**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 September] | |
| Source | [] [] [address] | Voice: [ ] Fax: [ ] E-mail: [ ] |
| Re: | TG12 Architecture: PDE and MMI operation | |
| Abstract | [Work in Progress – MMI has been started (pgs 2 – 4) but PDE has not.] | |
| 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

## 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

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

# MMI PURGE Service Primitive(s)

MMI-PURGE.request

MMI-PURGE.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 DstAd- drMode 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 KeyId- Mode parameter. The originator of the key purportedly used by the originator of the received frame. The Key- Source field, when present, indicates the origi- nator 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 concate- nated with the macShortAddress of the origina- tor 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 macEx- tendedAddress of the originator of the group key. This parameter is invalid if the KeyId- Mode parameter is invalid or set to 0x00 or set to 0x01.

# 

# Table 2—MPX-DATA.request parameters (continued)

# Name Type Valid range Description

# KeyIndex Integer 0x01–0xff The Key Index field allows unique identifica- tion 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 is- sues have distinct key indices and that the key indices are all different from 0x00.

# SendMultipur- pose Boolean TRUE, FALSE If TRUE, use the Multipurpose frame type. If FALSE, use Data frame type. See 8.3.1 MCPS-DATA.request of IEEE Std 802.15.4.

# 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.

# 

# Table 3—MPX-DATA.confirm parameters (continued)

# Name Type Valid range Description

# 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, TRANSAC- TION\_OVERFLOW, TRANSACTION\_EX- PIRED, CHANNEL\_AC- CESS\_FAILURE, INVALID\_ADDRESS, NO\_ACK, COUNTER\_ER- ROR, FRAME\_TOO\_LONG, UN- AVAILABLE\_KEY, UN- SUPPORTED\_SECURITY, INVALID\_PARAMETER. TRANSACTION\_ABORT- ED 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 SrcAd- drMode 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 DstAd- drMode 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 KeyId- Mode parameter. See Table 2.

# KeyIndex Integer 0x01–0xff See Table 2.

# 

# 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 particu- lar 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.

# KMP transport service

# KMP services provide two basic primitives: the first one is used to initiate SA creation, and the second one is used to indicate when the key management process is ready and the keys are ready to be used. In addition to those two primitives, there are some other management interfaces, but those might depend on the KMP used. A summary of KMP primitives can be found in Table 7.

# Table 7—Summary of KMP Primitives

# Name Request Indication Response Confirm

# KMP-CREATE 6.1.1 6.1.3 6.1.4 6.1.2

# KMP-FINISHED — 6.2.1 — —

# KMP-DELETE 6.3.1 6.3.3 — 6.3.2

# KMP-PURGE 6.4.1 — — 6.4.2

# The pairwise keys negotiated between the peers (i.e., using KeyIdMode 0x00) do not have a KeyIndex, i.e., there can be only one SA between the peers. This creates challenges for the rekeying of the SA, as a new a SA cannot be created before the old one is removed. On the other hand, as there is only one SA, if a peer creates a new pairwise key between the peers, both ends will know that the previous SA has been removed. Because the installation of the key can happen at different times in peers, some traffic might be lost between the peers during rekey, as it is possible that one end has already installed a new key but that the other end is still using the old one. Because of this, the upper layer, desiring no data loss, should cease transmitting data using the key to be rekeyed while the rekey is in progress and only continue transmitting data after the KMP- FINISHED.indication primitive has been received.

# The initial KMP frames need to be sent out using security level 0, as there is not yet an SA set up between the peers. In IEEE Std 802.15.4, there is a way to do security filtering based on the IEs, i.e., the security PIB can be configured to allow a MPX IE with security level 0 for suitable frame types. This allows the ability to process security level 0 MPX IEs even when the frames would require