**IEEE P802.15**

**Wireless Personal Area Networks**

|  |  |
| --- | --- |
| Project | IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs) |
| Title | **<ULI Mandatory Elements Operation>** |
| Date Submitted | [12 July 2018] |
| Source | [Pat Kinney][<Kinney Consulting>] | Voice: [+1.847.960.3715]E-mail: [pat.kinney@kinneyconsultingllc.com] |
| Re: | TG12 Architecture: PDE, MMI, MPM, PTM, and KMP operation |
| Abstract | [Description of the mandatory IEEE 802.15.12 elements: PDE, MMI, MPM, PTM, and KMP] |
| Purpose | [Provide a basis for drafting the 802.15.12 standard] |
| Notice | This document has been prepared to assist the IEEE P802.15. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein. |
| Release | The contributor acknowledges and accepts that this contribution becomes the property of IEEE and may be made publicly available by P802.15. |

# Table of Contents

Table of Contents 2

1 Overview of ULI Mandatory Elements 6

2 Profile Overview 7

2.1 Introduction 7

2.2 Profile Concept 8

2.3 Profile Hierarchies 8

2.4 Profile Type 9

2.4.1 Profile Capabilities 9

2.4.1.1 Security 9

2.4.1.2 Ranging 9

2.4.1.3 TSCH 9

2.4.2 MAC 9

2.4.3 PHY 9

2.4.4 Security 9

2.4.4.1 Security enabled 9

2.4.4.2 SecPolicy 9

2.4.4.2.1 secKeyIdLookupList [] 9

2.4.4.2.1.1 secKeyIdMode 9

2.4.4.2.1.2 secKeySource 9

2.4.4.2.1.3 secKeyIndex 9

2.4.4.2.1.4 secKey 9

2.4.4.2.1.5 secFrameCounterPerKey 9

2.4.4.2.1.6 secKeyUsageList [secKeyUsageFrameType, secKeyUsageCommandId] 9

2.4.4.2.1.6.1 secKeyIeUsageDescriptorList [secKeyIeType, secKeyIeId] 9

2.4.4.2.2 secDeviceList [secExtAddress] 9

2.4.4.2.2.1 secPanId 9

2.4.4.2.2.2 secShortAddress 9

2.4.4.2.2.3 secExempt 9

2.4.4.2.3 secSecurityLevelList [secFrameType, secCommandId] 9

2.4.4.2.3.1 secSecurityMinimum 9

2.4.4.2.3.2 secDeviceOverrideSecurityMinimum 9

2.4.4.2.3.3 secAllowedSecurityLevels 9

2.4.4.2.3.4 secIeSecurityLevelDescriptorList [secIeType, secIeId] 10

2.4.4.2.3.4.1 secIeSecurityMinimum 10

2.4.4.2.3.4.2 secIeDeviceOverrideSecurityMinimum 10

2.4.4.2.3.4.3 secIeAllowedSecurityLevels 10

2.4.4.3 SecTBD 10

2.4.4.3.1 10

2.4.5 Ranging 10

2.4.5.1 macRangingSupported 10

2.4.6 TSCH 10

2.4.6.1 enabled 10

2.4.6.2 macMinBe 10

2.4.6.3 macMaxBe 10

2.4.6.4 macDisconnectTime 10

2.4.6.5 macJoinMetric 10

2.4.6.6 macSlotframeTable [handle] 10

2.4.6.6.1 macSlotframeSize 10

2.4.6.6.2 macLinkTable [handle] 10

2.4.6.6.2.1 macTxType 10

2.4.7 Hopping 10

2.4.7.1 enabled 10

2.4.8 DSME 10

2.4.8.1 enabled 10

2.4.9 LE 10

2.4.9.1 enabled 10

2.4.10 Metrics 10

2.4.10.1 enabled 10

2.4.11 EBR 10

2.4.12 RCCN 11

2.4.12.1 enabled 11

2.4.13 PDE 11

2.4.14 MMI 11

2.4.15 MPM 11

2.4.16 PTM 11

2.4.17 KMP 11

3 PDE 11

3.1 Purpose: 11

3.2 Overview 11

3.3 Protocol Discrimination 12

3.3.1 EtherType 12

3.3.2 Dispatch code 13

3.3.3 Multiplex ID field 13

3.4 PDE Primitives 13

3.4.1 PDE-DATA 14

3.4.1.1 PDE-DATA Primitive Overview 14

3.4.1.2 PDE-DATA.request 14

3.4.1.3 PDE-DATA.confirm 15

3.4.1.4 PDE-DATA.indication 17

3.4.2 PDE-PROFILE 17

3.4.2.1 Overview 17

3.4.2.2 PDE-PROFILE-CREATE.request 17

3.4.2.3 PDE-PROFILE-CREATE.confirm 18

3.4.2.4 PDE-PROFILE-GET.request 18

3.4.2.5 PDE-PROFILE-GET.confirm 18

3.4.2.6 PDE-PROFILE-COMBINE.request 18

3.4.2.7 PDE-PROFILE-COMBINE.confirm 18

3.4.2.8 PDE-PROFILE-RCVEXEC.request 19

3.4.2.9 PDE-PROFILE-DELETE.request 20

3.4.3 PDE-MGMT 20

3.4.3.1 Overview 20

3.4.3.2 PDE-MGMT-SET.request 21

3.4.3.3 PDE-MGMT-SET.confirm 21

3.4.3.4 PDE-MGMT-GET.request 22

3.4.3.5 PDE-MGMT-GET.confirm 22

3.4.3.6 PDE-MGMT-ACTION.request 22

3.4.3.7 PDE-MGMT-ACTION.confirm 22

3.4.3.8 PDE-MGMT-NOTIFICATION.indication 23

3.4.3.9 PDE-MGMT-NOTIFICATION.response 23

3.4.4 PDE-PURGE 23

3.4.4.1 PDE-PURGE Overview 23

3.4.4.2 PDE-PURGE.request 24

3.4.4.3 PDE-PURGE.confirm 24

3.4.5 PDE-MODULE 24

3.4.5.1 PDE-MODULE Overview 24

3.4.5.2 PDE-MODULE -LIST.request 25

3.4.5.3 PDE-MODULE-LIST.confirm 25

3.4.5.4 PDE-MODULE-GETSTATUS.request 26

3.4.5.5 PDE-MODULE-GETSTATUS.confirm 26

3.4.5.6 PDE-MODULE-SETSTATE.request 27

3.4.5.7 PDE-MODULE-SETSTATE.confirm 28

3.4.5.8 PDE-MODULE.request 28

3.4.5.9 PDE-MODULE.confirm 28

3.4.5.10 PDE-MODULE.indication 29

3.4.5.11 PDE-MODULE.response 29

4 Multiplexed MAC Interface (MMI) 30

4.1 Purpose 30

4.2 Overview 30

4.3 MMI-DATA primitives 31

4.3.1 MMI-DATA.request 32

4.3.2 MMI-DATA.confirm 34

4.3.3 MMI-DATA.indication 36

4.4 MMI-PURGE primitives 37

4.4.1 MMI-PURGE.request 37

4.4.2 MMI-PURGE.confirm 38

4.5 MMI-MGMT primitives 39

4.5.1 Overview 39

4.5.2 MMI-MGMT.request 39

4.5.3 MMI-MGMT.confirm 39

4.5.4 MMI-MGMT.indication 39

4.5.5 MMI-MGMT.response 39

4.6 MMI-CONFIG primitives 40

4.7 MMI- SERVICE-DISCOVERY primitives 40

4.7.1 MMI-SERVICE-DISCOVERY Primitive Overview 40

4.7.2 MMI-SERVICE-DISCOVERY.request 40

4.7.3 MMI-SERVICE-DISCOVERY.confirm 40

5 Management Protocol Module 41

5.1 Purpose 41

6 PTM Description 42

6.1 Purpose 42

6.2 Overview 42

6.3 Design of (*and questions about*) the PTM 42

# Overview of ULI Mandatory Elements

As shown in Figure 2‑1, the mandatory elements of the ULI are the Protocol Discrimination Entity (PDE), Multiplexed MAC Interface (MMI), Management Protocols Module (MPM), Passthru Module (PTM), and the Key Management Protocol (KMP).



Figure 2‑1

# Profile Overview

## Introduction

A significant requirement for the ULI is the reduction of the complexity in configuring and using the IEEE 802.15.4 device.

The complexity in configuring IEEE 802.15.4devicesresults from having to select the correct configuration for the specific use case given all of the possible combinations of the following: 8 MAC modes with 13 distinct MAC behaviors, 9 PHY modulation types with 4 distinct PHY behaviors and 40 PHY data rates, and 20 PHY bands with greater than 35,390 channels.

Note: Some combinations of the above configurations will result in a non-operable IEEE 802.15.4 device.

The complexity in the use of IEEE 802.15.4 devices to send messages can be shown with a simple comparison to IEEE 802.3 and IEEE 802.11 devices. An IEEE 802.3 device (Ethernet) has 4 parameters in its data transmission primitive, i.e. MA\_DATA.request. The IEEE 802.11 device (e.g. WiFi) data transmission primitive, i.e. MA-UNITDATA.request, has 6 parameters. However, the IEEE 802.15.4 device data transmission primitive, i.e. MCPS-DATA.request, contains 28 parameters. Additionally, the IEEE 802.3 and the IEEE 802.11 data transmission primitive parameters are generic to all data transmissions while the IEEE 802.15.4 data transmission primitive includes parameters dealing with addressing modes, information element types and number, ranging, low energy, suppression of frame fields, security keys, etc. Eliding the parameters that are substantially different from those used by IEEE 802.3 and IEEE 802.11 devices would substantially reduce the complexity to use IEEE 802.15.4 devices.

## Profile Concept

The concept of the ULI profile is to concatenate one or more configuration parameters for the ULI’s PDE, MMI, and protocol module(s) along with the IEEE 802.15.4 MAC and PHY into an identifiable array via the use of the ProfileId. Table 3‑1 shows a ProfileId example.

Table 3‑1 ProfileId example

|  |
| --- |
| ProfileId = 0x0A |
| Frequency Band | 2450 MHz |
| PHY Modulation | O-QPSK |
| phyCurrentChannel | 15 |
| phyCCA Mode | 3 |
| phyTxPower | 10 |
| phyCurrentPage | 0 |

A ProfileId defining all necessary parameters for the MCPS-DATA.request primitive would be included as a parameter in the ULI data transmission primitive.

## Profile Hierarchies

To allow reuse of ProfileIds, the types of configuration parameters will be restricted to specific ProfileId types: Profile Capabilities, IEEE 802.15.4 MAC, IEEE 802.15.4 PHY, IEEE 802.15.4 SecPolicy, IEEE 802.15.4 SecTBD, IEEE 802.15.4 ranging, IEEE 802.15.4 TSCH, IEEE 802.15.4 hopping, IEEE 802.15.4 DSME, IEEE 802.15.4 LE, IEEE 802.15.4 metrics, IEEE 802.15.4 EBR, IEEE 802.15.4 RCCN, ULI MPM, ULI PTM, ULI PDE, ULI KMP, and ULI MMI. The ProfileId for the PDE-DATA.request or the PDE-DATA.indication would be some combination of all of the ProfileId types.

## Profile Type

### Profile Capabilities

#### Security

#### Ranging

##### macRangingSupported

#### TSCH

#### Metrics

#### Hopping

#### LE

#### SRM

##### macSrmCapable

#### TRLE

#### DSME

#### Extended DSME

### MAC

#### Addressing mode

##### macExtendedAddress

##### macShortAddress

##### macCoordExtendedAddress

##### macCoordShortAddress

#### PAN ID

##### macPanId

#### Commands

#### MAC Frame Types

#### Promiscuous

##### macPromiscuousMode

#### Beacon management

#### Superframe structure

##### Control, define, determine, and declare superframe structure

##### Follow and use superframe structure

##### Beacon-only Period (BOP)

#### Access Methods

#### Acknowledgments

#### Information Elements (IEs)

#### Priority

#### Metrics

##### Enabled

##### macCounterOctets

##### macRetryCount

##### macMultipleRetryCount

##### macTxFailCount

##### macTxSuccessCount

##### macFcsErrorCount

##### macSecurityFailure

##### macDuplicateFrameCount

##### macRxSuccessCount

##### macNackCount

#### Spectrum Resource Measurement (SRM)

##### Enabled

###### macSrmEnabled

##### SRM Request

##### SRM Response

##### SRM Report

##### SRM Information

###### macEd

###### macTxFailTime

###### macTxDeferredTime

###### macRetryHistogram

###### macChannelUtilization

###### macRcpi

###### macRsni

###### macRssi

###### macNoiseHistogram

###### macFrameErrorCount

###### macDeferredTxCount

###### macAverageBufferUtilization

###### macMaximumBufferUtilization

###### macAverageAccessDelay

###### macChannelPage

###### macChannelNumber

###### macRxAddrMode

###### macRxDeviceAddress

###### macSRMmeasurementMode

### PHY

#### Common

##### Transmit

###### phyTxPower

###### phyNomTxPower

###### phyCapableTxPower

###### phyBroadcastTxPower

###### phyUnicastTxPower

#### SUN

##### FSK

###### phyFskPreambleLength

###### phySunFskSfd

###### FEC

phyFskFecEnabled

phyFskFecScheme

###### Interleaving

phyFskFec InterleavingRsc

###### Data Whitening

phyFskScramblePsdu

###### Mode Switching

phyModeSwitch ParameterEntries

##### OFDM

###### Frequency spreading

###### phyOfdmInterleaving

##### O-QPSK

##### FSK Generic PHY

##### Operating mode

Options

#### TVWS

##### FSK

###### phyTvwsFskSpreadingEnabled

###### phyFskFecEnabled

###### phyTvwsFskFecScheme

###### phyFskFec InterleavingRsc

###### phyTvwsFskWhiteningEnabled

###### phyFskPreambleLength

###### phyTvwsSfdLength

##### OFDM

###### phyTvwsChannelAggregation

###### phyOfdmInterleaving

#### O-QPSK

#### BPSK

#### FSK

#### MSK

#### OFDM

#### CSS

#### UWB

##### phyHrpUwbDataRatesSupported†

##### phyHrpUwbCurrentPulseShape

##### phyHrpUwbLcpWeight1

##### phyHrpUwbLcpWeight2

##### phyHrpUwbLcpWeight3

##### phyHrpUwbLcpWeight4

##### phyHrpUwbLcpDelay2

##### phyHrpUwbLcpDelay3

##### phyHrpUwbLcpDelay4

##### phyCurrentCode

##### phyHrpUwbScanBinsPerChannel

##### phyHrpUwbInsertedPreambleInterval

#### ASK

#### Interleaver

#### Data Whitener

#### FEC

#### PSDU FRAK

##### phyPsduFragmentationEnabled

##### phyLmrCodingRate

##### phyFragmentSize

##### phyPsduFragSecure

##### phyFragmentFrameCounter

##### phyFrakProgressTimeout

##### phyPsduFragPadValue

#### Frame Check Sequence

##### macFcsType

### Security

#### Security enabled

#### SecPolicy

##### secKeyIdLookupList []

###### secKeyIdMode

###### secKeySource

###### secKeyIndex

###### secKey

###### secFrameCounterPerKey

###### secKeyUsageList [secKeyUsageFrameType, secKeyUsageCommandId]

secKeyIeUsageDescriptorList [secKeyIeType, secKeyIeId]

##### secDeviceList [secExtAddress]

###### secPanId

###### secShortAddress

###### secExempt

##### secSecurityLevelList [secFrameType, secCommandId]

###### secSecurityMinimum

###### secDeviceOverrideSecurityMinimum

###### secAllowedSecurityLevels

###### secIeSecurityLevelDescriptorList [secIeType, secIeId]

secIeSecurityMinimum

secIeDeviceOverrideSecurityMinimum

secIeAllowedSecurityLevels

#### SecTBD

#####

### Ranging

#### Enabled

#### Dynamic Preamble Selection (DPS)

### TSCH

#### enabled

#### macMinBe

#### macMaxBe

#### macDisconnectTime

#### macJoinMetric

#### macSlotframeTable [handle]

##### macSlotframeSize

##### macLinkTable [handle]

###### macTxType

### Hopping

#### Enabled

#### macHoppingSequenceId

#### macChannelPage

#### macNumberOfChannels

#### macPhyConfiguration

#### macExtendedBitmap

#### macHoppingSequenceLength

#### macHoppingSequenceList

#### macHopDwellTime

### DSME

#### DSME Enabled

#### Extended DSME Enabled

#### TMTCP

#### Multi-PHY Management (MPM)

#### DSME MAC management service

#### DSME multisuperframe structure

#### Association Request

#### Association Response

#### GTS Request

#### GTS Response

#### Information Request

#### Information Response

#### Beacon Allocation Notification

#### Collision Notification

#### Link Report

### LE

#### LE enabled

#### macCslPeriod

#### macCslMaxPeriod

#### macCslChannelMask

#### macCslFramePendingWait

#### macRitPeriod

#### macRitDataWaitDuration

#### macRitTxWaitDuration

#### macRitRequestPayload

#### macLowEnergySuperframeSupported

#### macLowEnergySuperframeSyncInterval

#### macCslInterval

#### macIRitOffsetInterval

#### macIRitListenDuration

#### macIRitEnabled

### RCCN

#### Enabled

#### macRccnNumTimeSlots

#### macRccnNumMgmtTs

#### macRccnNumGtsTs

#### macRccnNetId

#### macRccnDevType

#### macRccnCap

### LP-WAN

#### phyLecimCurrentBand

#### phyCurrentLecimPhyType

##### DSSS

###### phyLecimDsssPpduModulationRate

###### phyLecimDsssPpduTxAt

###### phyLecimDsssPsduSize

###### phyLecimDsssPreamble

###### phyLecimDsssSfdPresent

###### phyLecimDsssPsduSpreadingFactor

###### phyLecimFecTailBitingEnabled

###### phyLecimDsssPsduOvsf SpreadingFactor

###### phyLecimDsssPsduOvsf CodeIndex

##### FSK

###### phyLecimFskPreambleLength

###### phyLecimFskPsduPositionMod

###### phyLecimFskSpreading

###### phyLecimFskSpreading Factor

###### phyLecimFskSpreading Pattern

###### phyLecimFecEnabled

###### phyLecimFskInterleavingEnabled

###### phyLecimFskSymbolRate

### TRLE

#### Enabled

#### Relaying mode

#### Link extension for non-TRLE PAN

#### Link extension for TRLE-enabled PAN

#### TRLE MAC management service

### RFID

### Peers

#### phyPeersTxPower

#### phyMinSnr

#### phyMinLinkMargin

#### deviceAddrMode

#### deviceAddress

#### TxPower

### PDE

### MMI

### MPM

### PTM

### KMP

# PDE

## Purpose:

For information received via the 802.15.4 device; the PDE directs that information from the protocol module SAP to the appropriate higher layer SAP or to another protocol module SAP.

For information from a higher layer SAP, the PDE will direct that information to the designated protocol module.

## Overview

The PDE is responsible for determining if the higher layer entity’s SAP is a legacy SAP (i.e. intended to directly interface to the 802.15.4 MAC SAPs) or a ULI capable SAP. This allows the ULI to work with higher layer entities that were designed to interface directly to 802.15.4 MAC SAPs as well as higher layer entities designed to interface to the ULI.

For data from higher layer entity to be transmitted to a remote device via the 802.15.4 local device:

* If the higher layer entity’s SAP is a legacy SAP, the PDE will attach origination EtherType/Dispatch information to the data and then send the data to the protocol module dictated by a prior configuration by the MPM or the PTM.
* If the higher layer entity’s SAP is a ULI capable SAP, the PDE will attach origination EtherType/Dispatch information and Profile information to the data and then send that data to the protocol module designated by the EtherType/Dispatch code.
* For data received from a remote device via the 802.15.4 local device:
* From PTM: send frame payload to the application designated by the EtherType/Dispatch code or the default application as configured by the MPM
* From non-PTM: send frame payload to designated higher layer SAP as dictated by EtherType/Dispatch code attached by the protocol module

## Protocol Discrimination

Protocol discrimination is based on either EtherTypes or dispatch codes.

### EtherType

EtherType protocol identification values, examples are shown in Table 4‑1, are assigned by the IEEE RA and are used to identify the protocol that is to be invoked to process the user data in the frame. An EtherType is a sequence of 2 octets, interpreted as a 16-bit numeric value with the first octet containing the most significant 8 bits and the second octet containing the least significant 8 bits.

Table 4‑1 EtherType Examples

| **EtherType** | **Organization/Address** | **Protocol** |
| --- | --- | --- |
| 0800 | Xerox | IPv4 Internet Protocol Version A Standard for the Transmission of IP Datagrams over Ethernet Networks, RFC-Internet Society, Apr. 1984. http://www.ietf.org/rfc/rfc894.txt |
| 86DD | USC/ISI 4676 Admiralty Way, Marina del Rey, CA | IPv6 Internet Protocol Version 6Transmission of Packets over Ethernet Networks, RFC-2464, Internet Society, Dec. 1998. http://www.ietf.org/rfc/rfc2464.txt |
| 888E  | IEEE 802.1 802.1 Chairc/o IEEE Piscataway, NJ  | IEEE Std 802.1X - Port-based network access control |
| 88B7 | IEEE 802.1 IEEE 802.1 Chair c/o IEEE Piscataway, NJ | 802 - OUI Extended Ethertype. This Ethertype value is available for public use and for prototype and vendor-specific protocol development, as defined in Amendment 802a to IEEE Std 802.  |
| 88F0 | IEEE P1451.0700 King Farm Blvd., Rockville, MD | IEEE P1451.0 Smart Transducer Interface for Sensors and Actuators http://grouper.ieee.org/groups/1451/0/private/ |
| A0ED | IETF 6lo working group c/o Internet Society, Reston, VA | When carried over layer 2 technologies such as Ethernet, this EtherType will be used to identify IPv6 datagrams using LoWPAN encapsulation as defined in IETF RFC 4944 Transmission of IPv6 Packets over IEEE 802.15.4 Networks |

### Dispatch code

Values in the 0–1535 range are not available for use as EtherTypes, rather they designate dispatch codes. Dispatch codes are assigned by the IEEE 802.15 ANA.

### Multiplex ID field

The Multiplex ID field used in the MPX IE is used to multiplex different upper-layer protocols. The Multiplex ID field, as shown in Table 4‑2, takes one of two meanings, depending on its numeric value as follows:

1. If the value of this field is less than or equal to 1500, the Multiplex ID field takes Dispatch code values.
2. If the value of this field is greater than 1500, the Multiplex ID field indicates the EtherType of the MAC client protocol.

Table 4‑2

|  |  |  |
| --- | --- | --- |
| **Multiplex ID (decimal)** | **Multiplex ID (hex)** | **Description** |
| 1 | 0x0001 | KMP |
| 2 | 0x0002 | WiSUN |
| 3 - 30 | 0x0003 – 0x001e | Reserved |
| 31 | 0x001f | Vendor specific, OUI extended |
| 32 - 1535 | 0x0020 - 0x05ff | Reserved  |

The Multiplex ID field is present if the Frame number is 0x00 and the Transfer type is 0b010, or if the Transfer type is 0b000. If the Transfer type is 0b001, the Multiplex ID is stored inside the Transaction ID field and the Multiplex ID field is omitted.

## PDE Primitives

The PDE service consists of 14 primitives as shown in Table 4‑3.

Table 4‑3 Summary of PDE primitives

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Request | Indication | Response | Confirm |
| PDE-DATA | X | X |  | X |
| PDE-PROFILE-CREATE | X |  |  | X |
| PDE-PROFILE-COMBINE | X |  |  | X |
| PDE-PROFILE-DELETE | X |  |  | X |
| PDE-PROFILE-RCVEXEC | X |  |  | X |
| PDE-MGMT-GET | X |  |  | X |
| PDE-MGMT-SET | X |  |  | X |
| PDE-MGMT-ACTION | X | X | X | X |
| PDE-MGMT-NOTIFICATION |  | X | X |  |
| PDE-PURGE | X |  |  | X |
| PDE-MODULE-LIST | X |  |  | X |
| PDE-MODULE-GET | X |  |  | X |
| PDE-MODULE-SET | X |  |  | X |
| PDE-MODULE-OPERATION | X | X | X | X |

### PDE-DATA

#### PDE-DATA Primitive Overview

The PDE-DATA primitive supports the transport of data from the higher layer or to the higher layer. shows a message sequence diagram illustrating a use of the PDE-DATA.request.

Figure 4‑1

#### PDE-DATA.request

The PDE-DATA.request primitive is a request from a higher layer SAP to transport a data payload to a remote device. Specifically, the PDE-DATA.request primitive requests the transfer of a PDE payload (PdeData) to the designated protocol module (DstProtocolId). The semantics of this primitive are as follows:

(
DstAddr,

DstProtocolId,

ProfileId,

PdeData,

PdeHandle,

)

The primitive parameters are described in Table 4.4 PDE-DATA.request parameters

Table 4.4 PDE-DATA.request parameters

| **Name** | **Type** | **Valid range** | **Description** |
| --- | --- | --- | --- |
| DstAddr | — | Any valid extended address | The extended address of the receiving (destination) device.[[1]](#footnote-1) |
| DstProtocolId | Integer | 0x0000–0xffff | The destination protocol module’s ID of the remote device, i.e. either the protocol module’s EtherType or Dispatch code |
| ProfileId | Integer | 0x0000–0xffff | The ULI Profile ID identifying the desired configuration parameters for ULI protocol modules and MAC/PHY  |
| PdeData | Set of octets | — | The set of octets forming the PDE data payload. |
| PdeHandle | Integer | 0x00–0xff | An identifier that can be used to refer to the particular primitive transaction, used to match a confirm primitive with the corresponding request. |

#### PDE-DATA.confirm

The PDE-DATA.confirm primitive reports the results of a request to transport data from a higher layer SAP to a remote device. The semantics of this primitive are as follows:

(
PdeHandle,
MaxTransferSize,
Status
)

The primitive parameters are described in Table 4‑5. If there is no capacity to store the transaction, the Status will be set to TRANSACTION\_OVERFLOW. In case the remote device aborts the transaction then the status will be set to TRANSACTION\_ABORTED and the MaxTransferSize is set to the value returned from the remote device.

Table 4‑4 PDE-DATA.confirm parameters

| **Name** | **Type** | **Valid range** | **Description** |
| --- | --- | --- | --- |
| PdeHandle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction; used to match a confirm primitive with the corresponding request. |
| MaxTransfer- Size | Integer | 0x0000–0xffff | In case of an aborted transaction this parameter may be returned from the remote device to indicate the maximum size of transaction it can handle. In case the remote device did not give a maximum size, this is set to zero. |
| Status | Enumeration | SUCCESS,TRANSACTION\_OVERFLOW, TRANSACTION\_EXPIRED,CHANNEL\_ACCESS\_FAILURE, INVALID\_ADDRESS, NO\_ACK, COUNTER\_ERROR, FRAME\_TOO\_LONG, UNAVAILABLE\_KEY,UNSUPPORTED\_SECURITY, INVALID\_PARAMETER. TRANSACTION\_ABORTED | The status of the last PDE data transmission. |

#### PDE-DATA.indication

The PDE-DATA.indication primitive is invoked from a protocol module to deliver a payload to a higher layer SAP. The semantics of this primitive are as follows:

(
SrcAddr,
DstAddr,
DstProtocolId,
PdeData,

)

The primitive parameters are described in Table 4‑5.

Table 4‑5-PDE\_DATA.indication parameters

| **Name** | **Type** | **Valid range** | **Description** |
| --- | --- | --- | --- |
| SrcAddr | — | Any valid extended address | The extended address of the transmitting (source) device. |
| DstAddr | — | Any valid extended address | The extended address of the receiving (destination) device. |
| DstProtocolId | Integer | 0x0000–0xffff | The destination protocol module’s ID, i.e. either the protocol module’s EtherType or Dispatch code |
| PdeData | Set of octets | — | The set of octets forming the PDE data payload. |

### PDE-PROFILE

#### Overview

The PDE-PROFILE primitives support the transport of configuration information and profiles for the protocol modules or 802.15.4 MAC/PHY to the MPM.

#### PDE-PROFILE-CREATE.request

This request transports from a higher layer to the MPM, the configuration information (PdeMgmtData) for protocol modules or an 802.15.4 MAC/PHY. The MPM creates a profile and stores this information in its MIB. The identification of the Profile created, i.e. ProfileId, will be returned in the PDE-MGMT-CREATE.confirm primitive. The semantics of this primitive are as follows:

(

PdeMgmtData,

Handle

)

#### PDE-PROFILE-CREATE.confirm

This primitive reports the results of the PDE-MGMT-CREATE.request to the higher layer SAP. The semantics of this primitive are as follows:

(

ProfileId,

Handle,

Status

)

#### PDE-PROFILE-GET.request

This primitive requests the configuration information stored in ProfileId. The semantics of this primitive are as follows:

 (

ProfileId,

Handle

)

#### PDE-PROFILE-GET.confirm

This primitive reports the results of the PDE-PROFILE-GET.request to the higher layer SAP. The semantics of this primitive are as follows:

 (

PdeMgmtData,

Handle,

Status

)

#### PDE-PROFILE-COMBINE.request

This primitive creates a new ProfileId that is a combination of two or more existing ProfileIds. Changes to any of the ProfileIds used to create the new ProfileId will also change the new ProfileId. The semantics of this primitive are as follows:

 (

ProfileIdList,

Handle

)

#### PDE-PROFILE-COMBINE.confirm

This primitive reports the results of the PDE-MGMT-COMBINE.request to the higher layer SAP. The semantics of this primitive are as follows:

 (

ProfileId,

Handle,

Status

)

#### PDE-PROFILE-RCVEXEC.request

This primitive causes the configuration information stored in ProfileId to be loaded into the 802.15.4 MAC & PHY and other modules such as the PTM. After the configuration information has been loaded into the 802.15.4 MAC & PHY and other modules, the primitive enables the operation using the configuration information. Figure 3-2 shows a message sequence diagram illustrating a use of the PDE-PROFILE-RCVEXEC.request. The semantics of this primitive are as follows:

 (

ProfileId,

Handle

)

PDE-PROFILE-RCVEXEC.confirm

MPM-PROFILE-RCVEXEC.request

Management

Protocols sublayer

PDE

sublayer

Next

higher layer

PDE-PROFILE-RCVEXEC.request

MMI-MGMT.request

MAC PIB

Device

MAC

MLME-SRM.request

MMI

sublayer

MPM-PROFILE-RCVEXEC.response

PDE-PROFILE-

RCVEXEC.response

MMI-MGMT.response

MLME-SRM.response

IEEE802.15.12

MLME-SET.request

MLME-SET.response

IEEE802.15.4

**Figure 4‑2.2 PDE-PROFILE-RCVEXEC.request**

PDE-PROFILE-RCVEXEC.confirm

This primitive reports the results of the PDE-MGMT-RCVEXEC.request to the higher layer SAP. The semantics of this primitive are as follows:

 (

Handle,

Status

)

#### PDE-PROFILE-DELETE.request

This primitive deletes the configuration information stored in ProfileId. ProfileIds that are in use (either being used in a module or 802.15.4 MAC&PHY, or in use for other ProfileId) shall not be deleted. The semantics of this primitive are as follows:

(

ProfileId,

Handle

)

PDE-PROFILE-DELETE.confirm

This primitive reports the results of the PDE-PROFILE-DELETE.request to the higher layer SAP. The semantics of this primitive are as follows:

(

Handle,

Status

)

### PDE-MGMT

#### Overview

The PDE-MGMT primitives allow a higher layer to manipulate configuration data stored in the MPM or to be notified of configuration data in the MPM.

The parameters for the PDE-MGMT primitives are described in Table 4‑6.

Table 4‑6 PDE-MGMT primitive parameters

| **Name** | **Type** | **Valid range** | **Description** |
| --- | --- | --- | --- |
| ProfileId | Integer | 0x0000–0xffff | The Profile ID for the configuration parameters stored in the MPM |
| ProfileIdList | List of integers | 0x0000-0xffff | List of ProfileIds to be combined |
| ObjectId | Integer | 0x0000–0xffff | A specific parameter of the configuration parameters stored in the MPM |
| PdeMgmtData | Set of octets | --- | Configuration information for the 802.15.4 MAC and PHY and/or module(s) to be configured |
| PdeSapId | Integer | 0x0000–0xffff | The SAP ID for the configuration parameters being sent from the MPM |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to the particular primitive transaction; used to match a confirm primitive with the corresponding request. |
| Status | Enumeration | SUCCESS,TRANSACTION\_OVERFLOW, INVALID\_PARAMETER, TRANSACTION\_ABORTED, RESOURCES\_UNAVAILABLE | The status of the last primitive. |

#### PDE-MGMT-SET.request

This primitive sets a profile parameter, noted by its Profile Id and its ObjectId to a defined value. The semantics of this primitive are as follows:

(

ProfileId,

ObjectId,

Value,

Handle

)

#### PDE-MGMT-SET.confirm

This primitive reports the results of the PDE-MGMT-SET.request to the higher layer SAP. The semantics of this primitive are as follows:

 (

Value,

Handle

)

#### PDE-MGMT-GET.request

This primitive requests the value of a parameter, noted by ObjectId, stored in ProfileId. The semantics of this primitive are as follows:

(

ProfileId,

ObjectId,

Handle

)

#### PDE-MGMT-GET.confirm

This primitive reports the results of the PDE-MGMT-GET.request to the higher layer SAP. The semantics of this primitive are as follows:

(

Value,

Handle

)

#### PDE-MGMT-ACTION.request

Operation want the model to do, e.g. clear keys, rekey, reset module

 (

ProfileId,

ObjectId,

Handle

)

#### PDE-MGMT-ACTION.confirm

Operation want the model to do, e.g. clear keys, rekey, reset module

 (

ProfileId,

ObjectId,

Handle,

status

)

#### PDE-MGMT-NOTIFICATION.indication

This primitive sends configuration parameters received by the MPM to a higher layer. The semantics of this primitive are as follows:

(

PdeSapId,

ProfileId,

PdeMgmtData,

Handle

)

#### PDE-MGMT-NOTIFICATION.response

This primitive sends the response to the MPM’s PDE-MGMT-NOTIFICATION.indication. The semantics of this primitive are as follows:

(

ProfileId,

PdeMgmtData,

Handle

Status

)

### PDE-PURGE

#### PDE-PURGE Overview

The PDE-PURGE primitives provide a means to remove a request from the PDE transaction queue or abort data transfers.

On receipt of the PDE-PURGE.request primitive, the PDE data service attempts to find in the transaction queue the payload indicated by the Handle parameter. If a PDE payload has left the transaction queue, the handle will not be found, and the PDE payload can no longer be purged. If a PDE payload matching the given handle is found, the payload is discarded from the transaction queue, and optionally an abort message is sent to the other end, if the SendAbort parameter is TRUE. Sending an abort message to the other end will allow the other end to clear out its state immediately without waiting for the timeout. The PDE-PURGE.request will also cause a corresponding MCPS-PURGE.request to be issued to the MAC data service, provided it has an MCPS-DATA.request in process when the PDE-PURGE.request is called.The primitive parameters are described in Table 4‑7.

Table 4‑7 PDE-PURGE primitive parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Valid range | Description |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction; used to match a PDE-PURGE.request primitive with the corresponding PDE-DATA.confirm primitive. |
| SendAbort | Boolean | TRUE, FALSE | If this parameter is TRUE and the transaction is still active, the PDE data service requests that an abort code be sent to the remote device indicating that the transaction was aborted. If this parameter is FALSE, the transaction is just purged locally, and no information is sent to the remote device. |
| Status | Enumeration | SUCCESS, INVALID\_HANDLE | The status of the request to purge PDE data from the transaction queue. |

#### PDE-PURGE.request

The PDE-PURGE.request primitive allows the next higher layer to purge a PDE payload from the transaction queue.

The semantics of the PDE-PURGE.request are as follows:

(
Handle,
SendAbort
)

#### PDE-PURGE.confirm

The PDE-PURGE.confirm primitive allows the PDE service to notify the next higher layer of the success of its request to purge a PDE payload from the transaction queue of the PDE. The semantics of this primitive are as follows:

(
Handle,
Status
)

### PDE-MODULE

#### PDE-MODULE Overview

The PDE-MODULE primitives specify actions such as a command(s) to be done to a specified protocol module(s). For example, a command to the KMP module to generate keys would use a PDE-MODULE primitive

#### PDE-MODULE -LIST.request

This primitive requests a list of the ModuleIds present in the ULI. The primitive is directed to the MPM that keeps the list of protocol modules present in the ULI. The semantics of this primitive are as follows:

(
Handle
)

The primitive parameters are described in Table 4‑8.

Table 4‑8 PDE-MODULE-LIST.request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Valid range** | **Description** |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction; used to match the PDE-MODULE-LIST.request primitive with corresponding PDE-MODULE-LIST.confirm primitive. |

#### PDE-MODULE-LIST.confirm

This primitive returns a list of the modules present in IEEE 802.15.12. The semantics of this primitive are as follows:

(
ModuleIdList,

Handle,
Status
)

The primitive parameters are described in Table 4‑9.

Table 4‑9 PDE-MODULE-LIST.confirm parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Valid range** | **Description** |
| ModuleIdList | Set of modules’ IDs | — | The list of modules’ IDs present in the ULI. |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction; used to match a confirm with the corresponding request. |
| Status | Enumeration | SUCCESS, DENIED, UNAVAILABLE | The status of the request for the list of modules’ IDs. |

#### PDE-MODULE-GETSTATUS.request

This primitive requests the status of the protocol module indicated by ModuleId. The semantics of this primitive are as follows:

(
ModuleId,

StatusType,

Handle
)

The primitive parameters are described in Table 4‑10.

Table 4‑10 PDE-MODULE-GETSTATUS.request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Valid range | Description |
| ModuleId | Integer | 0x0000-0xffff | The ID of the targeted module. |
| StatusType |  |  | Type of information specific to the module |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction; used to match the PDE-MODULE-GETSTATUS.request primitive with corresponding PDE-MODULE-GETSTATUS.confirm primitive. |

#### PDE-MODULE-GETSTATUS.confirm

This primitive returns with the status of the protocol module indicated by ModuleId. The semantics of this primitive are as follows:

(

Handle,
StatusValue,

Status
)

The primitive parameters are described in Table 4‑11.

Table 4‑11 PDE-MODULE-GETSTATUS.confirm parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Valid range | Description |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction; used to match a confirm primitive with the corresponding request. |
| StatusValue | Enumeration | ENABLED, DISABLED,UNKNOWN | The status of the target module. |
| Status | Enumeration | SUCCESS, DENIED, UNAVAILABLE | The status of the request to the status of the module. |

#### PDE-MODULE-SETSTATE.request

This primitive sets the state of the protocol module indicated by ModuleId. The semantics of this primitive are as follows:

(
ModuleId,
StateValue,

Handle,
)

The primitive parameters are described in Table 4‑12

Table 4‑12 PDE-MODULE-SETSTATE.request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Valid range | Description |
| ModuleId | Integer | 0x0000-0xffff | The ID of the targeted module. |
| StateValue | Enumeration | ENABLE, RESET DISABLE, | The state value to be set to the targeted module. |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction; used to match the PDE-MODULE-SETSTATE.request primitive with corresponding PDE-MODULE-SETSTATUS.confirm primitive. |

#### PDE-MODULE-SETSTATE.confirm

This primitive reports the results of the request to set the state of the protocol module indicated by ModuleId. The semantics of this primitive are as follows:

(

Handle,

Status
)

The primitive parameters are described in Table 4‑13

Table 4‑13 PDE-MODULE-SETSTATUS.confirm parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Valid range | Description |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction; used to match a confirm primitive with the corresponding request. |
| Status | Enumeration | SUCCESS, DENIED, TIMEOUT | The result of the request to set the status of the target module. |

#### PDE-MODULE.request

Sends a command to a module

The PDE-MODULE.request primitive is a request from a higher layer SAP to perform an action. Specifically, the PDE-MODULE.request primitive requests the transfer of a command to the designated protocol module identified by ModuleId. The semantics of this primitive are as follows:

(

ModuleId,

Command,

Handle,

)

#### PDE-MODULE.confirm

The PDE-MODULE.confirm primitive reports the results of the PDE.MODULE.request to transport a command from a higher layer SAP to a module. The semantics of this primitive are as follows:

(

SapId,

Handle,

Status,

)

####  PDE-MODULE.indication

The PDE-MODULE.indication primitive indicates the presence of data from a designated protocol module that is to be sent to a higher layer. The semantics of this primitive are as follows:

 (

DstSapId,

ModuleId,

PdeOpData,

Status

)

The primitive parameters are described in Table 4‑15

Table 4‑14 PDE-MODULE.indication parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Valid range | Description |
| DstSapId | Integer | 0x00-0xffff | The ID of the destination SAP. |
| ModuleId | Integer | 0x0000-0xffff | The ID of the targeted module . |
| PdeOpData | Set of octets | — | Operation information (e.g., a command) specific to the module. |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction; used to match a confirm primitive with the corresponding request. |

#### PDE-MODULE.response

The PDE-MODULE.response primitive allows the next higher layer of a device to respond to the PDE-MODULE.indication primitive. The semantics of this primitive are as follows:

(

DstSapId,

ModuleId,

PdeOpData,

Status

)

The primitive parameters are described in Table 4‑15.

Table 4‑15 PDE-MODULE.response parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Valid range | Description |
| DstSapId | Integer | 0x00-0x0f | The ID of destination SAP. |
| ModuleId | Integer | 0x0000-0xffff | The ID of the targeted module. |
| PdeOpData | Set of octets | — | The response data of the operation. |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction; used to match a confirm primitive with the corresponding request. |
| Status | Enumeration | SUCCESS, DENIED,INVALID\_PARAMETER, TIMEOUT | The status of the indication to the module. |

# Multiplexed MAC Interface (MMI)

## Purpose

* Directs and may modify information from a protocol module SAP to the MCPS or MLME SAP or another protocol module’s SAP
* Directs and may modify information from the MCPS or MLME SAP to a protocol module SAP
* Provides a fragmentation/defragmentation service to the data sent to or from the MCPS-SAP, the MLME-SAP, or another protocol module SAP.

## Overview

The mechanism for the MMI, i.e. the ability to send the data to the proper SAP and fragmentation/defragmentation is similar to the mechanism defined in IEEE Std 802.15.9 for the multiplexed data service.

The MMI is responsible for determining the capability of the Remote Nodes, i.e. whether they support ULI IEs.

The process of sending the data to the MCPS-SAP includes possibly fragmenting the data and formatting the ULI IE, i.e. inserting the appropriate headers into the payload of the frame for transmission.

The process of receiving the data from the MCPS-SAP includes possibly defragmenting that data (buffering all fragments until the data is completely received), removing the ULI IE headers, and passing the data to the appropriate protocol module SAP.

The interface between the MMI and the ULI protocol modules includes the EtherType/Dispatch code and the payload.

Does the MMI need to include MLME primitives? Can’t protocol modules access MLME SAP on their own? The MLME section of 802.15.4 is very complicated and very often the MMI would add no value other than providing a protocol module with a defined MLME interface structure and providing a single point of control over access to the MLME SAP.

If a protocol module wishes to configure the MAC/PHY using profiles then it would need a primitive to request the MPM to send configuration data to the MAC/PHY

The MMI service consists of six primitives as shown in Table 5‑1.

Table 5‑1 Summary of MMI primitives

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Request | Indication | Response | Confirm |
| MMI-DATA | X | X |  | X |
| MMI-PURGE | X |  |  | X |
| MMI-MGMT | X | X |  | X |
| MMI-CONFIG | X | X |  | X |
| MMI-SERVICE-DISCOVERY | X |  |  | X |

## MMI-DATA primitives

The MMI data primitive delivers an MMI data payload from the protocol blocks to the MCPS-SAP after it packages them into a ULI IE or an MPX IE using the formats shown in Table 5‑2 and Table 5‑3. The dispatch or EtherType ID indicates the ULI destination of the data payload.

Table 5‑2

|  |  |
| --- | --- |
| Octets: 1 | Variable |
| ULI IE ID | Payload |

Table 5‑3

|  |  |  |
| --- | --- | --- |
| Octets: 1 | 2 | Variable |
| MPX IE ID | Dispatch/EtherType ID | Payload |

The formatted ULI IE or MPX IE is sent using the MCPS-DATA primitive via either Data or Multipurpose frames to the recipient device. At the recipient device, the ULI IE or MPX IE is delivered to the MCPS-SAP where the MMI data service delivers the data payload to the SAP of the protocol block or upper layer interface as identified by the dispatch/EtherType ID. Figure 5‑1 illustrates this message sequence.

ULI

MAC

Protocol Module

Protocol Module

MAC

ULI

MMI-DATA.request

MCPS-DATA.request

Data frame

MCPS-DATA.indication

ACK frame

MMI-DATA.indication

MCPS-DATA.confirm

MMI-DATA.confirm

Figure 5‑1 MMI message sequence

### MMI-DATA.request

The MMI-DATA.request primitive requests the transfer of an MMI payload to a remote device via the IEEE 802.15.4 MAC/PHY. The semantics of this primitive are as follows:

(

SrcAddrMode,

DstAddrMode,

DstPanId,

DstAddr,

Msdu,

MsduHandle,

HeaderIeList,

PayloadIeList,

HeaderIeIdList,

NestedIeSubIdList,

AckTx,

GtsTx,

IndirectTx,

SecurityLevel,

KeyIdMode,

KeySource,

KeyIndex,

UwbPrf,

Ranging,

UwbPreambleSymbolRepetitions,

DataRate,

LocationEnhancingInformationPostamble,

LocationEnhancingInformationPostambleLength,

PanIdSuppressed,

SeqNumSuppressed,

SendMultipurpose

FrakPolicy,

CriticalEventMessage
)

The primitive parameters are described in

Table 5‑4.

Table 5‑4 MMI-DATA.request parameters

| **Name** | **Type** | **Valid range** | **Description** |
| --- | --- | --- | --- |
| SrcAddrMode | Enumeration | NONE, SHORT, EXTENDED | The source addressing mode for this MMI data. |
| DstAddrMode | Enumeration | NONE, SHORT, EXTENDED | The destination addressing mode for this MMI data. |
| DstPanId | Integer | 0x0000–0xffff | The PAN identifier of the entity to which the MMI data is being transferred. |
| DstAddr | — | As specified by the DstAddrMode parameter. | The address of the receiving (destination) device. |
| MultiplexId | Integer | 0x0000–0xffff | The higher-layer protocol using the MMI data service. ULI IE ID or MPX IE ID and the EtherType/Dispatch code |
| MmiData | Set of octets | — | The set of octets forming the IE data payload. |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to the particular primitive transaction; used to match a confirm primitive with the corresponding request. |
| SecurityLevel | Integer | 0–7 | The combination of Message Integrity Check and Encryption to be applied to the payload of the MMI data service. For encoding see Table 9-6 in IEEE Std 802.15.4. |
| KeyIdMode | Integer | As defined in Table 9-7 of IEEE Std 802.15.4. | The mode used to identify the key purportedly used by the originator of the received frame. This parameter is invalid if the SecurityLevel parameter is set to 0x00. |
| KeySource | Set of octets | As indicated by the KeyIdMode parameter. | The originator of the key purportedly used by the originator of the received frame. The KeySource field, when present, indicates the originator of a group key. If the Key Identifier Mode field indicates a 4-octet Key Source field, then the Key Source field shall be the macPanId of the originator of the group key right concatenated with the macShortAddress of the originator of the group key. If the Key Identifier Mode field indicates an 8 octet Key Source field, then the Key Source field shall be set to the macExtendedAddress of the originator of the group key. This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00 or set to 0x01. |
| KeyIndex | Integer | 0x01–0xff | The Key Index field allows unique identification of different keys with the same originator. It is the responsibility of each key originator to make sure that the actively used keys that it issues have distinct key indices and that the key indices are all different from 0x00. |
| Send-Multipurpose | Boolean | TRUE, FALSE | If TRUE, use the 802.15.4 Multipurpose frame type. If FALSE, use 802.15.4 Data frame type.  |

### MMI-DATA.confirm

The MMI-DATA.confirm primitive reports the results of a request to transfer data to another device. The semantics of the MMI-DATA.confirm are as follows:

MMI-DATA.confirm

(
Handle,
MaxTransferSize,
Status
)

The primitive parameters are described in Table 5‑5. If there is no capacity to store the transaction, the Status will be set to TRANSACTION\_OVERFLOW. In case the other end aborts the transaction then the status will be set to TRANSACTION\_ABORTED and the MaxTransferSize is set to the value returned from the other end.

Table 5‑5 MMI-DATA.confirm parameters

| **Name** | **Type** | **Valid range** | **Description** |
| --- | --- | --- | --- |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction; used to match a confirm primitive with the corresponding request. |
| MaxTransfer- Size | Integer | 0x0000–0xffff | In case of an aborted transaction this parameter can be returned from the other end to indicate the maximum size of transaction it can handle. In case another end did not give a maximum size, this is set to zero. |
| Status | Enumeration | SUCCESS,TRANSACTION\_OVERFLOW, TRANSACTION\_EXPIRED,CHANNEL\_ACCESS\_FAILURE, INVALID\_ADDRESS, NO\_ACK,COUNTER\_ERROR, FRAME\_TOO\_LONG,UNAVAILABLE\_KEY,UNSUPPORTED\_SECURITY, INVALID\_PARAMETER. TRANSACTION\_ABORTED | The status of the last MMI data transmission. |

### MMI-DATA.indication

The MMI-DATA.indication primitive delivers a MMI payload from a remote device. The semantics of this primitive are as follows:

(
SrcAddrMode,
SrcPanId,
SrcAddr,
DstAddrMode,
DstPanId,
DstAddr,
MultiplexId,
MmiData,
SecurityLevel,
KeyIdMode,
KeySource,
KeyIndex
)

The primitive parameters are described in Table 5‑6.

Table 5‑6 MMI-DATA.indication parameters

| **Name** | **Type** | **Valid range** | **Description** |
| --- | --- | --- | --- |
| SrcAddrMode | Enumeration | NONE, SHORT, EXTENDED | The source addressing mode for this MMI data payload. |
| SrcPanId | Integer | 0x0000–0xffff | The PAN identifier of the entity from which MMI data is being transferred. |
| SrcAddr | — | As specified by the SrcAddrMode parameter. | The address of the transmitting (source) device. |
| DstAddrMode | Enumeration | NONE, SHORT, EXTENDED | The destination addressing mode for this MMI data payload. |
| DstPanId | Integer | 0x0000–0xffff | The PAN identifier of the entity to which the MMI data is being transferred. |
| DstAddr | — | As specified by the DstAddrMode parameter. | The address of the receiving (destination) device. |
| MultiplexId | Integer | 0x0000–0xffff | The higher-layer protocol using the MMI data service. See 7.2.3 |
| MmiData | Set of octets | — | The set of octets forming the MPX data payload. |
| SecurityLevel | Integer | 0–7 | See Table 2. |
| KeyIdMode | Integer | 0x00–0x03 | See Table 2. |
| KeySource | Set of octets | As specified by the KeyIdMode parameter. | See Table 2. |
| KeyIndex | Integer | 0x01–0xff | See Table 2. |

## MMI-PURGE primitives

The MMI-PURGE primitives provide a means to remove or abort pending transfers from the MMI transaction queue of the originator.

### MMI-PURGE.request

The MMI-PURGE.request primitive allows the next higher layer to purge an MMI payload from the transaction queue.

The semantics of the MMI-PURGE.request are as follows:

 (
Handle,
SendAbort
)

The primitive parameters are described in Table 5‑7.

Table 5‑7 MMI-PURGE.request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Valid range** | **Description** |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction used to match an MMI-PURGE.request primitive with the corresponding MMI-DATA.confirm primitive. |
| SendAbort | Boolean | TRUE, FALSE | If this parameter is TRUE and the transaction is still active, the MMI data service sends a MPX IE with an abort code to the other end indicating that the transaction was aborted. If this parameter is FALSE, the transaction is just purged locally, and no information is sent to the other end. |

On receipt of the MMI-PURGE.request primitive, the MMI data service attempts to find within the transaction queue the payload indicated by the MmiHandle parameter. If an MMI payload has left the transaction queue, the handle will not be found, and the MMI payload can no longer be purged. If an MMI payload matching the given handle is found, the payload is discarded from the transaction queue, and optionally an abort message is sent to the other end, if the SendAbort parameter is TRUE. Sending an abort message to the other end will allow the other end to clear out its state immediately without waiting for the timeout.

The MMI-PURGE.request will also issue a corresponding MCPS-PURGE.request to the MAC data service, provided it has an MCPS-DATA.request in process when the MMI-PURGE.request is called.

### MMI-PURGE.confirm

The MMI-PURGE.confirm primitive allows the MMI data service to notify the next higher layer of the success of its request to purge a MMI payload from the transaction queue.

The semantics of this primitive are as follows:

 (
Handle,
Status
)

The primitive parameters are described in Table 5‑8.

Table 5‑8 MMI-PURGE.confirm parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Valid range** | **Description** |
| Handle | Integer | 0x00–0xff | An identifier which can be used to refer to a particular primitive transaction; used to match a confirm primitive with the corresponding request. |
| Status | Enumeration | SUCCESS, INVALID\_HANDLE | The status of the request to purge MMI data from the transaction queue. |

## MMI-MGMT primitives

### Overview

The MMI-MGMT primitives takes an MMI management payload from the protocol modules, packages it into either a ULI IE or MPX IE, as shown in Figure 5‑2, delivers it to the MLME-SAP, and then using the MLME-IE-NOTIFY primitive it is sent via either Command or the Enhanced Ack frames to the remote device. At the remote device, the ULI IE or MPX IE is delivered to the MLME-SAP, where the MMI management service delivers the management payload to the identified IEEE 802.15.12 protocol module.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Octets: 1** | **0/1** | **0/2** | **0/2** | **Variable** |
| Transaction Control | Fragment Number | Total Upper-Layer Frame Size | Multiplex ID | Upper-Layer Frame Fragment |

Figure 5‑2 MMI MGMT IE

### MMI-MGMT.request

Send mgmt. payload to remote device

### MMI-MGMT.confirm

### MMI-MGMT.indication

### MMI-MGMT.response

## MMI-CONFIG primitives

The MMI-CONFIG primitive delivers an MMI configuration payload from the MPM to the MLME-SAP or other protocol modules. The configuration payload is formatted as per the appropriate IEEE 802.15.4 primitive accessed through the MLME-SAP.

## MMI- SERVICE-DISCOVERY primitives

### MMI-SERVICE-DISCOVERY Primitive Overview

The MMI-SERVICE-DISCOVERY primitive supports the request for and response citing the services contained within a remote device. The primitive parameters are described in Table 5‑9.

### MMI-SERVICE-DISCOVERY.request

This primitive serves to request service information from a remote device.

The semantics of this primitive are as follows:

(

ExtAddress,

Handle

)

### MMI-SERVICE-DISCOVERY.confirm

This primitive sends the remote device service information back to the originating device.

The semantics of this primitive are as follows:

(

Maximum frame size,

OptProtocolModulesPresent,

PdeSaps,

Handle,

Status

)

Table 5‑9 MMI-SERVICE-DISCOVERY parameters

| Name | Type | Valid range | Description |
| --- | --- | --- | --- |
| ExtAddress | — | Any valid extended address | The extended address of the remote device. |
| MaxTransferSize | Set of octets | --- | The maximum frame size that the remote device can properly receive |
| OptProtocolModulesPresent | Enumeration | 0x00–0xff | List of EtherType/Dispatch codes for the optional IEEE 802.15.12 protocol modules residing in the remote device |
| PdeSaps | Enumeration |  | List of EtherType/Dispatch codes for the IEEE 802.15.12 PDE SAPs present on the remote device |
| Handle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction; used to match a confirm primitive with the corresponding request. |
| Status | Enumeration | SUCCESS,TRANSACTION\_OVERFLOW, INVALID\_PARAMETER, TRANSACTION\_ABORTED, RESOURCES\_UNAVAILABLE | The status of the last primitive. |

# Management Protocol Module

## Purpose

The MPMprovides:

* Configuration parameters to the MAC and PHY using configuration data received from a higher layer or another module
* Configuration parameters to other protocol modules received from a higher layer or stored in the management protocol module

Note: IEEE 802.15.12 Profile IDs, used to identify the device/module configuration, shall be assigned by the 802.15 ANA for global profiles.

* Network device monitoring or management. The monitoring function defines managed objects to provide device monitoring metrics to a higher layer application. The management function uses data collected from the device to optimize the device’s configuration for better spectral use.

## Function

The MPM provides the information base for the 802.15.4 device and all modules.

The MPM provides the initialization of the 802.15.4 device and all modules

The MPM provides the information necessary and the execution/instruction sequence for the 802.15.4 device to provide network formation or to join a network

The MPM can act as the “higher layer” to the 802.15.4 device. Hence it provides short addresses, channel information, device authorization, error condition correction, etc.

The MPM shall be able to start scans, receive information from scans, and determine the appropriate action from the scan information.

The MPM of the PAN coordination shall provide resolution to a “PAN ID Conflict” by changing the coordinator’s PAN ID.

The MPM shall be able to handle transactions, both successful and failures.

The MPM shall be able to set up superframes.

The MPM shall be able to allocate and deallocate GTSs.

The MPM shall be able to allocate and deallocate DSME GTSs.

The MPM shall be able to set up CSL, RIT, and IRIT behaviors.

The MPM shall be able to set up the hopping behavior.

# PTM Description

## Purpose

The PTM provides:

* A conduit between the MMI and the PDE
* Allows applications/functions above the ULI to transparently access the 802.15.4 device
* Allows data from MCPS-SAP to be sent directly to those applications/functions above the ULI not using other protocol modules
* Allows legacy applications/functions (non-ULI capable) to be compatible with ULI devices
* Responds to primitives (i.e. MCPS.DATA.confirm and MCPS.DATA.indication) delivered via the data SAP, such as passing the MPDU to a higher layer function

## Overview

For data to be transmitted, the PTM allows a higher layer to supply the IEEE 802.15.4 parameters for presentation of the payload (via PTM-SAP) to the MCPS-SAP. In order to reduce the number of parameters required by the PTM, it will also be allowed for a higher layer protocol to pass a profile identifier as one of the parameters. Then, most of the parameters would be implicitly available to the PTM by referring to the definition and current configuration state for that profile within the ULI layer.

Upon reception of a well-formed 802.15.4 frame, the PTM should be able to dispatch the frame by inspection of the frame fields following the MHR. But this is what the ULI must anyway do upon frame reception, so that it is not clear whether the PTM actually has any responsibility for handling incoming frames.

## Design of (*and questions about*) the PTM

The interfaces of the PTM are shown in as PTH-SAP and PTM-SAP.

*From one point of view, there should be very little specification required for such a module, since it does not seem to impose any process requirements on the data that it passes through to the next lower layer (for transmission) or next higher layer (for reception). Nevertheless, there are several possibilities that sense. Do we provide for all of them? If only one, how do we choose?*

*Does the PTM bypass L2R routing operations? If so, then transmission could occur only to immediate neighbors. Similarly, if ranging is required prior to transmission, how does PTM know to make the request?*

*If profiles are not enabled for use with the PTM function, how does the function identify the proper MAC interface? Are the appropriate MAC and PHY layers associated with the source MAC address of the frame? Or, with the destination MAC address?*

*If profiles are enabled for use, then a table of profiles will be required, along with the MCPS-SAP interface parameters that each profile allows the PTM to infer from the profile definition. In this way, most of the PTM specification text actually would reside as part of the profile definitions. Each new profile will need a section within its specification for use by the generic PTM.*

*Some parameters aren’t simple data types, and can have substructure and refer to objects with nontrivial structure. We will need NULL (not present) parameters.*

1. Note: The ULI may convert the extended addresses to short addresses for reduced overhead packets. [↑](#footnote-ref-1)