 24 bits set to zero. The PADDING length is such that the VERSION parameter length is a multiple of 32 bits.

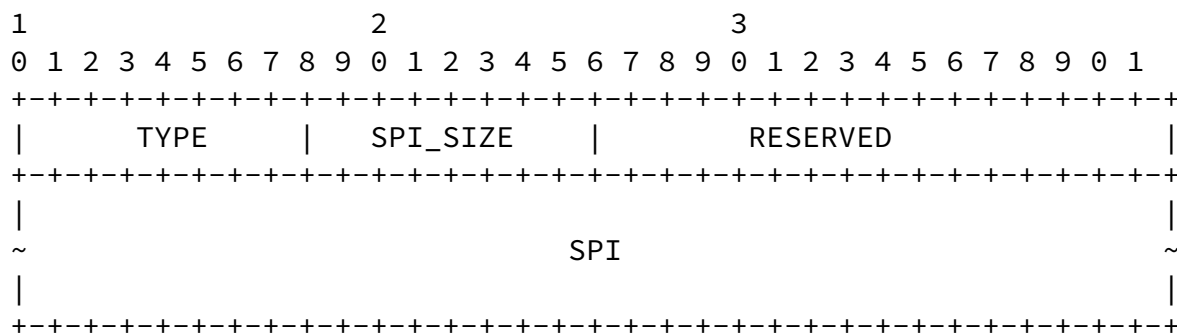


Its length is derived from NBR. Consider  $L = (NBR+1) \% 4$ . This value represents the PADDING number of bytes.

Name ----	Value -----	Reference -----
Reserved	0	
MOBIKE	1	
MOBIKE-X	2	
Reserved to IANA	3-254	
NONE	255	

#### VERSION

#### [11.4.2.](#) SPI



#### SPI size

Where :

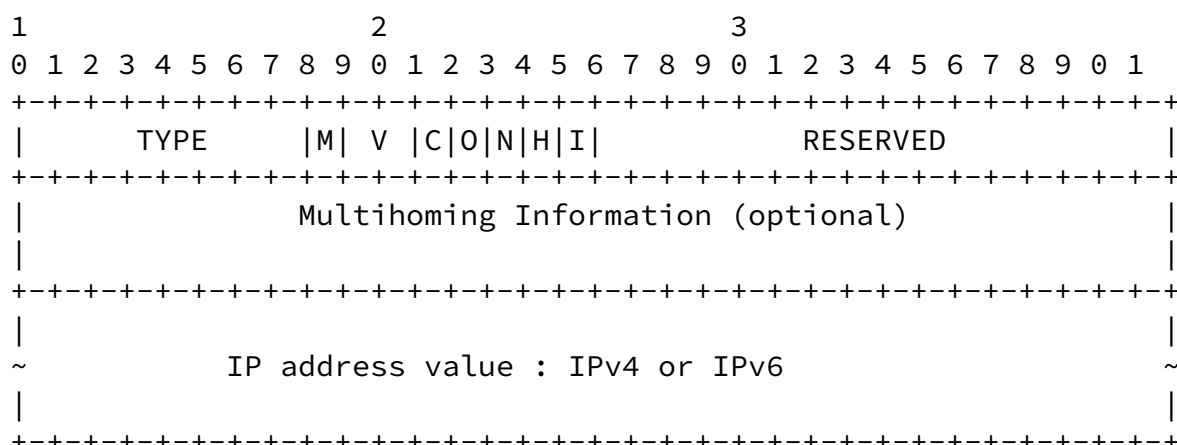
TYPE : 8 bits to define the SPI parameter.

SPI\_SIZE : 8 bits to define the size of the SPI. The size is expressed in number of bytes.

RESERVED : 16 bits set to zero and reserved for future use.

SPI : the SPI value.

### 11.4.3. IP



IP

Where :

TYPE : 8 bits to define the IP parameter.

M (Multihoming) bit : 1 bit that specifies whether Multihoming Information Payload (MIP) is present in the IP parameter. M set to 1 means that the MIP is present; otherwise M is set to 0. This bit is not optional and its value MUST be set properly.

V (Version) bits : 2 bits that specifies the version of the IP address. . V set to 0 means the IP is an IPv4, V set to 1 means the IP is an IPv6. Those bits MUST be set properly and their value is not optional.

C (CHECK) bit : 1 bit that specifies the Initiator does not want the action to be performed but does only want to check the IP address matches local and security policies. When this bit is set to 1, then the ACTION is only CHECKed, otherwise, the ACTION is performed.

O (Old) bit : 1 bit that specifies that the IP address is considered as the OLD\_IP, that is to say the address to replace. To specify the IP parameter is an OLD\_IP, the bit MUST be set to 1. O bit or N bit MUST be used when the IP address is a parameter of the UPDATE\_SA\_ADDRESSES Notify Payload. Other ACTION ignores those bits. If both OLD\_IP and NEW\_IP are explicitly specified and the CHECK bit has not the same value in both IP parameter, then a UNACCEPTABLE\_PARAMETER

will be sent by the Responder.

N (New) bit : 1 bit that specifies that the IP address is considered as the NEW\_IP, that is to say the value replacing the OLD\_IP. To specify the IP parameter is a NEW\_IP, the N bit MUST be set to 1. (see O bit for more information).

H (IP Header) bit : 1 bit that specifies the IP address is use for routing purpose. If the SA is using the transport mode, then setting H to 1 means the IP address has to be considered. If the SA is using the tunnel mode, the outer IP address is considered. The H and I bits are not optional and MUST be set for ADD, REMOVE and UPDATE actions. ADDITIONAL\_IP\_ADDRESS assumes H=1, I=0.

I (Inner IP) bit : 1 bit that specifies the IP address is a inner IP address. When set to 1, it means the IP address is the inner IP address of a tunnel mode. (see the H bit for more explanation).

RESERVED : 16 bits set to zero and reserved for future.

IP4 : 32 bits to define the IPv4 address.

IP6 : 128 bits to define the IPv6 address.

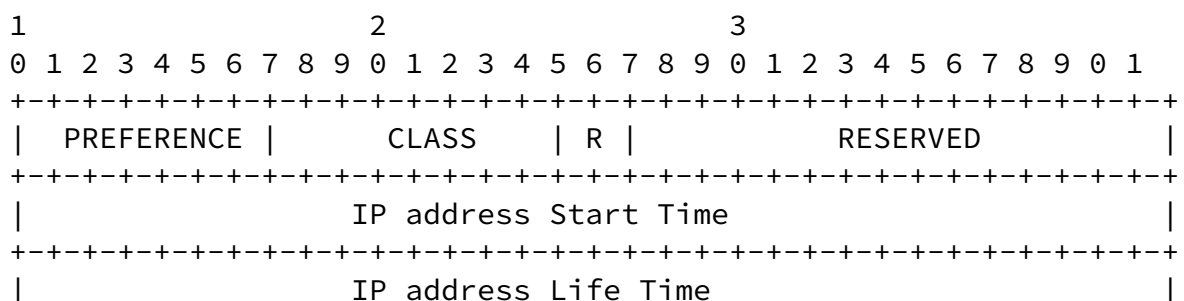
Name	Value	Reference
----	-----	-----
IPv6	0	
IPv4	1	
Reserved to IANA	2-3	

## Version

### [11.4.4.](#) Multihoming Information Payload (MIP)

This section defines the Multihoming Information Payload.

Multihoming Information can be useful for Multihoming purposes. That is to say such information is only useful for multihoming management entity. In this document the only Multihoming Management Entity is the IKEv2 application. In fact we do not consider here communication between IKEv2 and ULP. As a result, the MIP SHOULD be only used associated with ADDITIONAL\_IP\_ADDRESS Notify Payload.



[illegible]

Name	Value	Reference
NO_INFO	0	
REACHABLE	1	

R

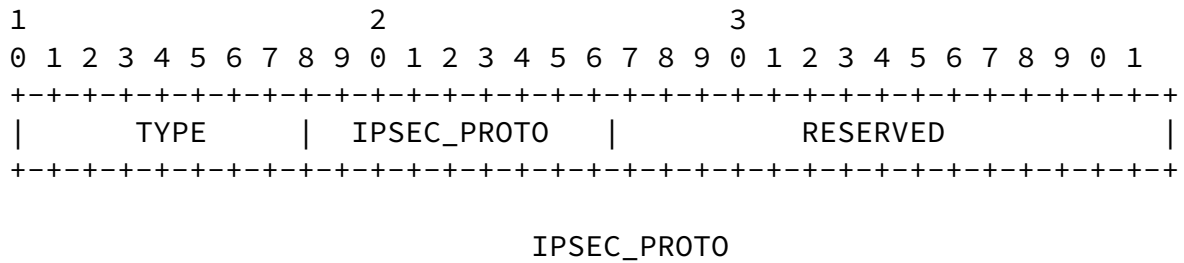


From <http://www.iana.org/assignments/isakmp-registry>, we get the following values :

Values 3-61439 are reserved to IANA. Values 61440-65535 are for private use.

IPSEC MODE

#### 11.4.6. IPSEC\_PROTO



Where :

Migault

Expires March 5, 2010

[Page 54]

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009

TYPE : 8 bits to define the IPSEC\_PROTO parameter.  
IPSEC\_MODE : 8 bits the IPsec protocol to consider. Protocols are defined [RFC4306] and in <http://www.iana.org/assignments/ikev2-parameters>  
RESERVED : 16 bits set to zero and reserved for future.

From <http://www.iana.org/assignments/ikev2-parameters>, we get the following values :

Protocol ID	Protocol	Reference
-----	-----	-----
0	Reserved	[RFC4306]
1	IKE	[RFC4306]
2	AH	[RFC4306]
3	ESP	[RFC4306]
4	FC_ESP_HEADER	[RFC4595]
5	FC_CT_AUTHENTICATION	[RFC4595]
6-200	Unassigned	[RFC4306]
201-255	Private use	[RFC4306]

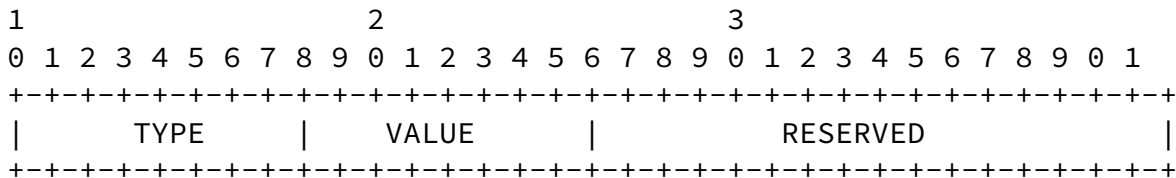
Registry Name: IKEv2 Traffic Selector Types

Reference: [RFC4306]

Registration Procedures: Expert Review

IPSEC\_PROTO

#### 11.4.7. SELECTOR\_SPECIFIC\_VALUE\_PARAMETER



## SELECTOR SPECIFIC VALUE PARAMETER

Where :

TYPE : 8 bits to define the SELECTOR\_PARAMETER parameter.

VALUE : 8 bits that specifies special SELECTORS values.

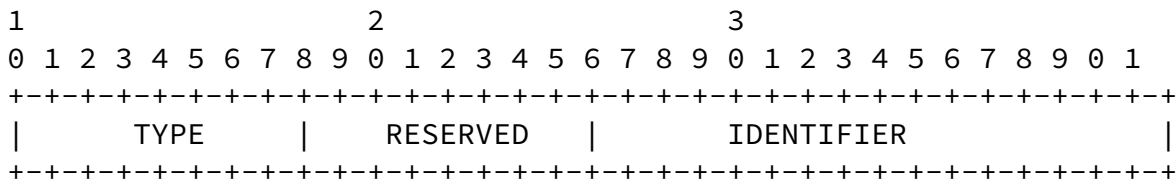
RESERVED : 16 bits set to zero and reserved for future.

The different values for VALUES are the following :

SELECTOR VALUES	Protocol	Reference
0	Reserved	
2	IKESA_AND_CHILD_SA	
3	ALL_IKE_SA	
4-255	Reserved to IANA	

## SELECTOR\_SPECIFIC\_VALUE\_PARAMETER

### 11.4.8. ID



ID parameter

Where :

TYPE : 16 bits to define the VERSION parameter.

RESERVED : 8 bits set to zero and reserved for future.

IDENTIFIER : 16 bits The identifier for the Notify Payload. The identifier does not need to be chosen randomly, and do not intend to provide any protection, it only reason is to distinguish different Notify Payloads.

#### 11.4.9. Parameter Code Type

Registry:

Value	NOTIFY PARAMETER - MOBIKE-X	Reference
-----	-----	-----
0	Reserved	
1	VERSION	
2	SPI	
3	IP	
4	IPSEC_MODE	
5	IPSEC_PROTO	
6	SELECTOR_SPECIFIC_VALUE_PARAMETER	
7	ID	
8-255	Reserved to IANA	

Parameter code types

## 12. Security Considerations

Security considerations are the same as those expressed in MOBIKE [[RFC4555](#)]. This document extends the Security Associations manipulation. More specifically, with the use of SELECTORs payloads, one user can select a random SPI, or other IKE\_SAs. For this reason, we recommend that unless specified otherwise for special uses, the SELECTORs SHOULD apply to only a subset of Security Associations. The subset of Security Association is Security Association that has been negotiated between the same peer's ID and with the same authentication method. By doing so, we protect peer from weak authentication method. An attacker will not be able to hijack a peer's identity using a weak authentication method. We also prevent, a peer to interfere with others peer Security Associations.



### [13.](#) IANA Considerations

The new Notify Message status Type to be added are :

Name	Value	Reference
----	-----	-----
SELECTOR	40961	
END_OF_SELECTOR	40962	

Notify Message -- status type -- Private values

The new Notify Message error Type to be added are :

Name	Value	Reference
----	-----	-----
MOBIKE_UNSUPPORTED_VERSION	8192	
DOES_NOT_FIT_SPD	8193	
UNACCEPTABLE_PARAMETER	8194	
UNEXPECTED_PAYLOAD_AFTER_SELECTOR	8195	
UNMATCHED_SELECTOR_SET	8196	

Notify Message -- error type -- Private values

A New field is created : MOBIKE-X parameters. The following values are :

#### Registry:

Value	NOTIFY PARAMETER - MOBIKE-X	Reference
-----	-----	-----
0	Reserved	
1	VERSION	
2	SPI	
3	IP	

4	IPSEC_MODE
5	IPSEC_PROTO
6	SELECTOR_SPECIFIC_VALUE_PARAMETER
7	ID

#### MOBIKE-X PARAMETERS

A New field is created : SELECTOR\_VALUE parameters. The following values are :

SELECTOR VALUES	Protocol	Reference
-----	-----	-----
0	Reserved	
2	IKESA_AND_CHILD_SA	
3	ALL_IKE_SA	
4-255	Reserved to IANA	

#### SELECTOR\_PARAMETER

Name	Value	Reference
----	-----	-----
NONE	0	
HoA	1	
CoA	2	
Reserved to IANA	3-255	

#### CLASS

Name	Value	Reference
----	-----	-----
NO_INFO	0	
REACHABLE	1	
Unreachable	2	
Reserved to IANA	4	

R

## 14. Acknowledgments

Daniel Migault is partly funded by 3MING, a research project supported by the French 'National Research Agency' (ANR).

## 15. References

### 15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", [RFC 4301](#), December 2005.
- [RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", [RFC 4306](#), December 2005.
- [RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", [RFC 4555](#), June 2006.

### 15.2. Informative References

- [I-D.mglt-ipsec-mm-requirements]  
Migault, D., "IPsec mobility and multihoming requirements : Problem statement", [draft-mglt-ipsec-mm-requirements-00](#) (work in progress), September 2009.
- [RFC2407] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", [RFC 2407](#), November 1998.
- [RFC3947] Kivinen, T., Swander, B., Huttunen, A., and V. Volpe, "Negotiation of NAT-Traversal in the IKE", [RFC 3947](#), January 2005.
- [RFC4595] Maino, F. and D. Black, "Use of IKEv2 in the Fibre Channel Security Association Management Protocol", [RFC 4595](#), July 2006.

Internet-Draft      MOBIKE-X for mobility and multihoming      September 2009

Author's Address

Daniel Migault  
Orange Labs R&D  
38 rue du General Leclerc  
92794 Issy-les-Moulineaux Cedex 9  
France

Phone: +33 1 45 29 60 52  
Email: [mglt.ietf@gmail.com](mailto:mglt.ietf@gmail.com)

Migault

Expires March 5, 2010

[Page 60]