**IEEE P802.15**

**Wireless Personal Area Networks**

|  |  |  |
| --- | --- | --- |
| Project | IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs) | |
| Title | **<MMI and PDE Operation>** | |
| Date Submitted | [15 November 2016] | |
| Source | [Pat Kinney] [<company>] [address] | Voice: [ ] Fax: [ ] E-mail: [ ] |
| Re: | TG12 Architecture: PDE and MMI operation | |
| Abstract | [Work in Progress – ] | |
| Purpose | [Description of what the author wants P802.15 to do with the information in the document.] | |
| 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. | |

# MMI and PDE Operation

## PDE Description

### Purpose:

Directs and optionally modifies information from protocol module SAP to the appropriate higher layer SAP or to another protocol module SAP.

### Overview

For frames going to the higher layer, the PDE determines the appropriate SAP for delivery, as determined by the ULI header, removes the ULI header, reconstitutes the appropriate header, and then directs the datagram to the SAP.

For datagrams coming from a higher layer, the PDE determines the SAP to which the datagram is to be sent based upon the configuration of the device as set by the Management Protocols entity, and forwards it to the appropriate SAP.

#### EtherType

| **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 6  Transmission of Packets over Ethernet Networks, RFC-2464, Internet Society, Dec. 1998. http://www.ietf.org/rfc/rfc2464.txt |
| 888E | IEEE 802.1  802.1 Chair c/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 P1451.0 700 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

##### Multiplex ID field

The Multiplex ID field is used to multiplex different upper-layer protocols. The Multiplex ID field 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 values as specified in Table 20.
2. If the value of this field is greater than 1500, the Multiplex ID field indicates the Ethertype of the MAC client protocol.

|  |  |  |
| --- | --- | --- |
| Multiplex ID (decimal) | Multiplex ID (hex) | Description |
| 1 | 0x0001 | KMP |
| 2–1279 | 0x0002–0x04ff | Reserved |
| 1280–1500 | 0x0500–0x05dc | Vendor specific |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Request | Indication | Response | Confirm |
| PDE-DATA | X | X |  | X |
| PDE-CONFIG | X | X |  | X |
| PDE-PURGE | X |  |  | X |

## MMI Description

The MMI service provides a multiplex mechanism to allow the protocol blocks to receive or send information to other blocks or either of the MAC SAPs.

The MMI service consists of four primitives as shown in Table 1.

Table 1—Summary of MMI primitives

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

The MMI data service delivers an MMI data payload from the protocol blocks to the MCPS-SAP after it packages them into a ULI IE using the format shown in Figure 1. The dispatch or EtherType ID indicates the ULI destination of the data payload.

Figure 1

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

The formatted ULI 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 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 2 illustrates this message sequence.

ULI

MAC This ULI 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 is delivered to the MCPS-SAP where the MMI data service delivers the data payload to the identified protocol block. Figure 2 illustrates this message sequence.

C

Protocol Bock

Protocol Block

MAC

ULI

MMI-DATA.request

MCPS-DATA.request

Data frame

MCPS-DATA.indication

ACK frame

MMI-DATA.indication

MCPS-DATA.confirm

MMI-DATA.confirm

The MMI management service takes an MMI management payload from the protocol blocks, packages it into an ULI IE as shown in Figure 1, 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 recipient device. At the recipient device, the ULI IE is delivered to the MLME-SAP, where the MMI management service delivers the management payload to the identified ULI protocol block.

The MMI configuration service delivers an MMI configuration payload from the Management protocol block to the MLME-SAP or other protocol blocks. Alternatively, the MMI configuration service may deliver an MMI configuration payload from another protocol block to the MLME-SAP The configuration payload is formatted as per the appropriate IEEE 802.15.4 primitive accessed through the MLME-SAP.

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

### MMI Data Service Primitives

# MMI-DATA.request

MMI-DATA.indication

MMI-DATA.confirm

The primitive parameters are described in Table 2.

# Table 2—MPX-DATA.request parameters

| Name | Type | Valid range | Description |
| --- | --- | --- | --- |
| SrcAddrMode | Enumeration | NONE, SHORT, EXTENDED | The source addressing mode for this MPX data. |
| DstAddrMode | Enumeration | NONE, SHORT, EXTENDED | The destination addressing mode for this MPX data. |
| DstPanId | Integer | 0x0000–0xffff | The PAN identifier of the entity to which the MPX 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 MPX data service. See 7.2.3. |
| MpxData | Set of octets | — | The set of octets forming the MPX data pay- load. |
| MpxHandle | Integer | 0x00–0xff | An identifier which 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 MPX 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. |

### MPX-DATA.confirm

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

# MPX-DATA.confirm

# ( MpxHandle, MaxTransferSize, Status )

# The primitive parameters are described in Table 3. 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 MaxTranferSize is set to the value returned from the other end.

# Table 3—MPX-DATA.confirm parameters

| Name | Type | Valid range | Description |
| --- | --- | --- | --- |
| MpxHandle | 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 an other 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 MPX data transmission. |

### MPX-DATA.indication

# The MPX-DATA.indication primitive delivers a MPX payload from another device. The semantics of this primitive are as follows:

# MPX-DATA.indication

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

# The primitive parameters are described in Table 4.

# Table 4—MPX-DATA.indication parameters

| Name | Type | Valid range | Description |
| --- | --- | --- | --- |
| SrcAddrMode | Enumeration | NONE, SHORT, EXTENDED | The source addressing mode for this MPX data payload. |
| SrcPanId | Integer | 0x0000–0xffff | The PAN identifier of the entity from which MPX 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 MPX data payload. |
| DstPanId | Integer | 0x0000–0xffff | The PAN identifier of the entity to which the MPX 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 MPX data service. See 7.2.3 |
| MpxData | 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 Service Primitive(s)

MMI-PURGE.request

MMI-PURGE.confirm

### MPX-PURGE primitives

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

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

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

# MPX-PURGE.request

# ( MpxHandle, SendAbort )

# The primitive parameters are described in Table 5.

# Table 5—MPX-PURGE.request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Valid range | Description |
| MpxHandle | Integer | 0x00–0xff | An identifier that can be used to refer to a particular primitive transaction; used to match a MPX-PURGE.request primitive with the corresponding MPX-DATA.confirm primitive. |
| SendAbort | Boolean | TRUE, FALSE | If this parameter is TRUE and the transaction is still active, the MPX 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 MPX-PURGE.request primitive, the MPX data service attempts to find in the transaction queue the payload indicated by the MpxHandle parameter. If a MPX payload has left the transaction queue, the handle will not be found, and the MPX payload can no longer be purged. If a MPX 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. If an abort message is sent to the other end that will allow the other end to clear out its state immediately without waiting for the timeout.

# The MPX-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 MPX-PURGE.request is called.

### MPX-PURGE.confirm

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

# The semantics of this primitive are as follows:

# MPX-PURGE.confirm

# ( MpxHandle, Status )

# The primitive parameters are described in Table 6.

# Table 6—MPX-PURGE.confirm parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Valid range | Description |
| MpxHandle | 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 MPX data from the transaction queue. |

### MMI Management Service Primitives

# MMI-MGMT.request

MMI-MGMT.indication

MMI-MGMT.confirm

MMI Configuration Service Primitives

# MMI-CONFIG.request

MMI-CONFIG.indication

MMI-CONFIG.confirm