**IEEE P802.21
Media Independent Handover Services**

|  |
| --- |
| **Draft D02 modified on 09/17/13** |
| **Date:** 2013-09-17 |
| **Author(s):** |
| **Name** | **Affiliation** | **Address** | **Phone** | **Email** |
| Toru Kambayashi | Toshiba |  |  | tooru.kamibayashi@ toshiba.co.jp |

Abstract

This document is an update of the draft D02.

P™/D
Draft for

Amendment 4: Multicast Group Management

Sponsor

**LAN/MAN Committee**of the **IEEE Computer Society**

Approved <XX MONTH 20XX>

**IEEE-SA Standards Board**

Copyright © 2013 by The Institute of Electrical and Electronics Engineers, Inc.

Three Park Avenue

New York, New York 10016-5997, USA

All rights reserved.

This document is an unapproved draft of a proposed IEEE Standard. As such, this document is subject to change. USE AT YOUR OWN RISK! IEEE copyright statements SHALL NOT BE REMOVED from draft or approved IEEE standards, or modified in any way. Because this is an unapproved draft, this document must not be utilized for any conformance/compliance purposes. Permission is hereby granted for officers from each IEEE Standards Working Group or Committee to reproduce the draft document developed by that Working Group for purposes of international standardization consideration. IEEE Standards Activities must be informed of the submission for consideration prior to any reproduction for international standardization consideration (stds.ipr@ieee.org). Prior to adoption of this document, in whole or in part, by another standards development organization, permission must first be obtained from the IEEE Standards Activities Department (stds.ipr@ieee.org). When requesting permission, IEEE Standards Activities will require a copy of the standard development organization’s document highlighting the use of IEEE content. Other entities seeking permission to reproduce this document, in whole or in part, must also obtain permission from the IEEE Standards Activities Department.

IEEE Standards Activities Department

445 Hoes Lane

Piscataway, NJ 08854, USA

Abstract: This amendment standard specifies additional mechanisms to enable the use of multicast transport for MIH communication. The specification defines new management primitives that enable a user to join, leave or update group membership and additional security mechanisms to secure multicast communication between MIH entities.

Keywords: group, multicast, group management, group security

[[1]](#footnote-2)•

**Important Notices and Disclaimers Concerning IEEE Standards Documents**

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

**Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents**

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (“IEEE-SA”) Standards Board. IEEE (“the Institute”) develops its standards through a consensus development process, approved by the American National Standards Institute (“ANSI”), which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

**Translations**

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

**Official Statements**

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

**Comments on Standards**

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

 Secretary, IEEE-SA Standards Board

 445 Hoes Lane

 Piscataway, NJ 08854 USA

**Laws & Regulations**

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

**Copyright**

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

**Photocopies**

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only.  To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

**Updating of IEEE Standards Documents**

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every ten years.  When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA website at <http://ieeexplore.ieee.org/xpl/standards.jsp> or contact IEEE at the address listed previously. For more information about the IEEE SA or IEEE’s standards development process, visit the [IEEE-SA website](http://standards.ieee.org/index.html) at <http://standards.ieee.org>.

 **Errata**

[Errata](http://standards.ieee.org/findstds/errata/index.html), if any, for all IEEE standards can be accessed on the IEEE-SA website at the following URL: <http://standards.ieee.org/findstds/errata/index.html>. Users are encouraged to check this URL for errata periodically.

**Patents**

IEEE standards are developed in compliance with the [IEEE Standards patent policy](http://standards.ieee.org/about/sasb/patcom/index.html) which is available at the following URL: <http://standards.ieee.org/about/sasb/patcom/patents.html>.

.

Participants

At the time this draft was submitted to the IEEE-SA Standards Board for approval, the Working Group had the following membership:

**Subir Das**, *Chair*

**Anthony Chan**, *Vice Chair*

Participant1

Participant2

Participant3

Participant4

Participant5

Participant6

Participant7

Participant8

Participant9

The following members of the <individual/entity> balloting committee voted on this. Balloters may have voted for approval, disapproval, or abstention.

***(to be supplied by IEEE)***

Balloter1

Balloter2

Balloter3

Balloter4

Balloter5

Balloter6

Balloter7

Balloter8

Balloter9

When the IEEE-SA Standards Board approved this on <XX MONTH 20XX>, it had the following membership:

***(to be supplied by IEEE)***

**<Name>,** *Chair*

**<Name>,** *Vice Chair*

 **<Name>,** *Past Chair*

**<Name>,** *Secretary*

SBMember1

SBMember2

SBMember3

SBMember4

SBMember5

SBMember6

SBMember7

SBMember8

SBMember9

\*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

<Name>, *DOE Representative*

<Name>, *NRC Representative*

<Name>

*IEEE Standards Program Manager, Document Development*

<Name>

*IEEE Standards Program Manager, Technical Program Development*

Introduction

This introduction is not part of IEEE P802.21d/D2, Draft Standard for Media Independent Handover Services—Amendment 4: Multicast Group Management.

This standard extends the communication mechanisms, defined in IEEE Std 802.21-2008, to support addressing group of nodes through multicast transport mechanisms. The need for this specification appears in scenarios where groups of nodes need to move simultaneously, such as sensor or actuator networks. This specification, hence, extends the communication mechanisms provided in IEEE Std 802.21-2008, by defining MIHF group identifiers, new primitives to manage group membership, support of multicast transport mechanisms and security extensions for group communication.

Contents

Draft for

Amendment 5: Multicast Group Management

***IMPORTANT NOTICE: IEEE*** ***Standards documents are not intended to ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks.*** ***Implementers of IEEE*** ***Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.***

NOTE—The editing instructions contained in this **<amendment/corrigendum>** define how to merge the material contained therein into the existing base standard and its amendments to form the comprehensive standard.

The editing instructions are shown in ***bold italic***. Four editing instructions are used: change, delete, insert, and replace. ***Change*** is used to make corrections in existing text or tables. The editing instruction specifies the location of the change and describes what is being changed by using ~~strikethrough~~ (to remove old material) and underscore (to add new material). ***Delete*** removes existing material. ***Insert*** adds new material without disturbing the existing material. Insertions may require renumbering. If so, renumbering instructions are given in the editing instruction. ***Replace*** is used to make changes in figures or equations by removing the existing figure or equation and replacing it with a new one. Editing instructions, change markings, and this NOTE will not be carried over into future editions because the changes will be incorporated into the base standard.

1. Definitions

For the purposes of this document, the following terms and definitions apply. The IEEE Standards Dictionary Online should be consulted for terms not defined in this clause. [[2]](#footnote-3)

***Insert the following definitions in alphabetically order:***

**Group command:** A command issued to members which belong to a group via a multicast channel. Group manipulation commands are explicitly excluded from this definition.

**Group manipulation command:** A command, sent to a group of nodes or to an individual node, that instructs the receivers to perform certain operations such as, joining and leaving a group, updating group membership and so on. By group manipulation command, we refer to the following commands:

* MIH\_MN\_Group\_Manipulate
* MIH\_Net\_Group\_Manipulate

**Command center (CC):** An MIH User which issues a group manipulation command and a group command.

**Device key**: A data element representing a key, assigned to an entity in order to de-capsulate a GKB.

**Group key block (GKB):**  A data entity which enables only those who have the corresponding device keys to decapsulate it and obtain a group key. It stores a Group key. See also: Annex P.

**Group manager (GM):** An entity that generates GKB.

**Media independent handover function Group identifier (MIHF Group ID):** An identifier for identifying a group of MIHF peers.

**Media Independent handover function Broadcast Identifier (MIHF Broadcast ID):** An MIHF Group ID of zero length.

1. Abbreviations and acronyms

***Insert the following abbreviations and acronyms in alphabetically order:***

CC Command center

GM Group manager

GKB Group Key Block

MGK Master Group Key

MIGEK Media Independent Group Encryption Key

MIGIK Media Independent Group Integrity Key

MIGMEK Media Independent Group Manipulation Encryption Key

MIGSK Media Independent Group Session Key

PRF Pseudorandom Function

1. General Architecture
	1. Introduction
		1. Multicast group communication

There are handover scenarios where a group of nodes are meant to move like a group. Examples of these scenarios are networks of sensors/actuators that move between production and management networks or nodes that move together due to some physical reason, such as all nodes traveling together in a transportation medium. MIHF supports the use of multicast means to convey a subset of all possible MIHF commands to group of users in a secure way. Hence, this specification provides primitives for managing the membership of nodes to multicast groups (join, leave and update group membership) and multicast group key mechanisms.

1. MIHF services
	1. Service management
		1. General

***Change list after first paragraph as follows:***

Prior to providing the MIH services from one MIHF to another, the MIH entities need to be configured properly. This is done through the following service management functions:

* MIH capability discovery
* MIH registration
* MIH service access authentication
* MIH event subscription
* MIH group configuration, manipulation and security
	+ 1. Service management primitives

***Insert new rows after last row in Table 3 as follows:***

1. —Services management primitives

|  |  |  |  |
| --- | --- | --- | --- |
| MIH command | (L) ocal / (R) emote | Defined in | Comments |
| MIH\_Configuration\_Update | R | 7.4.30 | This command is sent by a PoS to a group of MNs or other PoSes to update their configuration. |
| MIH\_MN\_Group\_Manipulate | R | 7.4.31 | This command is sent by an MN to a PoS to create, delete or update a group. |
| MIH\_Net\_Group\_Manipulate | R | 7.4.32 | This command is sent by a PoS to a group of MNs or other PoSes to create, delete or update a group. |
| MIH\_Pull\_Credential | R | 7.4.33 | This primitive is generated by an MN and it is used to request the sending of a certificate from the PoS to a destination PoS or MN. |
| MIH\_Push\_Credential | R | 7.4.34 | This command is sent by a PoS to a destination PoS or PoA |
| MIH\_Revoke\_Credential | R | 7.4.35 | This command is sent by a PoS to a group of PoSes and/or PoA to revoke a certificate previously issued by the PoS. |

* + 1.
		2. MIH group configuration, manipulation and security

The MIH group configuration, manipulation and security provide mechanisms for a PoS to manage groups of MNs, which are accessible through a multicast address in a secure way. The primitives used to manage the membership to the groups and their security properties are called group manipulation commands through this specification, and include the required functionality to manage the group membership (join, leave, update operations) and install appropriate credentials on the MNs belonging to the group. Details on which MIHF commands can be used for multicast communication can be found on subclause 8.3.1.

* 1. Media independent command service
		1. Command service flow model

***Insert the following paragraph after the 1st paragraph:***

When a command request frame is sent to a group of MIHF peers, it is transmitted using multicast transport and one or more remote MIHF(s) may receive the request frame. The local MIHF may receive one or more command response frame(s) from the remote MIHF(s). In this case, a CC who is an MIH User on an MIH PoS is the issuer of the group command and the MIH PoS is the sender of the group command request/indication frame, and MN(s) or other MIH PoS(es) are the recipient of the group command request/indication frame. MIH commands addressed to a group of MNs can be exchanged using request or indication messages. In case a request message is used, then each receiver must answer with a response message. In case the indication message is used, then the receivers generate no response message.

1. Service access points (SAPs) and primitives
	1. MIH\_SAP primitives
		1. MIH\_Register
			1. MIH\_Register.request
				1. Semantics of service primitive

***Change text as follows:***

MIH\_Register.request (

DestinationIdentifier,

LinkIdentifierList,

GroupLinkIdentifier,

RequestCode

)

***Insert and modify the following parameters:***

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| LinkIdentifierList | LIST(LINK\_ID) | (Optional) List of local link identifiers of the remote MIHF. This parameter shall be used if and only if DestinationIdentifier is an MIHF ID. |
| GroupLinkIdentifier | NET\_TYPE\_INC | (Optional) Identifier of a group of links. This parameter shall be used if and only if DestinationIdentifier is an MIHF Group ID. |

* + - 1. MIH\_Register.indication
				1. Semantics of service primitive

***Change text as follows:***

MIH\_Register.indication (

SourceIdentifier,

LinkIdentifierList,

GroupLinkIdentifier,

RequestCode

)

***Insert and modify the following parameters:***

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| LinkIdentifierList | LIST(LINK\_ID) | (Optional) List of local link identifiers of the remote MIHF. This parameter shall be used if and only if DestinationIdentifier is an MIHF ID. |
| GroupLinkIdentifier | NET\_TYPE\_INC | (Optional) Identifier of a group of links. This parameter shall be used if and only if DestinationIdentifier is an MIHF Group ID. |

* + - 1. MIH\_Register.response
				1. Semantics of service primitives

***Change text as follows:***

MIH\_Register.response (

DestinationIdentifier,

Status,

ValidTimeInterval,

MulticastCipherSuite,

Credential

)

***Add the following parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| MulticastCipherSuite | MULTICAST\_CAP | (Optional) Specifies the multicast ciphersuite to be used for securing multicast MIH messages. Only one ciphersuite shall be included. |
| Credential | CREDENTIAL | (Optional) X.509 certificate |

* + - 1. MIH\_Register.confirm
				1. Semantics of service primitives

***Change the text as follows:***

MIH\_Register.confirm (

SourceIdentifier,

Status,

ValidTimeInterval,

MulticastCipherSuite,

Credential

)

***Add the following parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| MulticastCipherSuite | MULTICAST\_CAP | (Optional) Specifies the multicast ciphersuite to be used for securing multicast MIH messages. Only one ciphersuite shall be included. |
| Credential | CREDENTIAL | (Optional) X.509 certificate |

* + 1. MIH\_Event\_Subscribe
			1. MIH\_Event\_Subscribe.request
				1. Semantics of service primitive

***Change the text as follows:***

MIH\_Event\_Subscribe.request (

DestinationIdentifier,

LinkIdentifier,

GroupLinkIdentifier,

RequestedMihEventList,

EventConfigurationInfoList

)

***Insert and modify the following parameters:***

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| LinkIdentifier | LINK\_TUPLE\_ID | (Optional) Identifier of the link for event subscription. For local event subscription, PoA link address need not be present if the link type lacks such a value. This parameter shall be used if and only if DestinationIdentifier is an MIHF ID. |
| GroupLinkIdentifier | NET\_TYPE\_INC | (Optional) Identifier of a group of links for event subscription. This parameter shall be used if and only if DestinationIdentifier is an MIHF Group ID. |

* + 1. MIH\_Event\_Unsubscribe
			1. MIH\_Event\_Unsubscribe.request
				1. Semantics of service primitive

***Change the text as follows:***

MIH\_Event\_Unsubscribe.request (

DestinationIdentifier,

LinkIdentifier,

GroupLinkIdentifier,

RequestedMihEventList

)

***Insert and modify the following parameters:***

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| LinkIdentifier | LINK\_TUPLE\_ID | (Optional) Identifier of the link for event unsubscription. For local event unsubscription, PoA address in the Link Identifier need not be present if the link type lacks such a value. This parameter shall be used if and only if DestinationIdentifier is an MIHF ID. |
| GroupLinkIdentifier | NET\_TYPE\_INC | (Optional) Identifier of a group of links for event unsubscription. This parameter shall be used if and only if DestinationIdentifier is an MIHF Group ID. |

* + 1. MIH\_Link\_Get\_Parameters
			1. MIH\_Link\_Get\_Parameters.request
				1. Semantics of service primitive

***Change the text as follows:***

MIH\_Link\_Get\_Parameters.request (

DestinationIdentifier,

DeviceStatesRequest,

LinkIdentifierList,

GroupLinkIdentifier,

GetStatusRequestSet

 )

***Insert and modify the following parameters:***

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| LinkIdentifierList | LIST(LINK\_ID) | (Optional) List of link identifiers for which status is requested. If the list is empty, return the status of all available links. This parameter shall be used if and only if DestinationIdentifier is an MIHF ID. |
| GroupLinkIdentifier | NET\_TYPE\_INC | (Optional) Identifier of a group of links for which status is requested. This parameter shall be used if and only if DestinationIdentifier is an MIHF Group ID. |

* + 1. MIH\_Link\_Configure\_Thresholds
			1. MIH\_Link\_Configure\_Thresholds.request
				1. Semantics of service primitive

***Change the text as follows:***

MIH\_Link\_Configure\_Thresholds.request (

DestinationIdentifier,

ResponseFlag,

LinkIdentifier,

GroupLinkIdentifier,

ConfigureRequestList

)

***Insert and modify the following parameters:***

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| ResponseFlaga | RESPONSE\_FLAG | (Optional) Flag which represents whether or not a response is needed. |
| LinkIdentifier | LINK\_TUPLE\_ID | (Optional) Identifier of the link to be configured. This parameter shall be used if and only if DestinationIdentifier is an MIHF ID. |
| GroupLinkIdentifier | NET\_TYPE\_INC | (Optional) Identifier of a group of links to be configured. This parameter shall be used if and only if DestinationIdentifier is an MIHF Group ID. |

a In case the ResponseFlag parameter is not present, the MIHF should always generate a request message, otherwise the MIHF generates either a request or an indication message, based on the ResponseFlag parameter.

* + - * 1. Effect of receipt

If the destination of the request is the local MIHF itself, the local MIHF issues a Link\_Configure\_Thresholds.request to the lower layer link to set the thresholds for the link according to the specified configuration parameters.

If the destination of the request is a remote MIHF, based on the ResponseFlag parameter, the local MIHF generates and sends an MIH\_Link\_Configure\_Thresholds request or an MIH\_Link\_Configure\_Thresholds indication message to the remote MIHF. Upon the receipt of the message, the remote MIHF then issues a Link\_Configure\_Thresholds request to the lower layer link to set the thresholds for the link according to the specified configuration parameters.

* + 1. MIH\_Link\_Actions
			1. MIH\_Link\_Actions.request
				1. Semantics of service primitive

***Change the text as follows:***

MIH\_Link\_Actions.request (

DestinationIdentifier,

ResponseFlag,

LinkActionsList,

GroupLinkActionsList

)

***Insert and modify the following parameters:***

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| ResponseFlaga | RESPONSE\_FLAG | (Optional) Flag which represents whether or not a response is needed. |
| LinkActionsList | LIST(LINK\_ACTION\_REQ) | (Optional) Specifies the suggested actions. This parameter shall be used if and only if DestinationIdentifier is an MIHF ID. |
| GroupLinkActionsList | LIST(MULTICAST\_ACTION\_REQ) | (Optional) Specifies the suggested actions for a group of links. This parameter shall be used if and only if DestinationIdentifier is an MIHF Group ID. |

a In case the ResponseFlag parameter is not present, the MIHF should always generate a request message, otherwise the MIHF generates either a request or an indication message, based on the ResponseFlag parameter.

* + 1. MIH\_Net\_HO\_Commit
			1. MIH\_Net\_HO\_Commit.request
				1. Semantics of service primitive

***Change the text as follows:***

MIH\_Net\_HO\_Commit.request (

DestinationIdentifier,

ResponseFlag,

LinkType,

TargetNetworkInfoList,

AssignedResourceSet,

LinkActionExecutionDelay,

LinkActionsList,

MulticastLinkActionList

)

***Insert and modify the following parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| ResponseFlaga | RESPONSE\_FLAG | (Optional) Flag which represents whether or not a response is needed. |
| LinkActionExecutionDelay | UNSIGNED\_INT(2) | (Optional) Time (in ms) to elapse before an action needs to be taken. A value of 0 indicates that the action is taken immediately. Time elapsed is calculated from the instance the command arrives until the time when the execution of the action is carried out.This parameter shall be used for non-group operation. |
| LinkActionsList | LIST(LINK\_ACTION\_REQ) | (Optional) Specifies the suggested actions. This parameter shall be used if and only if DestinationIdentifier is an MIHF ID. |
| GroupLinkActionsList | LIST(MULTICAST\_ACTION\_REQ) | (Optional) Specifies the suggested actions for a group of links. This parameter shall be used if and only if DestinationIdentifier is an MIHF Group ID. |

a In case the ResponseFlag parameter is not present, the MIHF should always generate a request message, otherwise the MIHF generates either a request or an indication message, based on the ResponseFlag parameter.

* + - * 1. Effect on receipt

Upon receipt of this primitive an MIHF shall send an MIH\_Net\_HO\_Commit request or indication message to the destination, based on the ResponseFlag parameter.

* + - 1. MIH\_Net\_HO\_Commit.indication
				1. Semantics of service primitive

***Change the text as follows:***

MIH\_Net\_HO\_Commit.indication (

SourceIdentifier,

ResponseFlag,

LinkType,

TargetNetworkInfoList,

AssignedResourceSet,

LinkActionExecutionDelay,

LinkActionsList,

MulticastLinkActionList

)

***Insert and modify the following parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| ResponseFlaga | RESPONSE\_FLAG  | (Optional) Flag which represents whether or not a response is needed. |
| LinkActionExecutionDelay | UNSIGNED\_INT(2) | (Optional) Time (in ms) to elapse before an action needs to be taken. A value of 0 indicates that the action is taken immediately. Time elapsed is calculated from the instance the command arrives until the time when the execution of the action is carried out.This parameter shall be used for non-group operation. |
| LinkActionsList | LIST(LINK\_ACTION\_REQ) | (Optional) Specifies the suggested actions. This parameter shall be used if and only if DestinationIdentifier is an MIHF ID. |
| GroupLinkActionsList | LIST(MULTICAST\_ACTION\_REQ) | (Optional) Specifies the suggested actions for a group of links. This parameter shall be used if and only if DestinationIdentifier is an MIHF Group ID. |

a In case the ResponseFlag parameter is not present, the MIHF should always generate a request message, otherwise the MIHF generates either a request or an indication message, based on the ResponseFlag parameter.

***Add the following primitives at the end of subclause 7.4:***

* + 1. MIH\_Configuration\_Update
			1. MIH\_Configuration\_Update.request
				1. Function

This primitive is generated by a PoS to update the configuration of one or more MN(s) or other PoS(es).

* + - * 1. Semantics of service primitive

MIH\_Configuration\_Update.request (

DestinationIdentifier,

ConfigurationData

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies MIHF ID of the remote MIHF(s) to be configured. |
| ConfigurationData | OCTET\_STRING | Configuration data. Examples of this parameter include firmware and management parameters. |

* + - * 1. When generated

The MIH user generates this primitive to update the configuration of one or more MN(s) and/or other PoS(es).

* + - * 1. Effect on receipt

Upon receipt of this primitive, MIHF on the PoS sends the corresponding MIH\_Configuration\_Update indication message to the MN(s) or other PoS(es).

* + - 1. MIH\_Configuration\_Update.indication
				1. Function

This primitive is generated by an MIHF to update the configuration of one or more MN(s) or other PoS(es).

* + - * 1. Semantics of service primitive

MIH\_Configuration\_Update.indication (

SourceIdentifier,

TargetIdentifier,

ConfigurationData

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| SourceIdentifier | MIHF\_ID | Specifies MIHF ID of the remote MIHF that sent the MIH\_Configuration\_Update indication message. |
| TargetIdentifier | MIHF\_ID | The target MIHF group identifier for the group operation. |
| ConfigurationData | OCTET\_STRING | Configuration data. Examples of this parameter include firmware and management parameters. |

* + - * 1. When generated

This primitive is generated by an MIHF on a MN or a PoS when receiving an MIH\_Configuration\_Update indication message from a remote MIHF.

* + - * 1. Effect on receipt

Upon receipt of this primitive, an MIH user on a MN or a PoS may modify its configuration using the ConfigurationData parameter.

* + 1. MIH\_MN\_Group\_Manipulate
			1. MIH\_MN\_Group\_Manipulate.request
				1. Function

This primitive is generated by an MN to manipulate its own group membership.

* + - * 1. Semantics of service primitive

MIH\_MN\_Group\_Manipulate.request (

DestinationIdentifier,

TargetIdentifier,

GroupAction

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies group MIHF-ID of the remote MIHF peers. DestinationIdentifier may be different from TargetIdentifier. |
| TargetIdentifier | MIHF\_ID | The target MIHF group identifier for the group operation. |
| GroupAction | GROUP\_MGT\_ACTION | The action to be taken: Join/Leave the group. |

* + - * 1. When generated

The MIH user generates this primitive to request joining or leaving a group.

* + - * 1. Effect on receipt

Upon receipt of this primitive, MIHF on the MN sends the corresponding MIH\_MN\_Group\_Manipulate request message to the PoS.

* + - 1. MIH\_MN\_Group\_Manipulate.indication
				1. Function

This primitive is used by an MIHF to notify an MIH User that a MIH\_MN\_Group\_Manipulate request message has been received.

* + - * 1. Semantics of service primitive

MIH\_MN\_Group\_Manipulate.indication (

SourceIdentifier,

TargetIdentifier,

GroupAction

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Specifies MIHF-ID of the remote MIHF that issued MIH\_MN\_Group\_Manipulate.request. |
| TargetIdentifier | MIHF\_ID | The target MIHF group identifier for the group operation. |
| GroupAction | GROUP\_MGT\_ACTION | The action to be taken: Join/Leave the group. |

* + - * 1. When generated

This primitive is generated by an MIHF on a PoS when receiving an MIH\_MN\_Group\_Manipulate request message from a remote MIHF.

* + - * 1. Effect on receipt

Upon receipt of this primitive, an MIH user on a PoS may take the required actions as the action specified in GroupAction.

* + - 1. MIH\_MN\_Group\_Manipulate.response
				1. Function

This primitive is generated by an MIH User to acknowledge result of an MIH\_MN\_Group\_Manipulate request from an MN.

* + - * 1. Semantics of service primitive

MIH\_MN\_Group\_Manipulate.response (

DestinationIdentifier,

TargetIdentifier,

MulticastAddress,

SubgroupRange,

VerifyGroupKey,

UserSpecificData,

CompleteSubtree,

GroupKeyData,

GroupStatus,

SecurityAssociationID

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies the MIHF ID of the destination of the primitive |
| TargetIdentifier | MIHF\_ID | The target MIHF group identifier for the group operation. |
| MulticastAddress | TRANSPORT\_ADDR | (Optional) Multicast address corresponding with the target group identifier. |
| SubgroupRange | SUBGROUP\_RANGE | (Optional) Subgroup to process the command |
| VerifyGroupKey | VERIFY\_GROUP\_KEY | (Optional) Verification data for group key. |
| UserSpecificData | OCTET\_STRING | (Optional) Auxiliary data. |
| CompleteSubtree | COMPLETE\_SUBTREE | (Optional) Complete Subtree data. |
| GroupKeyData | GROUP\_KEY\_DATA | (Optional )Encrypted group key. |
| GroupStatus | GROUP\_STATUS | Status of the group operation |
| SecurityAssociationID | SEQUENCE(ID\_TYPE, ID\_VALUE) | (Optional) ID of the GKB generated SA. |

* + - * 1. When generated

An MIH User at the PoS generates this primitive after receipt and processing of MIH\_MN\_Group\_Manipulate request. This primitive returns the status of the action asked in the request. Optionally, it may respond with the security mechanisms required by the group.

* + - * 1. Effect on receipt

MIH\_MN\_Group\_Manipulate response message is sent back to the requester.

* + - 1. MIH\_MN\_Group\_Manipulate.confirm
				1. Function

This primitive is generated by an MIHF that receives an MIH\_MN\_Group\_Manipulate response to indicate the status of the group manipulation. The status of the group manipulation provides information regarding the result of a group join or leave operation, indicating the status after the command execution.

* + - * 1. Semantics of service primitive

MIH\_MN\_Group\_Manipulate.confirm (

SourceIdentifier,

TargetIdentifier,

GroupStatus,

SecurityAssociationID

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Specifies the MIHF ID of the remote MIHF |
| TargetIdentifier | MIHF\_ID | The target MIHF group identifier for the group operation. |
| GroupStatus | GROUP\_STATUS | Status of the group operation |
| SecurityAssociationID | SEQUENCE(ID\_TYPE, ID\_VALUE) | (Optional) ID of the GKB generated SA. |

* + - * 1. When generated

This primitive is sent to the MIH User after the MIHF receives an MIH\_MN\_Group\_Manipulate response message.

* + - * 1. Effect on receipt

The status of the group operation is noted.

* + 1. MIH\_Net\_Group\_Manipulate
			1. MIH\_Net\_Group\_Manipulate.request
				1. Function

This primitive is generated by the MIH User of a PoS to manipulate group membership of one or more MN(s) or other PoS(es).

* + - * 1. Semantics of service primitive

MIH\_Net\_Group\_Manipulate.request (

DestinationIdentifier,

ResponseFlag,

GroupKeyUpdateFlag,

TargetIdentifier,

MulticastAddress,

SubgroupRange,

VerifyGroupKey,

UserSpecificData,

CompleteSubtree,

GroupKeyData,

SecurityAssociationID

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies group MIHF-ID of the remote MIHF peers. DestinationIdentifier may be different from TargetIdentifier. |
| ResponseFlaga | RESPONSE\_FLAG | (Optional) Flag which represents whether or not a response is needed. |
| GroupKeyUpdateFlag | GROUP\_KEY\_UPDATE\_FLAG | Flag which represents whether or not a group key in GroupKeyData is updated. |
| TargetIdentifier | MIHF\_ID | The target MIHF group identifier for the group operation. |
| MulticastAddress | TRANSPORT\_ADDR | (Optional) Multicast address corresponding with the target group identifier. |
| SubgroupRange | SUBGROUP\_RANGE | (Optional) Subgroup to process the command |
| VerifyGroupKey | VERIFY\_GROUP\_KEY | (Optional) Verification data for group key. |
| UserSpecificData | OCTET\_STRING | (Optional) Auxiliary data. |
| CompleteSubtree | COMPLETE\_SUBTREE | Complete Subtree data. |
| GroupKeyData | GROUP\_KEY\_DATA | (Optional) Encrypted group key. |
| SecurityAssociationID | SEQUENCE(ID\_TYPE, ID\_VALUE) | (Optional) ID of the GKB generated SA. |

a In case the ResponseFlag parameter is not present, the MIHF should always generate a request message, otherwise the MIHF generates either a request or an indication message, based on the ResponseFlag parameter.

* + - * 1. When generated

The MIH user generates this primitive to create, delete or modify groupmembership.

* + - * 1. Effect on receipt

Upon receipt of this primitive, MIHF on the PoS sends the corresponding MIH\_Net\_Group\_Manipulate indication message or MIH\_Net\_Group\_Manipulate request message to the MN(s) or other PoS(es). The ResponseFlag TLV indicates which message shall be sent.

* + - 1. MIH\_Net\_Group\_Manipulate.indication
				1. Function

This primitive is used by an MIHF to notify an MIH User that a MIH\_Net\_Group\_Manipulate indication message or an MIH\_Net\_Group\_Manipulate request message has been received.

* + - * 1. Semantics of service primitive

MIH\_Net\_Group\_Manipulate.indication (

SourceIdentifier,

ResponseFlag,

TargetIdentifier,

UserSpecificData,

GroupStatus,

SecurityAssociationID

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Specifies MIHF-ID of the remote MIHF that issued MIH\_Net\_Group\_Manipulate.request. |
| ResponseFlag | RESPONSE\_FLAG  | (Optional) Flag which represents whether or not a response is needed. |
| TargetIdentifier | MIHF\_ID | The target MIHF group identifier for the group operation. |
| MulticastAddress | TRANSPORT\_ADDR | (Optional) Multicast address corresponding with the target group |
| UserSpecificData  | OCTET\_STRING | (Optional) Auxiliary dataa.  |
| GroupStatus | GROUP\_STATUS | Status of the group. |
| SecurityAssociationID | SEQUENCE(ID\_TYPE, ID\_VALUE) | (Optional) ID of the GKB generated SA. |

a The UserSpecificData parameter can be used to convey additional information such as version information of the GKB used or additional credentials.

* + - * 1. When generated

This primitive is generated by an MIHF on a MN or a PoS when receiving an MIH\_Net\_Group\_Manipulate indication message or an MIH\_Net\_Group\_Manipulate request message from a remote MIHF.

* + - * 1. Effect on receipt

Upon reception of this primitive, an MIH user on an MN or a PoS may join or leave the group specified in the TargetIdentifier parameter. The MIH User may also decrypt and install the encrypted group key associated with the group and contained in the GroupKeyData. The detailed procedure is described in subclause 9.4.2.

* + - 1. MIH\_Net\_Group\_Manipulate.response
				1. Function

This primitive is generated by an MIH User to acknowledge the result of an MIH\_Net\_Group\_Manipulate request from a PoS.

* + - * 1. Semantics of service primitive

MIH\_Net\_Group\_Manipulate.response (

DestinationIdentifier,

TargetIdentifier,

GroupStatus

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies the requestor of the group manipulation. |
| TargetIdentifier | MIHF\_ID | The target MIHF group identifier for the group operation. |
| GroupStatus | GROUP\_STATUS | Status of the group |

* + - * 1. When generated

An MIH User generates this primitive after receipt and processing of MIH\_Net\_Group\_Manipulate.request.

* + - * 1. Effect on receipt

MIH\_Net\_Group\_Manipulate response message is sent back to the group manipulate requester.

* + - 1. MIH\_Net\_Group\_Manipulate.confirm
				1. Function

This primitive is generated by an MIHF that receives an MIH\_Net\_Group\_Manipulate response to indicate the status of the group manipulation.

* + - * 1. Semantics of service primitive

MIH\_Net\_Group\_Manipulate.confirm (

SourceIdentifier,

TargetIdentifier,

GroupStatus

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Specifies the responder of the group manipulation. |
| TargetIdentifier | MIHF\_ID | The target MIHF group identifier for the group operation. |
| GroupStatus | GROUP\_STATUS | Status of the group |

* + - * 1. When generated

An MIH User generates this primitive after receipt and processing an MIH\_Net\_Group\_Manipulate request.

* + - * 1. Effect on receipt

MIH\_Net\_Group\_Manipulate response message is sent back to the group manipulate requester.

* + 1. MIH\_Pull\_Credential
			1. MIH\_Pull\_Credential.request
				1. Function

This primitive is generated by an MN or a PoS and it is used to request sending of a certificate from the destination PoS to the requestor.

* + - * 1. Semantics of service primitive

MIH\_Pull\_Credential.request (

DestinationIdentifier,

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies the sender of the credential. |

* + - * 1. When generated

An MN generates this primitive for requesting a credential or for credential updates.

* + - * 1. Effect on receipt

Upon receipt of this primitive, the MIHF on the MN sends the corresponding MIH\_Pull\_Credential request message to the destination MN or PoS.

* + - 1. MIH\_Pull\_Credential.indication
				1. Function

This primitive is generated by an MIHF that receives an MIH\_Pull\_Credential request message in order to inform the MIH User.

* + - * 1. Semantics of service primitive

MIH\_Pull\_Credential.indication (

SourceIdentifier,

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Identifies the requester of the credential. |

* + - * 1. When generated

This primitive is generated by an MIHF when an MIH\_Pull\_Credential request message is received.

* + - * 1. Effect on receipt

Upon reception of this primitive, the MIH user generates an MIH\_Pull\_Credential.response to deliver a credential to the requester.

* + - 1. MIH\_Pull\_Credential.response
				1. Function

This primitive is generated by an MIH User in order to deliver a credential to an MN or other PoS for MIH protocol protection as described in IEEE Std 802.21a-2012 Section 9.

* + - * 1. Semantics of service primitive

MIH\_Pull\_Credential.response (

DestinationIdentifier,

EncryptedCredential

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies the requestor of the credential. |
| EncryptedCredential | ENCRYPTED\_KEY | Encrypted credential used for creating an EAP-generated MIH SA. |

* + - * 1. When generated

An MIH User generates this primitive using a leaf key corresponding with the credential requester.

* + - * 1. Effect on receipt

Upon receipt of this primitive, the MIHF on the PoS generates an MIH\_Pull\_Credential response message to the destination MN or PoS.

* + - 1. MIH\_Pull\_Credential.confirm
				1. Function

This primitive is generated by an MIHF that receives an MIH\_Pull\_Credential response, in order to inform of the credential received by the MIH User.

* + - * 1. Semantics of service primitive

MIH\_Pull\_Credential.confirm (

SourceIdentifier,

Credential

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Identifies the remote MIHF that invoked MIH\_Pull\_Credential response. |
| Credential | CREDENTIAL | A credential for MIH protection as described in IEEE Std 802.21a-2012 Section 9 |

* + - * 1. When generated

The MIHF that receives an MIH\_Pull\_Credential response message generates this primitive to indicate the credential.

* + - * 1. Effect on receipt

After verification, validated credential keys within their expiration period can be utilized for IEEE 802.21a.

* + 1. MIH\_Push\_Credential
			1. MIH\_Push\_Credential.request
				1. Function

This primitive is generated by an MIH User at the PoS to send a Credential to a destination PoS or MN.

* + - * 1. Semantics of service primitive

MIH\_Push\_Credential.request (

DestinationIdentifier,

Credential

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies the recipient of the credential. |
| Credential | CREDENTIAL | A credential for MIH protection as described in IEEE Std 802.21a-2012 Section 9 |

* + - * 1. When generated

A PoS generates this primitive for initial provisioning of credentials or for credential updates.

* + - * 1. Effect on receipt

Upon receipt of this primitive, the MIHF on the PoS sends the corresponding MIH\_Push\_Credential request message to the destination MN or PoS.

* + - 1. MIH\_Push\_Credential.indication
				1. Function

This primitive is generated by an MIHF to notify a local MIH User that an MIH\_Push\_Credential request message has been received.

* + - * 1. Semantics of service primitive

MIH\_Push\_Credential.indication (

SourceIdentifier,

Credential

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Identifies the sender of the credential. |
| Credential | CREDENTIAL | A credential for MIH protection as described in IEEE Std 802.21a-2012 Section 9 |

* + - * 1. When generated

This primitive is generated by an MIHF when an MIH\_Push\_Credential request message is received.

* + - * 1. Effect on receipt

Credential signature is verified and result of verification is provided back to push requester by CredentialStatus. After verification, the validated credential public keys can be utilized for multicast message exchange within their expiration period.

* + - 1. MIH\_Push\_Credential.response
				1. Function

This primitive is generated by an MIH User to acknowledge receipt of a credential from a PoS.

* + - * 1. Semantics of service primitive

MIH\_Push\_Credential.response (

DestinationIdentifier,

CredentialStatus

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies the requestor of the credential revocation. |
| CredentialStatus | CERT\_STATUS | Indicates whether a credential has been verified and is now in use by the recipient. |

* + - * 1. When generated

An MIH User generates this primitive after receipt and processing of credential.

* + - * 1. Effect on receipt

If the credential signature is valid, then an MIH\_Push\_Credential response message is sent back to the push requester. The result of the request is provided in the CredentialStatus.

* + - 1. MIH\_Push\_Credential.confirm
				1. Function

This primitive is generated by an MIHF that receives an MIH\_Push\_Credential response to indicate the status of the credential inspection.

* + - * 1. Semantics of service primitive

MIH\_Push\_Credential.confirm (

SourceIdentifier,

CredentialStatus

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Identifies the remote MIHF that invoked MIH\_Revoke\_Credential.response. |
| CredentialStatus | CERT\_STATUS | Indicates whether a credential has been verified and is now in use by the recipient. |

* + - * 1. When generated

The MIHF that receives an MIH\_Push\_Credential response message generates this primitive to indicate the status of the credential inspection.

* + - * 1. Effect on receipt

If Credential Status is success, then it indicates the device is capable of receiving signed multicast messages.

* + 1. MIH\_Revoke\_Credential
			1. MIH\_Revoke\_Credential.request
				1. Function

This primitive is generated by a PoS used to revoke a credential.

* + - * 1. Semantics of service primitive

MIH\_Revoke\_Credential.request (

DestinationIdentifier,

CredentialSerialNumber

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies an MIHF or a group of MIHF peers to revoke the credential. |
| CredentialSerialNumber | CERT\_SERIAL\_NUMBER | X.509 certificate subfield – serial number |
| CredentialRevocation | SIGNATURE | Digital signature for a revoked X.509 certificate serial number generated by CA. |

* + - * 1. When generated

The MIH user generates this primitive to revoke a credential.

* + - * 1. Effect on receipt

Upon receipt of this primitive, the MIHF on the PoS sends the corresponding MIH\_Revoke\_Credential request message to the destination MIHF(s).

* + - 1. MIH\_Revoke\_Credential.indication
				1. Function

This primitive is generated by an MIHF to revoke a credential stored in MN(s) and PoS(es).

* + - * 1. Semantics of service primitive

MIH\_Revoke\_Credential.indication (

SourceIdentifier,

CredentialSerialNumber

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Specifies the remote MIHF that invoked MIH\_Revoke\_Credential.request primitive. |
| CredentialSerialNumber | CERT\_SERIAL\_NUMBER | X.509 certificate subfield – serial number |
| CredentialRevocation | SIGNATURE | Digital signature for a revoked X.509 certificate serial number generated by CA. |

* + - * 1. When generated

This primitive is generated by an MIHF on a MN or a PoS when receiving an MIH\_Revoke\_Credential request message from a remote MIHF.

* + - * 1. Effect on receipt

Upon receipt of this primitive, an MIH user on a MN or a PoS verifies Credential Revocation signature, and if it is valid, then it deprecate the credential specified by the CredentialSerialNumber and invokes a MIH\_Revoke\_Credential.confirm primitive.

* + - 1. MIH\_Revoke\_Credential.response
				1. Function

This primitive is generated by an MIH User to acknowledge receipt of a credential revocation request from a PoS.

* + - * 1. Semantics of service primitive

MIH\_Revoke\_Credential.response (

DestinationIdentifier,

CredentialStatus

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Specifies the remote MIHF that invoked MIH\_Revoke\_Credential.request primitive. |
| Credential Status | CERT\_STATUS | Indicates whether a credential has been revoked. |

* + - * 1. When generated

This primitive is generated by an MIHF on a MN or a PoS when receiving an MIH\_Revoke\_Credential request message from a remote MIHF.

* + - * 1. Effect on receipt

Upon receipt of this primitive, an MIH user on an MN or a PoS deprecate the credential specified by the CredentialSerialNumber and invokes a MIH\_Revoke\_Credential.confirm primitive.

* + - 1. MIH\_Revoke\_Credential.confirm
				1. Function

This primitive is generated by an MIHF that receives an MIH\_Revoke\_Credential response to indicate the status of the credential revocation.

* + - * 1. Semantics of service primitive

MIH\_Revoke\_Credential.confirm (

SourceIdentifier,

CredentialStatus

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Identifies the remote MIHF that invoked MIH\_Revoke\_Credential.response. |
| Credential Status | CERT\_STATUS | Indicates whether a credential has been revoked. |

* + - * 1. When generated

The MIHF that receives an MIH\_Revoke\_Credential response message generates this primitive to indicate the status of the credential revocation.

* + - * 1. Effect on receipt

If Credential Status indicates success for all the MIHF peers to which credential revocation request was sent, the PoS can change status of the credential to revoked.

1. Media independent handover protocol
	1. MIH protocol description
		1. MIH protocol acknowledgement service

***Modify the following paragraphs:***

The acknowledgement service shall be used when the MIH transport used for remote communication does not provide reliable services. When the MIH transport is reliable, the use of the acknowledgement service is not needed. The acknowledgement service is particularly useful when the underlying transport used for remote communication does not provide reliable services. When the MIH transport is reliable, the acknowledgement service is optional. In case the destination of the communication is a MIHF Group ID, the acknowledgement service shall not be used, even in cases when underlying transport is not reliable.

The source MIHF requests for an acknowledgement message to ensure successful receipt of an MIH protocol message. This MIH message is used to acknowledge the successful receipt of an MIH protocol message at the destination MIHF.

The MIH acknowledgement service is supported by the use of two bits of information that are defined exclusively for acknowledgement (ACK) usage in the MIH header. The ACK-Req bit is set by the source MIH node and the ACK-Rsp bit is set by the destination MIH node to utilize the acknowledgement service. It is expected that the underlying transport layer would take care of ensuring the integrity of the MIH protocol message during delivery.

When seeking acknowledgement service, the source MIH node shall start a retransmission timer after sending an MIH protocol message with the ACK-Req bit set and saves a copy of the MIH protocol message while the timer is active. The algorithm defined in IETF RFC 2988 is used to calculate the value of the retransmission timer. If the acknowledgement message is not received before the expiration of the timer, the source MIH node immediately retransmits the saved message with the same Message-ID and with the same Transaction-ID (with ACK-Req bit set). If the source MIH node receives the acknowledgement before the expiration of the timer on the first or any subsequent retransmitted attempt, then the source MIH node has ensured the receipt of the MIH packet and therefore, resets the timer and releases the saved copy of the MIH protocol message. During retransmission, if the source MIH node receives the acknowledgement for any of the previous transmission attempts then the source MIH node determines successful delivery of the message and does not have to wait for any further acknowledgements for the current message. The source MIH node retransmits an MIH protocol message with ACK-Req bit set until it receives an acknowledgment or the number of retransmissions reaches its maximum value. The maximum number of retransmissions can be configured through a parameter defined in the MIB (see Annex J). The source MIH node does not attempt to retransmit a message with same Message-ID and Transaction-ID when the ACK-Req bit was not set in the first MIH message. Implementations may consider adjusting the retransmission time-out (RTO) when operating over links with power save mobile nodes.

When a destination MIH node receives an MIH protocol message with the ACK-Req bit set, then the destination MIH node returns an MIH message with the ACK-Rsp bit set and copying the Message-ID and Transaction-ID from the received MIH protocol message. The MIH message with the ACK-Rsp bit set has only the MIH header and no other payload. In instances where the destination MIH node immediately processes the received MIH protocol message and a response is immediately available, then the ACK-Rsp bit is set in the corresponding MIH protocol response message.

The destination MIH node responds with an acknowledgement message for duplicate MIH messages (messages with same transaction-ID) that have the ACK-Req bit set. However, the destination MIH node does not process these duplicate messages if it has already done so. If a destination MIH node receives an MIH protocol message with no ACK-Req bit set, then no action is taken with respect to the acknowledgement service.

In all cases, the MIH protocol message in a transaction is processed only once at the destination MIH node, irrespective of the number of received messages with the ACK-Req bit set. The destination MIH node sets the ACK-Rsp bit in an MIH protocol response message and additionally requests acknowledgement by setting the ACK-Req bit for the same MIH protocol response message.

In case an MIH protocol message with destination MIHF Group ID is received with the ACK-Req bit set, the receiving station should ignore this flag.

* + 1. MIH protocol transaction state diagram
			1. Inter-state-machine procedures

***Change item c) of IEEE 802.21 as amended by IEEE 802.21b, as follows:***

1. **BOOLEAN IsMulticastMsg(MIH\_MESSAGE)**—This procedure outputs TRUE if the input message has an MIHF Group ID ~~zero length~~ destination MIHF ID. Otherwise, it outputs FALSE.
	* + 1. Transaction source and destination state machines
				1. Intra-state-machine variables

***Change item a) and add item b) as follows:***

1. **IsMulticast**—This variable’s type is BOOLEAN. When its value is TRUE, it indicates that a message has a ~~zero length~~ destination MIHF Group ID. Otherwise, its value is FALSE.
2. **ResponseSent** – This variable’s type is BOOLEAN. When its value is TRUE, it indicates that a Response message has been sent. Otherwise, its value is FALSE.
	* + - 1. Transaction destination state machine

***replace Figure 24 as follows:***

1. —Transaction destination state machine
	* 1. Other considerations
			1. MIHF discovery
				1. Solicited MIH capability discovery

***Change first paragraph in 8.2.4.3.4 as follows:***

An MIHF (the requestor) discovers its peer MIH functions and capabilities by sending an MIH\_Capability\_Discover request message to either its multicast domain with ~~zero length~~ a MIHF Group ID or a known MIHF ID, respectively. Only MIH network entities respond to a multicast MIH\_Capability\_Discover request.

***Change last paragraph of 8.2.4.3.4 as follows:***

If the MIH capability discovery is invoked upon receiving MIH capability advertisement in unauthenticated state through media specific broadcast messages, such as beacon frames and DCD, destination MIHF ID is filled with ~~a zero length~~ MIHF Broadcast ID and this message is transmitted over the control plane using an L2 management frame, such as an IEEE 802.11 management action frame or an IEEE 802.16 MAC management message. This message contains the SupportedMihEventList, SupportedMihCommandList, SupportedISQueryTypeList, SupportedTransportList, and MBBHandoverSupport TLVs to enable the receiving MIHF to discover the sending MIHF’s capability. Therefore, peer MIHF entities can discover each other’s MIH capabilities by one MIH protocol message transaction. When the requestor receives the unicast MIH\_Capability\_Discover response message, which is embedded in the media specific control message, it retrieves the responder.'s MIHF ID by checking the source of the MIH\_Capability\_Discover response message.

* 1. MIH protocol identifiers
		1. MIHF ID

***Change subclause 8.3.1 as amended by IEEE 802.21b as follows:***

MIHF Identifier (MIHF ID) is an identifier that is required to uniquely identify ~~an MIHF entity~~ a specific MIHF or a group of MIHF peers for delivering the MIH services. MIHF ID is used in all MIH protocol messages. This enables the MIH protocol to be transport agnostic.

MIHF ID is assigned to the MIHF during its configuration process. The configuration process is outside the scope of the standard.

~~Broadcast~~ MIHF Broadcast ID is defined as an MIHF Group ID of zero length. A ~~zero length~~ broadcast (zero length)MIHF ID may be used in an MIH message when destination MIHF ID is not known to a source MIHF. MIHF Group ID is used when a message is addressed to a group of MIHF peers. The following MIH messages can use a ~~zero length~~ broadcast MIHF ID:

1. **MIH Messages for Management Service:**
	1. MIH\_Capability\_Discover request
2. **MIH Messages for Command Service:**
	1. MIH\_Link\_Get\_Parameters request
	2. MIH\_Link\_Configure\_Thresholds request
	3. MIH\_Net\_HO\_Bcst\_Commit indication
3. **MIH Messages for Information Service:**
	1. MIH\_Push\_Information indication

In addition the following rules apply to the case of messages addressed to a MIHF Group ID:

* Multicast transmission is not allowed for MIES.
* Multicast transmission in general is not allowed for messages sent by the MN. Hence, commands in the form of MIH\_MN\_\* cannot use multicast transmission.
* Multicast transmission is not allowed for MIH\_NET\_SAP primitives.
* Multicast transmission is not allowed for MIH\_LINK\_SAP primitives.

 In particular, the following MIH messages can use an MIHF Group ID. In the next list, when a message can be sent by a PoS and an MN, the only allowed multicast transmission is when the message is sent by the PoS:

1. **MIH Messages for Management Service:**
	1. MIH\_Registration request
	2. MIH\_DeRegister request
	3. MIH\_Net\_Group\_Manipulate request
	4. MIH\_Net\_Group\_Manipulate indication
	5. MIH\_Net\_Push\_Credential request
	6. MIH\_Push\_Credential request
	7. MIH\_Revoke\_Credential request
	8. MIH\_Pull\_Credential request
2. **MIH Messages for Command Service:**
	1. MIH\_Link\_Get\_Parameters request
	2. MIH\_Link\_Configure\_Thresholds request
	3. MIH\_Link\_Actions request
	4. MIH\_Net\_HO\_Candidate\_Query request
	5. MIH\_N2N\_HO\_Query\_Resources request
	6. MIH\_Net\_HO\_Commit request
	7. MIH\_Configuration\_Update indication
	8. MIH\_Event\_Subscribe request
	9. MIH\_Event\_UnSubscribe request
3. **MIH Messages for Information Service:**
	1. MIH\_Push\_Information indication

The MIHF ID is of type MIHF\_ID. (See F.3.11.)

* 1. MIH protocol frame format
		1. General frame format

**Change numberi****ng of subclause 8.4.1a to 8.4.2**

* + 1. Protected MIH protocol frame format

***Change second paragraph of subclause 8.4.2 as follows:***

A protected MIH PDU is an MIH PDU that has an MIH header with S bit set to one indicating that the MIH service specific TLVs in this PDU are ~~protected~~ encrypted or the PDU is digitally signed. When the MIH service specific TLVs in this PDU are encrypted, ~~Each~~ each security association is defined for a pair of MIHF identifiers and is identified by a security association identifier (SAID). Therefore, for a protected MIH PDU, when a security association identifier is defined and the PDU is not digitally signed, the Source and Destination MIHF identifier TLVs may not be present. In this case, an MIH header is followed by an SAID TLV, which is followed by a security TLV. When no SAID TLV is carried, Service Specific TLVs shall be carried without encryption and therefore no Security TLV is carried. A Signature TLV is carried when a multicast PDU is digitally signed. When an MIH message is multicast and the S bit is set, Source and Destination Identifier TLVs and an SAID TLV shall be carried in which the ID\_VALUE of the SAID TLV contains a NULL string.

***Change numbering of Figure 28a to Figure 29 and modify as follows:***

1. —Protected MIH frame format

***NOTE—Modify accordingly the reference num******ber of Figures 28x accordingly:***

* + - 1. MIH PDU protected by (D)TLS

***Add the following text at the end of subclause 8.4.2.1:***

A Signature TLV shall not be carried when MIH PDU is protected by (D)TLS.

* + - 1. MIH PDU protected through EAP-generated MIH SA

***Add the following text at the end of subclause 8.4.2.2:***

A Signature TLV shall not be carried when MIH PDU is protected through EAP-generated MIH SA.

***Add the following subclause:***

* + - 1. MIH PDU protected through GKB-generated MIH SA

When GKB is used the MIH SA must be protected through it. A group MIH SA is established among a group of MIHF peers. It includes a ciphersuite used for the protection. A security association identifier is assigned by the PoS as a result of successful GKB procedure. Figure 34 shows a protected MIH PDU for GKB-generated MIH SA with a Signature TLV. The protection procedure is specified in 9.4.1.

***Insert the following figure at the end of subclause 8.4.2.3:***

1. —MIH PDU protected by a GKB-generated MIH SA with a signature TLV

***Modify subclause numbering:***

* + - 1. Protected MIH PDU upon transport address change
		1. Fragmentation and reassembly
	1. Message parameter TLV encoding
	2. MIH protocol messages
		1. MIH messages for service management
			1. MIH\_Capability\_Discover request

***Change 8.6.1.1 as follows:***

If a requesting MIHF entity does not know the destination MIHF entity’s MIHF ID, the requesting MIHF entity may fill its destination MIHF ID with an ~~zero length~~ MIHF Broadcast ID to send this capability discover message.

* + - 1. MIH\_Capability\_Discover response

***Change 8.6.1.2 as follows:***

The corresponding MIH primitive of this message is defined in **Error! Reference source not found.**. This message is sent in response to an MIH\_Capability\_Discover request message that was destined to an MIHF ID or an ~~zero length~~ MIHF Broadcast ID.

* + - 1. MIH\_Register request

***Change 8.6.1.3 as follows:***

This message is transmitted to the remote MIHF to perform a registration or re-registration. The message must contain the Link identifier TLV or Multicast link identifier TLV.

***Insert and modify the following parameters:***

|  |
| --- |
| MIH Header Fixed Fields (SID=1, Opcode=1, AID=2) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| LinkIdentifier (Optional)(Link identifier TLV) |
| GroupLinkIdentifier (Optional)(Multicast link identifier TLV) |
| RequestCode(Register request code TLV) |

* + - 1. MIH\_Register response

***Insert the following parameters:***

|  |
| --- |
| MIH Header Fixed Fields (SID=1, Opcode=2, AID=2) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| Status (Status TLV) |
| ValidTimeInterval (not included if Status does not indicate “Success”) (Valid time interval TLV) |
| MulticastCipherSuite(Multicast Ciphersuite TLV) |
| Credential(Credential TLV) |

* + - 1. MIH\_DeRegister request
			2. MIH\_DeRegister response
			3. MIH\_Event\_Subscribe request

***Change 8.6.1.7 as follows:***

This message is sent by a remote MIHF (the subscriber) to subscribe to one or more event types from a particular event origination point. The message must contain the Link identifier TLV or Multicast link identifier TLV.

***Insert and modify the following parameters:***

|  |
| --- |
| MIH Header Fixed Fields (SID=1, Opcode=1, AID=4) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| LinkIdentifier (Optional)(Link identifier TLV) |
| GroupLinkIdentifier (Optional)(Multicast link identifier TLV) |
| RequestedMihEventList(MIH event list TLV) |
| EventConfigurationInfoList (Optional)(Event configuration info list TLV) |

* + - 1. MIH\_Event\_Subscribe response
			2. MIH\_Event\_Unsubscribe request

***Change 8.6.1.9 as follows:***

This message is sent by a remote MIHF (the subscriber) to unsubscribe from a set of link-layer events. The message must contain the Link identifier TLV or Multicast link identifier TLV.

***Insert and modify the following parameters:***

|  |
| --- |
| MIH Header Fixed Fields (SID=1, Opcode=1, AID=5) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| LinkIdentifier (Optional)(Link identifier TLV) |
| GroupLinkIdentifier (Optional)(Multicast link identifier TLV) |
| RequestedMihEventList(MIH event list TLV) |

* + - 1. MIH\_Event\_Unsubscribe response
			2. MIH\_Auth indication
			3. MIH\_Auth request
			4. MIH\_Auth response
			5. MIH\_Termination\_Auth request
			6. MIH\_Termination\_Auth response
			7. MIH\_Push\_key request
			8. MIH\_Push\_key response
			9. MIH\_LL\_Auth request
			10. MIH\_LL\_Auth response

***Add the following subclauses:***

* + - 1. MIH\_Configuration\_Update indication

The corresponding MIH primitive of this message is defined in 7.4.30.2.

This message is used by the MIHF to change configuration of the MIH node(s) identified by the Destination Identifier.

The Destination Identifier is passed to the local MIH User as a TargetIdentifier in a MIH\_Configuration\_Update.indication.

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=3, AID=10 ) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| ConfigurationData(Configuration Data TLV) |

* + - 1. MIH\_MN\_Group\_Manipulate request

The corresponding MIH primitive of this message is defined in 7.4.31.1.

This message is used by the MIHF to manipulate group membership of MIH node(s) identified by the Destination Identifier.

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=1, AID=11) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| TargetIdentifier(Group Identifier TLV) |
| GroupAction(Group Action TLV) |

* + - 1. MIH\_MN\_Group\_Manipulate response

The corresponding MIH primitive of this message is defined in 7.4.31.3.

This message is used by the MIHF to supply the group status of MIH node(s) identified by the Source Identifier.

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=2, AID=11 ) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| TargetIdentifier(Group Identifier TLV) |
| SequenceNumber (conditional)ª(Sequence Number TLV) |
| MulticastAddress (Optional)(Multicast Address TLV) |
| SubgroupRange (Optional)(Subgroup\_Range TLV) |
| VerifyGroupKey (Optional)(Verify Group Key TLV) |
| UserSpecificData (Optional)(Aux Data TLV) |
| CompleteSubtree (Optional)(Complete Subtree TLV) |
| GroupKeyData (Optional)(Group Key Data TLV) |
| GroupStatus(Group Status TLV) |
| SecurityAssociationID (Optional) (SAID TLV) |

ª This parameter is only used in the case CCM encryption method is used and the group key is not updated.

* + - 1. MIH\_Net\_Group\_Manipulate request

The corresponding MIH primitive of this message is defined in 7.4.32.1.

This message is used by the MIHF to manipulate group membership of MIH node(s) identified by the Destination Identifier.

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=1, AID=12 ) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| GroupKeyUpdateFlag(Group Key Update Flag TLV) |
| TargetIdentifier(Group Identifier TLV) |
| SequenceNumber (Optional)a(Sequence Number TLV) |
| MulticastAddress (Optional)(Multicast Address TLV) |
| SubgroupRange (Optional)(Subgroup Range TLV) |
| VerifyGroupKey (Optional)(Verify Group Key TLV) |
| UserSpecificData (Optional)(Aux Data TLV) |
| CompleteSubtree(Complete Subtree TLV) |
| GroupKeyData (Optional)(Group Key Data TLV) |
| SecurityAssociationID (Optional) (SAID TLV) |

a This parameter is only used in the case CCM encryption method is used and the group key is not updated.

* + - 1. MIH\_Net\_Group\_Manipulate indication

The corresponding MIH primitive of this message is defined in 7.4.32.2.

This message is used by the MIHF to manipulate group membership of MIH node(s) identified by the Destination Identifier.

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=3, AID=12 ) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| TargetIdentifier(Group Identifier TLV) |
| GroupKeyUpdateFlag(Group Key Update Flag TLV) |
| SequenceNumber (Optional)(Sequence Number TLV) |
| MulticastAddress (Optional)(Multicast Address TLV) |
| SubgroupRange (Optional)(Subgroup Range TLV) |
| VerifyGroupKey (Optional)(Verify Group Key TLV) |
| UserSpecificData (Optional)(Aux Data TLV) |
| CompleteSubtree(Complete Subtree TLV) |
| GroupKeyData (Optional)(Group Key Data TLV) |
| SecurityAssociationID (Optional) (SAID TLV) |

* + - 1. MIH\_Net\_Group\_Manipulate response

The corresponding MIH primitive of this message is defined in 7.4.32.3.

This message is used by the MIHF to inform group status of MIH node(s) identified by the Source Identifier.

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=2, AID=12 ) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| TargetIdentifier(Group Identifier TLV) |
| UserSpecificData (Optional)(Aux Data TLV) |
| GroupStatus(Group Status TLV) |

* + - 1. MIH\_Pull\_Credential request

The corresponding MIH primitive of this message is defined in 7.4.33.1.

This message is used by the MIHF to request a credential to the PoS identified by the Destination Identifier.

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=1, AID=13 ) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |

* + - 1. MIH\_Pull\_Credential response

The corresponding MIH primitive of this message is defined in 7.4.33.3.

This message is used by the MIHF to deliver a credential from a PoS used for creating an EAP-generated MIH SA. EncryptedCredential is decrypted by the leaf key of the MN.

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=2, AID=13 ) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| EncryptedCredential(EncryptedCredential TLV) |

* + - 1. MIH\_Push\_Credential request

The corresponding MIH primitive of this message is defined in 7.4.34.1.

This message is used by the MIHF to deliver a credential encrypted by the leaf key that the MIH node identified by the Destination Identifier holds to the MIH node.

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=1, AID=14 ) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| Credential(Credential TLV) |

* + - 1. MIH\_Push\_Credential response

The corresponding MIH primitive of this message is defined in 7.4.34.3.

This message is used by the MIHF to acknowledge receipt of a credential from a PoS.

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=2, AID=14) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| CredentialSerialNumber(Credential Serial Number TLV) |
| CredentialStatus(Credential Status TLV) |

* + - 1. MIH\_Revoke\_Credential request

The corresponding MIH primitive of this message is defined in 7.4.35.1.

This message is used by the MIHF to revoke a credential.

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=1, AID=15 ) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| CredentialSerialNumber(Credential Serial Number TLV) |
| CredentialRevocation(Credential Revocation Signature TLV) |

* + - 1. MIH\_Revoke\_Credential response

The corresponding MIH primitive of this message is defined in 7.4.35.3.

This message is used by the MIHF to acknowledge receipt of a credential revocation request from a PoS.

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=2, AID=15) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| CredentialStatus(Credential Status TLV) |

* + 1. MIH messages for event service
			1. MIH\_Link\_Detected indication
			2. MIH\_Link\_Up indication
			3. MIH\_Link\_Down indication
			4. MIH\_Link\_Parameters\_Report indication
			5. MIH\_Link\_Going\_Down indication
			6. MIH\_Link\_Handover\_Imminent indication
			7. MIH\_Link\_Handover\_Complete indication
		2. MIH messages for command service
			1. MIH\_Link\_Get\_Parameters request

***Change 8.6.3.1 as follows:***

This message is used to discover the status of currently available links. The message must contain the Link identifier list TLV or Multicast link identifier TLV.

***Insert and modify the following parameters:***

|  |
| --- |
| MIH Header Fixed Fields (SID=3, Opcode=1, AID=1) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| DeviceStatesRequest (Optional)(Device states request TLV) |
| LinkIdentifierList (Optional)(Link identifier list TLV) |
| GroupLinkIdentifier (Optional)(Multicast link identifier TLV) |
| GetStatusRequestSet(Get status request set TLV) |

* + - 1. MIH\_Link\_Get\_Parameters response
			2. MIH\_Link\_Configure\_Thresholds request

***Change 8.6.3.3 as follows:***

This message is used to configure thresholds of the lower layer link. The message must contain the Link identifier TLV or Multicast link identifier TLV.

***Insert and modify the following parameters:***

|  |
| --- |
| MIH Header Fixed Fields (SID=3, Opcode=1, AID=2) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| LinkIdentifier (Optional)(Link identifier TLV) |
| GroupLinkIdentifier (Optional)(Multicast link identifier TLV) |
| ConfigureRequestList (Configure request list TLV) |

* + - 1. MIH\_Link\_Configure\_Thresholds response

***Add the following subclause:***

* + - 1. MIH\_Link\_Configure\_Thresholds indication

The corresponding MIH primitive of this message is defined in 7.4.15.1.

This message is used to configure thresholds of the lower layer link when an MIHF Group ID is used as Destination Identifier. The message must contain the Link identifier TLV or Multicast link identifier TLV.

***Insert and modify the following parameters:***

|  |
| --- |
| MIH Header Fixed Fields (SID=3, Opcode=3, AID=2) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| LinkIdentifier (Optional)(Link identifier TLV) |
| GroupLinkIdentifier (Optional)(Multicast link identifier TLV) |
| ConfigureRequestList (Configure request list TLV) |

* + - 1. MIH\_Link\_Actions request

***Change the message format as follows:***

|  |
| --- |
| MIH Header Fields (SID=3, Opcode=1, AID=3) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| LinkActionsList (Optional)(Link actions list TLV) |
| MulticastLinkActionList (Optional)(Multicast link action list TLV) |

* + - 1. MIH\_Link\_Actions response

***Add the following subclause:***

* + - 1. MIH\_Link\_Actions indication

The corresponding MIH primitive of this message is defined in 7.4.16.1.

This message is used to control the behavior of a set of lower layer links when an MIHF Group ID is used as Destination Identifier. The message must contain the Link identifier TLV or Multicast link identifier TLV.

|  |
| --- |
| MIH Header Fields (SID=3, Opcode=3, AID=3) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| LinkActionsList (Optional)(Link actions list TLV) |
| MulticastLinkActionList (Optional)(Multicast link action list TLV) |

* + - 1. MIH\_Net\_HO\_Candidate\_Query request
			2. MIH\_Net\_HO\_Candidate\_Query response
			3. MIH\_MN\_HO\_Candidate\_Query request
			4. MIH\_MN\_HO\_Candidate\_Query response
			5. MIH\_N2N\_HO\_Query\_Resources request
			6. MIH\_N2N\_HO\_Query\_Resources response
			7. MIH\_MN\_HO\_Commit request
			8. MIH\_MN\_HO\_Commit response
			9. MIH\_Net\_HO\_Commit request

***Change the message format as follows:***

|  |
| --- |
| MIH Header Fields (SID=3, Opcode=1, AID=7) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| LinkType(Link type TLV) |
| TargetNetworkInfoList(List of target network info TLV) |
| AssignedResourceSet (Optional)(Assigned resource set TLV) |
| LinkActionExecutionDelay (Optional)(Time interval TLV) |
| LinkActionsList (Optional)(Link actions list TLV) |
| GroupLinkActionsList (Optional)(Multicast Link action list TLV) |

* + - 1. MIH\_Net\_HO\_Commit response
			2. MIH\_Net\_HO\_Commit indication

***Change the message format as follows:***

|  |
| --- |
| MIH Header Fields (SID=3, Opcode=3, AID=7) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| LinkType(Link type TLV) |
| TargetNetworkInfoList(List of target network info TLV) |
| AssignedResourceSet (Optional)(Assigned resource set TLV) |
| LinkActionExecutionDelay (Optional)(Time interval TLV) |
| LinkActionsList (Optional)(Link actions list TLV) |
| GroupLinkActionsList (Optional)(Multicast Link action list TLV) |

* + - 1. MIH\_N2N\_HO\_Commit request
			2. MIH\_N2N\_HO\_Commit response
			3. MIH\_MN\_HO\_Complete request
			4. MIH\_MN\_HO\_Complete response
			5. MIH\_N2N\_HO\_Complete request
			6. MIH\_N2N\_HO\_Complete response
			7. MIH\_Net\_HO\_Bcast\_Commit indication
		1. MIH messages for information service
			1. MIH\_Get\_Information request
			2. MIH\_Get\_Information response
			3. MIH\_Push\_Information indication
1. MIH protocol protection

***NOTE—Editor: The following subclauses are not changed, they appear in order to be able to reference them later.***

* 1. Protection established through MIH (D)TLS
	2. Key establishment through an MIH service access authentication
		1. MIH service access authentication
		2. Key derivation and key hierarchy
	3. MIH message protection mechanisms for EAP-generated SAs

***Insert the following subclause:***

* 1. Multicast MIH message protection mechanisms
		1. MIH message protection mechanisms using GKB-generated SAs

Group Key Block (GKB) is a data field used to distribute master group keys (MGKs) to protect MIH multicast/broadcast commands. A GKB contains GroupkeyVerificationCode, CompleteSubtree and GroupKeyData (see 7.4.31.3 and 7.4.32.1). A group manipulation command accompanies a target MIHF Group ID and a GKB. An MN follows the procedure described in 9.4.2.2.2 to determine whether it should try to recover the key from the GKB. If an MN succeeds in recovering an MGK, the MN will keep the pair of the target MIHF Group ID and the MGK, which means that the MN belongs to the group designated by the target MIHF Group ID. Otherwise, if an MN fails to derive an MGK from the GKB, it means that the MN does not belong to the group designated by the target MIHF Group ID. Then, the MN leaves the group discarding the stored pair of MIHF Group ID and MGK.

A series of group commands may follow a group manipulation command which defines a target group of MNs. A group command is issued, for instance, to instruct the group that the members should handover to a PoA or that they should update their configuration parameters. A payload of a group command can be protected (encrypted) using an SA derived from the MGK. The following two steps describe how group manipulation and command delivery are performed:

Step 1: A Command center, which is an MIH PoS, issues a group manipulation command to instruct MNs to join or leave a group. A group manipulation command may also be used to update a group key stored at a target MN. Group manipulation commands may be delivered to MNs through existing multicast channels. A multicast channel is associated with a group: If an MN joins a group then it starts listening to the multicast channel associated with the group. The address used by this multicast channel can be provided by the group manipulation command itself.

Step 2: A Command center issues to a group of MNs a group command (not a group manipulation command) to instruct the MNs in the group to take an action. The target group is designated by the MIHF Group ID field in the group command. A group command is delivered through the multicast channel associated with the MIHF Group ID. A group command may alternatively take two types of payload: Encrypted and non-encrypted. If a payload is encrypted, it is encrypted with a key derived from the current group key.

Each MN has a Device Key, which is a sequence of AES keys. The number of keys in a Device Key is 8, 16, 24 or 32, which is a system-wide constant. All MNs which are managed by a Group Manager have the same number of device keys. The format of Device Key may vary depending on its implementation and is out of the scope of this specification. For convenience, an example format of a Device Key is described in clause 9.4.2.2 and Annex U. When confidentiality is not required for group manipulation, a GKB without GroupKeyData should be used. Note that each MN need not have a Device Key if the GKBs always have no GroupKeyData.

A Command center has a module called GKB Generator. A GKB Generator receives combinations of a Device Key, a Leaf Number and an MGK, where the Device Keys cover all the MNs that constitute a group. A Leaf Number is uniquely associated with a Device Key. Definition of Leaf Number is given in Section 9.4.3. The MGK is the group key for the concerned group. On receiving a set of such combinations, a GKB Generator outputs a GKB, or several GKBs. An example of generation of GKBs is described in Section ???.

* + 1. Secure group manipulation with group key distribution

Figure 45 illustrates group manipulation command distribution initiated by a Command center via a multicast channel. The MIH User of the Command center generates an MIH\_Net\_Group\_Manipulate.request, described in 7.4.32, and then it passes the request to the MIHF of the Command center. Upon receiving the request, the MIHF generates MIH\_Net\_Group\_Manipulate indication (Note that the decision on sending an indication message or a request message depends on the ResponseFlag parameter of the MIH\_Net\_Group\_Manipulate.request primitive), described in 0, and sends it to the MNs via multicast mechanisms. When an MN receives the MIH\_Net\_Group\_Manipulate indication message, the MIHF of the MN processes the message. After processing the message, the MIHF sends MIH\_Group\_Manipulate.indication to the MIH User of the MN.

1. —Example of group manipulation distribution using multicast mechanisms
	* + 1. MIH User of a Command Center

Note that this section is informative. Required components in an MIH User of a Command Center relevant to group manipulation and group commands are listed as follows:

* A GKB Generator.
* All the MIHF IDs and all the Device Keys each of which is uniquely associated with one of the MIHF IDs. Each Device Key accompanies a Leaf Number.
* A Group Management Database which stores a groups table, a members table and a memberships table. The groups table stores the existing groups. It at least has the following two columns: mihf\_group\_id and mgk. A row of the table tells that a group designated by the mihf\_group\_id exists and has the mgk for the master group key. The members table stores the group members (MNs): It at least has the following three columns: mihf\_id, device\_key and leaf\_number. A row of the table tells that an MN designated by the mihf\_id exists and has the device\_key with the leaf\_number. And, the memberships table stores associations between the groups and the members: It at least has the following two columns: mihf\_group\_id and mihf\_id A row of the table tells that the MN designated by the mihf\_id belongs to the group designated by the mihf\_group\_id.

An MIH User generates MIH\_Net\_Group\_Manipulate.request described in 7.4.32 as follows:

1. Define a group to manipulate:
	1. If it is a new group, choose a TargetGroupIdentifier by consulting with the Group Management Database. A TargetGroupIdentifier should be an MIHF Group ID which is not currently in use as an MIHF Group ID for an existing group. Then, decide group members, i.e. MNs, which belongs to the group. Choose an MGK for the group. Add a row to the groups table: The row contains the chosen TargetGroupIdentifier and the MGK. And, add rows to the memberships table for all the group members: Each of the rows contains the TargetGroupIdentifier and the MIHF ID of a group member.
	2. For an already existing group, obtain all the group members in the group. As necessary, add new group members to them and remove group members from them so that the members of the group are updated. Choose an MGK for the group. It may be equal to the current MGK. If a new MGK is chosen, update by the new MKG the row in the groups table containing the TargetGroupIdentifier of the existing group. Add the rows to the memberships table where the rows have the TargetGroupIdentifier and the MIHF IDs of a new group members. Remove the rows from the memberships table where the rows have the TargetGroupIdentifier and the MIHF IDs of group members which are removed from the group.
2. Send to the GKB Generator all the Device Keys and the associated Leaf Numbers of the group members determined in a) and the MGK. Then, the MIH User receives from the GKB generator a GKB or a set of GKBs: A GKB contains a CompleteSubtree field, a GroupKeyData field and optionally a SubgroupRange field. A SubgroupRange is a pair of Leaf Numbers and defines a range of Leaf Numbers. A simple example which shows how to make those fields is given in ???. A GKB contains a SubgroupRange field if it is one of divided GKBs. Note that one MIH\_Net\_Group\_Manipulate.request contains one and only one GKB. Plural GKBs result in plural requests.
3. (Optional) Construct the UserSpecificData field.
4. Choose a DestinationIdentifier. A DestinationIdentifier is a Group MIHF ID which represents an existing group. The SubgroupRange indicates the MNs which are the distribution targets of the GKB. If an MN is in the range, it should receive the divided GKB. At least, an MIHF Broadcast Identifier is assumed to exist. Other initial groups may exist though they are out of the scope of this specification.
5. Generate an MIH\_Net\_Group\_Manipulate.request from the DestinationIdentifier, the TargetGroupIdentifier, the SubgroupRange (an option), the VerifyGroupKey (an option), the UserSpecificData (an option), the CompleteSubtree and the GroupKeyData (an option). Set the GroupKeyUpdateFlag if the MGK of the group designated by the TargetGroupIdentifier should be updated. Send it to the local MIHF.
	* + 1. MIHF of a Command Center

Required components relevant to group manipulation and group commands are listed as follows:

* A signing key. The key is for creation of a signature of the Command center.
* A Device Key to retrieve a group key from a GKB which is received from the local MIH User.
* A Multicast Address Database which stores a multicast addresses table which has the following four columns: mihf\_group\_id, multicast\_address, mgk and sa\_id. The multicast\_address on a row is associated with the group designated by the mihf\_group\_id recorded on the same row. Additionally, the multicast\_address may accompany an attribute which indicates if it is defined at Layer 2 or 3 of the protocol stack. The mgk is the one derived from the latest GKB targetted to the group of the mihf\_group\_id. The sa\_id is the SAID for the mgk. A Multicast Address Database may also have a saids table which has the following two columns at least: mihf\_group\_id and sa\_id. A saids table stores all the SAIDs which have ever been assigned to the group of the mihf\_group\_id. A saids table is used to check if a generated SAID is unique up to the group. If a SAID is chosen monotoneously increasing for instance, the saids table is not necessary.

It is assumed that the MIHF is able to obtain in some way a multicast address associated with a Group MIHF ID. In this case, if the TargetGroupIdentifier in the received request is not registered in the database, obtain the multicast address associated with the TargetGroupIdentifier and update the database with the DestinationIdentifier and the associated multicast address.

The MIHF of the Command center receives an MIH\_Net\_Group\_Manipulate.request which is generated by the MIH User, the MIHF generates and sends an MIH\_Net\_Group\_Manipulate indication message to a multicast group. Note that this behavior depends on the ResponseFlag parameter. When “ResponseFlag=1”, the MIHF will generate MIH\_Net\_Group\_Manipulate request message. When “ResponseFlag=0”, the MIHF will generate MIH\_Net\_Group\_Manipulate indication message. In this example, we assume “ResponseFlag=0”.

1. Generate a Source MIHF ID TLV using its own MIHF ID.
2. Generate a Destination MIHF ID TLV from the DestinationIdentifier in the received MIH\_Group\_Manipulate.request.
3. Generate a Group Identifier TLV from the TargetIdentifier in the received MIH\_Group\_Manipulate.request.
4. Generate, as needed, a Multicast Address TLV from the multicast address corresponding to the TargetIdentifier in the received MIH\_Net\_Group\_Manipulate.request. The Multicast Address Database can serve for the purpose of finding the multicast address.
5. (Optional) Generate a SubgroupRange TLV from the SubgroupRange in the received MIH\_Net\_Group\_Manipulate.request.
6. (Optional) Generate a Verify Group Key TLV from the VerifyGroupKey in the received MIH\_Net\_Group\_Manipulate.request.
7. (Optional) Generate an Aux Data TLV from the UserSpecificData in the received MIH\_Net\_Group\_Manipulate.request.
8. Generate a Complete Subtree TLV from the CompleteSubtree in the received MIH\_Net\_Group\_Manipulate.request.
9. Generate a Group Key Data TLV from the GroupKeyData in the received MIH\_Net\_Group\_Manipulate.request.
10. Process the GKB (the Complete Subtree TLV and the Group Key Data TLV) using the Device Key assigned to the MIHF, and obtain the MGK. If the MIHF fails to obtain the master group key, the MIHF shall cancel the rest of the process.
11. Ask the Multicast Address Database and obtain the current MGK and the current SAID for the TargetGroupIdentifier. If the obtained MGK is equal to the MKG derived in j), do nothing here. Otherwise, generate a new SAID which is unique up to the group, and update the mgk and the sa\_id on the row for the TargetGroupIdentifier in the multicast addresses table.
12.
13. Send the MIH\_Net\_Group\_Manipulate indication message to the multicast address corresponding to the DestinationIdentifier.
	* + 1. MIHF of an MN

Required components relevant to group manipulation and group commands are listed as follows:

* A Device Key.
* A certificate of a Command Center which contains a verification key. The verification key is for verification of a signature made by the Command Center.
* A Group Database which stores a groups table, which has the following three columns at least: mihf\_group\_id, mgk and multicast\_address. A row of the table tells that this MN belongs to the group designated by the mihf\_group\_id. The group has the mgk as the master group

When a client MN receives a group manipulation command, i.e., an MIH\_Net\_Group\_Manipulate indication message, issued by a Command center, the MIHF of the MN processes the command. Suppose at first that the GKB in the group manipulation command has a group key data element:

1. The MIHF obtains a Source Identifier from the Source MIHF ID TLV.
2. The MIHF verifies the Signature TLV using the verification key corresponding to the obtained SourceIdentifier. If the verification fails, the MIHF shall cancel the following steps and stop processing the command.
3. The MIHF checks the DestinationIdentifier in the Destination MIHF ID TLV. If the DestinationIdentifier does not match one of the following MIHF IDs, the MIHF shall cancel the following steps and stop processing the command: (i) An MIHF Group ID corresponding to a broadcast address, (ii) an MIHF Group ID which is registered with a multicast address in the Group Database, or (iii) the MN's own MIHF ID.
4. Decrypt the payload if it is encrypted, i.e., if it is a Security TLV. The decryption key is the one associated with the DestinationIdentifier in the Group Database.
	1. In case an MN cannot decrypt the Security TLV, the message will be silently discarded.
5. If a SubgroupRange TLV exists in the indication, the MIHF obtains a SubgroupRange and checks whether its own Leaf Number is contained in the SubgroupRange or not. If it is not, the MIHF shall cancel the following steps and stop processing.
6. The MIHF obtains the TargetIdentifier in the Group Identifier TLV.
7. A GKB is composed of the Complete Subtree TLV, the Group Key Data TLV and optionally the Verify Group Key TLV. The MIHF processes the Complete Subtree TLV and the Group Key Data TLV as described in 9.4.2.2.2. If a Verify Group Key TLV exists, the MIHF verifies the group key derived from the GKB. If an MGK is obtained (and verified), go to the next step. Otherwise, go to Step i). In case the Verify Group Key TLV is not present in the GKB, if an MGK is obtained, go to the next step without verification of the obtained MKG. Otherwise, go to Step i).
8. The MIHF checks whether the TargetIdentifier obtained in Step f) has already been registered or not in the Group Database. If it has been, go to Step j) [Stay]. Otherwise, go to Step k) [Join].
9. The MIHF checks whether the TargetIdentifier has already been registered or not in the Group Database. If it has been, go to Step m) [Leave]. Otherwise, go to Step j) [Stay].
10. [Stay] The MIHF throws an MIH\_Net\_Group\_Manipulate.indication described in 7.4.32.2 to the MIH User. The GroupStatus field of the indication shall be “Unchanged successful” (5). The procedure of command processing terminates.
11. [Join] The MIHF obtains a multicast address associated with the TargetIdentifier and starts listening to it. The messages come through the multicast channel may be encrypted with the group key obtained in Step g). The multicast address may be obtained from a server (Note that this operation is out of the scope of this specification). Or, the received indication may accompany it in the Multicast Address TLV. Save in the Group Database the TargetIdentifier, the associated multicast address and the group key obtained in Step f).
12. The MIHF throws an MIH\_Net\_Group\_Manipulate.indication described in 7.4.32.2 to the MIH User. The GroupStatus field must be “Join operation successful” (0). The procedure of command processing terminates.
13. [Leave] The MIHF finds the multicast address recorded on the same row as the TargetIdentifier obtained in Step f) and the MIHF stops listening to it. The MIHF discards the row which has the TargetIdentifier.
14. The MIHF throws an MIH\_Net\_Group\_Manipulate.indication described in 7.4.32.2 to the MIH User. The GroupStatus field must be “Leave operation successful” (3). The procedure of command processing terminates.

Then, suppose that the GKB in the group manipulation command has no group key data part:

1. The MIHF obtains a Source Identifier from the Source MIHF ID TLV.
2. The MIHF verifies the Signature TLV using the verification key corresponding to the obtained SourceIdentifier. If the verification fails, the MIHF shall cancel the following steps and stop processing the command.
3. The MIHF checks the DestinationIdentifier in the Destination MIHF ID TLV. If the DestinationIdentifier does not match one of the following MIHF IDs, the MIHF shall cancel the following steps and stop processing the command: (i) A Group MIHF ID corresponding to a broadcast address, (ii) a Group MIHF ID which is registered with a multicast address in the Group Database, or (iii) the MN's own MIHF ID.
4. Decrypt the payload if it is encrypted, i.e., if it is a Security TLV. The decryption key is the one associated with the DestinationIdentifier in the Group Database.
5. If a SubgroupRange TLV exists in the indication, the MIHF obtains a SubgroupRange and check whether its own Leaf Number is contained in the SubgroupRange or not. If it is not, the MIHF shall cancel the following steps and stop processing.
6. The MIHF obtains a TargetIdentifier in the Group Identifier TLV.
7. A GKB is composed of the Complete Subtree TLV. The MIHF processes the Complete Subtree TLV as described in 9.4.2.2.2. If the MIHF succeeds to find a matching pair of GKB Indices, go to the next step. Otherwise, go to Step i).
8. The MIHF checks whether the TargetIdentifier obtained in Step f) has already been registered or not in the Group Database. If it has been, go to the Step j) [Stay]. Otherwise, go to Step k) [Join].
9. The MIHF checks whether the TargetIdentifier obtained in Step f) has already been registered or not in the Group Database. If it has been, go to Step m) [Leave]. Otherwise, go to Step j) [Stay].
10. [Stay] The MIHF issues an MIH\_Net\_Group\_Manipulate.indication described in 7.4.32.2 to the MIH User. The GroupStatus field of the indication must be “Unchanged successful” (5). The process terminates.
11. [Join] The MIHF obtains a multicast address associated with the TargetIdentifier and starts listening to it. The multicast address may be obtained from a server. Or, the received indication may accompany it in the Multicast Address TLV. Save in the Group Database the TargetIdentifier, the associated multicast address.
12. The MIHF issues an MIH\_Net\_Group\_Manipulate.indication described in 7.4.32.2 to the MIH User. The GroupStatus field must be “Join operation successful” (0). The procedure of command processing terminates.
13. [Leave] The MIHF finds the multicast address recorded on the same row as the TargetIdentifier obtained in f) and the MIHF stops listening to it. The MIHF discards the row which has the TargetIdentifier.
14. The MIHF issues an MIH\_Net\_Group\_Manipulate.indication described in 7.4.32.2 to the MIH User. The GroupStatus field must be “Leave operation successful” (3). The procedure of command processing terminates.

Subclause 7.4.31 introduces a mechanism enabling the MN to trigger the Join/Leave operations controlled by the Command center. In order to do so, the MIH User located at the MN notifies the Command center of its desire to Join or Leave a group through the use of the MIH\_MN\_Group\_Manipulate primitive. The Command center, upon receiving the associated request message, performs the same process as defined in this subclause, for the use of the MIH\_Net\_Group\_Manipulate, although in this case, the group to be manipulated is provided by the MN. The resulting GKB parameters are returned to the MN in the MIH\_MN\_Group\_Manipulate response message.

* + 1. GKB operation by the complete subtree method

A GKB is generated based on a binary tree. It is assumed that a key is assigned to each node of the binary tree. The key is sometimes called Node Key. A node of the binary tree is naturally identified with a bit sequence. (See Figure 46 for an example of depth 3) The maximum length of bit sequences is equal to the Depth of the binary tree. The Depth shall be one of the following values: 8, 16, 24 or 32. The Depth depends on design of the system, and it is a system-wide constant.

1. —Example of GKB operation by the complete subtree method (this figure corresponds to a depth of 3)

A GKB contains a complete subtree part and a group key data part. A group key data part appears when a GKB is used to deliver a group key. A complete subtree part is, for instance, the field of CompleteSubtree in an MIH\_NET\_Group\_Manipulate.request defined in 7.4.32.1. And, a group key data part is, for instance, the field of GroupKeyData in an MIH\_Net\_Group\_Manipulate.request defined in 7.4.32.1. A complete subtree part is a list of GKB Indices. The data type of a complete subtree part is COMPLETE\_SUBTREE, and the data type of a GKB Index is GKB\_INDEX. A GKB Index is a pair of a Node Bit Length and a Node Index. The data type of a Node Index is NODE\_INDEX, and the data type of a Node Bit Length is NODE\_BIT\_LENGTH (as defined in **Error! Reference source not found.**). A Node Index stores the bit sequence which represents a node, and the Node Bit Length paired with the Node Index is an octet which stores the length of the bit sequence. A Node Index is left aligned in the Internet byte order. The size of a Node Index varies depending on the value of the preceding Node Bit Length. If the value of a Node Bit Length is not greater than 8, the size of the following Node Index is 1 octet (= 8 bits). If the value of a Node Bit Length is greater than 8 and not greater than 16, the size of the following Node Index is 2 octets. If the value of a Node Bit Length is greater than 16 and not greater than 24, the size of the following Node Index is 3 octets. And, if the value of a Node Bit Length is greater than 24 and not greater than 32, the size of the following Node Index is 4 octets. There is one-to-one correspondence between a GKB Index and a node of the binary tree. A Node Index shall have a zero padding added to the right. This GKB Index represents the node ‘10011’ in the binary tree of Depth 8. A Node Index for a leaf node is sometimes called Leaf Number. An example of GKB Index is (0x05, 0b10011000). Another example of GKB Index is (0x14, 0b1100101000011100), which represents the node ‘11001010000111’ in the binary tree of Depth 16. If a system adopts the binary tree of Depth 16, the size of a Node Index in the system is 1 octet or 2 octets.

Note that GKB Indices are sequenced following the ascending “dictionary order”: Let (L1, I1) and (L2, I2) be two GKB Indices, where L1 and L2 are Node Bit Lengths and I1 and I2 are Node Indices. The dictionary order on the set of GKB Indices is defined as follows: (L1, I1) <= (L2, I2) if and only if L1 < L2 or (L1 == L2 and I1 <= I2), where L1, L2, I1 and I2 are considered as natural numbers.

A group key data part of a GKB is a sequence of encrypted group keys, where a group key is encrypted by Node Keys. The data type of a group key data part is GROUP\_KEY\_DATA, and the data type of an encrypted group key is ENCRYPTED\_GROUP\_KEY. There is one-one correspondence between the complete subtree part and the group key data part in a GKB. The number of GKB Indices in the complete subtree part is equal to the number of encrypted group keys in the group key data part. And, the n-th encrypted group key is a group key encrypted by the Node Key assigned to the node designated by the n-th GKB Index. The encryption is made using the AES-ECB mode.

* + - 1. Encapsulation/Decapsulation

An MIHF has a Device Key. The format of a Device Key may vary depending on implementations and is out of the scope of this specification. The procedures of GKB encapsulation/decapsulation are explained here using an example format of Device Key. A Device Key is a sequence of Device Key Units. The number of Device Key Units in a Device Key is equal to the Depth of the tree. A Device Key Unit is a pair of a GKB Index and a Node Key. The GKB Index in a Device Key Unit represents the node of the binary tree to which the Node Key in the Device Key Unit is assigned. A Device Key stores nodes and the associated Node Keys along a path in the binary tree which starts at the root node and arrives at a leaf node descending the tree.

* + - * 1. Encapsulation

**Error! Reference source not found.** provides an example of creation of a GKB at a GKB Generator.

* + - * 1. Decapsulation

At first, the decapsulation procedure for a GKB with a group key data part is described as follows:

1. An MIHF finds a GKB Index in the complete subtree part of the GKB and a Device Key Unit in the Device Key that the MIHF itself owns such that the GKB Index and the GKB Index of the Device Key Unit are identical. Suppose that the GKB Index thus found is the n-th GKB Index in the complete subtree part. If the MIHF fails to find such GKB Indices, the procedure shall terminate.
* If the procedure terminates here, it means that the MN does not belong to the group designated by the TargetIdentifier defined in 7.4.32.1. The MN shall leave the group if it currently belongs to the group.
1. Using the Node Key in the Device Key Unit found in a), the MIHF decrypts the n-th encrypted group key in the group key data part. The result of the decryption is a group key KG.
* The group key KG is the group key for the group designated by the TargetIdentifier. The MN shall belong to the group.
1. If there exists a field of VerifyGroupKey in the MIH\_Net\_Group\_Manipulate.request defined in 7.4.32.1, check the MAC in the VerifyGroupKey field using the group key KG. If it fails, the decapsulation procedure shall abort.

If a GroupKeyData TLV is absent in an MIH\_Net\_Group\_Manipulate indication message, an MIHF recognizes that it carries a GKB without a group key data. The following procedures apply:

1. An MIHF tries to find a GKB Index in the complete subtree part of the GKB and a GKB Index in the Device Key which the MIHF itself owns such that the two GKB Indices are identical.
* If the MIHF fails to find a matching pair, it means that the MN does not belong to the group designated by the TargetIdentifier defined in 7.4.32.1. The MN leaves the group if it currently belongs to the group.
* If the procedure succeeds to find a matching pair, it means that the MN belongs to the group designated by the TargetIdentifier defined in 7.4.32.1. The MN joins in the group if it does not currently belong to the group.

Note that an MN need not necessarily have a Device Key when GKBs without keys are used. Then, an MN is only required to have a sequence of GKB Indices.

* + 1. Multicast message encryption based on group key

1. —Key derivation example

When an MN successfully recovers a GKB, it obtains a master group key (MGK). The following three keys are derived from MGK:

* Group key confirmation key (MIGKCK) used as a key confirmation key to confirm that the correct MGK is obtained through a Message Authentication Code (MAC);
* Group manipulation encryption key (MIGMEK) used to protect a group manipulation command;
* Group encryption key (MIGEK) used to protect the group command.”

The deriving key is specified by the different multicast ciphersuites described in 9.4.6. For the key derivation, the following notations and parameters are used.

* *K*: key derivation key. It is truncated from a master group key (MGK). The length of *K* is determined by the pseudorandom function (PRF) used for key derivation. If HMAC-SHA-1 or HMAC-SHA-256 is used as a PRF, then the full MGK is used as key derivation key, *K*. If CMAC-AES is used as a PRF, then the first 128 bits of MGK are used as derivation key, *K*.
* *L*: The binary length of derived keying material MIGSK. *L* is determined by selected multicast ciphersuites described in 9.4.6.
* *h*: The output binary length of PRF used in the key derivation. That is, *h* is the length of the block of the keying material derived by one PRF execution. Specifically, for HMAC-SHA-1, *h* = 160 bits; for HMAC-256, *h* = 256 bits; for CMAC-AES, *h* = 128 bits.
* *n*: The number of iterations of PRF in order to generate *L*-bits keying material.
* *c*: The multicast ciphersuite code is a one octet string specified for each ciphersuite. The code is defined in 9.4.6.
* *v*: The length of the binary representation of the counter and the length of keying material L. The default value for *v* is 32.
* “MIGSK”: 0x4D4947534B, ASCII code in hex for string “MIGSK.”
* [a]2: Binary representation of integer *a* with a given length.

For given PRF, the key derivation for MIGSK can be described in the following procedures:

**Fixed input values**: h and v.

**Input**: *K*, *L*, and multicast ciphersuite code.

**Process**:

1. If *n* > 2v-1, then indicate an error and stop.
2. Result(0) := empty string.
3. For i = 1 to *n*, do
	1. *K*(i) := PRF(*K*, “MIGSK” || [i]2 || *c* || [*L*]2).
	2. Result(i) = Result(i-1) || *K*(i).
4. Return Result(n) and MIGSK is the leftmost *L* bits of Result(n).

**Output**: MIGSK.

The MIGSK is parsed in such a way that

MIGSK = MIGIK || MIGMEK || MIGEK.

With the above procedure, a key hierarchy is derived as shown in Figure 47.

* + 1. Multicast message encryption based on group key

In order to issue an MIH\_Configuration\_Update indication message, the MIH User of the Command center generates an MIH\_Configuration\_Update.request described in 7.4.30.1 and delivers it to the local MIHF. Upon receiving the request, the MIHF of the Command center behaves as follows:

1. The MIHF generates a Source MIHF ID TLV based on its own MIHF ID.
2. The MIHF generates a Destination MIHF ID TLV based on the DestinationIdentifier in the received request.
3. The MIHF generates a Configuration Data TLV from the ConfigurationData in the received request.
4. Consulting with the Multicast Address Database, the MIHF finds the multicast address associated with the DestinationIdentifer in the received request.
5. The MIHF generates an MIH\_Configuration\_Update indication message described in 8.6.1.20, and it sends it to the multicast address found in Step d).
	1. The Configuration Data TLV in the MIH\_Configuration\_Update indication message may be encrypted to make a Security TLV if necessary in the scheme described in 8.4.2.

When an MIHF of an MN receives an MIH\_Configuration\_Update indication message, it issues an MIH\_Configuration\_Update.indication described in 7.4.30.2 to its MIH User, following the next steps:

1. The Destination Identifier is retrieved from the Destination MIHF ID TLV. The MIHF checks if the Destination Identifier is registered in the Group Database or not. If it is not, the message is not for the MN. Thus, it cancels the following steps and stops processing.
2. The Source Identifier is retrieved from the Source MIHF ID TLV.
3. The MIHF verifies the Signature TLV using the verification key corresponding with the preceding Source Identifier. If the verification fails, it cancels the following steps and abort.
4. The ConfigurationData is retrieved from the Configuration Data TLV. If it is encrypted, The MIHF decrypts the Security TLV with the group key associated with the Destination Identifier in the Group Database.
5. With this information, the MIHF generates an MIH\_Configuration\_Update.indication as described in 7.4.30.2.

1. —Example of configuration update distribution using multicast mechanisms
2. In order to fill the SourceIdentifer, TargetIdentifier and ConfigurationData fields of the MIH\_Configuration\_Update.indication primitive, the MIHF copies the Source Identifier, the Destination Identifier and the Configuration Data respectively of the MIH\_Configuration\_Update indication message.
3. Finally, the MIHF issues the MIH\_Configuration\_Update.indication created in f) to its MIH User.
	* 1. Signature and Credential Management

In order to enable signing functionality, the message source requests credentials for public key using an out-of-band mechanism that is not specified in this document. The message source provides the credentials to destination devices. Message signing procedure, signature verification procedure and certificate management procedure are described in 9.4.5.1, 9.4.5.2 and 9.4.5.3, respectively.

* + - 1. Multicast Message Signatures

Multicast Messages are signed with the message source using a private key of the message source. Integrity and proof of origin of a multicast message is verified by verifying the message signature with the public key of a message source.

On receipt of signed multicast message there is an optional response indicating the validity of signature. Message source requests credentials for key updates. Message source provides updates of credentials to destination devices (with overlap period).

The message content is signed using elliptical curve cryptography.

* + - 1. Signature Verification

The signature is verified using the message source signature verification key. The endpoints might have more than one key used for signature verification. This is to allow for key updates to happen in an efficient manner for large systems.

The message source will identify which key is to be used for the multicast message so that verification will utilize the correct key for signature verification.

* + - 1. Certificate Management

A root of trust will exist for the multicast nodes. The root of trust is envisioned to be a certificate authority. X.509 format certificates will be utilized. The root of trust will establish the binding between the identity of the message source and the public/private key pair used for signature generation and verification.

The certificate will include the identity of the certificate authority, the identity of the message source, the public key in use and the expiration date of the certificate and the certificate authority’s signature. For an endpoint (an MN or PoS) to trust the certificate it must have the certificate authority public key.

The initial certificates for multicast signature verification are distributed to multicast destinations as part of the provisioning process to the multi-node network. The certificates will include the certificate authority certificate used to verify the initial and updated certificates.

There will also be one or more certificates that are bound to the identity of the multicast source.

As part of the key update or revocation process, a new certificate will be provided to multicast destinations using the multicast mechanism. There needs to be a mechanism for multicast destinations to acknowledge the receipt of the multicast message.

When there is a suspicion that a certificate is compromised, a mechanism will be provided to revoke the certificate from service. This mechanism will utilize the multicast messaging mechanism. Multicast destinations will need to provide a reply that indicates they have successfully revoked the certificate.

***Insert the following subclause.***

* + 1. Multicast Ciphersuites

The ciphersuites used for securing multicast MIH message is defined in Table 26.

1.
2. —Multicast Ciphersuites

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Code | Encryption Algorithm for Group Manipulation | Encryption Algorithm for Group Command | Digital Signature Algorithm | MAC Algorithm for Verify Group Key |
| 10000000 | NULL | NULL | NULL | NULL |
| 10001000 | AES\_CCM-128 | AES\_CCM-128 | ECDSA-224 | AES\_CMAC-128 |
| 10001001 | AES\_CCM-128 | AES\_CCM-128 | ECDSA-256 | AES\_CMAC-128 |
| 10001100 | AES\_CCM-128 | NULL | ECDSA-224 | AES\_CMAC-128 |
| 10001101 | AES\_CCM-128 | NULL | ECDSA-256 | AES\_CMAC-128 |
| 10010000 | NULL | NULL | ECDSA-224 | NULL |
| 10010001 | NULL | NULL | ECDSA-256 | NULL |

* 1. Common procedures
		1. Sending

When a PoS issues an MIH Service Specific TLV, the MIHF of the PoA generates a signature of the TLV using the signing key of the PoS and creates a Signature TLV from the generated signature.

* + 1. Receiving

When an MN receives an MIH service specific TLV, the MIHF of the MN behaves as follows:

1. The MIHF verifies the signature in the Signature TLV using the verification key corresponding to the Source Identifier extracted from the received Source MIHF ID TLV. If the verification fails, it cancels the following steps and stops processing.
2. The Destination Identifier is extracted from the received Destination MIHF ID TLV. The MIHF checks if the Destination Identifier is registered as an MIH Group ID in the Group Database. If it is not, it cancels the following steps and stops processing.
3. If a Security TLV is found in the MIH Specific TLV, the MIHF decrypts the Security TLV using the MIGMEK derived from the MKG. The MGK is the group key corresponding to the Destination Identifier extracted in the previous step. The group key is found in the Group Database.

# (informative)Bibliography

Bibliographical references are resources that provide additional or helpful material but do not need to be understood or used to implement this standard. Reference to these resources is made for informational use only.

# (normative)Data type definition

## Derived data types

### Data types for link identification and manipulation

***Insert new row in Table F.4 as follows:***

Table F.4—Data types for links

|  |  |  |
| --- | --- | --- |
| Data type name | Derived from | Definition |
| MULTICAST\_ACTION\_REQ | SEQUENCE( NET\_TYPE\_INC, CHOICE(NULL, LINK\_ADDR), LINK\_ACTION, LINK\_AC\_EX\_TIME) | A set of handover action request parameters destined to a group of links. The choice of LINK\_ADDR is to provide PoA address information when the LINK\_ACTION contains the attribute for DATA\_FWD\_REQ. |

### Data type for MIHF identification

***Change the following row in Table F.19:***

Table F.19—Data type for MIH identification

|  |  |  |
| --- | --- | --- |
| Data type name  | Derived from | Definition |
| MIHF\_ID  | OCTET\_STRING | The MIHF Identifier: MIHF\_ID is a network access identifier (NAI). NAI shall be unique as per IETF RFC 4282. If L3 communication is used and MIHF entity resides in the network node, then MIHF\_ID isthe fully qualified domain name or NAI-encoded IP address (IP4\_ADDR or IP6\_ADDR) of the entity that hosts the MIH Services.If L2 communication is used then MIHF\_ID is the NAI-encoded linklayer address (LINK\_ADDR) of the entity that hosts the MIH services.In an NAI-encoded IP address or link-layer address, each octet of binary-encoded IP4\_ADDR, IP6\_ADDR and LINK\_ADDR data is encoded in the username part of the NAI as .“\.” followed by the octet value. ~~MIHF ID of zero length may be used when a destination MIHF ID is not known.~~ An MIHF broadcast identifier is defined as an MIHF ID of zero length. Ant MIHF group identifier is defined as a NAI-encoded multicast link-layer address in the case L2 communication is used, a NAI-encoded IP address (IP4\_ADDR or IP6\_ADDR) in case L3 communication is used or any other NAI that is not already used as an MIHF identifier. MIHFs and MIH Users that use MIH group identifiers shall maintain a list of MIH group identifiers allocated via MIH group manipulation primitives in order to distinguish them from MIHF identifiers.”When an MIH protocol message with ~~zero length~~ MIHF broadcast ID is transmitted over the L2 data plane, a group MAC address (01-80-C2-00-00-0E) shall be used (see IEEE P802.1aj/D2.2). The maximum length is 253 octets.  |

### Data type for security

***Insert new rows in Table F.24 as follows:***

Table F.24—Data type for security

|  |  |  |
| --- | --- | --- |
| Data type name  | Derived from | Definition |
| CREDENTIAL | OCTET\_STRING | Provides a X.509 Certificate |
| CERT\_SERIAL\_NUMBER | OCTET\_STRING | Provides X.509 formatted certificate serial number which are unique by certificate authority. |
| CERT\_STATUS | ENUMERATED | This indicates the status of the certificate being pushed or revoked0: Not Present – indicates that certificate is not present 1: Certificate Valid – indicates that certificate is present and that the associated public key is being used to verify signatures2: Certificate Revoked3: Certificate Expired4: Verification Failed – indicates that the signature validation of the credential failed |
| COMPLETE\_SUBTREE | LIST (GKB\_INDEX) | The data type for the complete subtree part of a GKB. See 9.4.2.1 for the details. |
| ENCRYPTED\_KEY | OCTET(16) | This is the base data type for GROUP\_KEY\_DATA. This store a key of 16 octets encrypted with an AES key of 16 octets. |
| ID\_TYPE | ENUMERATED | The type of security association.0: TLS-generated;1: EAP-generated2: GKB-generated |
| GKB\_INDEX | SEQUENCE( NODE\_BIT\_LENGTH, NODE\_INDEX) | This is the base data type for COMPLETE\_SUBTREE. |
| GROUP\_KEY\_DATA | LIST (ENCRYPTED\_KEY) | The data type for the key data part of a GKB. See 9.4.2.1 for the details. |
| GROUP\_KEY\_UPDATE\_FLAG | ENUMERATED | This indicates if the group key is to be updated0: Key is not to be updated1: Key is to be updated |
| GROUP\_MGT\_ACTION | ENUMERATED | This indicates a manipulation command.0: Join the group.1: Leave the group. |
| GROUP\_STATUS | ENUMERATED | This indicates a status of group manipulation command.0: Join operation successful1: Unauthorized to join the group2: Leave operation successful3: Unchanged |
| MIH\_SEC\_CAP | SEQUENCE(TLS\_CAP,EAP\_CAP,MULTICAST\_CAP,) | Represents the MIH security capabilities. |
| MULTICAST\_CAP | UNSIGNED\_INT(2) | A multicast ciphersuite. Available multicast ciphersuites are defined in 9.4.6.  |
| NODE\_BIT\_LENGTH | UNSIGNED\_INT(1) | This stores the bit length of the following NODE\_INDEX. |
| NODE\_INDEX | CHOICE ( UNSIGNED\_INT(1), UNSIGNED\_INT(2), UNSIGNED\_INT(3), UNSIGNED\_INT(4) ) | This stores the index of a node of the binary tree. See 9.4.2.1 for the details. |
| RESPONSE\_FLAG | ENUMERATED | This indicates if an answer is required0: No response is needed1: Response is needed |
| SIGNATURE | OCTET\_STRING | A digital signature data. |
| SUBGROUP\_RANGE | CHOICE( SEQUENCE( UNSIGNED\_INT(1), UNSIGNED\_INT(1)),SEQUENCE( UNSIGNED\_INT(2), UNSIGNED\_INT(2)),SEQUENCE( UNSIGNED\_INT(3), UNSIGNED\_INT(3)),SEQUENCE( UNSIGNED\_INT(4), UNSIGNED\_INT(4))) | A range of valid leaf identifiers in a complete subtree of a GKB. The first integer indicates the lowest value of the range. The second integer indicates the highest value of the range. |
| VERIFY\_GROUP\_KEY | SEQUENCE ( OCTETS(16), OCTETS(16)) | The first OCTET(16) is arbitrary data, which is an input message to AES-CMAC (defined in RFC-4493). The second OCTET(16) is the MAC value for the first OCTET(16) to be verified. |

# (normative)MIH protocol message code assignments

***Modify Table L.1 as follows:***

Table L.1 —AID assignment

|  |  |
| --- | --- |
| MIH messages | AID |
| MIH messages for Service Management |
| MIH\_Configuration\_Update | 10 |
| MIH\_MN\_Group\_Manipulate | 11 |
| MIH\_Net\_Group\_Manipulate | 12 |
| MIH\_Pull\_Credential | 13 |
| MIH\_Push\_Credential | 14 |
| MIH\_Revoke\_Credential | 15 |

***Modify Table L.2 as follows:***

Table L.2 —Type values for TLV encoding

|  |  |  |
| --- | --- | --- |
| TLV type name | TLV type value | Data Type |
| Aux Data | 79 | OCTET\_STRING |
| Configuration Data | 80 | OCTET\_STRING |
| Credential Revocation Signature | 81 | SIGNATURE |
| Credential | 82 | CREDENTIAL |
| Credential Serial Number | 83 | CERT\_SERIAL\_NUMBER |
| Credential Status | 84 | CERT\_STATUS |
| Complete Subtree | 85 | COMPLETE\_SUBTREE |
| Encrypted Credential | 86 | ENCRYPTED\_KEY |
| Group Action  | 87 | GROUP\_MGT\_ACTION |
| Group Identifier | 88 | MIHF\_ID |
| Group Key Data | 89 | GROUP\_KEY\_DATA  |
| Group\_Status | 90 | GROUP\_STATUS |
| Multicast Address | 91 | TRANSPORT\_ADDRESS |
| Multicast Ciphersuite | 92 | MULTICAST\_CAP |
| Multicast Link Action List | 93 | LIST(MULTICAST\_ACTION\_REQ) |
| Multicast Link Identifier | 94 | NET\_TYPE\_INC |
| Response Flag  | 95 | RESPONSE\_FLAG |
| Sequence Number | 96 | OCTET\_STRING |
| Signature | 97 | SIGNATURE |
| Subgroup Range | 98 | SUBGROUP\_RANGE |
| Verify Group Key | 99 | VERIFY\_GROUP\_KEY |

# (informative)GKB toy example

An example is introduced to explain the basic principle of GKB and how to make a GKB. Consider a binary tree of Depth 4. The nodes other than the root node are labeled ‘0’, ‘1’, ‘00’, ‘01’, ‘10’, ‘11’, …, ‘0000’, ‘1111’, up to down and left to right. (See Figure U.1).

The label is sometimes called Node Index. A Node Index assigned to a leaf is especially called Leaf Number. Each node is assigned a key: k(0), k(00), k(01), …, k(0000), k(0001), …, k(1110), k(1111). Let the keys be called Node Keys. An MN is associated with a unique leaf. Thus, sixteen MNs are associated with the leaves of the tree: Call them ‘MN0’, ‘MN1’, …, ‘MN15’, left to right. Each MN is assigned a set of pairs of a Node Index and a Node Key, which is called Device Key: An MN is assigned the pairs along the path that is descending from the root to the leaf associated with the MN. For instance, MN3 is assigned the following Device Key: {(0, k(0)), (00, k(00)), (001, k(001)), (0011, k(0011))}.



Figure U.1— An Example Tree

A set of MNs is called group if and only if they share an MIHF Group ID and a group key. At first, make all the sixteen MNs constitute one group, say, GA. Then, make the GKB such that {{0, 1}, {<k(0)>[KG], <k(1)>[KG]}}, where KG is the group key for the group GA and <k>[D] denotes data D encrypted by a key k. {0, 1} is the complete subtree part of the GKB and {<k(0)>[KG], <k(1)>[KG]} is the group key data part. Check if all the MNs can share the group key. Any Device Key has one of the Node Keys: k(0) or k(1). Therefore, any MN can decrypt the preceding GKB to derive the group key KG. The group key is shared by all the MNs as expected.

Let MN1, MN4 and MN5 be removed from the group GA: Then the GKB will cover {MN0, MN2, MN3, MN6, …, MN16}. The GKB required for this is as follows:

GKB1 = {{1, 001, 011, 0000}, {<k(1)>[KGA], <k(001)>[KGA], <k(011)>[KGA], <k(0000)>[KGA]}},

where KGA is a new group key for the group GA. Check that any MN in GA can decrypt one of the elements of the group key data part and derive the group key. Also note that the complete subtree part of the GKB is ordered in the ascending dictionary order defined in 9.4.2.1. And, let GB be a group which is composed of MN3, MN4, MN8, MN9 and MN12, MN13, MN14 and MN15. The GKB to create group GB is the following GKB2:

GKB2 = {{11, 100, 0011, 0100}, {<k(11)>[KGB], <k(100)>[KGB], <k(0011)>[KGB], <k(0100)>[KGB]}},

where KGB is a group key for the group GB. Note that multiple groups with their own group keys may exist on one tree. An MN with one Device Key Set may belong to multiple groups at the same time.

It may be that the size of a GKB is too large for an MIH service specific TLV. The followings show one example of the ways how a GKB is divided into smaller pieces: Suppose here that the capacity of the TLV allows only two encrypted group keys. GKB2 can be divided into two GKBs: GKB2-1 and GKB2-2 such that GKB2-1 = {{11, 100}, {<k(11)>[KGB], <k(100)>[KGB]}} and GKB2-2 = {{0011, 0100}, {<k(0011)>[KGB], <k(0100)>[KGB]}}. Suppose that GKB2-2 arrives at MN15 after GKB2-1 does first. MN15 joins in the group GB when it receives GKB2-1. Then, MN15 leaves GB receiving GKB2-2 because it has no key to successfully decapsulate GKB2-2. This is clearly not the expected behavior for MN15. This problem can be avoided if the Subgroup Ranges of the GKBs are appropriately set. Let GKB2-1 = {R1, {11, 100}, {<k(11)>[KGB], <k(100)>[KGB]}} and GKB2-2 = {R2, {0011, 0100}, {<k(0011)>[KGB], <k(0100)>[KGB]}}, where R1 = [8, 15] and R2 = [0, 7]. The Leaf Number of MN15 is 15, which means that it is in the range of R1. Thus, it processes GKB2-1 and derives the group key. MN15, however, does not process GKB2-2 because it is out of the range of R2. Thus, MN15 joins in GB and stays there as expected.

There is a version of GKB without a group key data part, which are used when confidentiality is not necessary for group commands. Creation of such a GKB is the same. For instance, just remove the GroupKeyData field from a GKB having a group key data part.

1. The Institute of Electrical and Electronics Engineers, Inc.

3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 20XX by The Institute of Electrical and Electronics Engineers, Inc.

All rights reserved. Published <XX MONTH 20XX>. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics
Engineers, Incorporated.

PDF: ISBN 978-0-XXXX-XXXX-X STDXXXXX

Print: ISBN 978-0-XXXX-XXXX-X STDPDXXXXX

*IEEE prohibits discrimination, harassment and bullying. For more information, visit* [*http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html*](http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html)*.*

*No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.*  [↑](#footnote-ref-2)
2. IEEE Standards Dictionary Online subscription is available at:

<http://www.ieee.org/portal/innovate/products/standard/standards_dictionary.html>. [↑](#footnote-ref-3)