IEEE P802.11
Wireless LANs

|  |
| --- |
| Resolution of Flow Control related CIDs |
| Date: 2018-02-13 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Oren Kedem | Intel |  |  | oren.kedem@intel.com |
| Idan Maor | Intel  |  |  | idan.maor@intel.com |
| Solomon, Trainin | Qualcomm |  |  | strainin@qti.qualcomm.com |
|  |  |  |  |  |

Abstract

This submission proposes resolutions to 1070, 1187, 1709, 1710, 1071, 1073, 2116, 1286, 1713, 1712, 1135, 2115, 2117, 2240, 2241, 1125, 1713, 2130, 2131, 2133, 1108, 1219, 1876, 1983, 2137, 2267, 2276, 2277, 2278, 2280, 2281, 2268, 2282 CIDs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 1070 | 9.3.1.9.1 | "No\_Mem\_Kept" Why not simply name it "No Memory Kept"? | As in comment. Change throughout the draft. | AcceptedName was changed to  No Memory Kept |
| 1187 | 9.3.1.9.1 | "No\_Mem\_Kept" - this is a stylistic departure from the 802.11 baseline. Also, don't abrvt unless absltly nccsry. | Change subfield globally to "No Memory Kept" |
| 1709 | 9.3.1.9.1 | Undescores in subfield name inconsistent with established style | Remove underscores |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 1710 | 9.3.1.9.1 | "at successive BlockAck agreement establishements": clumsy wording and inaccurate since it does not define what happens when only one block ack agreement is established. Also, the relationship between the BlockAck frame (which carries the subfield) and the block ack agreement establishment is not clear. | "The No\_Mem\_Kept subfield is set to 0 in the first BlockAck frame sent under a block ack agreement." | Revised: |

**Discussion**

Since EDMG Flow Control Extension Configuration element included the RBUFCAP but didn’t include the No Memory Kept, the intention of the rule is to indicate it should be considered as 0 after Block Ack agreement. Proposed solution is to add Flow Control status field that include it together with Memory Configuration Tag so ambiguity is removed and there is no need for this rule. Rule is removed.

**Revised Text**

**9.3.1.9 BlockAck frame format
9.3.1.9.1 Overview***Change Figure 9-33 as follows*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | B0 | B1 – B4 | B5 – D8 | B9 | B10  |  B11 | B12 - B15 |
|  | BA ACK Policy | BA Type | Reserved | No MemoryKept | Memory Configuration Tag | ManagementACK | TID\_INFO |
| Bits: | 1 | 4 | 4 | 1 | 1  | 1 | 4 |

The No Memory Kept subfield set to one indicates that the free memory space indicated in the last
RBUFCAP is not kept at the start of the next frame exchange sequence; otherwise if set to zero, free
memory space as indicated by RBUFCAP is kept by the receiver for the next frame exchange sequence of the same TID or group of TIDs.

The Memory Configuration Tag subfield indicates one out of two memory configurations as indicated in
Memory Configuration Tag field in the recipient’s EDMG Flow Control Extension Configuration element (9.4.2.263).

The Management ACK subfield is set to one to indicate that frames of type Management that are not
Action No Ack are acknowledged. This subfield is reserved if the BlockAck variant used is not the EDMG
Multi-TID BlockAck variant.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 1071 | 9.3.1.9.7 | "RBUFCAP field" To be strict, it is a subfield. | Change field to subfield. | Accepted.   |
| 1073 | 9.3.1.9.8 | "The RBUFCAP subfield is defined in 9.3.1.9.5." It should be 9.3.1.9.7. | As in comment. | Accepted.   |
| 2116 | 9.3.1.9.7 | The RBUFCAP field in 11ay seems to be dffierent from what specified in 802.11-2016. From 802.11-2016,, "The RBUFCAP field contains an unsigned integer that is the number of MPDU buffers available to store received MPDUs at the time of transmission of the Extended Compressed BlockAck frame (10.39)." | Make the definition of the field consistent | Revised.There are two definitions one for DMG indicated in section 9.3.1.9.5 and EDMG defined in 9.3.1.9.7.The EDMG point to 9.3.1.9.7.  |
| 1286 | 9.3.1.9.7 | "Indicates that the recipient's memory has enough space to receive A-MPDUs with a length that is not less than indicated by Maximum A-MPDU Length Exponent (Table 3) "" | Change to "Indicates that the recipient's memory space length is not less than indicated by Maximum A-MPDU Length Exponent (Table 3) " | Accepted |
| 1713 | 9.3.1.9.7 | Use decimal values in RBUFCAP value column. Change "1 through 0xFE" to "1-254". Remove "RBUFCAP (RBUF\_Unit\_Size)" from last row, second column since a name is not needed here. | As in comment | Accepted  |
| 1712 | 9.3.1.9.7 | Unlimited\_space: not implementable. Zero\_space: space is uncountable so it can't be zero. | Unlimited\_space -> RBUF\_EMPTY. Zero\_space -> RBUF\_FULL | Revised  |

**Discussion**

11ay extends the DMG RBUFCAP definition for EDMG 11ay STA.

**9.3.1.9.7 EDMG Compressed BlockAck variant**

*Change the following subclause in the end of the paragraph*

The RBUFCAP field is defined in Table 1.

**Table 1— RBUFCAP encoding for the EDMG Compressed BlockAck variant**

|  |  |  |
| --- | --- | --- |
| **RBUFCAP value** | **RBUFCAP value name** | **Definition** |
| 0  | Receiver Buffer Empty  | Indicates that the recipient’s memory is not less than indicated by Maximum A-MPDU Length Exponent (Table 3) |
| 255 | Receiver Buffer Full  | Indicates no space in the recipient’s memory |
| 1 through 254 | Receiver Buffer Available | Indicates the size of recipient’s current memory that the originator can use to transmit MPDUs to the recipient; measured in units of Buffer Unit Size (9.4.2.263) |

**9.3.1.9.8 EDMG Multi-TID BlockAck variant**

*Change the following subclause in the end of the paragraph*

The RBUFCAP subfield is defined in 9.3.1.9.7.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 1135 | 9.4.2.263 | The Element ID, Length, and Element ID fields are defined in 9.4.2.1. | The Element ID, Length, and Element ID Extension fields are defined in 9.4.2.1. | Accepted |
| 2115 | 9.4.2.263 | """The Memory Configuration Tag subfield indicates one out of two memory configurations as indicated in Memory Config Tag field in the recipient's EDMG Flow Control Extension Configuration element (9.4.2.263)."" This sentece is incomplete, what is actaully indicated by the one out of two memeory configs ? what is Memory Config Tag field? I could not find it throughout the text" |  | Revised  |

**Discussion**

The field indicates that the STA is capable to support two memory configuration parameters each is indicated by different value (0 or 1) of Memory Configuration Tag (IE) or Memory Configuration Tag (Block Ack). Recipient capability is reflected by advertising two set of memory configurations and indicating the index in the Block Ack. Originator capability is reflected by the ability to calculate the transmitted frames byte count according to the recipient two set of memory configurations as indicated by the Memory Configuration Tag in the Block Ack.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 2117 | 9.4.2.263 | The Recipient Memory Capabilities field which conntains the Advanced Recipient Memory Length Capable subfield which reads "The Advanced Recipient Memory Length Capable subfield is set to 1 to indicate support of Advanced Recipient Memory Length Exponent (Figure 62) and is set to 0 otherwise." Shouldn't the Advanced Recipient Memory Length Exponent that is non-zero already indicated the support of Advanced Recipient Memory Length Exponent? It looks strage that a byte is already used to indicate the Advanced Recipient Memory Length Exponent in the element and then a bit later within the optional subelement is used to indicate whether the Advanced Recipient Memory Length Exponent is supported or not | clarify | Revised  |

**Discussion**

The Recipient Memory Capabilities field was moved to the EDMG Flow Control Extension configuration element as suggested. The need for indication in the capability is to allow the originator and the responder to advertise if they want to support it. Advanced Recipient Memory Length Exponent in value 0 is valid number that indicates 2^13 bytes. Advanced Recipient Memory Length Exponent is now limited by the value Maximum A-MPDU Length Exponent.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 2240 | 9.4.2.263 | Not clear what is the use of Advanced Recipient Memory Length Exponent/RBUFCAP from the originator | Add ''This field is reserved when transmitted by the originator' | Accepted.Field is used only in ADDBA Response  |
| 2241 | 9.4.2.263 | Should clarify that Recipient Memory Configuration subelement could be included twice in an EDMG Flow Control Extension Configuration Element | add a note indicating such | AcceptedField is used only in ADDBA Response |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 1125 | 10.24.7.7 | Mult\_Buff\_MPDU should be clarified further. Under current definition, if this field value is 0, the Mem\_Unit\_Size must to be in the size of max supported MSDU and it is not reflected in the text equations. | As suggested  | Revised  |

**Discussion**

The definition doesn’t restrict for recipient to indicate smaller buffer size than MAX MPDU size with "MPDU Split in Buffer" = 0 and by that to force the originator to discard MPDUs larger than Memory Unit Size, it is up to the implementation.

*Please change section as follow*

**9.4.2.263 EDMG Flow Control Extension Configuration element**

The EDMG Flow Control Extension Configuration element is defined in Figure 62.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Octets: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | Variable  |
|  | ElementID | Length  | Element IDExtension | RBUFCAP  | Flow Control Status | Advanced Recipient MemoryLength Exponent | Recipient Memory Capabilities field | OptionalSubelements |
|  |  |  |  |  |  |  |  |  |

**Figure 62 — EDMG Flow Control Extension Configuration element format**

The Element ID, Length, and Element ID Extension fields are defined in 9.4.2.1.

When sent in ADDBA response, the RBUFCAP field is as defined in 9.3.1.9.7. This field is reserved when it sent in ADDBA request frame.

When sent in ADDBA response, the Flow Control Status field is defined in figure 63, this field is reserved when it sent in ADDBA request frame.

|  |  |  |  |
| --- | --- | --- | --- |
|  | B0 | B1 | B2 – B7 |
|  | No MemoryKept | Memory Configuration Tag | Reserved |
| Bits: | 1 | 1 | 6 |

The Memory Configuration Tag and No Memory Kept subfields are defined in 9.3.1.9.1.

When sent in ADDBA response, the Advanced Recipient Memory Length Exponent field indicates the amount of free space at the recipient’s memory at the start of a frame exchange sequence. This field is an integer in the range 0 to 9. The length defined by this subfield is equal to 2(13 + Advanced Recipient Memory Length Exponent) – 1 octets. The value of 2(13 + Advanced Recipient Memory Length Exponent) is smaller than or equal to the value of the 2(13 + Maximum A-MPDU Length Exponent) as advertised by the STA’s EDMG Capabilities element, only one value of Advanced Recipient Memory Length Exponent may be present in all Block Ack agreements and this value is applies to all successfully established BlockAck agreements identified by the same pair of Address 1 and Address 2 fields. This field is reserved when it sent in ADDBA request frame

The Recipient Memory Capabilities field is defined in Figure 64.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | B0  | B1  | B2  | B3  | B4  | B5 B7 |
|  | RBUFCAPQuantityCapable | Advanced RecipientMemory LengthCapable | Recipient MemoryMultiple Buffer UnitsCapable | TIDGroupingCapable | Two Memory Config Tag Capable | Reserved |
| Bits: | 1 | 1 | 1 | 1 | 1  | 3 |

**Figure 64 — Recipient Memory Capabilities field format**

The RBUFCAP Quantity Capable subfield is set to 1 to indicate support of RBUFCAP values in the range 1 through 254 and is set to 0 otherwise (Table 1).

The Advanced Recipient Memory Length Capable subfield is set to 1 to indicate support of Advanced
Recipient Memory Length Exponent (Figure 62) and is set to 0 otherwise.

The Recipient Memory Multiple Buffer Units Capable subfield is set to 1 to indicate support of the Recipient Memory Unit Size, Maximum MPDU per Memory Unit, and MPDU Split in Buffer values (Figure 65) and is set to 0 otherwise.

The TID Grouping Capable subfield is set to 1 to indicate support of TID Grouping values (Figure 64) and
set to 0 otherwise.

The Two Memory Config Tag Capable subfield is set to 1 to indicate capability to support two Memory Configuration Tag values (Figure 65) and is set to 0 otherwise.

The Optional Subelements field is defined in Table 13. An EDMG Flow Control Extension Configuration
element contains no more than two Recipient Memory Configuration subelements.

**Table 13 — Optional subelement IDs for the EDMG Flow Control Extension Configuration
element**

|  |  |  |
| --- | --- | --- |
| **Subelement ID**  | **Name**  | **Extensible** |
|  |  |  |
| 0  | Recipient Memory Configuration  | Yes |
| 1-220  | Reserved |  |
| 221  | Vendor specific |  |
| 222-225  | Reserved |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

The Recipient Memory Configuration subelement is defined in Figure 65.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  | SubelementID | Length  | Memory Configuration Tag  | Buffer Unit Size | Recipient Memory MultipleBuffer Units Parameters | TIDGrouping |
|  | Memory UnitSize | Maximum MPDUper Memory Unit | MPDU Split in Buffer |
| Octets | 1 | 1 | 1 | 2 | 2 | 1 | 1  | 2 |

**Figure 65 — Recipient Memory Configuration subelement format**

The Subelement ID field is defined in Table 13.

The Length field is defined in 9.4.2.1.

The Memory Configuration Tag subfield indicates one of two Recipient Memory Configurations indicated in ADDBA Response frame within the EDMG Flow Control Extension Configuration element. Allowed values are 0 and 1.

The Buffer Unit Size subfield indicates the size, in units of bytes, of the RBUFCAP to deliver information of the recipient’s available memory space to the originator for MPDU delivery. It is set to value greater than 0 in case Recipient Quantity Capable subfield set to 1 and to value of zero otherwise. The recipient’s available memory space, in bytes, is equal to RBUFCAP × Buffer Unit Size.

Recipient Memory Multiple Buffer Units Parameters indicates altogether a recipient memory structure constructed from fixed size memory buffers used to store incoming MPDUs.

The Memory Unit Size subfield indicates the size, in units of bytes, of each buffer unit in the recipient’s memory.
The Maximum MPDU per Memory Unit subfield indicates the maximum number of MPDUs that can be held in a
single buffer. Valid values are 1 through 255, where value 255 indicates that there is no limitation on the number of
MPDUs can be held in a single buffer.

The MPDU Split in Buffer subfield is set to 1 to indicate that a single MPDU can be split between memory
buffer units in the recipient’s memory and is set to 0 otherwise.

The TID Grouping subfield is a bitmap where each bit corresponds to a TID/TSID (Figure 66). By setting a
bit to one in the TID Grouping subfield, it indicates the TID/TSID that correspond to a TID/TSID of an
ADDBA Response frame within which the Recipient Memory Configuration subelement(s) is transmitted.
The recipient memory configuration becomes applicable to all TIDs/TSIDs that have their bit in the TID
Grouping subfield set to one. The RBUFCAP, No Memory Kept and Memory Configuration Tag fields delivered in an EDMG Flow Control Extension Configuration element within an ADDBA Response frame and the RBUFCAP field delivered in a BlockAck frame are applicable to all TIDs/TSIDs that correspond to the TIDs/TSIDs of the ADDBA Response frame which contained the TID Grouping subfield. The Advanced Recipient Memory Length Exponent field delivered in an EDMG Flow Control Extension Configuration element of an ADDBA Response frame is applicable to all TIDs/TSIDs that correspond to the TIDs/TSIDs of the ADDBA Response frame which
contained the TID Grouping subfield.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | B0 | B1 | B2 |  | B14 | B15 |
|  | TID 0 | TID 1 | TID 2 | … | TID 14 | TID 15 |
| Bits: | 1 | 1 | 1 |  | 1 | 1 |

**Figure 66 — TID Grouping subfield format**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 1713 | 9.3.1.9.7 | If the STA uses EDMG flow control, parameters for EDMG flow control need to be added to MLME-ADDBA primitives. | Add parameters relating to flow control to MLME-ADDBA primitives in 6.3.29. | MLME-ADDBA was updated with Flow Control information per CID Solomon please add the relevant CID  |

*Change the section as follows*

**6.3.29.2 MLME-ADDBA.request
6.3.29.2.1 Function**This primitive requests the initiation (or modification) of block ack with a peer MAC entity.
**6.3.29.2.2 Semantics of the service primitive**The primitive parameters are as follows:

MLME-ADDBA.request(

PeerSTAAddress,
DialogToken,
TID,
BlockAckPolicy,
BufferSize,
BlockAckTimeout,
BlockAckStartingSequenceControl,
GCRGroupAddress,
Multi-band,
TCLAS,
ADDBA Extension,
VendorSpecificInfo,

EDMG Segmentation-Reassembly Configuration,

EDMGFlowControl
)

|  |  |  |  |
| --- | --- | --- | --- |
| **Name**  | **Type**  | **Valid range**  | **Description** |
| PeerSTAAddress  | MACAddress  | N/A  | Specifies the address of the peer MAC entity withwhich to perform the block ack initiation (ormodification). |
| ….. | …. | ….. | ……. |
| VendorSpecificInfo  | A set of elements  | As defined in9.4.2.26 | Zero or more elements. |
| EDMG Segmentation-Reassembly Configuration  | SegmentationandReassembly Configuration element | As defined in9.4.2.266 | Specify the Segmentation and Reassembly parameters  |
| EDMGFlowControl | EDMG Flow Control Extension Configuration element | As defined in9.4.2.263 | Specify the EDMG flow control parameters  |

**6.3.29.3 MLME-ADDBA.confirm
6.3.29.3.1 Function**The primitive reports the results of initiation (or modification) of the block ack attempt with the specified
peer MAC entity.
**6.3.29.3.2 Semantics of the service primitive**The primitive parameters are as follows:
MLME-ADDBA.confirm(

PeerSTAAddress,
DialogToken,
TID,
ResultCode,
BlockAckPolicy,
BufferSize,
BlockAckTimeout,
GCRGroupAddress,
Multi-band,
TCLAS,
ADDBA Extension,
VendorSpecificInfo,

EDMG Segmentation-Reassembly Configuration,

EDMGFlowControl
)

|  |  |  |  |
| --- | --- | --- | --- |
| **Name**  | **Type**  | **Valid range**  | **Description** |
| PeerSTAAddress  | MACAddress  | N/A  | Specifies the address of the peer MAC entity withwhich to perform the block ack initiation (ormodification). |
| ….. | …. | ….. | ……. |
| VendorSpecificInfo  | A set of elements  | As defined in9.4.2.26 | Zero or more elements. |
| EDMG Segmentation-Reassembly Configuration  | SegmentationandReassembly Configuration element | As defined in9.4.2.266 | Specify the Segmentation and Reassembly parameters  |
| EDMGFlowControl | EDMG Flow Control Extension Configuration element | As defined in9.4.2.263 | Specify the EDMG flow control parameters  |

**6.3.29.4 MLME-ADDBA.indication
6.3.29.4.1 Function**This primitive reports the initiation (or modification) of block ack by a peer MAC entity.
**6.3.29.4.2 Semantics of the service primitive**

The primitive parameters are as follows:
MLME-ADDBA.indication(

PeerSTAAddress,
DialogToken,
TID,
ResultCode,
BlockAckPolicy,
BufferSize,
BlockAckTimeout,
GCRGroupAddress,
Multi-band,
TCLAS,
ADDBA Extension,
VendorSpecificInfo,

EDMG Segmentation-Reassembly Configuration,

EDMGFlowControl
)

|  |  |  |  |
| --- | --- | --- | --- |
| **Name**  | **Type**  | **Valid range**  | **Description** |
| PeerSTAAddress  | MACAddress  | N/A  | Specifies the address of the peer MAC entity withwhich to perform the block ack initiation (ormodification). |
| ….. | …. | ….. | ……. |
| VendorSpecificInfo  | A set of elements  | As defined in9.4.2.26 | Zero or more elements. |
| EDMG Segmentation-Reassembly Configuration  | SegmentationandReassembly Configuration element | As defined in9.4.2.266 | Specify the Segmentation and Reassembly parameters  |
| EDMGFlowControl | EDMG Flow Control Extension Configuration element | As defined in9.4.2.263 | Specify the EDMG flow control parameters  |

**6.3.29.5 MLME-ADDBA.response
6.3.29.5.1 Function**The primitive responds to the initiation (or modification) by a specified peer MAC entity.
**6.3.29.5.2 Semantics of the service primitive**

The primitive parameters are as follows:
MLME-ADDBA.response(

PeerSTAAddress,
DialogToken,
TID,
ResultCode,
BlockAckPolicy,
BufferSize,
BlockAckTimeout,
GCRGroupAddress,
Multi-band,
TCLAS,
ADDBA Extension,
VendorSpecificInfo,

EDMG Segmentation-Reassembly Configuration,

EDMGFlowControl
)

|  |  |  |  |
| --- | --- | --- | --- |
| **Name**  | **Type**  | **Valid range**  | **Description** |
| PeerSTAAddress  | MACAddress  | N/A  | Specifies the address of the peer MAC entity withwhich to perform the block ack initiation (ormodification). |
| ….. | …. | ….. | ……. |
| VendorSpecificInfo  | A set of elements  | As defined in9.4.2.26 | Zero or more elements. |
| EDMG Segmentation-Reassembly Configuration  | SegmentationandReassembly Configuration element | As defined in9.4.2.266 | Specify the Segmentation and Reassembly parameters  |
| EDMGFlowControl | EDMG Flow Control Extension Configuration element | As defined in9.4.2.263 | Specify the EDMG flow control parameters  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 2130 | 10.24.2 | "The Advanced Recipient Memory Length Capable subfield in the STA's EDMG Flow Control Extension Configuration element is equal to one and the value of the Advanced Recipient Memory Length Exponent field in the STA's EDMG Flow Control Extension Configuration element is greater than or equal to the value of the Maximum A-MPDU Length Exponent field in the STA's EDMG Capabilities element."The Advanced Recipient Memory Length Capable subfield is within the optional subelement EDMG Flow COntrol Capabilities for the EDMG Flow Control Extension Configuration element. change to"The Advanced Recipient Memory Length Capable subfield in the EDMG Flow Control Capabilities subelement of the STA's EDMG Flow Control Extension Configuration element is equal to one..." | As suggested  | Revised: |
| 2131 | 10.24.2 | "The Advanced Recipient Memory Length Capable subfield and the RBUFCAP Quantity Capable subfield are both equal to one in the STA's EDMG Flow Control Extension Configuration element;"The Advanced Recipient Memory Length Capable subfield and the RBUFCAP Quantity Capable subfield are within the optional subelement EDMG Flow Control Capabilities for the EDMG Flow Control Extension Configuration element. change to"The Advanced Recipient Memory Length Capable subfield and the RBUFCAP Quantity Capable subfield in the EDMG Flow COntrol Capabilities subelement of the the STA's EDMG Flow Control Extension Configuration element are both equal to one;" | As suggested  | Revised  |

**Discussion**

D1.0 text provided the recipient to indicate Advanced Recipient Memory Length Exponent greater than Maximum A-MPDU Length Exponent size, hence this rule was added. Since the updated definition of the Advanced Recipient Memory Length Exponent is limited by the Maximum A-MPDU Length Exponent, the support of it doesn’t tied to the value of the Advanced Recipient Memory Length Exponent value hence the rule was changed

**10.24.2 Setup and modification of the block ack parameters***Change the 11th paragraph as follows*

An EDMG originator STA may insert an EDMG Flow Control Extension Configuration element in an
ADDBA Request frame. In this case, no EDMG Flow Control Extension Configuration element subelements shall be present.

An EDMG recipient STA that responds to an ADDBA Request frame that contains an EDMG Flow
Control Extension Configuration element should insert an EDMG Flow Control Extension Configuration
element in the ADDBA Response frame sent as response. A Recipient Memory Configuration subelement shall be included in the EDMG Flow Control Extension Configuration element sent by the recipient if at least one subfield except of Advanced Recipient Memory Length Capable in the Recipient Memory Capabilities field of the EDMG Flow Control Capabilities subelement is not equal to 0.

EDMG STAs that established a block ack agreement with or without exchange of an EDMG Flow Control
Extension Configuration element shall follow the flow control operation rules defined in 10.24.3, 10.24.4, 10.24.7.5 and 10.24.7.7.

The following negotiation rules apply to EDMG STAs (EDMG originator and EDMG recipient) that
exchange ADDBA Request and ADDBA Response frames.

An EDMG originator or EDMG recipient support those recipient memory capabilities for which the
corresponding subfields in the Recipient Memory Capabilities field of the EDMG originator or EDMG
recipient, respectively, are set to one in ADDBA Request and ADDBA Response frames of the block ack
agreement established between the EDMG originator and EDMG recipient, and do not support otherwise.

An EDMG recipient shall not respond with Status Code = SUCCESS in an ADDBA Response frame if the
EDMG recipient sets to one at least one of the subfields within the recipient’s Recipient Memory Capabilities field and:

* The same subfield is set to 0 in the Recipient Memory Capabilities field in the corresponding
ADDBA Request frame received from EDMG originator; or
* No EDMG Flow Control Extension Configuration element is present in the corresponding ADDBA
Request frame.

NOTE—Status Code values REFUSED, REFUSED\_REASON\_UNSPECIFIED, REQUEST\_DECLINED, or
INVALID\_PARAMETERS can be used in the aforementioned case.

Recipient Memory Multiple Buffer Units Capability is supported in a successfully established block ack agreements if both the originator and recipient have set the RBUFCAP Quantity Capable and Recipient Memory Multiple Buffer Units Capable subfields to 1 in the ADDBA Request and ADDBA Response and not supported otherwise.

Memory Config Tag Capability is supported in a successfully established block ack agreements if both the originator and recipient have set the RBUFCAP Quantity Capable and Two Memory Config Tag Capable subfields to 1 in the ADDBA Request and ADDBA Response and not supported otherwise.

A TID Grouping Capable capability is supported in a successfully established block ack agreements if both the originator and recipient have set the TID Grouping Capable and the RBUFCAP Quantity Capable subfields to 1 and not supported otherwise. The Recipient Memory Capabilities field and Recipient Memory Configuration subelement fields of TIDs were set to 1 in TID Grouping subfields sent in ADDBA Request or ADDBA Response shall be identical.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 2133 | 10.24.3 | "If the block ack agreement is between a pair of EDMG STAs, the memory occupied by the frames shall not exceed the maximum between the value indicated in the Advanced Recipient Memory Length Exponent field if the advanced recipient memory length capability is supported and the value indicated in the RBUFCAP field in the associated ADDBA Response frame....""change to""if the block ..... shall not exceed both the values indicated in the......ADDBA Response frame.""" | As suggested  | Rejected  |

**Discussion**

D0.1 text defined that in case Advanced Recipient Memory Length is supported, the originator may transmit up to the maximum between RBUFCAP and the Advanced….

However, initial intention of the feature was to provide size limitation on the originator in case no information other is provided. In case the recipient sent Non Memory Kept = 1, the originator should refer to the relevant RBUFCAP updated value since it is more updated than the general Advanced Recipient Memory Length Exponent value. The rules are defined in details in Table 22 and Table 23, the rule refer to those tables

**10.24.3 Data and acknowledgment transfer using immediate block ack policy and delayed
block ack policy***modify the D0.1 section as follows and add it after the second paragraph in 802.11 2016*

After setting up either an immediate block ack agreement or a delayed block ack agreement following the
procedure in 10.24.2, and having gained access to the medium and established protection, if necessary, the
originator may transmit a block of QoS Data frames separated by SIFS, with the total number of frames:

* Not exceeding the Buffer Size subfield value in the associated ADDBA Response frame and
subject to any additional duration limitations based on the channel access mechanism; and
* If the block ack agreement is between a pair of EDMG STAs, the memory occupied by the frames shall not exceed the value indicated in the Table 22 and Table 23. The actual RBUFCAP value is delivered by the EDMG Flow Control Extension Configuration element in the ADDBA Response frame or the RBUFCAP update for same or other TIDs as indicated in TID Grouping field of the Recipient Memory Configuration subelement, whichever comes later. If the ADDBA Response frame does not contain an EDMG Flow Control Extension Configuration element, the relevant originator parameters shall be considered as receiving an RBUFCAP of Receiver Buffer Empty (9.3.1.9.7).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 2136 | 10.24.7.5 | change "The RBUFCAP subfield in the transmitted BlockAck frame shall be computed as defined in Table 21" to"The RBUFCAP subfield value in the transmitted BlockAck frame shall be computed as defined in Table 21"Also change column "RBUFCAP field value" to "RBUFCAP subfield value" in Table 21 | as suggested | Accepted  |

**10.24.7.5 Generation and transmission of BlockAck frames by an HT STA or DMG STA***Modify at the end of the subclause as follow*

If an EDMG STA transmits a BlockAck frame in response to a BlockAckReq frame or an A-MPDU with
Ack Policy equal to Normal Ack (i.e. implicit block ack request) during either full-state or partial-state
operation, the EDMG STA shall calculate the Free Memory Space at the generation and transmission of
the BlockAck frame. The RBUFCAP subfield value in the transmitted BlockAck frame shall be computed as defined in Table 21. The Free Memory Space is an estimation of the amount of free memory available at the recipient to collect MPDUs at the time of and during reception of a forthcoming A-MPDU.

**Table 21 — RBUFCAP value calculation**

|  |  |  |
| --- | --- | --- |
| **Free Memory Space comparison**  | **RBUFCAP Quantity Capability (10.24.2)** | **RBUFCAP field value** |
| Free Memory Space ≥ (2 (13 + Maximum A-MPDU Length Exponent) –1) | Supported  | Receiver Buffer Empty |
| Free Memory Space ≥ (2(13 + Maximum A-MPDU Length Exponent) – 1) | Not supported  | Receiver Buffer Empty |
| Free Memory Space < (2 (13 + Maximum A-MPDU Length Exponent) –1) | Supported  | Receiver Buffer Available:Int [Free Memory Space/Buffer Unit Size] |
| Free Memory Space < (2 (13 + Maximum A-MPDU Length Exponent) –1) | Not supported  | Receiver Buffer Full |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 2276 | 10.24.7.7 | Is A-MPDU Byte Count Limit and AMPDU\_Data\_frames\_Limit only applicable to AMPDU, but not applicable to APPDU? | specify that A-MPDU Byte Count Limit and AMPDU\_Data\_frames\_Limit are applicable to byte count and frame limit before a BA is received regardless the mechanisms by which frames are aggregated |  |

**Discussion**

Definition " EDMG STA shall not transmit subsequent frames belonging to Block Ack agreement " applies to all frame sent under Block Ack agreement, it applied to A-MPDU, A-PPDU or subsequent frames sent with IFS in between.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 2277 | 10.24.7.7 | !COND1 & No\_Mem\_Kept==0 is not described in the table 22 | Chane No\_Mem\_Kept to '0 or 1' on row 3 (!COND1) | Revised Table was updated, this Option was added as first row in the relevant table |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 1108 | 10.24.7.7 | the count limit column of the last row of table 21 and the second row in table 22 should be excluded to the case 1>=RBUFCAP=<FE | Add the following 4th NOTE below the table:Equation is valid for the case 1>=RBUFCAP=<FE | RevisedTable and calculation were updated to differentiate the two cases  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 2280 | 10.24.7.7 | It is not clear what +4 means in the table 24 last column if the AMPDU byte count limit does not include delimiterIf byte count limit does not include delimeter, then some values in the last column in table 22, 23 will make the final A-MPDU size greater than the maximum allowed | Chane this sentence to "The A-MPDU Byte Count Limit does not include A-MPDU EOF padding field or A-MPDU subframes carrying Block Ack Schedule frames with EOF subfield set to 1" | Revised Calculation was revised with the assumption that MPDU size is aligned according to Delimiter requirements if needed.  |
| 2281 | 10.24.7.7 | The A-MPDU\_Data\_frames\_Limit equations in table 24 is not entirely accurate because it does not account for min MPDU Start Spacing.The equation also does not seem to cover the case for multi-TID A-MPDU that 2 or more AMPDU byte count limits are calculated from RBUFCAPs of 2 or more TIDs which are not sharing the buffer. | Specify table 24 as sentences of requirements because the equation in a table may not cover all cases | Revised Calculation was revised with the assumption that MPDU size includes the padding for MPDU Start spacing requirements if needed. |
| 2268 | 10.24.2 | "It is not clear why the support of advanced recipient memory length is tied to the RBUFCAP quatity capabilityFor example, why originator cannot use RBUFCAP with values unlimited or 0 on the last 3 rows of Table 22?" | clarify this bullet with a note | Revised Dependency was removed  |
| 2282 | 10.24.7.7 | N\_MPDUs and TotalMEM in Figure 87 are not defined | Define the variables | Revised Calculation is revised, variable were removed |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Comment**  | **Proposed change** | **Resolution**  |
| 1219 | 10.24.7.7 | "Figure 87 needs to be redrawn to provide adequate quality artwork for publication. When that happens, please replace the glyph that looks like capital lambda with something like ""and"", or explain it in the text.The figure also mixes subscript notation (BC\_p) with functional index notation (BUFsize(q)). Pick one." | As in comment. | RevisedFigure was removed and replaced with pseudocode |
| 1876 | 10.24.7.7 | Equations format | Figure 87 is not a figure. It presents some equations. The text and equations should be converted from image caption. | RevisedFigure was removed and replaced with pseudocode |
| 1983 | 10.24.7.7 | Equations format | Figure 87 is not a figure. It presents some equations. The text and equations should be converted from image caption. | RevisedFigure was removed and replaced with pseudocode |
| 2137 | 10.24.7.7 | The calculation of the A-MPDU Byte Count Limit involves multiple parameters and conditions and is extremley compex to parse. The relationship among different parameters are also unclear. This also applies to Table 23 and 24.Some examples should be provided. Also does N/A mean the value is irrelevent? | as suggested | RevisedFigure was removed and replaced with pseudocode |
| 2267 | 10.24.7.7 | The calculation of the Table 24 1st row (recipient multiple buffer unit not capable/supported) and the other rows (recipient multiple buffer unit capable/supported) are unrelated to RBUFCAP QuatityIt is not clear why RBUF Quantity capable capability affects recipient multiple buffer unit suopport of the BA agreement. Even RBUFCAP quantity not supported (by either originator or recipient), the originator can still calculate table 24 row 2,3,4 | clarify this requirement with a note | Table 24 was removed and replaced with pseudocode |
| 2278 | 10.24.7.7 | What is the A-MPDU\_Data\_frames\_Limit for (1, M, N, Yes)? | add a row in Table 24 describing the limit of this case | Figure was removed and replaced with pseudocode |
|  |  |  |  |  |

**10.24.7.7 Originator’s behaviour**

*Replace the 4th paragraph (starting from P126L17 in D0.1) through the end of the section as follow:*

An originator that is an EDMG STA shall not transmit subsequent frames belonging to Block Ack agreement at the start of transfer sequence of size greater than Flow Control Byte Count Limit per the configuration obtained during the Block Ack Agreement for the respective TID as described in Table 22 and per computation as described in section 10.24.7.7.1.

**Table 22 — Flow Control Byte Count Limit calculation at the start of a data transfer sequence**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| **According to BlockAck agreement** | **As received via BlockAck or ADDBA Response frame** | **Flow Control ByteCount Limit**  |
| Advanced RecipientMemory LengthCapable | Recipient BufferQuantityCapable | Updated RBUFCAP | No Memory Kept  |
| 0 | 0 | Receiver Buffer Full  | N/A | Zero (see NOTE1) |
| N/A | 0 | Receiver Buffer Empty  | 0 | 2 (13 + Maximum A-MPDU Length Exponent) –1) |
| 1 | 0 | N/A | 1 | 2 (13 + Advanced Recipient Memory Length Exponent) –1) |
| N/A | 1 | Receiver Buffer Available  | 0 | Receiver Buffer Available × Buffer Unit Size  |
|  |  |  |  |  |

An originator that is an EDMG STA shall not transmit subsequent frames belonging to Block Ack agreement at the middle of transfer sequence in size greater than Flow Control Byte Count Limit per the configuration obtained during the Block Ack Agreement for the respective TID as described in Table 23 and per computation as described in section 10.24.7.7.1.

**Table 23 — Flow Control Byte Count Limit calculation in the middle of a data transfer sequence**

|  |  |  |
| --- | --- | --- |
| **According to BlockAck agreement** | **As received via BlockAck or ADDBA Response frame** | **Flow Control ByteCount Limit**  |
| Recipient BufferQuantityCapable | Updated RBUFCAP |
| N/A | Receiver Buffer Full  | Zero (see NOTE1) |
| N/A | Receiver Buffer Empty  | 2 (13 + Maximum A-MPDU Length Exponent) –1) |
| 1 | Receiver Buffer Available | Receiver Buffer Available × Buffer Unit Size  |

NOTE1 – Originator may poll the responder RBUFCAP Value.

**10.24.7.7.1 Flow Control Byte Count Limit computation by EDMG Originator**

*numOfMpdusForTx* indicates the Number of pending MPDUs in TX Queue that are within the transmission window.

In case Recipient Memory Multiple Buffer Units Capability is not supported parameters *maxMpduInMem* and *mpduSplitInBuffer* are assigned with the values 255 and 1 respectivly.

Parameters *unitBufferSize, rbufcap, memoryUnitSize and maxMpduInMem, mpduSplitInBuffer* are the recent EDMG Flow Control parameters as received from a TID within the Multi TID Group and with the respective Memory Configuration Tag.

*mpduForTx[k]* contains the size of MPDU at location k in TX Queue with the padding for Minimum A-MPDU Spacing and A-MPDU delimiter alignment if required.

*FlowControlByteCountLimit* derived from Table 22 and Table 23.

int calcAggregationMemory(IN int FlowControlByteCountLimit, IN int memoryUnitSize, IN int maxMpduInMem, IN int mpduSplitInBuffer, IN int mpduForTx[], IN int numOfMpdusForTx, OUT int mpduToSend[])

{

       int           memoryToUse = FlowControlByteCountLimit;

       int           freeMemory  = memoryUnitSize;

       int           k = 0;

       int           numOfMpdusInMemoryUnit = 0;

       bool   bIsMpduInserted;

//Adding MPDUs to the queue as long as there are MPDUs in TX Queue and recipient memory buffer is not full

       while (k < numOfMpdusForTx && (memoryToUse >= mpduForTx[k]))

       {

              bIsMpduInserted = true;

// Handle the case when MPDU[k] has enough memory in one Memory Buffer Unit hence it is added to the aggregaton

              if (freeMemory >= mpduForTx[k])

              {

                      mpduToSend[k] = mpduForTx[k];

                      k++;

                      numOfMpdusInMemoryUnit++;

                      freeMemory -= mpduForTx[k];

                      memoryToUse -= mpduForTx[k];

              }

// Handle the case when MPDU[k] doesn’t have enough memory in one Memory Buffer Unit however it can be spiltted among several buffers hence it is added to the aggregaton

              else if (mpduSplitInBuffer == 1)

                                   {

                      mpduToSend[k] = mpduForTx[k];

                      k++;

                     //Calculating the free memory space from the last used Memory Buffer Unit

                      freeMemory = memoryUnitSize - ((mpduForTx[k] - freeMemory) % memoryUnitSize);

                      //The case where MPDU was placed in whole in the previuse buffer

                      if (freeMemory == memoryUnitSize)

                      {

                             numOfMpdusInMemoryUnit = 0;

                      }

                      else

                      {

                             numOfMpdusInMemoryUnit = 1;

                      }

                      memoryToUse -= mpduForTx[k];

              }

              else

              {

                      bIsMpduInserted = false;

              }

//Handle the case where the MPDU cannot be inserted to current Memory Buffer Unit, free memory is deacresed and new Memory Buffer Unit is allocated

              if ((maxMpduInMem != 255 && numOfMpdusInMemoryUnit == maxMpduInMem) ||

                      ((false == bIsMpduInserted) && (mpduSplitInBuffer == 0)))

              {

                      memoryToUse -= freeMemory;

                      freeMemory = memoryUnitSize;

                      numOfMpdusInMemoryUnit = 0;

              }

       }

       return k;

}

**SP/M:** Do you accept the resolutions given in