

Errata for “Universal Serial Bus Communications Class Subclass
Specifications for Network Control Model Devices Revision 1.1 May 22, 2025”
as of February 5, 2026.

Issue A

Related to:

Table 8-27: Capability-specific data structure for ARP offload, Table 8-28: Capability-specific data structure for NS offload

Issue:

Functions should inform the host about the number of MAC addresses the ARP and NS presence offloads support. The host may send multiple SET_EXTENDED_FEATURE requests to the function to configure multiple offloads.

Resolution Description:

Add a field to specify the number of MAC addresses the device supports in the capability specific offload structures. The host shall set a unique identifier for the SET_EXTENDED_FEATURE and GET_EXTENDED_FEATURE requests in the “wValue” field. Add text to explain the usage of the unique identifier.

Correct as shown below:

Table 8-29. Capability-specific data structure for ARP offload

| Offset | Field | Size | Value | Description |
|--------|-------------------------------|------|--------|--|
| 0 | <i>bMaxIpv4Addresses</i> | 1 | Number | Maximum number of IPv4 addresses supported for the offload. |
| 1 | <i>bMaxLocalIpv4Addresses</i> | 1 | Number | Maximum number of local IPv4 addresses. Must be smaller than or equal to <i>bMaxIpv4Addresses</i> . <i>An address is considered local if it belongs to the device managed by the NCM function. The NCM function shall only use 'Local' addresses for ARP offload. 'Other' addresses (those which are not 'Local'), shall be used by the NCM function for mDNSResponder name conflict resolution.</i> |
| 2 | <i>bMaxMacAddresses</i> | 1 | Number | Maximum number of MAC addresses supported for the offload. |

Table 8-30. Capability-specific data structure for NS offload

| Offset | Field | Size | Value | Description |
|--------|--------------------------|------|--------|---|
| 0 | <i>bMaxIpv6Addresses</i> | 1 | Number | Maximum number of IPv6 addresses supported for the offload. |

| | | | | |
|---|-------------------------------|---|--------|---|
| 1 | <i>bMaxLocalIpv6Addresses</i> | 1 | Number | Maximum number of local IPv6 addresses. Must be smaller than or equal to <i>bMaxIpv6Addresses</i> . <i>An address is considered local if it belongs to the device managed by the NCM function. The NCM function shall only use 'Local' addresses for NS offload. 'Other' addresses (those which are not 'Local'), shall be used by the NCM function for mDNSResponder name conflict resolution.</i> |
| 2 | <i>bMaxMacAddresses</i> | 1 | Number | Maximum number of MAC addresses supported for the offload. |

| bmRequestType | bRequest | wValue | wIndex | wLength | Data |
|---------------|----------------------|---|---|--------------------------|--|
| 00100001B | SET_EXTENDED_FEATURE | OffloadType Bits 0..7: Unique Identifier/Reserved Bits 8..15: NCM_PRESENCE_OFFLOAD_TYPE | Bits 0..7: NCM Communications Interface Bits 8..15: NCM_PRESENCE_OFFLOAD | Number of bytes to write | Configuration data for the specified offload (Table 8 30, Table 8 31, Table 8 32) |

The host must send a separate SET_EXTENDED_FEATURE with a unique identifier in the *wValue* for each unique MAC address that is to be offloaded for the ARP and NS offloads. The unique identifier must be an integer between 0 and *bMaxMacAddresses*-1. Bits 0-7 are reserved for the mDNS offload. If the *wValue* is not one of the defined values, or if the capability represented by *wValue* is not one of the supported capabilities, according to Table 8 25, the function shall return an error response (a STALL PID) and shall otherwise not change state.

| bmRequestType | bRequest | wValue | wIndex | wLength | Data |
|---------------|----------------------|---|---|-------------------------|--|
| 10100001B | GET_EXTENDED_FEATURE | OffloadType Bits 0..7: Unique Identifier/Reserved Bits 8..15: NCM_PRESENCE_OFFLOAD_TYPE | Bits 0..7: NCM Communications Interface Bits 8..15: NCM_PRESENCE_OFFLOAD | Number of bytes to read | Configuration data for the specified offload (Table 8 30, Table 8 31, Table 8 32) |

The host must send a separate GET_EXTENDED_FEATURE with a unique identifier in the *wValue* for each unique MAC address that is offloaded for the ARP and NS offloads. The unique identifier must be an integer between 0 and *bMaxMacAddresses*-1. Bits 0-7 are reserved for the mDNS offload. If the *wValue* is not one of the defined values, or if the capability represented by *wValue* is not one of the supported capabilities, according to **Table 8 25**, the function shall return an error response (a STALL PID) and shall otherwise not change state.

Compatibility Impact: No change in intent. Hosts that wish to support devices made before the publication of these errata may assume there is only one MAC address supported and send the SET_EXTENDED_FEATURE request with a unique identifier of 0 for each presence offload.

Issue B

Related to:

Table 4-6: Meta-info structure for receive NDPX, Table 4-3: Meta-info structure for transmit NDPX

Issue:

Functions should inform the host about the coalesced segment count for receive coalesced packets. For TCP packets, the function should include the duplicate acknowledgement count and for UDP packets, the function should include the coalesced segment size in the meta-info structure for receive NDPX.

Resolution Description:

Add two 16-bit fields to specify the coalesced segment count and either duplicate acknowledgement count or coalesced segment size, depending on the layer 4 protocol. Extend the receive meta-info structure size to 32 bytes. Add reserved fields to the transmit meta-info structure to keep it the same size as receive. Update Table 4-2 accordingly.

Correct as shown below:

Table 4-6: Meta-info structure for receive NDPX

| Receive Datagram Meta-info | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------------------|--|----|----|----|----|----|----|----|---------|----|----|----|--------|----|----|----|-----------------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | Flags | | | | | | | | RSSType | | | | Length | | | | | | | | | | | | | | | | | | | |
| 1 | RxEErrors | | | | | | | | | | | | | | | | RxStatus | | | | | | | | | | | | | | | |
| 2 | Reserved | | | | | | | | | | | | | | | | VLAN | | | | | | | | | | | | | | | |
| 3 | RSSHash | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | DuplicateAckCount/CoalescedSegmentSize | | | | | | | | | | | | | | | | CoalescedSegmentCount | | | | | | | | | | | | | | | |
| 5 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| DWORD | Bits | Field | Value | Description |
|-------|--------|----------|-----------|---|
| 0 | [19:0] | Length | Number | Total length of the datagram to which this Meta-info applies Current implementations are not expected to use more than 17 bits, due to the packet length limits of TCP/IP stacks. |
| ... | | | | |
| 1 | [15:0] | RxStatus | Bit field | Receive Status Flags (16 bits) |

| DWORD | Bits | Field | Value | Description |
|-------|---------|--|--------|--|
| | | | | A value of 1 in the following fields denotes that the device performed the specific offload/feature on this datagram 0 – L2 FCS (MAC CRC) stripping 1 – L3 Header Checksum (IPv4 Checksum) verification 2 – L4 Checksum (TCP/UDP Checksum) verification 3 – VLAN Tag stripping 4 – Large Receive Offload (Receive coalescing) 5 – CoalescedSegmentCount and DuplicateAckCount/CoalescedSegmentSize Large Receive Offload information available 15:6 – Reserved |
| ... | | | | |
| 3 | [31:0] | RSSHash | Number | 32-bit RSS Hash Value as computed over the packet This value is valid only if <i>RSSType</i> is not zero and not one of the reserved values. The fields over which the RSS hash is computed are described in Table 10- 1 . |
| 4 | [15:0] | CoalescedSegmentCount | Number | Number of TCP/UDP segments coalesced This value is valid only if the Large Receive Offload receive status flag is set. |
| 4 | [31:16] | DuplicateAckCount/ CoalescedSegmentSize | Number | For TCP packet: number of duplicate acknowledgements coalesced For UDP packet: The size of the UDP data segments that were coalesced, in bytes This value is valid only if the Large Receive Offload receive status flag is set. |
| 5-7 | [31:0] | Reserved | 0 | Should be set to 0. |

Table 4-3: Meta-info structure for transmit NDPX

| Transmit Datagram Meta-info | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------------------------|----------|----|----|----|----|----|----|----|----------|----|----|----|--------|----|----|----|-----------|----|----|----|----|----|---|---|----------|---|---|---|---|---|---|---|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | Flags | | | | | | | | Reserved | | | | Length | | | | | | | | | | | | | | | | | | | |
| 1 | MSS | | | | | | | | | | | | | | | | TxRequest | | | | | | | | | | | | | | | |
| 2 | Reserved | | | | | | | | L3Length | | | | | | | | VLAN | | | | | | | | | | | | | | | |
| 3 | Reserved | | | | | | | | | | | | | | | | L2Length | | | | | | | | L4Length | | | | | | | |
| 4 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| DWORD | Bits | Field | Value | Description |
|-------|--------|--------|--------|---|
| 0 | [19:0] | Length | Number | Total length of the datagram to which this Meta-info applies Current implementations are not expected to use more than 17 bits, due to the packet length limits of TCP/IP stacks. |

| DWORD | Bits | Field | Value | Description |
|-------|---------|----------|-------|---------------------|
| ... | | | | |
| 3 | [31:15] | Reserved | 0 | Should be set to 0. |
| 4-7 | [31:0] | Reserved | 0 | Should be set to 0. |

Table 4-2: Extended NCM Datagram Pointer (NDPX)

| Offset | Field | Size | Value | Description |
|--------|------------------|------|--|--|
| 0 | dwSignature | 4 | Number (0x7478636E or 0x7278636E) | Signature of this Extended NDP. This is transmitted in little-endian form. For a transmit NDPX (host to device): 0x6E, 0x63, 0x78, 0x74 or as the character sequence "ncxt". For a receive NDPX (device to host): 0x6E, 0x63, 0x78, 0x72 or as the character sequence "ncxr". |
| 4 | dwNextNdpOffset | 4 | Number | Byte Offset, in little endian, of the next Extended NDP. This value must be a multiple of 4, and ≥ 44 , because it is counted from byte 0 of this NDPX. A value of zero denotes that this is the last Extended NDP within an NTB. |
| 8 | dwDatagramOffset | 4 | Number | Byte Offset, in little endian, of the datagram described by this Extended NDP. The offset is from byte zero of this Extended NDP, and so must be greater than 44. A value of zero denotes that this Extended NDP does not contain a datagram and shall only be used for the last Extended NDP in an NTB. If the value of this field is zero, then dwNextNdpOffset shall also be zero. |
| 12 | stMetaInfo | 32 | Structure | Extended Datagram Meta-info structure. This field shall contain meta-information for the datagram transmitted by this Extended NDP. The content of the meta-info for a transmit NDPX is described in Section 4.4.1. The content of the meta-info for a receive NDPX is described in Section 4.4.3. |

Issue C

Related to:

Table 7 1: Networking Control Model Requests

Issue:

Some of the optional requests of NCM 1.0 are not relevant when an NCM 1.1 function operates in Extended capability mode, either because they are not applicable, or because NCM 1.1 provides other means for configuring these parameters.

Resolution Description:

Clarify which requests are for NCM 1.0 mode only.

Correct as shown below:

Table 7-1 lists the class-specific requests that are valid for an NCM Communications Interface. This table includes those defined in [USBECM12]. Commands marked as “required” must be implemented by any conforming NCM function. Commands marked as “optional” must be implemented in some circumstances; see the Reference section for each command for details. Commands marked as “1.0 only” are supported by NCM 1.1 functions operating in NCM 1.0 mode, but not in NCM 1.1 (extended) mode. For a description of NCM 1.1 specific requests see Section 8.2.

| Request | Description | Req'd/Opt | reference |
|---|---|---------------------|------------|
| <i>SendEncapsulatedCommand</i> | Issues a command in the format of the supported control protocol. The intent of this mechanism is to support networking functions (e.g. host-based cable modems) that require an additional vendor-defined interface for media specific hardware configuration and management. | Optional | [USBCDC12] |
| <i>GetEncapsulatedResponse</i> | Requests a response in the format of the supported control protocol. | Optional | [USBCDC12] |
| <i>SetEthernetMulticastFilters</i> | Controls the receipt of Ethernet frames that are received with “multicast” destination addresses. | Optional | [USBECM12] |
| <i>SetEthernetPowerManagement-PatternFilter</i> | Some hosts are able to conserve energy and stay quiet in a “sleeping” state while not being used. NCM functions may provide special pattern filtering hardware that enables the function to wake up the attached host on demand when something is attempting to contact the host (e.g. an incoming web browser connection). This command allows the host to specify the filter values that detect these special frames. | Optional (1.0 only) | [USBECM12] |
| <i>GetEthernetPowerManagement-PatternFilter</i> | Retrieves the status of the above power management pattern filter setting | Optional (1.0 only) | [USBECM12] |
| <i>SetEthernetPacketFilter</i> | Controls the types of Ethernet frames that are to be received via the function. | Optional | [USBECM12] |
| <i>GetEthernetStatistic</i> | Retrieves Ethernet statistics such as frames transmitted, frames received, and bad frames received. | Optional | [USBECM12] |
| <i>GetNtbParameters</i> | Requests the function to report parameters that characterize the Network Control Block | Required | 7.2.1 |
| <i>GetNetAddress</i> | Requests the current EUI-48 network address | Optional | 7.2.2 |

CDC NCM 1.1 Errata 1

| Request | Description | Req'd/Opt | reference |
|---------------------------|---|---------------------|-----------|
| <i>SetNetAddress</i> | Changes the current EUI-48 network address | Optional | 7.2.3 |
| <i>GetNtbFormat</i> | Get current NTB Format | Optional (1.0 only) | 7.2.4 |
| <i>SetNtbFormat</i> | Select 16 or 32 bit Network Transfer Blocks | Optional (1.0 only) | 7.2.5 |
| <i>GetNtbInputSize</i> | Get the current value of maximum NTB input size | Required | 7.2.6 |
| <i>SetNtbInputSize</i> | Selects the maximum size of NTBs to be transmitted by the function over the bulk IN pipe. | Required | 7.2.7 |
| <i>GetMaxDatagramSize</i> | Requests the current maximum datagram size (see 3.6 and 4.7) | Optional | 7.2.8 |
| <i>SetMaxDatagramSize</i> | Sets the maximum datagram size to a value other than the default (see 3.6 and 4.7) | Optional | 7.2.9 |
| <i>GetCrcMode</i> | Requests the current CRC mode | Optional (1.0 only) | 7.2.10 |
| <i>SetCrcMode</i> | Sets the current CRC mode | Optional (1.0 only) | 7.2.11 |

Issue D

Related to:

Table 8-36. Capability-specific data structure for transmit segmentation offload

Issue:

Clarification is required for “UDP segmentation” terminology, which historically could refer to two different offloads – at OSI layer 4 (segmentation) and OSI layer 3 (fragmentation).

Resolution Description:

Add a clarifying footnote.

Correct as shown below:

The segmentation offloads that an NCM function may support are performed at layer 4. Each segmented packet will contain a layer 4 (TCP or UDP) header, a layer 3 (IPv4 or IPv6) and a layer 2 (Ethernet) header. The IP fragment offset is not used and shall be set to 0 for all packets. This is not to be confused with the (mostly deprecated) UDP/IP fragmentation offload, where the layer 4 header is present only in the first packet (fragment), and the fragment offset field in the IP header is incremented with the payload size for each subsequent fragment.

Issue E

Related to:

4.4.1 - Transmit meta-info structure, 4.4.2 - Transmit parameters combinations and dependencies

Issue:

Clarification is required about the dependencies between the various transmit offloads, especially L2 FCS (CRC) offload.

Resolution description:

Clarify that L2 CRC offload should be enabled (set to 1) in the typical case. Explain that if L2 CRC offload is disabled, other offloads shall be disabled (reserved).

Correct as shown below:

in Table 4-3: Meta-info structure for transmit NDPX:

| | | | | |
|---|--------|-----------|-----------|---|
| 1 | [15:0] | TxRequest | Bit field | Transmit Request Flags (16 bits) A value of 1 in the following fields denotes the Host request for Device to perform the requested offload/feature on this datagram 0 – Append L2 FCS (MAC CRC). Normally, datagrams are sent by upper software layers without L2 FCS attached, so this bit should be set. 1 – Insert L3 Header Checksum (IPv4 Checksum) 2 – Insert L4 Checksum (TCP/UDP Checksum) 3 – Insert VLAN Tag 4 – Perform Large Send Offload (TCP/UDP Segmentation) 15:5 – Reserved |
|---|--------|-----------|-----------|---|

Table 4-4: Dependencies between TxRequest Insert L2 FCS and other fields

| | |
|--|---|
| L2 FCS Disabled (TxRequest[0]=0) | The datagram shall include the correct FCS (CRC) value as computed by the software. <i>Length</i> refers to the length of the datagram, including the FCS. All other offload bits in <i>TxRequest</i> are reserved (set to 0). <i>VLAN</i> , <i>MSS</i> and <i>L2/L3/L4 Length</i> fields are reserved (set to 0). <i>Flags</i> field is reserved (set to 0). |
| L2 FCS Enabled (TxRequest[0]=1) | The datagram shall not include the FCS (CRC). <i>Length</i> refers to the length of the datagram without the 4 FCS (CRC) bytes that shall be appended by the NCM function. Other offloads may be set as described below. |

Issue F

Related to:

8.3.1 - Medium Handling, 8.4 - NCM 1.1 Notifications

Issue:

NCM 1.1 unified medium notification cannot always be mandatory – because an NCM 1.1 function may not, in the general case, support the Medium Handling extended feature. Additionally, the existing wording is confusing since an explicit enable bit for the notification is present in the SET_EXTENDED_FEATURE request for Medium Handling.

Resolution description:

Clarify that *UnifiedMediumNotification* is optional. Remove any wording referring to it being mandatory. Explain that it may be enabled via the SET_EXTENDED_FEATURE request. Explain that NCM 1.0 notifications shall be issued if *UnifiedMediumNotification* is disabled.

Correct as shown below:

8.3.1 Medium Handling (footnote 2):

See 8.4.1 for description of the UnifiedMediumNotification structure.

Before Table 8-11:

If the medium specified in the *Data* field is not the supported medium, or if the medium structure attempts to enable an unsupported medium feature or capability, the function shall return an error response (a STALL PID) and shall otherwise not change state.

8.4 NCM 1.1 Notifications

When running in NCM 1.0 mode, an NCM 1.1 function shall support all standard NCM notifications as described in section 7.4. When running in NCM 1.1 extended capability mode, an NCM function may support the extended notifications below:

Table 8-48: NCM 1.1 Extended Notifications

| Notification | Description | Req'd/Opt | reference |
|----------------------------------|-------------------------------|-----------|-----------|
| <i>UnifiedMediumNotification</i> | Reports link status and speed | Optional | 8.4.1 |

The medium structure included in the data of this notification command is the same structure that would be returned in response to a GET_EXTENDED_FEATURE request for *NCM_MEDIUM_HANDLING*. Refer to section 8.3.1 for details.

The unified medium notification is supported if the function supports the Medium Handling extended feature, and bit D0 in the *bmFeatureFlags* field of the *NCM_MEDIUM_HANDLING* extended feature capabilities structure is set.

The host may enable or disable the unified medium notification, by sending a *SET_EXTENDED_FEATURE* request with *bNcmFeatureSelector* set to *NCM_MEDIUM_HANDLING*, and the appropriate value of bit D0 in the *bmFeatureFlags* field. When the unified medium notification is disabled, the NCM function shall issue the NCM 1.0 notifications as described in section 7.4.

Issue G

Related to:

8.3.1 - Medium Handling

Issue:

The existing definition of the GET_EXTENDED_FEATURE request offers no way to distinguish between retrieving the last configuration passed by the host and retrieving the active device state (which may be different from the last configuration due to operational parameters of the device or the link partner (if present).

Resolution description:

Add Table 8-13: NCM_MEDIUM_QUERY_TYPE Codes. Explain the two query types (last configuration vs current status). Explain that wValue field in the request is used to determine the query type. Clarify the contents of the data structure returned in each case.

Correct as shown below:

The host may send a GET_EXTENDED_FEATURE request with *bNcmFeatureSelector* set to NCM_MEDIUM_HANDLING to query either the current medium status or the last configuration set by the host. The actual status may differ from the requested configuration, as certain medium features advertised by the host depend also on the capabilities of the link partner. For example, in case of link speed auto-negotiation, the host may pass a combination of several possible speed settings, which will be resolved to a single active link speed/duplex in the process of auto-negotiation with the link partner.

Table 8-13: NCM_MEDIUM_QUERY_TYPE Codes

| Value | Name | | Description | | |
|---------------|------------------------------|-----------------------|--|-------------------------|------------------|
| 00h | NCM_MEDIUM_GET_CONFIGURATION | | Retrieve the last configuration issued by the host | | |
| 01h | NCM_MEDIUM_GET_STATUS | | Retrieve the status of the medium | | |
| 02h-FFh | Reserved | | Reserved for future use | | |
| bmRequestType | bRequest | wValue | wIndex | wLength | Data |
| 10100001B | GET_EXTENDED_FEATURE | NCM_MEDIUM_QUERY_TYPE | Bits 0..7: NCM Communications Interface Bits 8..15: NCM_MEDIUM_HANDLING | Number of bytes to read | Medium structure |

Table 8-14. GET_EXTENDED_FEATURE Medium Structure for Virtual Wire (Medium Type 00h)

| Offset | Field | Size | Value | Description |
|--------|-----------------------|------|-----------|--|
| 0 | <i>bMediumType</i> | 1 | Type code | NCM_MEDIUM_VIRTUAL_WIRE |
| 1 | <i>bmFeatureFlags</i> | 1 | Bit field | D7:D1 – Reserved D0 – Medium Notification Enabled |

| Offset | Field | Size | Value | Description |
|--------|---------------------------|------|---------------------------|---|
| 2 | <i>dwSpeed</i> | 4 | Number (little endian) | A value of all zeros – Medium is disabled A value of all ones – Medium is enabled All other values are reserved and shall be interpreted by the Host as the medium being enabled. |
| 6 | <i>bmMediumParameters</i> | 4 | Bit field | D31:D0 - Reserved (set to 0) |

The medium structure for Virtual Wire is the same, regardless of NCM_MEDIUM_QUERY_TYPE, as there is no difference between last configuration set and current medium state.

Replacing Table 8-14:

Table 8-15. GET_EXTENDED_FEATURE Medium structure for Ethernet (Medium Type 01h) and NCM_MEDIUM_QUERY_TYPE == NCM_MEDIUM_GET_CONFIGURATION

| Offset | Field | Size | Value | Description |
|--------|---------------------------|------|---------------------------|---|
| 0 | <i>bMediumType</i> | 1 | Type code | NCM_MEDIUM_ETHERNET |
| 1 | <i>bmFeatureFlags</i> | 1 | Bit field | D7:D1 – Reserved (set to 0) D0 – Unified Medium Notification Enabled |
| 2 | <i>dwSpeed</i> | 4 | Number (little endian) | A value of all zeros – No link A value of all ones – Link is up in auto-negotiation mode, according to the enabled speed bits in <i>bmMediumParameters</i> . Otherwise - Medium speed in Kbps (kilobits per second). |
| 6 | <i>bmMediumParameters</i> | 4 | Bit field | Specifies the active capabilities. A value of '1' in the corresponding bit indicates that a capability is active. A value of '0' indicates that it is inactive. The interpretation of the bits is according to the value of <i>bmMediumParameters</i> in Table 8-10. The values represent the previous configuration passed by the host via the SET_EXTENDED_FEATURE command. |

Table 8-16. GET_EXTENDED_FEATURE Medium structure for Ethernet (Medium Type 01h) and NCM_MEDIUM_QUERY_TYPE == NCM_MEDIUM_GET_STATUS

| Offset | Field | Size | Value | Description |
|--------|---------------------------|------|---------------------------|---|
| 0 | <i>bMediumType</i> | 1 | Type code | NCM_MEDIUM_ETHERNET |
| 1 | <i>bmFeatureFlags</i> | 1 | Bit field | D7:D1 – Reserved (set to 0) D0 – Unified Medium Notification Enabled |
| 2 | <i>dwSpeed</i> | 4 | Number (little endian) | Zero – no link Non-zero – Link speed in Kbps (kilobits per second). The speed shall be equal to the value implied by the speed selection bits in <i>bmMediumParameters</i> . |
| 6 | <i>bmMediumParameters</i> | 4 | Bit field | Specifies the active capabilities. A value of '1' in the corresponding bit indicates that a capability is active. A value of '0' indicates that it is inactive. The interpretation of the bits is according to the value of <i>bmMediumParameters</i> in Table 8-10. The values represent the actual status of the medium. In particular, at most one of the bits D13:D0 shall be set, indicating the active link speed. No bit shall be set if there is no link. |

Issue H

Related to:

9.4 Using the Interface Association Descriptor

Issue:

The Interface Association Descriptor (IAD) is optional in NCM 1.0, however, in practice, hosts expect it, and do not operate correctly if it is not present.

Resolution description:

Mandate IAD at least for NCM 1.1-compliant functions to reduce interoperability issues

Correct as shown below:

USB Device containing NCM functions may include an Interface Association Descriptor (IAD) to indicate that the Communication Class interface and the Data Class Interface are to be treated as a single function. (This is an alternative to using the WMC WHCM Union descriptor.) The IAD is mandatory for an NCM function complying with version 1.1 of this specification.