Universal Serial Bus
Communications Class
Subclass Specification for
Mobile Broadband Interface Model

Revision 1.0
Errata-1

May 1, 2013

Revision History

Rev	Date	Filename	Comments
1.0	2011-10-25	MBIM v1.0	Initial release This specification was developed as USB Communications Class Subclass Specifications for Network Control Model (NCM) Devices v2.0 and subsequently renamed as Mobile Broadband Interface Model (MBIM) v1.0
Errata-1	2012-12-06	MBIM v1.0 – errata-1	Corrected some errors and added some clarifications. For details see Errata for “MBIM 1.0”

Please send comments or questions to: admin@usb.org

Contributors

Björn Boden	Ericsson AB
Patrik Olesen	Ericsson AB
Ulrich Leucht-Roth	Intel
Ygal Blum	Jungo
Yoav Nissim	Jungo
Terry Moore	MCCI
Greg Scaffidi	MCCI
Mats Webjörn	MCCI
Randy Aull	Microsoft
Mitesh Desai	Microsoft
Nazan Kurt	Microsoft
Brian Lieuallen	Microsoft
Srinivasan Malayala	Microsoft
Ryan Menezes	Microsoft
Gabriel Montenegro	Microsoft
Trideepraj Roychoudhury	Microsoft
Mukund Sankaranarayan	Microsoft
Vineet Venugopal	Microsoft
Kenji Oguma	NEC Casio
Eugene Gryazin	Nokia

Roman Pak	Nokia
Richard Petrie	Nokia
Vladimir Semenyuk	Smith Micro Software
Morten Christiansen	ST-Ericsson
Alexey Orishko	ST-Ericsson
Brian Duddie	Qualcomm
Liron Manor	Qualcomm

Table of Contents:

List of Figures

List of Tables

1 OVERVIEW

This specification is a protocol by which USB hosts and Mobile Broadband devices can efficiently exchange control commands and data frames.

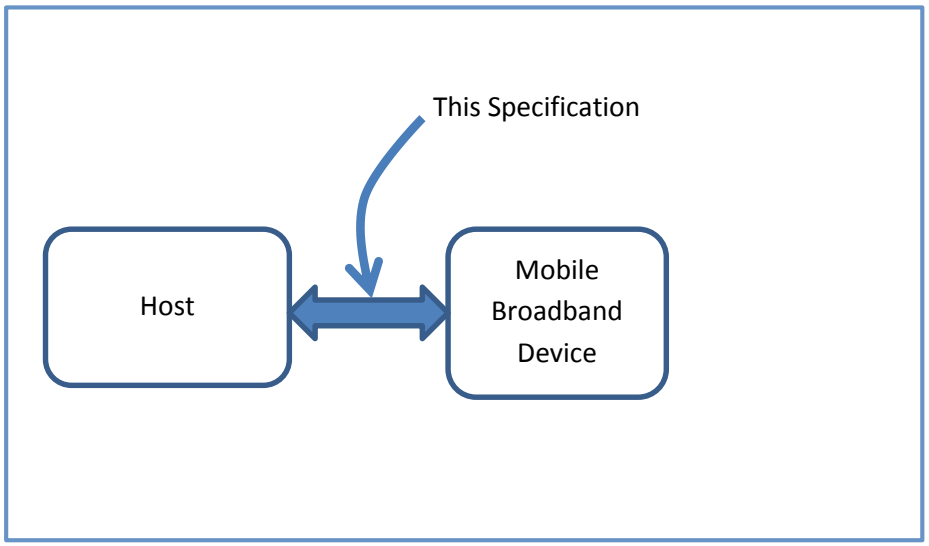

Figure 1: Specification scope

The Mobile Broadband Interface Model (MBIM) is based on the Communications Class Network Control Model (NCM 1.0) Subclass.

MBIM extends the NCM as a protocol between host and USB devices. USB devices referred in this specification are defined as capable of transmitting packet data ("IP" – Internet Protocol) over the cellular technology standards defined by 3GPP or 3GPP2.
This specification defines control commands, raw IP usage and Communications Device Class subclass specification for Network Control Model [USBNCM10] for multiple IP packet aggregations into a single USB transfer.

1.1 TERMS AND ABBREVIATIONS

Table 1-1: Terms and Abbreviations

AKA	Authentication and Key Agreement
APN	Access Point Name

CDC	Communications Devices Class specification [USBCDC12]
CID	Command Identifier
Downlink	Direction from mobile network towards the function
DSS	Device Service Stream
ECM	Ethernet Control Model (ECM) [USBECM12]
IAD	Interface Association Descriptor
IMSI	International Mobile Subscriber Identity
HLR	Home Location Register
MB	Mobile Broadband
MBIM	Mobile Broadband Interface Model. [Formerly called NCM 2.0.]

MNO	Mobile Network Operator One essential characteristic that defines an MNO is that it must have acquired a radio spectrum license from the government before it can offer mobile phone service within a country. The precise spectrum obtained depends on the type of mobile phone technology the operator intends to deploy.
MSB	Most Significant Bit (e.g., in a field)
Multi-mode	Applied to a function or to a device that has both GSM and CDMA capabilities, e.g. a device that can seamlessly swap between GSM and CDMA for a single provider.
MVNO	Mobile Virtual Network Operator Provides the same kind of services as an MNO but are dependent on an MNO to provide the radio network.
NA	Not applicable
NCM or NCM 1.0	Network Control Mode [USBNCM10]

NCM/MBIM function	A function that supports both NCM 1.0 and MBIM operation modes, as described in section 3.2.
OTA	Over-the-air
PDU	Protocol Data Unit
PLMN	Public land mobile network
PPL	Preferred provider list
PRL	Preferred roaming list
Provider	Also known as MNO and MVNO
RSSI	Received signal strength indication
R-UIM	Removable User Identity Module A card developed for CDMA handsets, equivalent to the GSM SIM card. Sometimes referred to as U-RIM.
SC	Service Center
SMS	Short Message Service
SMSC	Short Message Service Center
STK	SIM Application Toolkit

STK PAC	STK Proactive Command
Transaction	A message sent by the host and the corresponding response from the function.
Uplink	Direction from the function towards the mobile network
USB	Universal Serial Bus
USSD	Unstructured Supplementary Service Data
U-RIM	See R-UIM
VLR	Visitor Location Register

This specification defines the 1.0 version of MBIM. 'NCM' is used interchangeably with 'NCM 1.0'.

1.2 SCOPE

This document specifies a new control and data plane for mobile broadband based networking devices, as a subclass based on the Universal Serial Bus Class Definitions for Communications Devices specification [USBCDC12]. This specification defines the following material applicable to MBIM functions:

- The class-specific contents of standard descriptors.
- Additional class-specific descriptors.
- Required interface and endpoint structure.
- Class-specific commands.
- Class-specific notifications or events.
- The format of data that is exchanged between device and host.
- The required operational behavior.

Behavior that is required of all USB devices and hosts is defined by [USB20] and [USB30]. Behavior that is required of all Communications devices is defined by [USBCDC12]. Some commands and notifications may be based on material defined in [USBNCM10].

This specification imposes requirements on devices not on hosts. Accordingly, normative language throughout (see section 1.4) applies strictly to devices and not to hosts.

1.3 OTHER USB NETWORKING SPECIFICATIONS

At time of writing, four other USB networking subclasses were defined:

- Ethernet Control Model (ECM) [USBECM12]
- Ethernet Emulation Model (EEM) [USBEEM10]
- ATM Networking Control Model [USBATM12]
- CDC NCM Subclass Revision [USBNCM10]

ECM and NCM are both applicable to IEEE 802.3 type Ethernet networking functions that can carry IP traffic to an external network. ECM was designed for USB full speed devices, especially to support DOCSIS 1.0 Cable Modems. Although ECM is functionally complete, it does not scale well in throughput or efficiency to higher USB speeds and higher network speeds. NCM draws on the experience gained from ECM implementations, and adjusts the data transfer protocol to make it substantially more efficient. EEM is intended for use in communicating with devices, using Ethernet frames as the next layer of transport. It is not intended for use with routing or Internet connectivity devices, although this use is not prohibited. [USBATM12] is applicable to USB-connected devices like ADSL modems which expose ATM traffic directly. Rather than transporting Ethernet frames, [USBATM12] functions send and receive traffic that is broken up into ATM cells and encoded using AAL-2, AAL-4 or AAL-5. Although some of the insights that led to the design of NCM came from experience with [USBATM12] implementations, the two specifications address substantially different target applications.

Although NCM was targeted for high-speed network attachments like HSPA and LTE, it lacks the definition of control commands and has packet (dis)assembly overhead to carry the raw IP encapsulated in an Ethernet frame. Primarily, NCM is based on 802.3 frames which doesn't apply to mobile broadband devices covered in this specification (there are no MAC addresses nor meaning to the fields in an Ethernet frame).

MBIM addresses these shortcomings in NCM.

1.4 EDITORIAL NOTES

In this specification, the words 'shall' and 'must' are used for mandatory requirements, the word 'should' is used to express recommendations and the word 'may' is used for options.

In some cases, text from [USBNCM10] is repeated or referred to for clarity. In such cases, the MBIM specification shall be treated as the controlling document.

2 REFERENCES

[PCI-Express]	PCI Express, Mini Card Electromechanical Specification, Revision 1.2, October 26, 2007.
[USB20]	Universal Serial Bus Specification, revision 2.0. http://www.usb.org
[USB30]	Universal Serial Bus Specification, revision 3.0. http://www.usb.org. Unless otherwise specified, any reference to [USB30] includes [USB20] by reference, especially when referring to full- and high-speed devices.
[USBATM12]	USB Subclass Specification for ATM Control Model, Revision 1.2 http://www.usb.org
[USBCDC12]	Universal Serial Bus Class Definitions for Communications Devices, Revision 1.2. http://www.usb.org.
[USBECM12]	USB Subclass Specification for Ethernet Control Model, Revision 1.2 http://www.usb.org
[USBEEM10]	USB Subclass Specification for Ethernet Emulation Model, Revision 1.0 http://www.usb.org
[USBIAD10]	USB Interface Association Descriptor ECN in [USB20] http://www.usb.org
[USBWMC11]	Universal Serial Bus Subclass Specification for Wireless Mobile Communications, Version 1.1. http://www.usb.org

[USBNCM10]	Universal Serial Bus Subclass Specification for Network control Model (NCM), Version 1.0 http://www.usb.org
[3GPP1111]	3GPP TS 24.11; Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) Interface; Mobile Equipment (SIM - ME) interface (Release 1999).
[3GPP31102]	3GPP TS 31.102; Characteristics of the Universal Subscriber Identity Module (USIM) application (Release 9).
[3GPP27005]	3GPP TS 27.005; Use of Data Terminal Equipment - Data Circuit terminating Equipment (DTE - DCE) interface for Short Message Service (SMS) and Cell Broadcast Service (CBS)
[3GPP24008]	3GPP TS 24.008; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3.
[3GPP23040]	3GPP TS 23.040; Technical realization of the Short Message Service (SMS).
[3GPP23038]	3GPP TS 23.038; Technical Specification Group Terminals; Alphabets and language-specific information.
[3GPP22090]	3GPP TS 22.090; Unstructured Supplementary Service Data (USSD); Stage 1.
[3GPP33402]	3GPP TS 33.402; System Architecture Evolution (SAE); Security aspects of non-3GPP accesses.

[RFC 4186]	Haverinen, H., Ed., and J. Salowey, Ed., "Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)", RFC 4186, January 2006.
[RFC 4282]	Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.
[RFC 5448]	Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA'), May 2009
[MBIMRegistry]	MBIM Registry at www.usb.org For registration requests, please contact admin@usb.org
[ETSITS102220920]	Smartcards; ETSI numbering system for telecommunication application providers (Release 9), ETSI TS 102 220 V9.2.0
[ETSITS102223900]	Smart Cards; Card Application Toolkit (CAT) (Release 9), ETSI TS 102 223 V9.0.0

3 COMPATIBILITY WITH NCM 1.0

3.1 SUMMARY OF DIFFERENCES BETWEEN MBIM AND NCM 1.0

The largest deviation from NCM 1.0 [USBNCM10] is that devices transfer raw IP packets instead of packets with 802.3 headers. The Identification of raw IP packets is described in section 7.1 on Data formatting. The following table shows the changes for the class-specific request that is reused from [USBNCM10].

Table 3-1: NCM 1.0 request applicability to MBIM devices

Request	Changes from NCM 1.0	Comment
SendEncapsulatedCommand	Yes	Required for MBIM functions
GetEncapsulatedResponse	Yes	Required for MBIM functions
GetNtbParameters	No	
GetNtbFormat	No	
SetNtbFormat	No	
GetNtbInputSize	No	
SetNtbInputSize	No	
GetMaxDatagramSize	No	
SetMaxDatagramSize	No	

3.2 IMPLEMENTING FUNCTIONS THAT SUPPORT NCM 1.0 AND MBIM

For backwards compatibility, a device vendor may need to build a device that will work with hosts that support only NCM 1.0, and with hosts that support MBIM. Although a complete list of approaches is beyond the scope of this specification, the following approach is recommended. In the remainder of this section, “shall” or “must” are to be interpreted as conditionally mandatory requirements: if the device implements the recommendation in this section, the device must follow the approach given here, or the approach is not guaranteed to work. In addition, compliance test suites may enforce these requirements as part of their test process, if they detect a device that seems to follow this recommendation.

3.2.1 OVERVIEW

Functions that implement both NCM 1.0 and MBIM (an "NCM/MBIM function") according to this recommendation shall provide two alternate settings for the Communication Interface. Alternate setting 0, and the associated class and endpoint descriptors, shall be constructed according to the rules given for the Communication Interface in section 5 of [USBNCM10]. Alternate setting 1, and the associated class and endpoint descriptors, shall be constructed according to the rules given in section 6 (USB Device Model) of this specification.

When alternate setting 0 of the Communication Interface is selected, the function shall operate according to the rules given in [USBNCM10]. In particular, NTBs shall transport Ethernet frames, not IP datagrams. When alternate setting 1 of the Communication Interface is selected, the function shall operate according to the rules given in this specification.

For some hosts, devices with NCM 1.0 and MBIM functions may have to implement additional device-level features in order to take maximum advantage of the features built into the host.

3.2.2 DESCRIPTOR REQUIREMENTS

3.2.2.1 INTERFACE ASSOCIATION DESCRIPTOR

If an interface association descriptor is used, its interface class, subclass, and protocol codes shall match those given in alternate setting 0 of the Communication Interface.

3.2.2.2 COMMUNICATION INTERFACE ALTERNATE SETTING 0

The function shall have two interface descriptor bundles for the Communication Interface. The first bundle shall start with an interface descriptor. The descriptor shall have bAlternateSetting == 0, bInterfaceClass == 02h, bInterfaceSubClass == 0Dh, and bInterfaceProtocol == XXh.

This descriptor is followed by the required descriptors from section 5.2 of [USBNCM10], and optional descriptors as needed. The collection of descriptors ends with an endpoint descriptor (possibly followed by an endpoint companion descriptor) for the NCM1.0 notification pipe.

3.2.2.3 COMMUNICATION INTERFACE ALTERNATE SETTING 1

After the Communication Interface, but before the first Data Interface alternate setting, the function shall have an additional interface descriptor. bInterfaceNumber for the interface shall be the same as that used in the interface descriptor given in 3.2.2.1 above. bAlternateSetting shall be 01h. The remaining fields of the descriptor shall follow the requirements given in section 6 (USB Device Model) of this specification. In particular, bInterfaceClass shall be 02h, and bInterfaceSubClass and bInterfaceProtocol shall be as specified in Table 6-1: Salient fields of Communication Class Interface Descriptor.

This descriptor is followed by the required descriptors (and optional descriptors as needed) from section 6 (USB Device Model) of this specification. The collection of descriptors ends with an endpoint descriptor (possibly followed by an endpoint companion descriptor) for the MBIM notification pipe.

Note that there's no requirement that wMaxPacketSize for the interrupt pipe be the same in alternate setting 1 as in alternate setting 0.

3.2.2.4 DATA CLASS

Functions that implement both NCM 1.0 and MBIM (an "NCM/MBIM function") according to this recommendation shall provide three alternate settings for the Data Class Interface. Accordingly, the data class descriptors shall have three alternate settings. The first two alternate settings shall be formatted as given in [USBNCM10] section 5.3. The third shall be almost identical to the second setting with the sole difference being that the bInterfaceProtocol shall be set to 02h.

As for [USBNCM10], the first alternate setting (the default interface setting, alternate setting 0) shall include no endpoints and therefore no networking traffic can be exchanged when the default interface setting is selected. The second and third alternate settings (alternate settings 1 and 2) are used for normal operation, and shall include one bulk IN endpoint and one bulk OUT endpoint.

For Communication Interface setting 0 the only valid settings for the data interface is setting 0 and 1, i.e., [USBNCM10] compatible mode.

For Communication Interface setting 1 the only valid settings for the data interface is setting 0 and 2, i.e., MBIM mode.

For any other combination the result is unknown.

3.2.2.5 FUNCTIONAL DESCRIPTORS FOR NCM AND MBIM FUNCTIONS

Table 3-2: NCM/MBIM Communication Interface Descriptor Requirements

Descriptor	Description	Required/Opt	Order	Reference
HEADER	CDC Header functional descriptor	Required	First	[USBCDC12], 5.2.3.1
UNION	CDC Union functional descriptor	Required	Arbitrary	[USBCDC12] 5.2.3.2
Ethernet	CDC Ethernet Networking Functional Descriptor	Required	Arbitrary	[USBECM12], 5.4
NCM	NCM Functional Descriptor	Required	Arbitrary	[USBNCM10] 5.2.1

Descriptor	Description	Required/Opt	Order	Reference
COMMAND SET	Command Set Functional Descriptor	Required if NCM Communications Interface bInterfaceProtocol is 0FEh	Arbitrary	[USBNCM10] 5.2.2
COMMAND SET DETAIL	Command Set Detail Functional Descriptor	Optional if Command Set Functional Descriptor is present; otherwise prohibited	After Command Set Functional Descriptor	[USBNCM10] 5.2.3
MBIM	MBIM Functional Descriptor	Required	Arbitrary	Table 6-3: MBIM FUNCTIONAL DESCRIPTOR
MBIM Extended	MBIM Extended Functional Descriptor	Optional	After MBIM Functional Descriptor	Table 6-4: MBIM EXTENDED FUNCTIONAL DESCRIPTOR

3.2.3 FUNCTION BEHAVIORAL REQUIREMENTS

The overall behavior of the function is determined by the alternate setting currently selected for the Communication Interface. When alternate setting 0 is selected, the function shall operate in NCM1.0 mode. When alternate setting 1 is selected, the function shall operate in MBIM mode.

The effect of changing the Communication Interface alternate setting while the Data Interface is set to a non-zero alternate setting is not specified. Best practice is for the host to ensure that the Data Interface is set to Alternate Setting 0 prior to changing the alternate setting of the Communication Interface.

Regardless of the previous sequence of SET_INTERFACE commands targeting the Communication Interface, the host is able to put the function into a defined state by the following sequence:

1. SET_INTERFACE(Data Interface, 0)
2. SET_INTERFACE(Communication Interface, desired alternate setting)
3. Sending the required class commands for the targeted alternate setting
4. SET_INTERFACE(Data Interface, 1 if Communication Interface,0 and Data Interface,2 Communication Interface 1)

4 CLASS SPECIFIC CODES

Table 4-1 defines the interface subclass code used in the MBIM Communications Interface Descriptor.

Table 4-1: MBIM Communications Interface Subclass Code

Code	Subclass
0Eh	Mobile Broadband Interface Model

Table 4-2: MBIM Communications Interface Protocol Code

Code	Protocol Code
00h	Mobile Broadband Interface Model Protocol

Table 4-3: MBIM Descriptor Codes

Value	Descriptor Code
1Bh	Mobile Broadband Interface Model Functional Descriptor
1Ch	Extended Mobile Broadband Interface Model Functional Descriptor

Table 4-4: MBIM Class-Specific Request Codes

Value	Request code
05h	RESET_FUNCTION

5 FUNCTIONAL CHARACTERISTICS

5.1 OVERVIEW

The MBIM function consists of two logical channels. One is the data channel for transporting payloads, similar to the [USBNCM10] data transport. The only difference is that instead of transporting Ethernet frames, MBIM transports raw IP data streams and Device Service Streams. The other channel is used for the control plane, i.e., for transporting MBIM control messages.

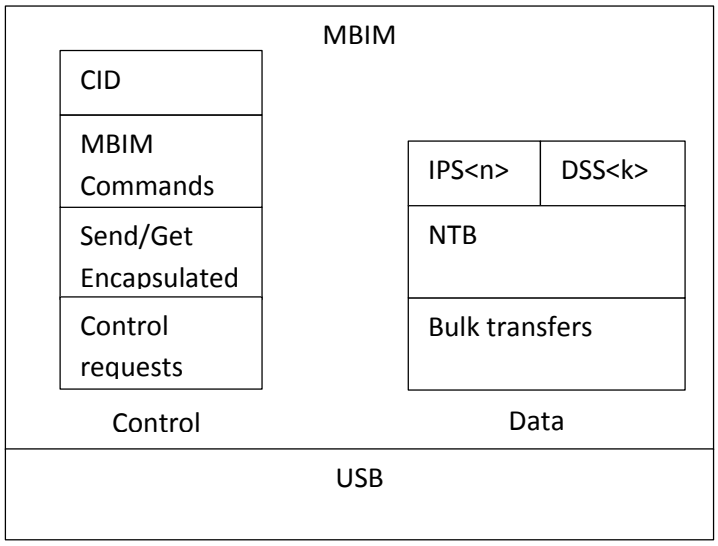

Figure 2: MBIM Logical Layers for Control and Data Channels

5.2 COMMUNICATION PIPES

In addition to the default pipe, each MBIM function shall support one interrupt, one bulk-in and one bulk-out pipe.

5.2.1 DEFAULT PIPE

Default-pipe messages are generally used to control a USB device. These messages include standard requests, such as GET_DESCRIPTOR and SET_CONFIGURATION. Commands that are sent on the default pipe report information back to the host on the default pipe. If an error occurs, they generate a standard USB error token. This applies to all class-specific requests in addition to the general requests described in the Universal Serial Bus Specification [USB30] .

The class-specific control messages are defined and described in Section 9 of this document.

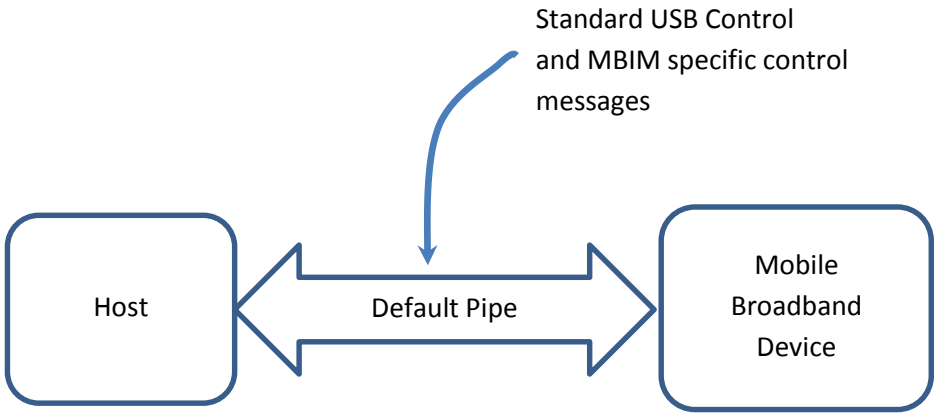

Figure 3: Default Pipe

5.2.2 INTERRUPT PIPE

Interrupt pipe messages are used to alert the host of an asynchronous event from the device. The event could either be an unsolicited event (see section 9.4.4), or a response to a previously issued class specific control messages (see section 9.4.3),

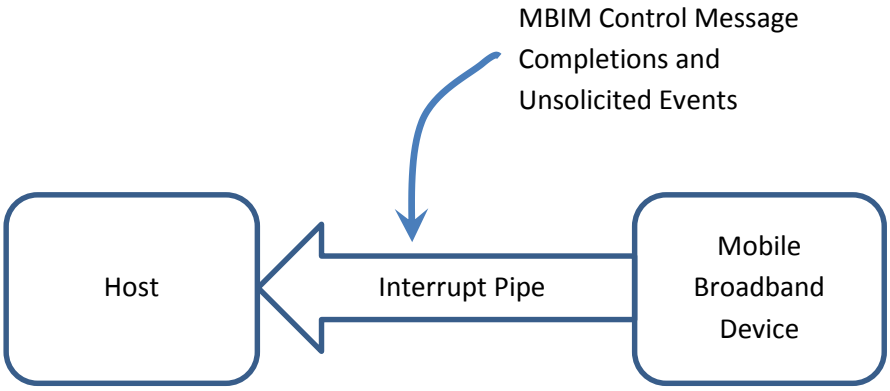

Figure 4: Interrupt Pipe

5.2.3 BULK-IN AND BULK-OUT PIPES

The host sends data to the MBIM function using the bulk-OUT pipe. The MBIM function sends data to the host using the bulk-IN pipe.

It is recommended that the values of dwNtbInMaxSize and dwNtbOutMaxSize (which dictate the maximum transfer sizes for the bulk-in and bulk-out pipes, respectively) be multiples of the wMaxPacketSize of the respective bulk pipes. If the transfer in either direction is a less than the configured Max NTB size and is a multiple of the wMaxPacketSize the sender must terminate the transfer with a ZLP. The sender may choose to pad the transfer out max NTB size to avoid the ZLP at its discretion.

5.3 OPERATIONAL MODEL

The MBIM control messages from the host to the device are asynchronous. The host sends a control message to the function on the device's default pipe. The function asynchronously completes the messages and signals availability of the result by sending a notification to the host over the interrupt end-point. The host, then, fetches the response using the default pipe. This model allows the host to issue multiple control messages to the device without waiting for previous message to complete. This flow is show in the figure below.

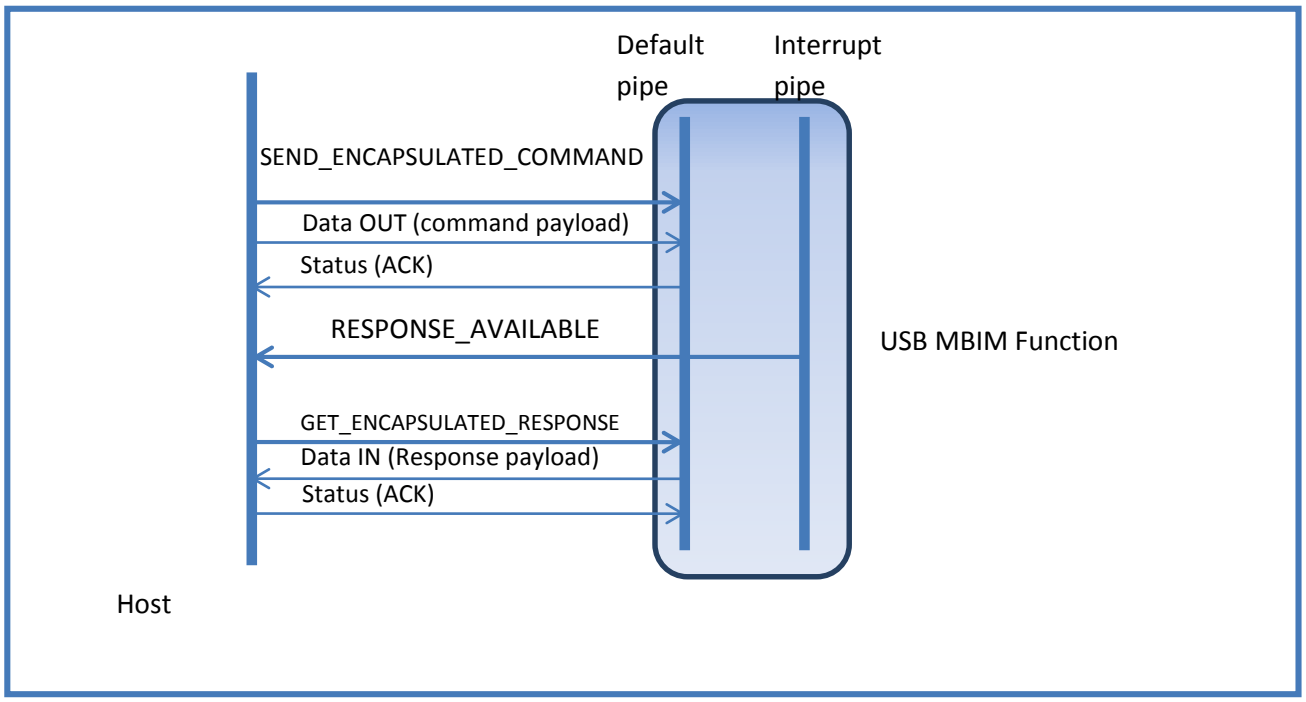

Figure 5: Encapsulated Command Response sequence

Similarly, whenever the device needs to notify the host of an unsolicited device events described in section 9.4.4, it uses the interrupt end-point. The host, then, retrieves the information from the control endpoint. This flow is shown below.

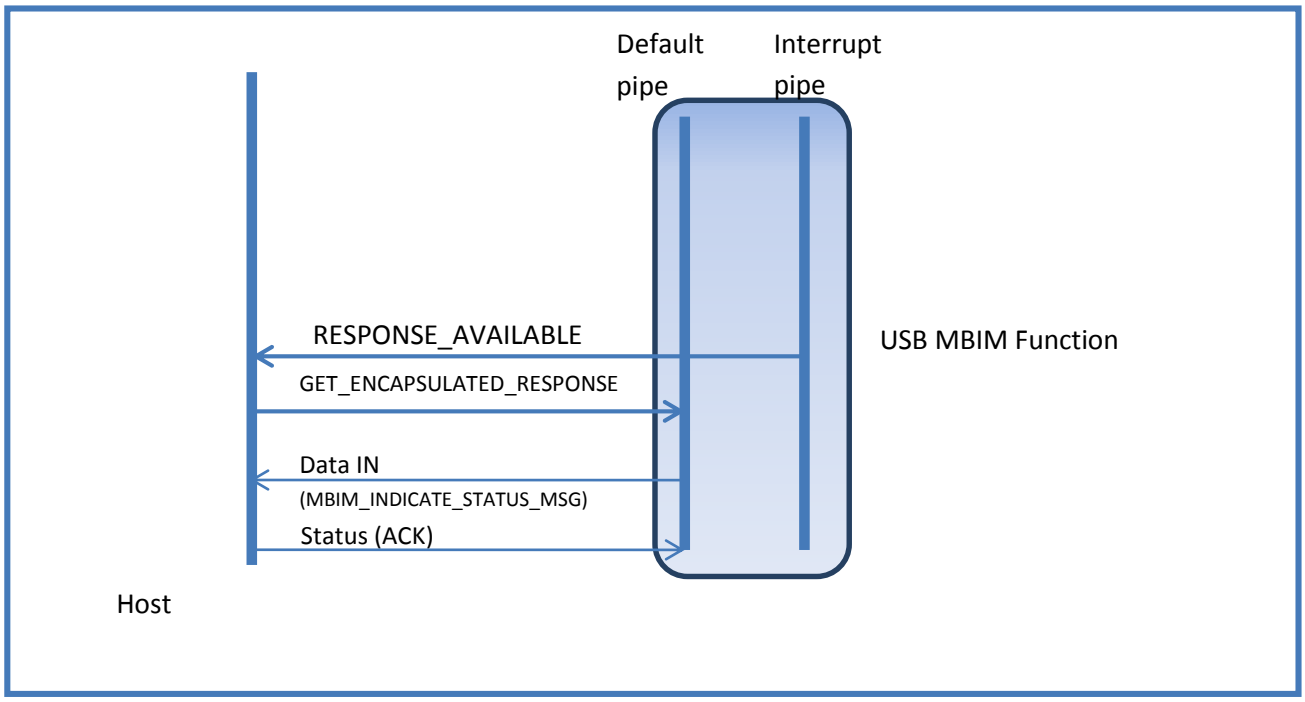

Figure 6: Unsolicited Event Sequence

5.4 DEVICE SERVICE

A logical group of functionality supported by the device constitutes a "device service." Each device service is uniquely identified by a 128-bit UUID (Universally Unique Identifier). All CIDs exchanged between the host and the device carry the UUID to identify the device service associated with the transfer.

5.4.1 DEVICE SERVICE STREAMS

The transfer of high-bandwidth data over the control channel is not supported. Instead, Device Service Streams over bulk are recommended.

Device Service Stream is a best effort service.

The protocol layer and/or application above MBIM shall be responsible for flow control and recovery from data loss.

Doing flow control by throttling the bulk IN/OUT pipes is strongly discouraged, because the pipes are shared by multiple services. Throttling the bulk OUT pipe to benefit one service, for example, may result in deadlock, by preventing delivery of important data to another service.

5.4.2 CUSTOM DEVICE SERVICES

Vendors are free to define additional device services by generating a UUID of their own. These services shall not be IP-protocol based.

This mechanism allows vendors to extend the functionality beyond the standard services defined in this document. If a vendor extension changes any of the states in this specification, the function needs to send the necessary indication (section 9.4.4) so that the Host state machine becomes synchronized. See section 10.6 “MBIM Service and CID extensibility” for more information on how to extend the protocol with new device services.

6 USB DEVICE MODEL

6.1 OVERVIEW

A USB MBIM function is implemented as a USB CDC function with two interfaces. Functions shall provide a CDC Union functional descriptor to group these two interfaces. See [USBWMC11].

A Communication Class interface, with class 02h and subclass 0Eh, and a Data Class interface combine to form a single functional unit representing the USB MBIM device. The Communication Class interface includes a single endpoint for event notification; it also uses the device's default pipe for control messages. The Data Class interface includes two bulk endpoints for data traffic.

There are two ways to group interfaces: the WHCM Union Functional Descriptor (see [USBWMC11]) and the IAD. Devices may also provide an IAD. If an IAD is provided, the information in the IAD for MBIM functions shall be consistent with the information in the CDC Union descriptor and Communication Class interface descriptor.

6.2 STANDARD USB DESCRIPTOR DEFINITIONS

The device returns a USB Device Descriptor per Chapter 9, "USB Device Framework", in the USB Specification [USB20] . See [USBCDC12] or [USBIAD10] for additional details.

6.3 COMMUNICATION CLASS INTERFACE DESCRIPTORS

The MBIM Communication Class interface is described by a USB interface descriptor, functional descriptors for the MBIM subclass, and an endpoint descriptor for the notification endpoint. Table 6-1 defines the salient fields of the Communication Class interface descriptor.

Table 6-1: Salient fields of Communication Class Interface Descriptor

Offset	Field	Size	Value	Description
5	bInterfaceClass	1	02h	Communication Interface Class code
6	bInterfaceSubClass	1	0Eh	MBIM
7	bInterfaceProtocol	1	00h	MBIM Protocol

MBIM functions must implement class-specific descriptors ("functional descriptors") for the MBIM Communications Interface. The framework for these is defined in [USBCDC12]. MBIM Communications Interfaces must implement the following functional descriptor described in [USBCDC12]:

- Header Functional Descriptor (describing the level of compliance to [USBCDC12] Interface and the Data interface).
- Union Functional Descriptor (containing the interface numbers of the Communications Interface and the Data interface).

MBIM functions shall supply an MBIM Functional Descriptor, as defined in Table 6-3: MBIM FUNCTIONAL DESCRIPTOR.

The class-specific descriptors must be followed by an Interrupt IN endpoint descriptor.

Descriptor requirements are summarized in Table 6-2: MBIM Communication Interface Descriptor Requirements.Table 6-2:

Table 6-2: MBIM Communication Interface Descriptor Requirements

Descriptor	Description	Required/Opt	Order	Reference
HEADER	CDC Header functional descriptor	Required	First	[USBCDC12], 5.2.3.1
UNION	CDC Union functional descriptor	Required	Arbitrary	[USBCDC12] 5.2.3.2
MBIM	MBIM Functional Descriptor	Required	Arbitrary	Table 6-3: MBIM FUNCTIONAL DESCRIPTOR
MBIM Extended	MBIM Extended Functional Descriptor	Optional	After MBIM Functional Descriptor	Table 6-4: MBIM EXTENDED FUNCTIONAL DESCRIPTOR

6.4 MBIM FUNCTIONAL DESCRIPTOR

This descriptor provides information about the implementation of the MBIM function. It is mandatory, and must appear after the Header Functional Descriptor.

Table 6-3: MBIM FUNCTIONAL DESCRIPTOR

Offset	Field	Size	Value	Description
0	bFunctionLength	1	12	Size of Descriptor in bytes
1	bDescriptorType	1	Constant	CS_INTERFACE (0x24)
2	bDescriptorSubtype	1	Constant	MBIM Functional Descriptor code (see Table 4-3)

3	bcdMBIMVersion	2	Number 0x0100	Release number of this specification in BCD, with implied decimal point between bits 7 and 8. 0x0100 == 1.00 == 1.0. This is a little-endian constant, so the bytes will be 0x00, 0x01.
5	wMaxControlMessage	2	Number	Maximum segment size in bytes a function can handle from a SEND_ENCAPSULATED_COMMAND or can return from a GET_ENCAPSULATED_RESPONSE. This number must not be smaller than 64. However, in order to avoid frequent fragmentation (e.g., with full SMS or USSD messages), it is recommended that this value be no smaller than 512.
7	bNumberFilters	1	Number	Contains the number of PacketFilters that are available in total for all SessionIds. These filters are potentially capable of causing wake-up of the host. This number must not be smaller than 16.
8	bMaxFilterSize	1	Number	The maximum size of the PacketFilters in bytes. This number must not exceed 192.

9	wMaxSegmentSize	2	Number	The Maximum segment size in bytes that the function is capable of supporting. This number has to be larger than the MTU set for IP traffic by the network. This number must not be smaller than 2048. Note: This shall not be used for IP MTU. For configuring IP MTU use either MBIM extended functional descriptor or IP MTU handling MBIM_CID_IP_CONFIGURATION
11	bmNetworkCapabilities	1	Bitmap	Specifies the capabilities of this function. A bit value of zero indicates that the capability is not supported. This is patterned after the NCM Functional Descriptor [USBNCM10] section 5.2.1 except that Ethernet-specific fields D4(CRC) and D1(NetAddress) are not used. D7,D6: Reserved (zero) D5: Function can process 8-byte forms of GetNtbInputSize and SetNtbInputSize requests D4: Reserved (zero) D3: Function can process SetMaxDatagramSize and GetMaxDatagramSize requests. D2: Reserved (zero). D1: Reserved (zero) D0: Reserved (zero)

MBIM departs from NCM1.0 in the specification for wMaxSegmentSize: NCM 1.0 takes it from the Ethernet functional descriptor, whereas MBIM takes it from its own MBIM functional descriptor above (Table 6-3: MBIM FUNCTIONAL DESCRIPTOR). Accordingly, in MBIM the default and minimum value of

wMaxSegmentSize is 2048. Furthermore, this is the value of wMaxSegmentSize that applies to both GetMaxDatagramSize and SetMaxDatagramSize in MBIM.

It is important to note that whereas MBIM's bmNetworkCapabilities uses NCM's as a template, the definition in Table 6-3: MBIM FUNCTIONAL DESCRIPTOR is entirely an MBIM construct. Thus, any similarities that may remain between the two have no normative implications. In particular, MBIM's version of bmNetworkCapabilities only uses two bits: D3 and D5. The rest are reserved and set to zero.

6.5 MBIM EXTENDED FUNCTIONAL DESCRIPTOR

This descriptor provides extended information about the implementation of the MBIM function. The descriptor is optional, but if it exists the descriptor must appear after the MBIM Functional Descriptor.

Table 6-4: MBIM EXTENDED FUNCTIONAL DESCRIPTOR

Offset	Field	Size	Value	Description
0	bFunctionLength	1	8	Size of Descriptor in bytes
1	bDescriptorType	1	Constant	CS_INTERFACE (0x24)
2	bDescriptorSubtype	1	Constant	MBIM Extended Functional Descriptor code (see Table 4-3)
3	bcdMBIMExtendedVersion	2	Number 0x0100	Release number of MBIM extensions in BCD, with implied decimal point between bits 7 and 8. 0x0100 == 1.00 == 1.0. This is a little-endian constant, so the bytes will be 0x00, 0x01.
5	bMaxOutstandingCommandMessages	1	Number	Max number of outstanding Command Messages the device can handle simultaneously. Shall be greater than 0.

6	wMTU	2	Number	Operator preferred MTU for home network. wMTU applies to IP Data Streams. If no specific requirements exist the recommended value should be 1500.

6.6 DATA CLASS INTERFACE

The Data Class interface is described by a standard USB Interface Descriptor followed by two endpoint descriptors.

The Data Interface of an MBIM networking function shall have two alternate settings. The first alternate setting (the default interface setting, alternate setting 0) shall include no endpoints and therefore no networking traffic can be exchanged when the default interface setting is selected. The second alternate setting (alternate setting 1) is used for normal operation, and shall include one bulk IN endpoint and one bulk OUT endpoint.

The interface descriptors for alternate settings 0 and 1 shall have bInterfaceSubClass set to 0, and bInterfaceProtocol set to 02h (see Table 7 of [USBCDC12]).

Table 6-5: Data Class Interface Descriptor Requirements defines the salient fields of the Data Class Interface Descriptor.

Table 6-5: Data Class Interface Descriptor Requirements

Offset	Field	Size	Value	Description
5	bInterfaceClass	1	0Ah	Data Interface Class code
6	bInterfaceSubClass	1	00h	Data Class SubClass code
7	bInterfaceProtocol	1	02h	Data Class Protocol code, Network transfer block-MB

7 DATA TRANSPORT

The data packets will be passed to and from the device using the standard NCM 1.0 transfer headers and datagram pointers as described in section 3.2 of [USBNCM10]. Additional considerations are described in section 5.2.3 Bulk-in and bulk-out pipes.

[USBNCM10] defines a method to transport IEEE 802.3 frames between the host and device. For MBIM functions the same method is used but the payload inside the NDP has changed. IEEE 802.3 frames are not supported anymore. Instead "raw" IP (both IPv4 and IPv6) and Device Service Streams are transported between the host and the device without any type of MAC layer header. MBIM devices also introduce the concept of multiple data streams in the NDP packets for both IP and Device Service Streams.

Figure 7: Logical layers for the data transport, shows a logical layering of the data transport mechanisms.

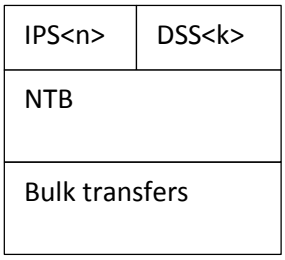

Figure 7: Logical layers for the data transport

In order to identify the frames as containing payload of either raw IP or Device Service Streams, the valid signature values for NDP structures (section 3.3 of [USBNCM10]) are extended as follows.

To distinguish among the data streams, the last character of the dwSignature in the NDP header shall be coded with the index SessionId or DssSessionId specified by the host in the MBIM_CID_CONNECT or MBIM_CID_DSS_OPEN command, respectively. The index value is assigned separately for IP and Device Service Streams. The first three symbols are encoded as ASCII characters in little-endian form plus a last byte in HEX (binary) format:

Table 7-1: MBIM Data Transport signature values for NDP structures

Protocol	NDP16	NDP32	Description
Raw IPv4 or IPv6	"IPS"<SessionId>	"ips"<SessionId>	Raw IPv4 or IPv6 payload
Device Service Stream	"DSS"<DssSessionId>	"dss"<DssSessionId>	Device Service Stream payload

The minimum values for wDatagramLength (corresponding to the field wDatagramLength[0] in Table 3-3 of [USBNCM10]) and dwDatagramLength (corresponding to the field dwDatagramLength[0] in Table 3-4 of [USBNCM10]) are as follows:

- For IPv4: >= 20
- For IPv6: >= 40
- For DSS: >= 0

Because the signature applies to all the datagrams in the NDP, the formats must not be mixed within the NDP, but it can add different NDPs within an NTB (see [USBNCM10]).

7.1 DATA FORMATTING

The format of the datagrams described by a given NDP depends on the protocol.

7.1.1 IPV4

The datagrams starts with the IPv4 header, and then continues with the appropriate payload.

7.1.2 IPV6

The datagrams starts with the IPv6 header, and then continues with the appropriate payload.

7.1.3 DEVICE SERVICE STREAM

The format of the Device Service Stream payload depends on the device service (as identified by the corresponding UUID) that is used when opening the data stream.

7.1.4 MBIM FRAME ALIGNMENT

Many network stacks in embedded devices benefit from careful alignment of the payload to system-defined memory boundaries. MBIM allows a function to align transmitted datagrams on any convenient boundary within the NTB. Functions indicate how they intend to align their transmitted datagrams to the host in the NTB Parameter Structure ([USBNCM10] Table 6-3)

Similarly, for data transmitted from the host, functions indicate their preferred alignment requirements to the host. The host then formats the NTBs to satisfy this constraint.

MBIM assumes that hosts are more flexible and powerful than devices, and that hosts honor the constraints given by the device when preparing OUT NTBs.

Alignment requirements are met by controlling the location of the payload. This alignment is specified by indicating a constraint as a divisor and a remainder. The agent formatting a given NTB aligns the payload of each datagram by inserting padding, such that the offset of each datagram satisfies the constraint:

Offset % wNdpInDivisor == wNdpInPayloadRemainder (for IN datagrams)

Or

Offset % wNdpOutDivisor == wNdpOutPayloadRemainder (for OUT datagrams)

Two use cases are anticipated (although the specification does not restrict the user to these two cases).

In one use case, the function wants to align the beginning of an IP packet to a cache line boundary. Cache lines are generally much smaller than the maximum IP packet size. In this case, the wNdpXxDivisor is set to the size of a cache line (in bytes), and the wNdpXxPayloadRemainder is set to zero. The effect is shown schematically in Figure 7-8.

In another use case, the function wants to place each IP packet in a fixed sized buffer. (This is primarily intended for use on the OUT pipe.) For this to work, each buffer must be bigger than the maximum IP packet size. In this case, wNdpXxDivisor is set to the size of the buffer, and wNdpXxPayloadRemainder is set to the desired offset of the IP packet in the fixed size buffer. This situation is shown schematically in Figure 7-9.

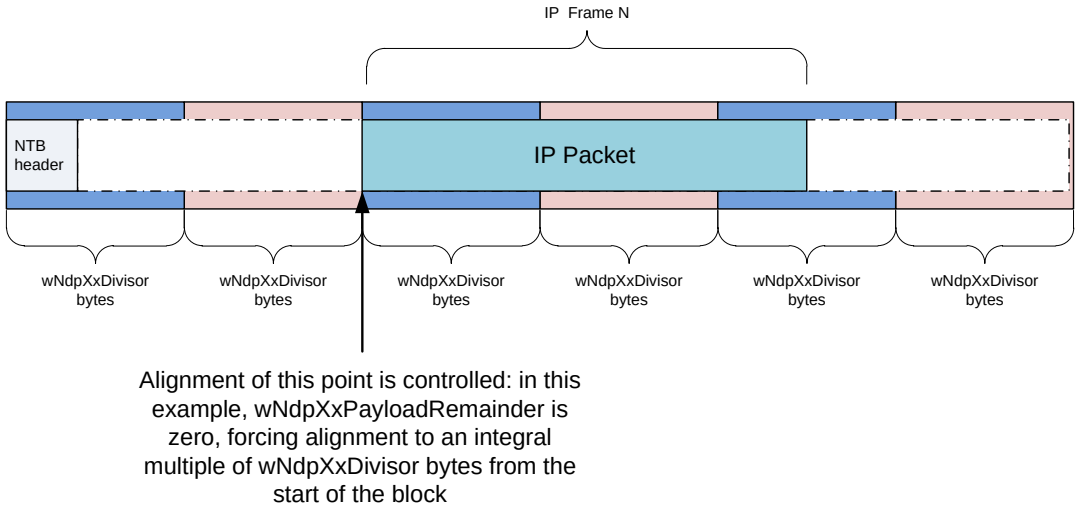

Figure 7-8: Alignment to a cache line

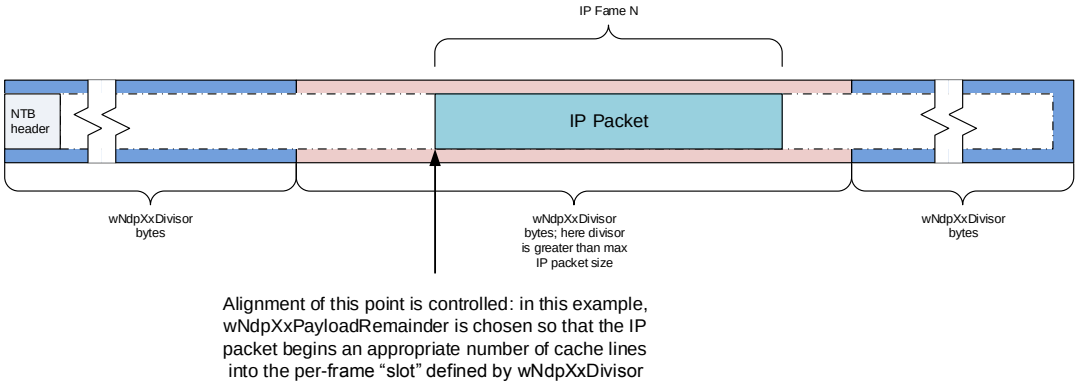

Figure 7-9: Alignment for fixed-size internal buffers

7.1.5 MBIM ALLOWS MULTIPLE NDPS WITHIN ONE NTB

Today the [NCM10] specification does not allow the possibility to send multiple NDPs within one NTB. MBIM allows the usage of this feature and hence the value wNextNdpIndex in Table 3-3 (NDP16) and Table 3-4 (NDP32) will not be reserved anymore. The new definition is

6	wNextNdpIndex	2	Number	Byte index, in little endian, of the next NDP. The index is from byte zero of the NTB.

8 CONTROL CHANNEL CHARACTERISTICS

The control channel for the MBIM function is the device's default pipe.

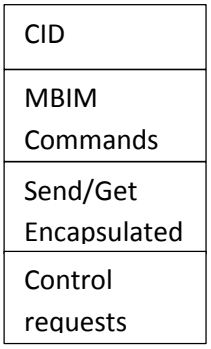

Figure 10: Logical layers for the control channel

8.1 CONTROL REQUESTS

The MBIM Interface Class supports class-specific requests, defined in this document and in the communications data class specifications. This section describes the requests that are specific to the MBIM Interface Class. These requests are sent over the default pipe and can apply to different functions as defined by the Communications Class interface codes.

Table 8-1: Networking Control Model Requests lists all of the class-specific requests that are valid for an MBIM Communications Interface. This table includes those defined in [USBECM12] and [USBNCM10]. Commands marked as "required" must be implemented by any conforming MBIM function (see the reference section for each command for details).

Table 8-1: Networking Control Model Requests

Request	Description	Required/Opt	reference
SendEncapsulatedCommand	Issues a command in the format of the supported control protocol.	Required	See 8.1.1 SendEncapsulatedCommand
GetEncapsulatedResponse	Requests a response in the format of the supported control protocol.	Required	See 8.1.2 GetEncapsulatedResponse

Request	Description	Required/Opt	reference
GetNtbParameters	Requests the function to report parameters that characterize the Network Control Block	Required	[USBNCM10]
GetNtbFormat	Get current NTB Format	Optional	[USBNCM10]
SetNtbFormat	Select 16 or 32 bit Network Transfer Blocks	Optional	[USBNCM10]
GetNtbInputSize	Get the current value of maximum NTB input size	Required	[USBNCM10]
SetNtbInputSize	Selects the maximum size of NTBs to be transmitted by the function over the bulk IN pipe	Required	[USBNCM10]
GetMaxDatagramSize	Requests the current maximum datagram size	Optional	8.1.3 Max Datagram Size
SetMaxDatagramSize	Sets the maximum datagram size to a value other than the default	Optional	8.1.3 Max Datagram Size
ResetFunction	Resets the function so it returns to the same state it has after a set config	Required	See 8.1.5 Reset

8.1.1 SENDENCAPSULATEDCOMMAND

The SendEncapsulatedCommand is in principle the same as described in [USBCDC12] chapter 6.2.1, but with the following limitation: the host honors the wMaxControlMessage from the MBIM descriptor in such a way that wLength is never chosen to be longer than wMaxControlMessage. If the CID is longer than wMaxControlMessage, then the host fragments the message according to section 9.5 Fragmentation of messages. In case of fragmented messages, the host sends maximum-length transfers (full transfers for which wLength equals wMaxControlMessage) up to the very last fragment. The last fragment could, of course, be smaller than a maximum-length transfer.

This request is used to issue a command in the format of the supported control protocol of the Communications Class interface. The following table is copied from [USBCDC12] as a reference.

Table 8-2: Send Encapsulated Command

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	SEND_ENCAPSULATED_COMMAND	Zero	Interface	Amount of data, in bytes, associated with this recipient.	Control protocol-based command

8.1.2 GETENCAPSULATEDRESPONSE

The GetEncapsulatedResponse is in principle the same as described in [USBCDC12] chapter 6.2.2, but with the following limitation: the host honors the wMaxControlMessage from the MBIM descriptor in such a way that wLength always is chosen to be MaxControlTransfer (see Table 9-4: MBIM_OPEN_MSG). Then the function can send MaxControlTransfer or less depending on data avaibility. The following table is copied from [USBCDC12] as a reference.

Table 8-3: Get Encapsulated Response

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
10100001B	GET_ENCAPSULATED_RESPONSE	Zero	Interface	Amount of data, in bytes, associated with this recipient.	Protocol-dependent data

The host does not continuously poll the default pipe for input control messages. When the MBIM function is ready to send a control message to the host, the function must return a RESPONSE_AVAILABLE notification on the Communication Class interface's Interrupt IN endpoint. The transfer from the function's interrupt IN endpoint to the host is a standard USB Interrupt IN transfer

Upon receiving the RESPONSE_AVAILABLE notification, the host reads the control message from the Control endpoint using a GET_ENCAPSULATED_RESPONSE transfer, defined in [USBCDC12] chapter 6.2.2. The function must use a separate GET_ENCAPSULATED_RESPONSE transfer for each control message it has to send to the host (that is, the function must not concatenate multiple messages into a single GET_ENCAPSULATED_RESPONSE transfer). The function must send a RESPONSE_AVAILABLE notification for each available ENCAPSULATED_RESPONSE to be read from the default pipe. For example, if the function has 2 ENCAPSULATED_RESPONSES available and the first ENCAPSULATED_RESPONSE consists of 3 fragments, the second one consists of one only fragment, the function should send 4 RESPONSE_AVAILABLE notifications over the interrupt IN pipe. Per [USB30] section 8.12.2.2, each RESPONSE_AVAILABLE notification must be Zero Length Packet (ZLP) terminated if the length is an exact multiple of the wMaxPacketSize of the Endpoint Descriptor (see [USB30], table 9-

18) for interrupt endpoint. The ENCAPSULATED_RESPONSE must also be ZLP terminated if the size returned is a multiple of the bMaxPacketSize0 and is not equal to wLength in the GET_ENCAPSULATED_RESPONSE request (see [USB30] Table 9-8).

8.1.3 MAX DATAGRAM SIZE

For both GetDatagramSize and SetDatagramSize, per table Table 8-1: Networking Control Model Requests, these optional requests are implemented per [USBNCM10]. However, there is an exception: the value of wMaxSegmentSize used is now specified in the MBIM functional descriptor Table 6-3: MBIM FUNCTIONAL DESCRIPTOR. In MBIM, the default and minimum value of wMaxSegmentSize is 2048.

8.1.4 NOTIFICATIONS

The only notification used from chapter 6.3 in [USBNCM10] is ResponseAvailable. This notification is mandatory in this specification, whereas it was optional in [USBNCM10].

8.1.5 RESET FUNCTION

RESET_FUNCTION resets the function to its initial state. All IP data stream connections are disconnected; all Device Service Streams are closed. If the function is open using MBIM_OPEN, it is returned to the closed state. The function shall abandon all outstanding transactions that are awaiting completion. No notifications shall be sent.

The data toggles of bulk and interrupts pipes are not affected. The alternate settings of interfaces are not changed. [USB30] remote-wakeup enable for the function is not affected. Other state controlled by the [USB30] core specification is similarly not affected.

The intended use of this request is to allow a host to return the function to a known state. It differs from USB reset, in that only the targeted function is affected by this request. Other functions in a composite device are not affected by this request. USB transport-level state is not affected by this request.

Table 8-4: Reset Function

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	RESET_FUNCTION	Zero	Interface	Zero	Zero

8.1.6 ERROR HANDLING

See specification [USB20] and [USB30]

9 MBIM CONTROL MESSAGES

MBIM control messages must be supported by an MBIM function. In the more detailed descriptions below, the messages include a TransactionId field. This is used to match sent messages with responses. With this mechanism, a host can send multiple MBIM messages to a function concurrently without concern for the ordering of responses. An MBIM function must maintain the TransactionId field when returning a response. For notifications, the TransactionId must be set to 0 by the function.

9.1 MBIM MESSAGE HEADER

All MBIM control messages must start with the following header.

Table 9-1: MBIM_MESSAGE_HEADER

Offset	Size	Field	Type	Description
0	4	MessageType	UINT32	Specifies the MBIM message type. See Table 9-3: Control messages sent from the host to the function, and Table 9-9: Control Messages sent from function to host The Most Significant Bit in the MessageType indicates direction: • MSB set to 0: from host to function • MSB set to 1: from function to host
4	4	MessageLength	UINT32	Specifies the total length of this MBIM message in bytes.
8	4	TransactionId	UINT32	Specifies the MBIM message id value. This value is used to match host-sent messages with function responses. This value must be unique among all outstanding transactions. For notifications, the TransactionId must be set to 0 by the function. TransactionId 0 is reserved for exclusive use by notifications and by MBIM_FUNCTION_ERROR_MSG if the TransactionId of the control message causing the error cannot be determined.

9.2 MBIM FRAGMENT HEADER

If the size of a command or a response is larger than MaxControlTransfer, the message may be split across multiple messages. This header indicates how many fragments there are in total. For fragmentation considerations, see section 9.5 (Fragmentation of messages).

Table 9-2: MBIM_FRAGMENT_HEADER

Offset	Size	Field	Type	Description
0	4	TotalFragments	UINT32	This field indicates how many fragments there are in total. For fragmentation considerations, see section 9.5 (Fragmentation of messages).
4	4	CurrentFragment	UINT32	This field indicates which fragment this message is. Values are 0 to TotalFragments-1

9.3 CONTROL MESSAGES SENT FROM THE HOST TO THE FUNCTION

The following MessageType values are defined for messages sent from the host to an MBIM function.

Table 9-3: Control messages sent from the host to the function

Message Identifier	Value	Description
MBIM_OPEN_MSG	1	Initialize the function.
MBIM_CLOSE_MSG	2	Closes the function.
MBIM_COMMAND_MSG	3	Send a 'COMMAND' CID.
MBIM_HOST_ERROR_MSG	4	Indicates an error in the MBIM communication.

A control message from the host to the device is sent in the payload of a SEND_ENCAPSULATED_COMMAND (see 8.1.1 on SendEncapsulatedCommand).

The function is either in a Closed or Opened global state (as opposed to the Ready States in Table 10-16: Table 10-16: MBIM_SUBSCRIBER_READY_STATE), and these transitions occur as follows:

A device is initially (upon enumeration) in the Closed state.

The function transitions from the Closed to the Opened state when it responds to an MBIM_OPEN_MSG with an MBIM_OPEN_DONE whose Status field is set to MBIM_ERROR_STATUS_SUCCESS.

The function transitions from the Opened state to the Closed state when it sends an MBIM_CLOSE_DONE to the host.

9.3.1 MBIM_OPEN_MSG

This message is sent by the host to begin interacting with the function. The message is only sent when the device is in the Closed state.

The function indicates that it has been successfully opened by responding with MBIM_OPEN_DONE.

Right after the function has been successfully opened, the host cannot make any assumptions about the function's current state other than what is implied by the function completing the MBIM_CLOSE_MSG. Once the open message has been successfully completed it is recommended that the host make sure that state-change notifications are enabled for all states the host finds relevant. The host can then query the function's current state for all relevant states. This will ensure that the host and function are synchronized.

In case this message is sent to a function that is already opened, the function shall interpret this as that the host and the function are out of synchronization. The function shall then perform the actions dictated by the MBIM_CLOSE_MSG before it performs the actions dictated by this command. The function shall not send the MBIM_CLOSE_DONE when the transition to the Closed state has been completed. Only the MBIM_OPEN_DONE message is sent upon successful completion of this message.

See Table 9-4 for details of the message.

9.3.1.1 MBIM_OPEN_MSG FORMAT

Table 9-4: MBIM_OPEN_MSG

Offset	Size	Field	Type	Description
0	12	MessageHeader	MBIM_MESSAGE_HEADER	A message header as specified in Table 9-1: MBIM_MESSAGE_HEADER with MessageType set to MBIM_OPEN_MSG.

Offset	Size	Field	Type	Description
12	4	MaxControlTransfer	UINT32	Maximum control transfer the host supports. This is the size the host will use for GetEncapsulatedResponse. If the function does not support this, it returns MBIM_ERROR_MAX_TRANSFER per Table 9-8: MBIM_PROTOCOL_ERROR_CODES. This must not exceed wMaxControlMessage from Table 6-3: MBIM FUNCTIONAL DESCRIPTOR. If the MaxControlTransfer is less than 64 bytes the device behavior is unspecified. For test purposes the function shall support a MaxControlTransfer of 64 bytes. In addition any other value shall be a multiple of the maximum control endpoint size

9.3.2 MBIM_CLOSE_MSG

This message is sent by the host in order to terminate to the host's session with the function.

Between the host's sending this message and the function's completing this message (acknowledged with MBIM_CLOSE_DONE), the function can expect that no MBIM control messages are sent by the host on the control plane or data on the bulk pipes, and the function shall ignore any such messages.

The function shall not send any MBIM control messages on the control plane or data on the bulk pipes after completing this message (acknowledging it with the MBIM_CLOSE_DONE message) with one exception and that is MBIM_ERROR_NOT_OPENED. MBIM_ERROR_NOT_OPENED shall be sent by the function if the host sends an MBIM_COMMAND_MSG message to the function while the function is in a "Closed" state (e.g., prior to MBIM_OPEN_DONE or after an MBIM_CLOSE_DONE),

When the function has completed this message and acknowledged it, the only MBIM control message the host may send is the MBIM_OPEN_MSG.

Any active context between the function and the host shall be terminated. Note: this is specific to the link between device and host.

This message may be sent at any time the device is in the Opened state.

Table 9-5: MBIM_CLOSE_MSG Message

Offset	Size	Field	Type	Description
0	12	MessageHeader	MBIM_MESSAGE_HEADER	A message header as specified in Table 9-1: MBIM_MESSAGE_HEADER with MessageType set to MBIM_CLOSE_MSG.

9.3.3 MBIM_COMMAND_MSG

MBIM_COMMAND_MSG is sent to a MBIM device from a host when it needs to query or set state on device. The parameter or statistics being queried for is identified by means of a Command Identifier (CID). Whether the command is setting or querying is determined by the CommandType. The host may send MBIM_COMMAND_MSG to the device via the control channel at any time that the device is in the Opened state. A MBIM device will respond to MBIM_COMMAND_MSG with information about the desired capabilities or status.

9.3.3.1 MBIM_COMMAND_MSG FORMAT

Table 9-6: MBIM_COMMAND_MSG Message

Offset	Size	Field	Type	Description
0	12	MessageHeader	MBIM_MESSAGE_HEADER	A message header as specified in Table 9-1: MBIM_MESSAGE_HEADER with MessageType set to MBIM_COMMAND_MSG.

Offset	Size	Field	Type	Description
12	8	FragmentHeader	MBIM_FRAGMENT_HEADER	A fragmentation header as specified in Table 9-2: MBIM_FRAGMENT_HEADER For fragmentation considerations, see section 9.5 (Fragmentation of messages).
20	16	DeviceServiceId	UUID	A 16-byte UUID that identifies the device service the following CID value applies.
36	4	CID	UINT32	Specifies the CID that identifies the parameter being queried for
40	4	CommandType	UINT32	0 for a query operation, 1 for a Set operation.
44	4	InformationBufferLength	UINT32	Size of the Total InformationBuffer, may be larger than current message if fragmented.
48		InformationBuffer	DATABUFFER	Data supplied to device specific to the CID

9.3.4 MBIM_HOST_ERROR_MSG OR MBIM_FUNCTION_ERROR_MSG

An MBIM_X_ERROR_MSG (referring to either MBIM_HOST_ERROR_MSG or MBIM_FUNCTION_ERROR_MSG) shall be sent to indicate an error in the MBIM protocol. MBIM_HOST_ERROR_MSG is used for messages that are sent from the host to the function. MBIM_FUNCTION_ERROR_MSG is used for error messages that are sent from the function to the host. These messages can be sent from host and function to indicate that a received message was formatted incorrectly. These messages are meant to be used for debugging as there is no guarantee that any corrective action triggered by them will be effective at all. An MBIM_X_ERROR_MSG shall not be sent in response to an MBIM_X_ ERROR_MSG. An MBIM_X_ERROR_MSG shall not make use of a DataBuffer, so it cannot send any data payload.

Table 9-7: MBIM_HOST_ERROR_MSG or MBIM_FUNCTION_ERROR_MSG

Offset	Size	Field	Type	Description

Offset	Size	Field	Type	Description
0	12	MessageHeader	MBIM_MESSAGE_HEADER	A message header as specified in Table 9-1: MBIM_MESSAGE_HEADER with MessageType set to MBIM_X_ERROR_MSG. If the message is sent from the device to the host the MSB will also be set.
12	4	ErrorStatusCode	MBIM_PROTOCOL_ERROR_CODES	See Table 9-8: MBIM_PROTOCOL_ERROR_CODES.

Table 9-8: MBIM_PROTOCOL_ERROR_CODES

Message Identifier	Value	Description
MBIM_ERROR_TIMEOUT_FRAGMENT	1	Shall be sent by the receiver if the time between the fragments exceeds the max fragment time. See 9.3.4.1
MBIM_ERROR_FRAGMENT_OUT_OF_SEQUENCE	2	Shall be sent by the receiver if a fragmented message is sent out of sequence See 9.3.4.2
MBIM_ERROR_LENGTH_MISMATCH	3	Shall be sent by the receiver if the InformationBufferLength with required padding does not match the total of MessageLength minus headers See 9.3.4.3
MBIM_ERROR_DUPLICATED_TID	4	Shall be sent by the receiver if two MBIM commands are sent with the same TID See 9.3.4.4

Message Identifier	Value	Description
MBIM_ERROR_NOT_OPENED	5	The function shall respond with this error code if it receives any MBIM commands prior to an open command or after a close command. See 9.3.4.5.
MBIM_ERROR_UNKNOWN	6	Shall be sent by the function when an unknown error is detected on the MBIM layer. Expected behavior is that the host resets the function if a MBIM_ERROR_UNKNOWN is received.
MBIM_ERROR_CANCEL	7	Can be sent by the host to cancel a pending transaction. See section 9.3.4.6.
MBIM_ERROR_MAX_TRANSFER	8	Shall be sent if the function does not support the maximum control transfer the host supports. See 9.3.1.1.

9.3.4.1 MBIM_ERROR_TIMEOUT_FRAGMENT

A host or function that receives fragmented messages shall send an MBIM_X_ERROR_MSG if the time between the fragments exceeds 1250 ms. A host or function that receives fragmented messages shall not send an MBIM_X_ERROR_MSG if the time between the fragments is less than 750 ms. The time between fragments is defined as the time between the end of the Status Stage of one fragment and the beginning of the Setup Stage of the next fragment (see section 8.5.3 in [USB2.0] and section 8.12.2 in [USB3.0]). The TransactionId of the responding message must match the TransactionId in the faulty fragmented sequence. In case of a timeout error, the receiver shall discard all the packets with the same TransactionId as the faulty message sequence.

9.3.4.2 MBIM_ERROR_FRAGMENT_OUT_OF_SEQUENCE

If a host or function detects a fragmented message that is out of order, e.g. , the CurrentFragment is not 0 or in-order, the receiver shall send a MBIM_X_ERROR_MSG with status code MBIM_ERROR_FRAGMENT_OUT_OF_SEQUENCE. The TransactionId of the responding message must

match the TransactionId in the faulty fragmented sequence. In case of an out of a sequence error, the receiver shall discard all the packets with the same TransactionId as the faulty message sequence.

The sender shall stop transmitting the remaining packets with that TransactionId as soon as it receives the error message. However, if the receiver gets one more message that is out of order for the same TransactionId, it shall send a new error message with the same TransactionId once more.

9.3.4.3 MBIM_ERROR_LENGTH_MISMATCH

If the host or the function receives a message where the InformationBufferLength does not match the total of the MessageLength, minus headers, of all fragmented packets the receiver shall send a MBIM_X_ERROR_MSG with status code MBIM_ERROR_LENGTH_MISMATCH. The TransactionId of the responding message must match the TransactionId of the faulty message. In case of a MBIM_ERROR_LENGTH_MISMATCH all packets with the same TransactionId shall be discarded.

9.3.4.4 MBIM_ERROR_DUPLICATED_TID

If the host or the function receives a message with the same TransactionId as another message that is still being processed, the receiver shall send a MBIM_X_ERROR_MSG with status code MBIM_ERROR_DUPLICATED_TID. The TransactionId of the responding message shall match the TransactionId in the faulty message. In case of an MBIM_ERROR_DUPLICATED_TID error, the receiver shall discard the newly arrived message.

9.3.4.5 MBIM_ERROR_NOT_OPENED

The three following paragraphs talk about three different scenarios. The order is roughly chronological, according to what the host is sending to the function: MBIM_COMMAND_MSG (via the control channel), data (via the data channel) and MBIM_CLOSE_MSG (via the control channel):

If the host sends an MBIM_COMMAND_MSG message to the function while the function is in a “Closed” state (e.g. prior to MBIM_OPEN_DONE or after a MBIM_CLOSE_DONE), the function shall respond with an MBIM_FUNCTION_ERROR_MSG message whose status code is set to MBIM_ERROR_NOT_OPENED. This status code gives an indication to the host that the function and the host are not synchronized, and that the function is in “Closed” state.

If the host sends data traffic to the function, the function shall send an MBIM_FUNCTION_ERROR_MSG with status code MBIM_ERROR_NOT_OPENED if it is in “Closed” state.

If the host sends MBIM_CLOSE_MSG while the function is still powering up (and hence not yet out of the “Closed” state), the function shall respond with an MBIM_FUNCTION_ERROR_MSG with MBIM_ERROR_NOT_OPENED status code. The host can send an MBIM_OPEN_MSG immediately after receiving such a response.

9.3.4.6 MBIM_ERROR_CANCEL

In case of a cancel, the receiver shall try to cancel the outstanding operations indicated by the MBIM_COMMAND_MSG with the same TransactionId as specified in the cancel message. The receiver

must not send any MBIM_COMMAND_DONE message after it has received a MBIM_ERROR_CANCEL message. However due to timing issues the sender must handle that a MBIM_COMMAND_DONE could have been sent from the receiver in the time between sending and receiving the MBIM_ERROR_CANCEL.

The TransactionId of the responding message must match the TransactionId in the previous message in the sequence (if available). In case of a cancel error, the receiver shall discard all the packets with the same TransactionId as indicated in the MBIM_ERROR_CANCEL message.

9.4 CONTROL MESSAGE SENT FROM FUNCTION TO HOST

Table 9-9: Control Messages sent from function to host

Message Identifier	Value	Description
MBIM_OPEN_DONE	80000001h	Device response to initialization request (MBIM_OPEN_MSG).
MBIM_CLOSE_DONE	80000002h	Device response to Close request (MBIM_CLOSE_MSG).
MBIM_COMMAND_DONE	80000003h	Device response to command' CID request (MBIM_COMMAND_MSG)
MBIM_FUNCTION_ERROR_MSG	80000004h	Indicates an error in the MBIM communication.
MBIM_INDICATE_STATUS_MSG	80000007h	Indicates unsolicited network or devices status changes

Control message layout for messages sent from the function to the host.

The data of the control messages is retrieved by sending a GET_ENCAPSULATED_RESPONSE to the function. The information will be contained in the data returned from this command. See section 8.1.2 on GetEncapsulatedResponse.

9.4.1 MBIM_OPEN_DONE

The function initializes itself. Statistics per section 10.5.34 must be reset at this point. Notifications for CIDs defined in this specification are enabled upon OPEN_DONE completion. Vendor extension

notifications are off by default. All filters in section 10.5.30 (see Table 10-142: MBIM_DEVICE_SERVICE_SUBSCRIBE_LIST) and section 10.5.37 (MBIM_CID_IP_PACKET_FILTERS) are to be cleared at this point.

The function shall respond to the MBIM_OPEN_MSG message with an MBIM_OPEN_DONE message in which the TransactionId must match the TransactionId in the MBIM_OPEN_MSG.

The Status field shall be set to MBIM_STATUS_SUCCESS if the function initialized successfully; otherwise it shall be set to an error code indicating the failure.

Table 9-10: MBIM_OPEN_DONE

Offset	Size	Field	Type	Description
0	12	MessageHeader	MBIM_MESSAGE_HEADER	A message header as specified in Table 9-1: MBIM_MESSAGE_HEADER with MessageType set to MBIM_OPEN_DONE.
12	4	Status	MBIM_STATUS_CODES	Specifies the status of processing the Open request. See Table 9-15: MBIM_STATUS_CODES

9.4.2 MBIM_CLOSE_DONE

The device will respond to the MBIM_CLOSE_MSG message with an MBIM_CLOSE_DONE message. The TransactionId of the responding message must match the TransactionId in the MBIM_CLOSE_MSG.

Table 9-11: MBIM_CLOSE_DONE

Offset	Size	Field	Type	Description
0	12	MessageHeader	MBIM_MESSAGE_HEADER	A message header as specified in Table 9-1: MBIM_MESSAGE_HEADER with MessageType set to MBIM_CLOSE_DONE.

12	4	Status	MBIM_STATUS_CODES	Specifies the status of processing the Close request. This field shall always be set to MBIM_STATUS_SUCCESS. See Table 9-15: MBIM_STATUS_CODES.

9.4.3 MBIM_COMMAND_DONE

The function shall send MBIM_COMMAND_DONE to the host in response to MBIM_COMMAND_MSG. This message is used to relay the result of executing the command, and may also contain additional result values.

Table 9-12 defines the structure of the MBIM_COMMAND_DONE message. The TransactionId of the responding message must match the TransactionId in the MBIM_COMMAND_MSG.

Table 9-12: MBIM_COMMAND_DONE

Offset	Size	Field	Type	Description
0	12	MessageHeader	MBIM_MESSAGE_HEADER	A message header as specified in Table 9-1: MBIM_MESSAGE_HEADER with MessageType set to MBIM_COMMAND_DONE.
12	8	FragmentHeader	MBIM_FRAGMENT_HEADER	A fragmentation header as specified in Table 9-2: MBIM_FRAGMENT_HEADER For fragmentation considerations, see section 9.5 (Fragmentation of messages).
20	16	DeviceServiceId	UUID	A 16-byte UUID that identifies the device service the following CID value applies.
36	4	CID	UINT32	Specifies the CID that identifies the parameter being that this message is completing

Offset	Size	Field	Type	Description
40	4	Status	MBIM_STATUS_CODES	Specifies the status of processing the CID set/query request. See section 9.4.5.
44	4	InformationBufferLength	UINT32	Length of the InformationBuffer. May exceed the length of this message if fragmented.
48		InformationBuffer	DATABUFFER	Holds any information retuned from the function. May be empty.

9.4.4 MBIM_INDICATE_STATUS_MSG

The function may send MBIM_INDICATE_STATUS_MSG to the host via the control channel in an unsolicited fashion at any time that the device has been opened. This message is used to indicate a change in the status of the device. The specific event that is being indicated is identified DeviceServiceId and CID. The contents of the InformationBuffer are specific to the CID.

9.4.4.1 MBIM_INDICATE_STATUS_MSG FORMAT

Table 9-13: MBIM_INDICATE_STATUS_MSG Message

Offset	Size	Field	Type	Description
0	12	MBIM_Message_Header	MBIM_MESSAGE_HEADER	A message header as specified in Table 9-1: MBIM_MESSAGE_HEADER with MessageType set to MBIM_INDICATE_STATUS.
12	8	FragmentHeader	MBIM_FRAGMENT_HEADER	A fragmentation header as specified in Table 9-2: MBIM_FRAGMENT_HEADER For fragmentation considerations, see section 9.5 (Fragmentation of messages).

20	16	DeviceServiceId	UUID	A 16 byte UUID that identifies the device service to which the following Status ID applies.
36	4	CID	UINT32	Specifies the CID that this indication matches.
40	4	InformationBufferLength	UINT32	Length of the InformationBuffer.
44		InformationBuffer	DATABUFFER	Variable length InformationBuffer. Possibly empty.

9.4.5 USE OF THE STATUS CODES

For a complete list of the Status codes and their descriptions, please refer to Table 9-15: MBIM_STATUS_CODES. If the CID is successful, the function shall set the Status field to MBIM_STATUS_SUCCESS in the MBIM_COMMAND_DONE, and send back whatever information is specified by the CID in the InformationBuffer field (Table 9-12: MBIM_COMMAND_DONE) and, possibly, in the dynamic DataBuffer field as well. If the Status field returned to the host is not equal to MBIM_STATUS_SUCCESS, the device must set the InformationBufferLength to 0, indicating an empty InformationBuffer. However, there are exceptions to this rule. In some cases called out explicitly below, a Status value other than MBIM_STATUS_SUCCESS could be accompanied by further information in the field NwError within the InformationBuffer. In these cases, the function must follow the specification for the CID in question and send back the required information, while zeroing out all irrelevant fields. Such cases are specified for the following CIDs:

- MBIM_CID_REGISTER_STATE
- MBIM_CID_PACKET_SERVICE
- MBIM_CID_CONNECT
- MBIM_CID_SERVICE_ACTIVATION
- MBIM_CID_PIN

Of course, future CIDs defined using the extensibility defined in this specification (see section 10.6 on MBIM Service and CID extensibility) may possibly create other similar exceptions.

If the function does not implement the CID, then the function shall fail the request with MBIM_STATUS_NO_DEVICE_SUPPORT.

Section 10.3 provides more detailed guidance about the failure status codes to be returned for each CID. Within Section 10.3, generic status codes that apply to all or many CIDs are not repeated for each CID. These Generic Status Codes, along with the CIDs to which they apply, are listed in Table 9-14: Generic Status Codes and their Relevant CIDs. These Generic Status codes are usable by all CIDs. Nevertheless, some of them are still mentioned explicitly in the CID subsections for completeness sake, even if such mention is not necessary (e.g., MBIM_STATUS_BUSY).

In the absence of a better status code to describe a failure, the function must return the generic failure code MBIM_STATUS_FAILURE.

Table 9-14: Generic Status Codes and their Relevant CIDs

Generic Status Codes	Relevant Commands
MBIM_STATUS_SUCCESS	All CIDs
MBIM_STATUS_BUSY	All CIDs
MBIM_STATUS_FAILURE	All CIDs
MBIM_STATUS_SIM_NOT_INSERTED	MBIM_CID_RADIO_STATE MBIM_CID_PIN MBIM_CID_PIN_LIST MBIM_CID_HOME_PROVIDER MBIM_CID_PREFERRED_PROVIDERS MBIM_CID_VISIBLE_PROVIDERS MBIM_CID_REGISTER_STATE MBIM_CID_PACKET_SERVICE MBIM_CID_CONNECT MBIM_CID_PROVISIONED_CONTEXTS MBIM_CID_SERVICE_ACTIVATION MBIM_CID_SMS_CONFIGURATION MBIM_CID_SMS_READ MBIM_CID_SMS_SEND MBIM_CID_SMS_DELETE MBIM_CID_SMS_MESSAGE_STORE_STATUS

Generic Status Codes	Relevant Commands
MBIM_STATUS_BAD_SIM	MBIM_CID_PIN MBIM_CID_PIN_LIST MBIM_CID_HOME_PROVIDER MBIM_CID_PREFERRED_PROVIDERS MBIM_CID_REGISTER_STATE MBIM_CID_PACKET_SERVICE MBIM_CID_CONNECT MBIM_CID_PROVISIONED_CONTEXTS MBIM_CID_SERVICE_ACTIVATION MBIM_CID_SMS_CONFIGURATION MBIM_CID_SMS_READ MBIM_CID_SMS_SEND MBIM_CID_SMS_DELETE MBIM_CID_SMS_MESSAGE_STORE_STATUS
MBIM_STATUS_PIN_REQUIRED	MBIM_CID_PIN MBIM_CID_PIN_LIST MBIM_CID_HOME_PROVIDER MBIM_CID_PREFERRED_PROVIDERS MBIM_CID_VISIBLE_PROVIDERS MBIM_CID_REGISTER_STATE MBIM_CID_PACKET_SERVICE MBIM_CID_CONNECT MBIM_CID_PROVISIONED_CONTEXTS MBIM_CID_SERVICE_ACTIVATION MBIM_CID_SMS_CONFIGURATION MBIM_CID_SMS_READ MBIM_CID_SMS_SEND MBIM_CID_SMS_DELETE MBIM_CID_SMS_MESSAGE_STORE_STATUS MBIM_CID_PHONEBOOK_CONFIGURATION MBIM_CID_PHONEBOOK_READ MBIM_CID_PHONEBOOK_DELETE MBIM_CID_PHONEBOOK_WRITE MBIM_CID_STK_PAC MBIM_CID_STK_TERMINAL_RESPONSE MBIM_CID_STK_ENVELOPE
MBIM_STATUS_NO_DEVICE_SUPPORT	All CIDs

Generic Status Codes	Relevant Commands
MBIM_STATUS_NOT_INITIALIZED	MBIM_CID_DEVICE_CAPS MBIM_CID_PIN MBIM_CID_PIN_LIST MBIM_CID_HOME_PROVIDER MBIM_CID_REGISTER_STATE MBIM_CID_PACKET_SERVICE MBIM_CID_CONNECT MBIM_CID_PROVISIONED_CONTEXTS MBIM_CID_SERVICE_ACTIVATION MBIM_CID_SMS_CONFIGURATION MBIM_CID_SMS_READ MBIM_CID_SMS_SEND MBIM_CID_SMS_DELETE MBIM_CID_SMS_MESSAGE_STORE_STATUS MBIM_CID_PHONEBOOK_CONFIGURATION MBIM_CID_PHONEBOOK_READ MBIM_CID_PHONEBOOK_DELETE MBIM_CID_PHONEBOOK_WRITE MBIM_CID_STK_PAC MBIM_CID_STK_TERMINAL_RESPONSE MBIM_CID_STK_ENVELOPE

Table 9-15: MBIM_STATUS_CODES

Name	Value	Description
MBIM_STATUS_SUCCESS	0	The operation succeeded.
MBIM_STATUS_BUSY	1	The operation failed because the device is busy. In the absence of any explicit information from the function to clear this condition, the host can use subsequent actions by the function (e.g., notifications or command completions) as a hint to retry the failed operation.
MBIM_STATUS_FAILURE	2	The operation failed (a generic failure).
MBIM_STATUS_SIM_NOT_INSERTED	3	The operation failed because the SIM card was not fully inserted in to the device.
MBIM_STATUS_BAD_SIM	4	The operation failed because the SIM card is bad and cannot be used any further.
MBIM_STATUS_PIN_REQUIRED	5	The operation failed because a PIN must be entered to proceed.
MBIM_STATUS_PIN_DISABLED	6	The operation failed because the PIN is disabled.

MBIM_STATUS_NOT_REGISTERED	7	The operation failed because the device is not registered with any network.
MBIM_STATUS_PROVIDERS_NOT_FOUND	8	The operation failed because no network providers could be found.
MBIM_STATUS_NO_DEVICE_SUPPORT	9	The operation failed because the device does not support the operation.
MBIM_STATUS_PROVIDER_NOT_VISIBLE	10	The operation failed because the service provider is not currently visible.
MBIM_STATUS_DATA_CLASS_NOT_AVAILABLE	11	The operation failed because the requested data-class was not available.
MBIM_STATUS_PACKET_SERVICE_DETACHED	12	The operation failed because the packet service is detached.
MBIM_STATUS_MAX_ACTIVATED_CONTEXTS	13	The operation failed because the maximum number of activated contexts has been reached.
MBIM_STATUS_NOT_INITIALIZED	14	The operation failed because the device is in the process of initializing. Retry the operation after the ReadyState of the device changes to MBIMSubscriberReadyStateInitialized.
MBIM_STATUS_VOICE_CALL_IN_PROGRESS	15	The operation failed because a voice call is in progress.
MBIM_STATUS_CONTEXT_NOT_ACTIVATED	16	The operation failed because the context is not activated.
MBIM_STATUS_SERVICE_NOT_ACTIVATED	17	The operation failed because service is not activated.
MBIM_STATUS_INVALID_ACCESS_STRING	18	The operation failed because the access string is invalid.
MBIM_STATUS_INVALID_USER_NAME_PWD	19	The operation failed because the username and/or password supplied are invalid.
MBIM_STATUS_RADIO_POWER_OFF	20	The operation failed because the radio is currently powered off.
MBIM_STATUS_INVALID_PARAMETERS	21	The operation failed because of invalid parameters.
MBIM_STATUS_READ_FAILURE	22	The operation failed because of a read failure.
MBIM_STATUS_WRITE_FAILURE	23	The operation failed because of a write failure.
Reserved	24	
MBIM_STATUS_NO_PHONEBOOK	25	The phonebook operation failed because there is no phone book.
MBIM_STATUS_PARAMETER_TOO_LONG	26	A parameter with dynamic size is larger than the function can handle
MBIM_STATUS_STK_BUSY	27	The SIM Toolkit application on the SIM card is busy and the command could not be processed
MBIM_STATUS_OPERATION_NOT_ALLOWED	28	The operation failed because the operation is not allowed.

MBIM_STATUS_MEMORY_FAILURE	29	The phonebook or sms operation failed because the because of device or SIM memory failure.
MBIM_STATUS_INVALID_MEMORY_INDEX	30	The phonebook or sms operation failed because of an invalid memory index.
MBIM_STATUS_MEMORY_FULL	31	The phonebook or sms operation failed because the device or SIM memory is full.
MBIM_STATUS_FILTER_NOT_SUPPORTED	32	The phonebook or sms operation failed because the filter type is not supported.
MBIM_STATUS_DSS_INSTANCE_LIMIT	33	Attempt to open a device service failed because the number of opened streams has reached the device service instance limit for this service.
MBIM_STATUS_INVALID_DEVICE_SERVICE_OPERATION	34	The device service operation attempted is invalid.
MBIM_STATUS_AUTH_INCORRECT_AUTN	35	The device sets this error on an AKA or AKAPrime challenge response when the AKA or AKAPrime challenge sent has incorrect AUTN.
MBIM_STATUS_AUTH_SYNC_FAILURE	36	The device sets this error on a an AKA or AKAPrime challenge response when the AKA or AKAPrime challenge sent has synchronization failure. When this error code is returned the AUTS field would be set.
MBIM_STATUS_AUTH_AMF_NOT_SET	37	The device sets this error on an AKAPrime challenge response when the AKAPrime challenge sent does not have the AMF bit set to 1.
MBIM_STATUS_CONTEXT_NOT_SUPPORTED	38	The operation failed because it could not support the type of context identified by ContextType.
MBIM_STATUS_SMS_UNKNOWN_SMSC_ADDRESS	100	The SMS operation failed because the service center address is either invalid or unknown.
MBIM_STATUS_SMS_NETWORK_TIMEOUT	101	The SMS operation failed because of a network timeout.
MBIM_STATUS_SMS_LANG_NOT_SUPPORTED	102	The SMS operation failed because the SMS language is not supported.This applies to CDMA based devices only.
MBIM_STATUS_SMS_ENCODING_NOT_SUPPORTED	103	The SMS operation failed because the SMS encoding is not supported. This applies to CDMA based device only.
MBIM_STATUS_SMS_FORMAT_NOT_SUPPORTED	104	The SMS operation failed because the SMS format is not supported.
Device service specific status commands	80000000h-FFFFFFFFh	

If the MSB is set in the status message the DeviceServiceId (see section 9.4.4) shall be used to identify the status message namespace. This namespace is thus unique for each DeviceServiceId.

9.5 FRAGMENTATION OF MESSAGES

MBIM_COMMAND_MSG, MBIM_COMMAND_DONE and MBIM_INDICATE_STATUS_MSG may be split across multiple encapsulated commands and responses due to limitation in the maximum size of a USB transfer across the control channel.

Fragmented messages can be identified by the TotalFragment field having a value greater than 1. The first fragment contains all of the message headers; the following fragments only contain the headers up to the current fragment field followed by the continuation of the InformationBuffer field.

See the below example of a fragmented MBIM_COMMAND_DONE message:

Assume the max USB control transfer is 4096.
The message payload is 14000 bytes.
The message will be split across 4 messages.
Only the first fragment will contain DeviceServiceId, CID, Status, and InformationBufferLength.
The remaining fragments will hold InformationBuffer data following the CurrentFragment Field.
MessageLength is the actual length of each fragment.
InformationBufferLength is the length the whole InformationBuffer.

MessageType	MessageIdType	MessageType	MessageType
MessageLength(4096)	MessageLength(4096)	MessageLength(4096)	MessageLength(1820)
TransactionId(99)	TransactionId(99)	TransactionId(99)	TransactionId(99)
TotalFragments(4)	TotalFragments(4)	TotalFragments(4)	TotalFragments(4)
CurrentFragment(0)	CurrentFragment(1)	CurrentFragment(2)	CurrentFragment(3)
DeviceServiceId	InformationBuffer	InformationBuffer	InformationBuffer
CID			
Status			
InformationBufferLength (14000)			
InformationBuffer			

Figure 11: MBIM Message Fragmentation

In a fragment, MessageLength is the length of the current fragment, whereas InformationBufferLength is the total length of the InformationBuffer prior to fragmentation. Fragments shall be transmitted in back to back transfers without intermixing fragments from other messages.

A host or function that receives fragmented messages shall send an MBIM_X_ERROR_MSG with error code MBIM_ERROR_TIMEOUT_FRAGMENT; if the time between the fragments is too long (see 9.3.4.1).

If a host or function receives a fragment that is out-of-order or is improperly formatted or in any other way is faulty, the receiver shall send an MBIM_X_ERROR_MSG with the corresponding error code to the sender (see Table 9-8: MBIM_PROTOCOL_ERROR_CODES).

If a host or function receives a faulty fragment (e.g incorrect length, out of sequenced fragment, etc.), all the fragments with the same TransactionId shall be discarded. If the function receives a first fragment of a new command with a new TransactionId, the function shall discard the previous command and start handling the new command. If the function receives a fragment that is not the first fragment of a new command with a new TransactionId, both commands shall be discarded. One MBIM_FUNCTION_ERROR_MSG message per TransactionId shall be sent.

Per section 1.2, this specification does not mandate behavior on hosts, so normative language above (and elsewhere) does not apply to hosts. Accordingly, devices must be prepared to handle any eventuality (for example, intermixed fragments from different messages).

10 MBIM-SPECIFIC COMMAND MECHANISM

Each CID allows either or both Query and Set operations from the host to the function. A Query enables the host to inquire as to the value or status of a given capability at the function. A Set enables the host to change a given capability at a function. CIDs may also allow indications from the function to the host, both to notify about command completions as well as for unsolicited events (also known as “unsolicited indications”, “notifications” or “events”).

To retrieve the information associated with a CID, the HOST sends a SEND_ENCAPSULATED_COMMAND with a payload of an MBIM_COMMAND_MSG. The CID field of the structure will have the value corresponding to the CID to be queried or set. When the function has retrieved the information in question, it notifies the host via the interrupt channel. The host then sends a GET_ENCAPSULATED_RESPONSE to the function to retrieve the information. If there is undue delay in either the function’s notification via the interrupt channel, or in the host’s response by issuing the GET_ENCAPSULATED_RESPONSE, this can be handled via an MBIM_ERROR_CANCEL (see section 9.3.4.6).

The information is formatted as an MBIM_COMMAND_DONE message, and the CID-specific information is located within the InformationBuffer.

A CID request to a function is identified via two parts: the DeviceServiceId and the CID value. The DeviceServiceId is the UUID which uniquely identifies the service for which the CID value or notification (section 10.1) is relevant. The CID value is a 32 bit value.

For transmission between a host and a function, a UUID is encoded as a 128-bit object, as follows: The fields are encoded as 16 octets, with the sizes and order of the fields defined below, and with each field encoded with the Most Significant Byte first (known as network byte order or big-endian order).

Table 10-1: Transmission structure of a UUID

Offset	Size	Field	Description
0	4	Data1	The 8 character portion of the UUID string (“a”) as a bigendian 32 bit value
4	2	Data2	The first 4 character portion of the UUID string (“b”) as a big-endian 16 bit value
6	2	Data3	The first 4 character portion of the UUID string (“c”) as a big-endian 16 bit value
8	8	Data4	This field is composed of the third 4 character portion (“d”) of the UUID string and the 12 character portion (“e”) of the UUID string. The byte values are transmitted in the order listed in the string.

For example, a UUID string as follows would be transmitted as shown in Table 10-2: UUID transmission example.

{a1a2a3a4-b1b2-c1c2-d1d2-e1e2e3e4e5e6}

Table 10-2: UUID transmission example

a1	a2	a3	a4	b1	b2	c1	c2	d1	d2	e1	e2	e3	e4	e5	e6

10.1 MBIM SERVICES AND THEIR CID VALUES

This specification defines the following services:

Table 10-3: Services Defined by MBIM

Service Name	UUID	UUID Value
Basic IP Connectivity	UUID_BASIC_CONNECT	a289cc33-bcbb-8b4f-b6b0-133ec2aae6df
SMS	UUID_SMS	533fbeeb-14fe-4467-9f90-33a223e56c3f
USSD (Unstructured Supplementary Service Data)	UUID_USSD	e550a0c8-5e82-479e-82f7-10abf4c3351f
Phonebook	UUID_PHONEBOOK	4bf38476-1e6a-41db-b1d8-bed289c25bdb
STK (SIM Toolkit)	UUID_STK	d8f20131-fcb5-4e17-8602-d6ed3816164c
Authentication	UUID_AUTH	1d2b5ff7-0aa1-48b2-aa52-50f15767174e
Device Service Stream	UUID_DSS	c08a26dd-7718-4382-8482-6e0d583c4d0e

The following table specifies the command code for each CID. Additionally, it also shows informatively (normative information is contained in each detailed per-CID explanation under section 10.5), whether the CID supports Query, Set or Events (notifications).Per-CID sections (see section 10.5) have details about what (if any) is the structure used for input parameters on a Query (Query InformationBuffer Payload), on a Set (Set InformationBuffer Payload), on command completion (Completion InformationBuffer payload), and on unsolicited Indications (asynchronous events also known as Notifications). Unsolicited indications are delivered via MBIM_INDICATE_STATUS_MSG from the device to the host.

Table 10-4: Defined CIDs and Message Formats for Commands and Results

CID	UUID_	Command Code	Set	Query	Notification
MBIM_CID_DEVICE_CAPS	BASIC_CONNECT	1	N	Y	N
MBIM_CID_SUBSCRIBER_READY_STATUS	BASIC_CONNECT	2	N	Y	Y
MBIM_CID_RADIO_STATE	BASIC_CONNECT	3	Y	Y	Y
MBIM_CID_PIN	BASIC_CONNECT	4	Y	Y	N
MBIM_CID_PIN_LIST	BASIC_CONNECT	5	N	Y	N
MBIM_CID_HOME_PROVIDER	BASIC_CONNECT	6	Y	Y	N
MBIM_CID_PREFERRED_PROVIDERS	BASIC_CONNECT	7	Y	Y	Y
MBIM_CID_VISIBLE_PROVIDERS	BASIC_CONNECT	8	N	Y	N
MBIM_CID_REGISTER_STATE	BASIC_CONNECT	9	Y	Y	Y
MBIM_CID_PACKET_SERVICE	BASIC_CONNECT	10	Y	Y	Y
MBIM_CID_SIGNAL_STATE	BASIC_CONNECT	11	Y	Y	Y
MBIM_CID_CONNECT	BASIC_CONNECT	12	Y	Y	Y
MBIM_CID_PROVISIONED_CONTEXTS	BASIC_CONNECT	13	Y	Y	Y
MBIM_CID_SERVICE_ACTIVATION	BASIC_CONNECT	14	Y	N	N
MBIM_CID_IP_CONFIGURATION	BASIC_CONNECT	15	N	Y	Y
MBIM_CID_DEVICE_SERVICES	BASIC_CONNECT	16	N	Y	N
MBIM_CID_DEVICE_SERVICE_SUBSCRIBE_LIST	BASIC_CONNECT	19	Y	N	N
MBIM_CID_PACKET_STATISTICS	BASIC_CONNECT	20	N	Y	N
MBIM_CID_NETWORK_IDLE_HINT	BASIC_CONNECT	21	Y	Y	N
MBIM_CID_EMERGENCY_MODE	BASIC_CONNECT	22	N	Y	Y
MBIM_CID_IP_PACKET_FILTERS	BASIC_CONNECT	23	Y	Y	N
MBIM_CID_MULTICARRIER_PROVIDERS	BASIC_CONNECT	24	Y	Y	Y
MBIM_CID_SMS_CONFIGURATION	SMS	1	Y	Y	Y
MBIM_CID_SMS_READ	SMS	2	N	Y	Y
MBIM_CID_SMS_SEND	SMS	3	Y	N	N
MBIM_CID_SMS_DELETE	SMS	4	Y	N	N
MBIM_CID_SMS_MESSAGE_STORE_STATUS	SMS	5	N	Y	Y
MBIM_CID_USSD	USSD	1	Y	N	Y
MBIM_CID_PHONEBOOK_CONFIGURATION	PHONEBOOK	1	N	Y	Y
MBIM_CID_PHONEBOOK_READ	PHONEBOOK	2	N	Y	N
MBIM_CID_PHONEBOOK_DELETE	PHONEBOOK	3	Y	N	N
MBIM_CID_PHONEBOOK_WRITE	PHONEBOOK	4	Y	N	N
MBIM_CID_STK_PAC	STK	1	Y	Y	Y
MBIM_CID_STK_TERMINAL_RESPONSE	STK	2	Y	N	N
MBIM_CID_STK_ENVELOPE	STK	3	Y	Y	N
MBIM_CID_AKA_AUTH	AUTH	1	N	Y	N

CID	UUID_	Command Code	Set	Query	Notification
MBIM_CID_AKAP_AUTH	AUTH	2	N	Y	N
MBIM_CID_SIM_AUTH	AUTH	3	N	Y	N
MBIM_CID_DSS_CONNECT	DSS	1	Y	N	N

10.2 DATA TYPES

MBIM uses the Data Types shown in Table 10-5: Data Types.

Arrays use the following notation: <base-type>[number-of-elements]. For example, an array with 16 elements of type UINT8 is denoted by: UINT8[16]

Table 10-5: Data Types

Types	Definition	Comments
UINT8	8-bit unsigned number	Used for 8-bit quantities and as a base type for arrays.
UUID	UINT8[16]	Used for UUID entities like Device Service IDs. See Table 10-1: Transmission structure of a UUID
UINT16	16-bit unsigned number	Used for 16-bit quantities.
UINT32	32-bit unsigned number	Used for 32-bit quantities.
UINT64	64-bit unsigned number	Used for 64-bit quantities.
DATABUFFER	UINT8[variable-size]	The second part of the CID storage structure used for variable size fields (see section 10.3 CID Storage for Fixed and Variable Fields).
OFFSET	UINT32	Offset in bytes, calculated from the beginning of the CID structure, to a variable-sized data in the DATABUFFER. See section 10.4 Byte ordering and String Format.
SIZE (min..max)	UINT32	Size in bytes of the variable-sized data in the DATABUFFER. See section 10.4 Byte ordering and String Format.

OL_PAIR_LIST	List of Offset-Length Pairs	This data type is preceded by count (UINT32) of the number of pairs in the list. The first element in each pair is a 4 byte field with the Offset into the DataBuffer, calculated from the beginning (offset 0) of the enclosing structure to some data. The second element of easy pair is a 4 byte field with the size of the data pointed at by the first element.

10.3 CID STORAGE FOR FIXED AND VARIABLE FIELDS

The CID structures are stored in the InformationBuffer in an MBIM message see Table 9-6: MBIM_COMMAND_MSG Message. The CID structure consists of two contiguous structures: the first one, a static structure, is used for fixed size fields; the second part is a data buffer ("DataBuffer") used for variable size fields.

All fields (either static or variable size) start at a 4-byte boundary. Hence, all field offsets shall be multiples of 4. Between the end of a field (not necessarily at a 4-byte boundary) and the beginning of the next field (always at a 4-byte boundary), padding may be required. Padding bytes must be set to 0 upon transmission and ignored upon reception.

Strings, being variable in size, are stored in the DataBuffer. On the other hand, offsets to strings in the DataBuffer and sizes of strings, being fixed fields, are stored in the static part of the CID structures.

The offsets into the DataBuffer are counted from the beginning of the CID structure. Offsets shall be linearly increasing, non-overlapping and non-sparse except for alignment. An offset of zero indicates that the string is NULL (for example, to indicate the default APN per Table 10-67: MBIM_SET_CONNECT). If the offset is zero the size must also be set to zero.

The size field indicates the size in bytes of the variable field in the DataBuffer. The value for the size field shall only include the real size of the variable field's payload, without padding.

If the size of the payload in the variable field is not a multiple of 4 bytes, the field shall be padded up to the next 4 byte multiple. This shall be true even for the last payload in DataBuffer.

The length of an array, as reflected by the size field, shall be a multiple of the size of the base type. For example, the size of UNICODE strings shall be a multiple of 2 bytes.

The function shall reject incoming messages that don't follow the rules for variable-length encoding by setting MBIM_STATUS_INVALID_PARAMETERS as the status code in the MBIM_COMMAND_DONE message. For malformed notifications sent from the function to the host the behavior is unspecified.

10.4 BYTE ORDERING AND STRING FORMAT

Unless otherwise specified, all fields of MBIM control messages and CID related structures are in little-endian format per [USB30] section 7.1. Unless otherwise specified, all strings use UNICODE UTF-16LE encodings limited to characters from the Basic Multilingual Plane. Strings shall not be terminated by a NULL character. Instead, their length is specified explicitly.

10.5 MBIM CIDS DETAILED DESCRIPTIONS

This section contains the detailed CID descriptions for the defined device services.

All offsets in the CID sections below are calculated from the beginning of the InformationBuffer in Table 9-6: MBIM_COMMAND_MSG Message.

10.5.1 MBIM_CID_DEVICE_CAPS

10.5.1.1 DESCRIPTION

Query only:

The function prepares and returns an MBIM_DEVICE_CAPS_INFO structure in response to a MBIM_COMMAND_MSG with UUID_BASIC_CONNECT and CID MBIM_CID_DEVICE_CAPS.

10.5.1.2 PARAMETERS

Table 10-6: Parameters

	Set	Query	Notification
Command	NA	Empty	NA
Response	NA	MBIM_DEVICE_CAPS_INFO	NA

10.5.1.3 DATA STRUCTURES

Table 10-7: MBIM_DEVICE_TYPE

Types	Value	Description
MBIMDeviceTypeUnknown	0	The function type is unknown.

Types	Value	Description
MBIMDeviceTypeEmbedded	1	The function type is embedded in the system
MBIMDeviceTypeRemovable	2	The function is implemented within a device that is removable, for example, a pluggable USB device.
MBIMDeviceTypeRemote	3	The function type is remote, for example, a tethered cellular phone modem.

The MBIM_CELLULAR_CLASS bitmap is used to indicate the relevant cellular technologies. For example, dual-mode functions with both GSM and CDMA capabilities could turn both corresponding bits on in the context of function capabilities (Table 10-14: MBIM_DEVICE_CAPS_INFO), whereas only one technology would typically be indicated in the context of MB service providers (Table 10-37: MBIM_PROVIDER). A multi-mode device (both GSM and CDMA) with a single carrier must use MBIMCellularClassGsm as the primary mode. The device should use the primary mode to choose a single service where multiple exist among the cellular technologies. Specific requirements are highlighted where applicable in the following sections.

Table 10-8: MBIM_CELLULAR_CLASS

Types	Mask
MBIMCellularClassGsm	1
MBIMCellularClassCdma	2

Table 10-9: MBIM_VOICE_CLASS

Types	Value	Description
MBIMVoiceClassUnknown	0	The device uses an unknown method to support voice connections.
MBIMVoiceClassNoVoice	1	The device does not support voice connections
MBIMVoiceClassSeparateVoiceData	2	The device supports separate voice and data connections

MBIMVoiceClassSimultaneousVoiceData	3	The device supports simultaneous voice and data connections

This is the voice class of the device. This member informs the host about the presence of circuit voice service, and how such service interacts with data service.

Table 10-10: MBIM_SIM_CLASS

Types	Mask	Description
MBIMSimClassSimLogical	1	Functions that do not have a physical SIM, must set this bit. For example, devices supporting embedded SIM (embedded on QFN8 chips) must report this bit.
MBIMSimClassSimRemovable	2	Functions supporting physical SIMs that are removable by the end-user must set this bit. E.g., 3GPP devices supporting SIM, U-SIM, I-SIM, 3GPP2 devices supporting R-UIM, CSIM, etc., must report this bit as 1.

A bitmap that represents which radio technologies are supported by the function. The following table shows the possible values for this member.

Table 10-11: MBIM_DATA_CLASS

Types	Mask	Description
MBIMDataClassNone	0h	
MBIMDataClassGPRS	1h	
MBIMDataClassEDGE	2h	
MBIMDataClassUMTS	4h	
MBIMDataClassHSDPA	8h	
MBIMDataClassHSUPA	10h	
MBIMDataClassLTE	20h	
Reserved	40h-8000h	Reserved for future GSM classes
MBIMDataClass1XRTT	10000h	
MBIMDataClass1XEVDO	20000h	
MBIMDataClass1XEVDORevA	40000h	

MBIMDataClass1XEVDV	80000h	
MBIMDataClass3XRTT	100000h	
MBIMDataClass1XEVDORevB	200000h	
MBIMDataClassUMB	400000h	
Reserved	800000h-40000000h	Reserved for future CDMA classes
MBIMDataClassCustom	80000000h	

This is a bitmap that represents the type of SMS messages and directional flow that the device supports. The following table shows the valid SMS capabilities settings. Table 10-12: MBIM_SMS_CAPS

Types	Mask
MBIMSmsCapsNone	0
MBIMSmsCapsPduReceive	1
MBIMSmsCapsPduSend	2
MBIMSmsCapsTextReceive	4
MBIMSmsCapsTextSend	8

This is a bitmap that represents the control capabilities that the device supports. The following table shows the valid MBIM_CTRL_CAPS settings for GSM-based and CDMA-based devices. Devices that support CDMA must specify MBIMCtrlCapsCdmaMobileIp, or MBIMCtrlCapsCdmaSimpleIp, or both flags to inform the host about the type of IP that the device supports.

Table 10-13: MBIM_CTRL_CAPS

Types	Mask	Description
MBIMCtrlCapsNone	0h	No flags set
MBIMCtrlCapsRegManual	1h	Indicates whether the device allows manual network selection. Functions for GSM-based devices may specify this flag. Functions for CDMA-based devices must not specify this flag.
MBIMCtrlCapsHwRadioSwitch	2h	Indicates the presence of a hardware radio power switch

MBIMCtrlCapsCdmaMobileIp	4h	Indicates that the CDMA-based function is configured to support mobile IP. This flag applies only to CDMA-based functions
MBIMCtrlCapsCdmaSimpleIp	8h	Indicates that the CDMA-based function is configured for simple IP support. This flag applies only to CDMA-based functions
MBIMCtrlCapsMultiCarrier	10h	Indicates that the device can work with multiple-providers

As the connection credentials (AccessString, UserName, and Password) for simple IP are pre-configured, function firmware that supports both simple IP and mobile IP must report both capabilities, regardless of the runtime configuration or network capability in the current location.

Functions that support secured connection credentials, such as those using R-UIM, CSIM or SIM-less implementation, shall use a blank placeholder provisioned context.

These are capabilities of the device, and do not reflect what may be possible at any given point in time, as this is often determined by other factors such as network characteristics, technology, etc.

10.5.1.4 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero.

10.5.1.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-14: MBIM_DEVICE_CAPS_INFO

Offset	Size	Field	Type	Description
0	4	DeviceType	MBIM_DEVICE_TYPE	Device type. See Table 10-7: MBIM_DEVICE_TYPE.
4	4	CellularClass	MBIM_CELLULAR_CLASS	Cellular class. See Table 10-8: MBIM_CELLULAR_CLASS.
8	4	VoiceClass	MBIM_VOICE_CLASS	Voice class. See Table 10-9: MBIM_VOICE_CLASS

Offset	Size	Field	Type	Description
12	4	SimClass	MBIM_SIM_CLASS	SIM class. See Table 10-10: MBIM_SIM_CLASS
16	4	DataClass	MBIM_DATA_CLASS	Data class. See Table 10-11: MBIM_DATA_CLASS.
20	4	SmsCaps	MBIM_SMS_CAPS	SMS capabilities. See This is a bitmap that represents the type of SMS messages and directional flow that the device supports. The following table shows the valid SMS capabilities settings. Table 10-12: MBIM_SMS_CAPS
24	4	ControlCaps	MBIM_CTRL_CAPS	Control capabilities. See Table 10-13: MBIM_CTRL_CAPS
28	4	MaxSessions	UINT32	The maximum number of activated IP data stream sessions that are supported by the function. The protocol allows up to 256 simultaneous IP streams. See Section 7 on Data Transport.
32	4	CustomDataClassOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string CustomDataClass that represents the name of the custom data class if the DataClass bitmask contains 80000000h. Max length is 11 (16-bit) characters. If the DataClass bitmask does not contain 80000000h, then this field is reserved and shall be encoded as zero by the sender, and ignored by the receiver.
36	4	CustomDataClassSize	SIZE (0..22)	Size used for CustomDataClass

Offset	Size	Field	Type	Description
40	4	DeviceIdOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string DeviceId that represents the function ID. For GSM-based devices, the string must conform to the International Mobile Equipment Identity (IMEI) format (up to 15 digits). For CDMA-based devices, the string must conform to either the Electronic Serial Number (ESN, 8 or 11 digits) or Mobile Equipment Identifier (MEID, 14 or 18 digits) formats. This value should be stored in the device's memory and must be available even when the device/SIM requires a PIN to unlock. Max length is 18 (16-bit) characters. For multi-mode devices, only the GSM based DeviceId must be reported.
44	4	DeviceIdSize	SIZE (0..36)	Size used for DeviceId
48	4	FirmwareInfoOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string FirmwareInfo that represents firmware-specific information about the function. Max length is 30 (16-bit) characters
52	4	FirmwareInfoSize	SIZE (0..60)	Size used for FirmwareInfo

Offset	Size	Field	Type	Description
56	4	HardwareInfoOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string HardwareInfo that represents hardware-specific information about the function. Max length is 30 16-bit characters.
60	4	HardwareInfoSize	SIZE (0..60)	Size used for HardwareInfo
64		DataBuffer	DATABUFFER	CustomDataClass DeviceId FirmwareInfo HardwareInfo

10.5.1.6 STATUS CODES

This CID only uses Generic Status Codes (see Use of the Status Codes section 9.4.5).

10.5.2 MBIM_CID_SUBSCRIBER_READY_STATUS

10.5.2.1 DESCRIPTION

The function prepares and returns an MBIM_SUBSCRIBER_READY_INFO structure in response to an MBIM_COMMAND_MSG with UUID_BASIC_CONNECT and MBIM_CID_SUBSCRIBER_READY_STATUS. This structure contains the Subscriber Identity Module (SIM) card ready state as well as some information on the SIM card (SubscriberId, IccId, Telephone numbers). The information on the SIM card may not be accessible in all SIM-card states: some fields may not be accessible to the function before the SIM card is unlocked (see [3GPP31102] for access conditions to fields on the SIM card).

The initial ReadyState for the SIM card when the function is opened may be any of the ReadyStates as the function may have communicated with the SIM card prior to the MBIM_OPEN_DONE message is sent. The host is responsible to query for the ReadyState when the device has been opened. After the MBIM_OPEN_DONE message has been sent, the function shall always notify the host whenever the SIM ReadyState changes, using MBIM_INDICATE_STATUS_MSG with UUID_BASIC_CONNECT and MBIM_CID_ SUBSCRIBER_READY_INFO.

For the states in the MBIM_SUBSCRIBER_READY_STATE structure, the following shall be noted:

- Before the function knows the state of the SIM card or before it has finalized initializating the SIM card, the ReadyState is MBIMSubscriberReadyStateNotInitialized
- When the SIM card has been initialized and no other of the ReadyStates applies, the ReadyState is MBIMSubscriberReadyStateInitialized

- If the SIM card has been initialized and the SIM requires PIN1 or PUK1 to be entered, the ReadyState is MBIMSubscriberReadyStateLocked. The host can then unlock the SIM using a MBIM_CID_PIN set request. Note that the function may need another unlock code to be entered before the PIN1/PUK1 code can be entered. Hence, a query of the MBIM_CID_PIN command is needed prior to sending the set command. After the SIM is unlocked, the function must send a MBIM_CID_SUBSCRIBER_READY_STATUS event notification with ReadyState set to the SIM card's new state.
- If the SIM card is PUK1 locked but no unblock attempts remain, the ReadyState is set to MBIMSubscriberReadyStateBadSim. This state may be entered as soon as the SIM is initialized or after an unsuccessful unlock attempt of PUK1 in which case the ReadyState is changed from MBIMSubscriberReadyStateLocked to MBIMSubscriberReadyStateBadSim
- If there is no SIM card present, the ReadyState is MBIMSubscriberReadyStateNotInserted. If the SIM card is removed while the function is open and the function has the capability to detect a hot removal/insertion of a SIM card, a notification that the ReadyState has changed to MBIMSubscriberReadyStateNotInserted shall be sent. The function shall send event notifications (1) if a new SIM card is inserted (with ReadyState set to MBIMSubscriberReadyStateNotInitialized); and (2) whenever new ReadyStates apply.
- If the function can detect that a SIM card is not activated, the ready state of an inactive SIM card is MBIMSubscriberReadyStateNotActivated. If the function supports service activation, and the host successfully sends the function a MBIM_CID_SERVICE_ACTIVATION set request, on successful completion of service activation, the function shall send another event notification with updated ReadyState.
- If any failure of the SIM card has been detected, the ReadyState is MBIMSubscriberReadyStateFailure

Query:

No input parameters, so InformationBuffer on Query message is not used.

MBIM_SUBSCRIBER_READY_INFO returned in the InformationBuffer of MBIM_COMMAND_DONE.

Set:

Not applicable.

Unsolicited Event:

The Event InformationBuffer contains an MBIM_SUBSCRIBER_READY_INFO structure.

Functions must report all ready-state changes as an unsolicited event. When the function initializes it sets MBIMSubscriberReadyStateNotInitialized. Thereafter, the function must report any ready-state change to the host through this notification. For example, the function must report a ready-state change when the state changes from MBIMSubscriberReadyStateNotInitialized to

MBIMSubscriberReadyStateDeviceLocked, or MBIMSubscriberReadyStateBadSim, or MBIMSubscriberReadyStateSimNotInserted, or any other different function ready-state.

Most ready-state changes happen when the function initializes the radio stack and the SIM card (if required). A change can also happen during the course of a session between the host and the function, such as user changing the SIM card.

10.5.2.2 PARAMETERS

Table 10-15: Parameters

	Set	Query	Notification
Command	NA	Empty	NA
Response	NA	MBIM_SUBSCRIBER_READY_INFO	MBIM_SUBSCRIBER_READY_INFO

10.5.2.3 DATA STRUCTURES

Table 10-16: MBIM_SUBSCRIBER_READY_STATE

Types	Value	Description
MBIMSubscriberReadyStateNotInitialized	0	The SIM has not yet completed its initialization
MBIMSubscriberReadyStateInitialized	1	The SIM is initialized. When this state is reported all the subscriber fields in Table 10-18: MBIM_SUBSCRIBER_READY_INFO must be valid.
MBIMSubscriberReadyStateSimNotInserted	2	The SIM card is not inserted into the device
MBIMSubscriberReadyStateBadSim	3	The SIM card inserted into the device is invalid
MBIMSubscriberReadyStateFailure	4	A general SIM failure has occurred
MBIMSubscriberReadyStateNotActivated	5	The subscription is not activated
MBIMSubscriberReadyStateDeviceLocked	6	The SIM is locked and requires PIN1 or PUK1 to unlock

Table 10-17: MBIM_UNIQUE_ID_FLAGS

Types	Value	Description
MBIMReadyInfoFlagsNone	0	The device is in normal mode
MBIMReadyInfoFlagsProtectUniqueID	1	When this flag is specified, the host will not display the SubscriberId specified in the same CID.

10.5.2.4 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.2.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-18: MBIM_SUBSCRIBER_READY_INFO

Offset	Size	Field	Type	Description
0	4	ReadyState	MBIM_SUBSCRIBER_READY_STATE	One of the values in Table 10-16: MBIM_SUBSCRIBER_READY_STATE

4	4	SubscriberIdOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string SubscriberId of digits that represents the identity of the subscriber. For GSM-based functions, this member represents the International Mobile Subscriber Identity (IMSI) string (up to 15 digits in length). For CDMA-based functions, this represents the Mobile Identification Number (MIN) string or the International Roaming MIN (IRM) string (both 10 digits in length). For single-carrier multi-mode functions, the GSM SubscriberId format must be used. This does not apply to multi-carrier multi-mode functions as the SubscriberId may change. The function must provide a valid SubscriberId when the device ready-state changes to MBIMSubscriberReadyStateInitialized. Functions should also specify this string when the device ready-state changes to MBIMSubscriberReadyStateBadSim, MBIMSubscriberReadyStateFailure, or MBIMSubscriberReadyStateDeviceLocked, if possible, for identification purposes.
8	4	SubscriberIdSize	SIZE (0..30)	Sized used for the SubscriberId

12	4	SimIccIdOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string SimIccId of digits that represents the International Circuit Card (ICC) ID of the SIM. The ICC ID varies from between 15 to 20 digits in length and is represented in alphanumeric characters. Functions must provide a valid SimIccId for any of the following conditions: • the function's ready-state changes to MBIMSubscriberReadyStateInitialized • the function is locked waiting for entry of PIN1 or PUK1 keys. Functions must specify this value for all devices where MBIMCellularClass equals MBIMCellularClassGsm. Functions of CDMA-based devices must specify this value for devices where SimClass equals MBIMSimClassSimRemovable.
16	4	SimIccIdSize	SIZE (0..40)	Sized used for the SimIccId
20	4	ReadyInfo	MBIM_UNIQUE_ID_FLAGS	Set per Table 10-17: MBIM_UNIQUE_ID_FLAGS.
24	4	ElementCount (EC)	UINT32	Count of Telephone numbers following this element

28	8*EC	TelephoneNumbersRefList	OL_PAIR_LIST	First element in the pair is a 4 byte field with the Offset into the DataBuffer, calculated from the beginning (offset 0) of this MBIM_SUBSCRIBER_READY_INFO structure, to a telephone number (TN) TelephoneNumber assigned to the subscriber identity. Each TN in the list is a string, with a fixed maximum size of 22 characters. In GSM-based devices the TNs are called Mobile Station ISDN Number (MSISDNs). In CDMA-based devices they are called Mobile Directory Numbers (MDNs). Functions shall not return telephone numbers until the device ready-state changes to MBIMSubscriberReadyStateInitialized. If the function is not initialized, the function shall set ElementCount to zero, and shall not return any TNs Second element of the pair is a 4 byte field with the size of the record element
28+8*EC		DataBuffer	DATABUFFER	SubscriberId SimIccid TelephoneNumber

10.5.2.6 NOTIFICATION

See Table 10-18: MBIM_SUBSCRIBER_READY_INFO

10.5.2.7 STATUS CODES

This CID only uses Generic Status Codes (see Use of the Status Codes section 9.4.5).

10.5.3 MBIM_CID_RADIO_STATE

10.5.3.1 DESCRIPTION

The command sets or returns information about a MB device's radio power state.

Query:

InformationBuffer on MBIM_COMMAND_MSG is not used. MBIM_RADIO_STATE_INFO returned in InformationBuffer of MBIM_COMMAND_DONE.

Set:

InformationBuffer on MBIM_COMMAND_MSG contains MBIM_SET_RADIO_STATE. MBIM_RADIO_STATE_INFO is returned in InformationBuffer of MBIM_COMMAND_DONE.

Unsolicited Event:

The Event InformationBuffer contains an MBIM_RADIO_STATE_INFO structure.

10.5.3.2 PARAMETERS

Table 10-19: Parameters

	Set	Query	Notification
Command	MBIM_SET_RADIO_STATE	Empty	NA
Response	MBIM_RADIO_STATE_INFO	MBIM_RADIO_STATE_INFO	MBIM_RADIO_STATE_INFO

10.5.3.3 DATA STRUCTURES

Table 10-20: MBIM_RADIO_SWITCH_STATE

Types	Value
MBIMRadioOff	0
MBIMRadioOn	1

10.5.3.4 SET

The following structure shall be used in the InformationBuffer

Table 10-21: MBIM_SET_RADIO_STATE

Offset	Size	Field	Type	Description

0	4	RadioState	MBIM_RADIO_SWITCH_STATE	Sets the software controlled radio state. See Table 10-20: MBIM_RADIO_SWITCH_STATE

10.5.3.5 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.3.6 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-22: MBIM_RADIO_STATE_INFO

Offset	Size	Field	Type	Description
0	4	HwRadioState	MBIM_RADIO_SWITCH_STATE	The state of the W_DISABLE switch (see Section 3.2.5.2 of [PCI-Express]). If the device does not have a W_DISABLE switch The function must return MBIMRadioOn in this field. See Table 10-20: MBIM_RADIO_SWITCH_STATE
4	4	SwRadioState	MBIM_RADIO_SWITCH_STATE	Software configured radio state. See Table 10-20: MBIM_RADIO_SWITCH_STATE

10.5.3.7 NOTIFICATION

See Table 10-22: MBIM_RADIO_STATE_INFO

10.5.3.8 STATUS CODES

This CID only uses Generic Status Codes (see Use of the Status Codes section 9.4.5).

10.5.4 MBIM_CID_PIN

10.5.4.1 DESCRIPTION

During the initialization process, the host does not proceed to registration until PIN1 is successfully unlocked, if enabled.

The host provides a PIN value, entered by the end user, in the Pin member of the MBIM_SET_PIN structure when processing set requests. In case the PIN member provided in the MBIM_SET_PIN structure is incorrect, the function shall fail the set request with status code MBIM_STATUS_FAILURE.

CDMA-based devices that have a power-on device lock must report it as PIN1.

For multi-mode functions, the PIN operations for a given mode must be supported even when operating in a different mode.

For all supported PIN types, functions must support the MBIMPinOperationEnter operation. Additionally, if PIN1 is supported, functions must support the MBIMPinOperationEnable, MBIMPinOperationDisable, and MBIMPinOperationChange operations.

If a PIN disable operation for a PIN type is tried when that PIN type is locked, functions can either fail the request with MBIM_STATUS_PIN_REQUIRED or they can successfully complete the request. If the function completes the request successfully, the disable operation must also unlock the PIN.

If multiple PINs are enabled and reporting multiple PINs is also enabled, and only one PIN can be reported at a time, then functions must report PIN1 first. For example, if reporting of Subsidy lock and SIM PIN1 is enabled, then the Subsidy lock PIN should be reported (in a subsequent query request) only after PIN1 has been successfully verified.

Query and Set:

For **Query**, the InformationBuffer is empty. For **Set**, it contains an MBIM_SET_PIN.

MBIM_PIN_INFO is returned with both Set Complete and Query Complete messages in the InformationBuffer.

10.5.4.2 PARAMETERS

Table 10-23: Parameters

	Set	Query	Notification
Command	MBIM_SET_PIN	Empty	NA
Response	MBIM_PIN_INFO	MBIM_PIN_INFO	NA

10.5.4.3 DATA STRUCTURES

Table 10-24: MBIM_PIN_TYPE

Types	Value	Description
MBIMPinTypeNone	0	No PIN is pending to be entered
MBIMPinTypeCustom	1	The PIN type is a custom type and is none of the other PIN types listed in this enumeration
MBIMPinTypePin1	2	The PIN1 key

MBIMPinTypePin2	3	The PIN2 key
MBIMPinTypeDeviceSimPin	4	The device to SIM key
MBIMPinTypeDeviceFirstSimPin	5	The device to very first SIM key
MBIMPinTypeNetworkPin	6	The network personalization key
MBIMPinTypeNetworkSubsetPin	7	The network subset personalization key
MBIMPinTypeServiceProviderPin	8	The service provider (SP) personalization key
MBIMPinTypeCorporatePin	9	The corporate personalization key
MBIMPinTypeSubsidyLock	10	The subsidy unlock key
MBIMPinTypePuk1	11	The Personal Identification Number1 Unlock Key (PUK1).
MBIMPinTypePuk2	12	The Personal Identification Number2 Unlock Key (PUK2
MBIMPinTypeDeviceFirstSimPuk	13	The device to very first SIM PIN unlock key
MBIMPinTypeNetworkPuk	14	The network personalization unlock key
MBIMPinTypeNetworkSubsetPuk	15	The network subset personalization unlock key
MBIMPinTypeServiceProviderPuk	16	The service provider (SP) personalization unlock key
MBIMPinTypeCorporatePuk	17	The corporate personalization unlock key

Table 10-25: MBIM_PIN_STATE

Types	Value	Description
MBIMPinStateUnlocked	0	The device does not require a PIN
MBIMPinStateLocked	1	The device requires the user to enter a PIN

Table 10-26: MBIM_PIN_OPERATION

Types	Value	Description
MBIMPinOperationEnter	0	Enter the specified PIN into the device.
MBIMPinOperationEnable	1	Enable the specified PIN
MBIMPinOperationDisable	2	Disable the specified PIN
MBIMPinOperationChange	3	Change the specified PIN

10.5.4.4 SET

The following structure shall be used in the InformationBuffer

Table 10-27: MBIM_SET_PIN

Offset	Size	Field	Type	Description

0	4	PinType	MBIM_PIN_TYPE	See Table 10-24: MBIM_PIN_TYPE
4	4	PinOperation	MBIM_PIN_OPERATION	See Table 10-26: MBIM_PIN_OPERATION
8	4	PinOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string Pin that represents the PIN value to perform the action with, or the PIN value required to enable/disable PIN settings. This member applies for all values of PinOperation.
12	4	PinSize	SIZE (0..32)	Size in bytes used for the Pin
16	4	NewPinOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string NewPin that represents the new PIN value to set when PinOperation is MBIMPinOperationChange or MBIMPinOperationEnter for PinTypeMBIMPinTypePuk1 or MBIMPinTypePuk2.
20	4	NewPinSize	SIZE (0..32)	Size in bytes used for the NewPin
24		DataBuffer	DATABUFFER	Pin NewPin

10.5.4.5 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.4.6 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-28: MBIM_PIN_INFO

Offset	Size	Field	Type	Description
0	4	PinType	MBIM_PIN_TYPE	See Table 10-24: MBIM_PIN_TYPE
4	4	PinState	MBIM_PIN_STATE	See Table 10-25:

				MBIM_PIN_STATE
8	4	RemainingAttempts	UINT32	Number of remaining attempts for any pin-related operations such as enter, enable, disable. Devices that do not support this information must set this member to 0xffffffff

10.5.4.7 STATUS CODES

Table 10-29: Status codes

Status code	Description
MBIM_STATUS_PIN_DISABLED	The operation failed because the PIN is disabled.
MBIM_STATUS_PIN_REQUIRED	The operation failed because a PIN must be entered to proceed.
MBIM_STATUS_NO_DEVICE_SUPPORT	The operation failed because a set on a corresponding PIN type is not supported by the device.

10.5.5 MBIM_CID_PIN_LIST

10.5.5.1 DESCRIPTION

This command returns a list of all the different types of Personal Identification Numbers (PINs) that are supported by the MB device and additional details for each PIN type, such as the length of the PIN (minimum and maximum lengths), PIN format, PIN-entry mode (enabled/disabled/not-available). This CID also specifies the current mode of each PIN supported by the function.

Functions must report all the PINs they support. However, PIN1 for multi-mode devices must only be reported once.

A PIN reported as PIN1 must comply with PIN1 guidelines: for CDMA-based devices this is a PIN that provides power-on verification or identification functionality, and for GSM-based devices this is a Subscriber Identity Module (SIM) PIN.

Functions must be able to return this information when the ready-state changes to MBIMSubscriberReadyStateInitialized or when the ready-state is

MBIMSubscriberReadyStateDeviceLocked (PIN locked). Functions should also return this information in other ready-states, wherever possible.

Query only:

InformationBuffer of Query message is empty. InformationBuffer of MBIM_COMMAND_DONE contains an MBIM_PIN_LIST_INFO.

10.5.5.2 PARAMETERS

Table 10-30: Parameters

	Set	Query	Notification
Command	NA	Empty	NA
Response	NA	MBIM_PIN_LIST_INFO	NA

10.5.5.3 DATA STRUCTURES

Table 10-31: MBIM_PIN_MODE

Types	Value
MBIMPinModeNotSupported	0
MBIMPinModeEnabled	1
MBIMPinModeDisabled	2

Table 10-32: MBIM_PIN_FORMAT

Types	Value
MBIMPinFormatUnknown	0
MBIMPinFormatNumeric	1
MBIMPinFormatAlphaNumeric	2

Table 10-33: MBIM_PIN_DESC

Offset	Size	Field	Type	Description

0	4	PinMode	MBIM_PIN_MODE	See Table 10-31: MBIM_PIN_MODE. This shows if the lock is enabled or not. It does not show if the lock state is locked or unlocked.
4	4	PinFormat	MBIM_PIN_FORMAT	See Table 10-32: MBIM_PIN_FORMAT
8	4	PinLengthMin	UINT32	The minimum number of characters in the PIN. Devices should not specify a value that is greater than 16. Devices should specify 0xffffffff, if the PIN length is not available.
12	4	PinLengthMax	UINT32	The maximum number of characters in the PIN. Devices should not specify a value that is greater than 16. Devices should specify 0xffffffff, if the PIN length is not available.

10.5.5.4 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.5.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-34: MBIM_PIN_LIST_INFO

Offset	Size	Field	Type	Description
0	16	PinDescPin1	MBIM_PIN_DESC	MBIM_PIN_DESC structure describing PIN1. For GSM-based devices, this is a Subscriber Identity Module (SIM) PIN. For CDMA-based devices, power-on device lock is reported as PIN1
16	16	PinDescPin2	MBIM_PIN_DESC	MBIM_PIN_DESC structure describing PIN2. This is a SIM PIN2 that protects certain SIM functionality.

32	16	PinDescDeviceSimPin	MBIM_PIN_DESC	MBIM_PIN_DESC structure describing the device-to-SIM-card PIN. This is a PIN that locks the device to a specific SIM.
48	16	PinDescDeviceFirstSimPin	MBIM_PIN_DESC	MBIM_PIN_DESC structure describing device-to-very-first-SIM-card PIN. This is a PIN that locks the device to the very first inserted SIM.
64	16	PinDescNetworkPin	MBIM_PIN_DESC	MBIM_PIN_DESC structure describing the network personalization PIN. This is a PIN that allows the device to be personalized to a network. For more information about this PIN type, see 3GPP specification 22.022.
80	16	PinDescNetworkSubsetPin	MBIM_PIN_DESC	MBIM_PIN_DESC structure describing the network subset personalization PIN. This is a PIN that allows the device to be personalized to a subset of a network. For more information about this PIN type, see 3GPP specification 22.022.
96	16	PinDescServiceProviderPin	MBIM_PIN_DESC	MBIM_PIN_DESC structure describing the Service Provider (SP) personalization PIN. This is a PIN that allows the device to be personalized to a service provider. For more information about this PIN type, see 3GPP specification 22.022

112	16	PinDescCorporatePin	MBIM_PIN_DESC	MBIM_PIN_DESC structure describing the corporate personalization PIN. This is a PIN that allows the device to be personalized to a specific company. For more information about this PIN type, see 3GPP specification 22.022.
128	16	PinDescSubsidyLock	MBIM_PIN_DESC	MBIM_PIN_DESC structure describing the subsidy unlocks PIN. This is a PIN that allows the device to be restricted to operate on a specific network. For more information about this PIN type, see 3GPP specification 22.022
144	16	PinDescCustom	MBIM_PIN_DESC	MBIM_PIN_DESC structure describing the custom PIN. This is a custom vendor-defined PIN type. It is not included in the above list

10.5.5.6 STATUS CODES

10.5.5.7 STATUS CODES

Status code	Description
MBIM_STATUS_PIN_REQUIRED	The PIN list operation failed because a PIN must be entered before this operation can proceed.

10.5.6 MBIM_CID_HOME_PROVIDER

10.5.6.1 DESCRIPTION

This command sets or returns information about the home provider of the cellular service subscription. For GSM-based devices and CDMA-based device with U-RIM, this information should be stored on the

Subscriber Identity Module (SIM card). For CDMA-based devices without U-RIM, this information should be stored in auxiliary device memory.

Query and Set:

InformationBuffer of **Query** message is empty. InformationBuffer of MBIM_COMMAND_ DONE contains an MBIM_PROVIDER. If the device does not report MBIM_CTRL_CAPS_MULTI_CARRIER, this CID only supports **Query** unless the device reports MBIM_CTRL_CAPS_MULTI_CARRIER. If so, this CID may also be **Set**. The InformationBuffer of a **Set** message contains an MBIM_ PROVIDER structure. Once set, the device must select the specified providers as its home provider, including reporting it to on a query of this CID.

The following bitmap table (Table 10-36: MBIM_PROVIDER_STATE) represents the various states with which the network provider's entry can be tagged. It shows the possible values that a function should specify (one or more values can be specified). Some values in this table apply only to specific CIDs. The "Relevant Commands" column shows the associations between the different CIDs and the relevant states.

10.5.6.2 PARAMETERS

Table 10-35: Parameters

	Set	Query	Notification
Command	MBIM_PROVIDER	Empty	NA
Response	MBIM_PROVIDER	MBIM_PROVIDER	NA

10.5.6.3 DATA STRUCTURES

Table 10-36: MBIM_PROVIDER_STATE

Types	Value	Description	Relevant Commands
MBIM_PROVIDER_STATE_UNKNOWN	0h	The network provider state is unknown	
MBIM_PROVIDER_STATE_HOME	1h	The network provider is a home operator	MBIM_CID_HOME_PROVIDER, MBIM_CID_ VISIBLE_PROVIDERS

Types	Value	Description	Relevant Commands
MBIM_PROVIDER_STATE_FORBIDDEN	2h	The network provider is on the blocked list	MBIM_CID_PREFERRED_PROVIDERS, MBIM_CID_ VISIBLE_PROVIDERS
MBIM_PROVIDER_STATE_PREFERRED	4h	The network provider is on the preferred list	MBIM_CID_PREFERRED_PROVIDERS, MBIM_CID_ VISIBLE_PROVIDERS
MBIM_PROVIDER_STATE_VISIBLE	8h	The network provider is visible	MBIM_CID_ VISIBLE_PROVIDERS
MBIM_PROVIDER_STATE_REGISTERED	10h	The network provider is currently registered by the device	MBIM_CID_ VISIBLE_PROVIDERS
MBIM_PROVIDER_STATE_PREFERRED_MULTICARRIER	20h	The network provider is a preferred multicarrier network.	MBIM_CID_MULTICARRIER_PROVIDERS MBIM_CID_ VISIBLE_PROVIDERS

10.5.6.4 SET

The following structure shall be used in the InformationBuffer

Table 10-37: MBIM_PROVIDER

Offset	Size	Field	Type	Description

0	4	ProviderIdOffset	OFFSET	Offset in bytes, calculated from the beginning (offset 0) of this MBIM_PROVIDER structure, to a numeric (0-9) string ProviderId that represents the network provider identity. For GSM-based networks, this string is a concatenation of a three-digit Mobile Country Code (MCC) and a two or three-digit Mobile Network Code (MNC). GSM-based carriers may have more than one MNC, and hence more than one ProviderId. However, only one is expected to be used at any given point in time (e.g., as home provider). For CDMA-based networks, this string is a five-digit System ID (SID). Generally a CDMA-based carrier has more than one SID. Typically, the carrier has one SID for each market, which is usually divided geographically within a nation by regulations, such as Metropolitan Statistical Areas (MSA) in the United States of America. Devices of CDMA-based devices must specify MBIM_CDMA_DEFAULT_PROVIDER_ID if this information is not available.
4	4	ProviderIdSize	SIZE (0..12)	Size used for the ProviderId
8	4	ProviderState	MBIM_PROVIDER_STATE	See Table 10-36: MBIM_PROVIDER_STATE. This state tracks whether the network provider is currently the home provider, a preferred provider, whether it's visible, etc.

12	4	ProviderNameOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string ProviderName that represents the network provider's name. This member is limited to, at most, 20 characters. For GSM-based networks, if the Preferred Presentation of Country Initials and Mobile Network Name (PPCI&N) is longer than 20 characters, the device should abbreviate the network name. This member is ignored when the Host sets the preferred provider list. Note: This is not the long version (256 characters maximum) but the short version, so string length is short (limited to 20 characters or so by both 3GPP and 3GPP2).
16	4	ProviderNameSize	SIZE (0..40)	Size used for the ProviderName
20	4	CellularClass	MBIM_CELLULAR_CLASS	The cellular class that the provider uses. See Table 10-8: MBIM_CELLULAR_CLASS. A device that supports MBIM_CTRL_CAPS_MULTI_CARRIER must correctly populate this field for any provider that is currently visible. This field must be valid for any SET requests. Devices that do not support MBIM_CTRL_CAPS_MULTI_CARRIER should set this field to zero.

24	4	Rssi	UINT32	Represents the strength of the wireless signal (when available). See Table 10-58: MBIM_SIGNAL_STATE_INFO. A device that supports MBIM_CTRL_CAPS_MULTI_CARRIER must correctly populate this field for any provider that is currently visible. This field should be ignored on set requests. Devices that do not support MBIM_CTRL_CAPS_MULTI_CARRIER should set this field to zero.
28	4	ErrorRate	UINT32	A coded value that represents a percentage range of error rates. See Table 10-58: MBIM_SIGNAL_STATE_INFO. A devices that supports MBIM_CTRL_CAPS_MULTI_CARRIER must correctly populate this field for any provider that is currently visible. This field should be ignored on set requests. Devices that do not support MBIM_CTRL_CAPS_MULTI_CARRIER should set this field to zero.
32		DataBuffer	DATABUFFER	ProviderId ProviderName

10.5.6.5 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.6.6 RESPONSE

See Table 10-37: MBIM_PROVIDER

10.5.6.7 STATUS CODES

Status code	Description
MBIM_STATUS_READ_FAILURE	Home provider operation failed because the information could not be read from the device or SIM card. The error may be returned in instances where the SIM card does not have home provider information provisioned.

10.5.7 MBIM_CID_PREFERRED_PROVIDERS

10.5.7.1 DESCRIPTION

This command returns information about the list of preferred providers for GSM-based devices. CDMA-based devices do not need to support this command.

Set and Query:

For **Query**, the InformationBuffer is empty, for **Set,** it contains an MBIM_PROVIDERS structure. **Set** can only operate on user-controlled entries, if such a list is available. If user-controlled entries are available, a **Set** replaces them with the list passed in the MBIM_PROVIDERS structure. For functions that do not support user-controlled entries, a **Set** results in the error status, MBIM_STATUS_OPERATION_NOT_ALLOWED.

An MBIM_PROVIDERS structure is returned from both **Set** and **Query** complete messages in the InformationBuffer, and in it, the function returns to the host all entries found. Thus, the list of entries returned by the function includes both the operator list as well as, whenever available, the user-controlled list.

Unsolicited Event:

The Event InformationBuffer contains an MBIM_PROVIDERS structure.

In some cases, the PPL (for GSM-based devices) is updated by the network either Over-The-Air (OTA) or by Short Message Service (SMS). When this happens devices must notify the HOST about the updates using this INDICATION with the updated PPL

10.5.7.2 PARAMETERS

Table 10-38: Parameters

	Set	Query	Notification
Command	MBIM_PROVIDERS	Empty	NA
Response	MBIM_PROVIDERS	MBIM_PROVIDERS	MBIM_PROVIDERS

10.5.7.3 SET

The following structure shall be used in the InformationBuffer

Table 10-39: MBIM_PROVIDERS

Offset	Size	Field	Type	Description

0	4	ElementCount (EC)	UINT32	Number of structures that follow.
4	8*EC	ProvidersRefList	OL_PAIR_LIST	The first element of the pair is a 4 byte Offset in bytes, calculated from the beginning (offset 0) of this MBIM_PROVIDERS structure, to an MBIM_PROVIDER structure (see Table 10-37: MBIM_PROVIDER). The second element of the pair is a 4 byte size of the record element.
4+8*EC		DataBuffer	DATABUFFER	MBIM_PROVIDER

10.5.7.4 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.7.5 RESPONSE

See Table 10-39: MBIM_PROVIDERS

10.5.7.6 NOTIFICATION

See Table 10-39: MBIM_PROVIDERS

10.5.7.7 STATUS CODES

Status code	Description
MBIM_STATUS_READ_FAILURE	Reading information from device or SIM card, or both, failed. For example, the SIM card does not have preferred providers information provisioned.
MBIM_STATUS_OPERATION_NOT_ALLOWED	A Set was attempted on a preferred provider that is not settable (e.g., because it may have come from an operator-controlled list or whatever reason).

10.5.8 MBIM_CID_VISIBLE_PROVIDERS

10.5.8.1 DESCRIPTION

This command returns a list of network providers currently visible within the MB device's range.

For CDMA-based networks, device should return only the home provider, if any of the networks in the Preferred Roaming List (PRL) is currently visible. For GSM-based networks, more than one provider may be present in the visible provider list.

Devices that do not support scanning for visible providers while connected should return the MBIM_STATUS_BUSY error value in the Status member.

Both GSM-based and CDMA-based devices must support scanning for visible providers while in registered mode. However, devices are not required to support scanning for visible provider while a Packet Data Protocol (PDP) context is active (for example, the device is connected to the provider's network).

Query Only:

The InformationBuffer of the MBIM_COMMAND_MSG contains an MBIM_VISIBLE_PROVIDERS_REQ structure (see Table 10-42: MBIM_VISIBLE_PROVIDERS_REQ). InformationBuffer of MBIM_COMMAND_DONE contains an MBIM_PROVIDERS structure (see Table 10-39: MBIM_PROVIDERS)

10.5.8.2 PARAMETERS

Table 10-40: Parameters

	Set	Query	Notification
Command	NA	MBIM_VISIBLE_PROVIDERS_REQ	NA
Response	NA	MBIM_PROVIDERS	NA

10.5.8.3 DATA STRUCTURES

Table 10-41: MBIM_VISIBLE_PROVIDERS_ACTION

Types	Value	Description
MBIMVisibleProvidersActionFullScan	0	Device should perform a full scan.

MBIMVisibleProvidersActionRestrictedScan	1	Device should perform a restricted scan to locate preferred multicarrier providers. The device may also report a static list in case a scan is not possible.

10.5.8.4 QUERY

The following structure shall be used in the InformationBuffer

Table 10-42: MBIM_VISIBLE_PROVIDERS_REQ

Offset	Size	Field	Type	Description
0	4	Action	MBIM_VISIBLE_PROVIDERS_ACTION	Defines which type of scan to perform. See Table 10-41: MBIM_VISIBLE_PROVIDERS_ACTION

10.5.8.5 RESPONSE

See Table 10-39: MBIM_PROVIDERS

10.5.8.6 STATUS CODES

Status code	Description
MBIM_STATUS_READ_FAILURE	Reading information from device or SIM card, or both, failed. For example, the SIM card does not have preferred providers information provisioned.

10.5.9 MBIM_CID_REGISTER_STATE

10.5.9.1 DESCRIPTION

This command selects a network provider with which to register.

MBIM supports two registration methods: automatic and manual. For CDMA-based networks, MBIM supports only automatic registration.

Functions that support manual registration must set the ControlCaps member in MBIM_DEVICE_CAPS_INFO (Table 10-14: MBIM_DEVICE_CAPS_INFO) structure to MBIM_CTRL_CAPS_REG_MANUAL

If the registration state is automatic, functions select a network provider based on the selection algorithm specific to the cellular technology and proceed with registration.

The semantics of RegisterAction values are defined as follows:

- The MBIMRegisterActionAutomatic flag is used by the Host to set the function to automatic registration mode, and to let the function select the best provider network. The function must ignore the ProviderId parameter. This setting is persistent across radio states (ON/OFF), and device power cycles, until it is explicitly change by the Host.
- The MBIMRegisterActionManual flag is used by the Host to instruct the function to register with the provider network identified by the ProviderId parameter. The ProviderId value shall come from the ProviderId member of MBIM_PROVIDER data structure of one of the visible providers. This setting is persistent across radio states (ON/OFF), and device power cycles, until it is explicitly changed by the Host.
- Changing between the different RegisterAction values are allowed even if the device is currently registered to a provider. If the device need to deregister before switching between the Automatic and Manual registration modes, the device must ensure that the device is set to deregistration before setting to the new registration mode.
- The Manual and Automatic registration mode only affects the network selection mode. The MB device should try to register to selected network whenever the radio is turned on.
- When RegisterAction is set to MBIMRegisterActionManual, if the provider is not visible when the function receives the request, the function shall return error code MBIM_STATUS_PROVIDER_NOT_VISIBLE. The function must not switch to automatic registration because of a failure in setting the manual mode. If the function was earlier set to manually register to another network, this request should change the function to register to the network specified in the request. The value of RegisterState in response to the request should be set to MBIMRegisterStateDeregistered.
- When RegisterAction is set to MBIMRegisterActionManual, if the function has already registered with the same network that is being requested, it shall respond with MBIM_STATUS_SUCCESS.
- When RegisterAction is set to MBIMRegisterActionManual, and the Radio is OFF, the function must set itself to manual registration mode and complete the request with the transaction notification. The RegisterState should be set to MBIMRegisterStateDeregistered. The function must attempt a manual registration when the Radio changes to ON state and the event notification must be sent.
- When RegisterAction is set to MBIMRegisterActionAutomatic, and the Radio is OFF, the function must set itself to automatic registration mode and must complete the request with the transaction notification. The RegisterState should be set to MBIMRegisterStateDeregistered. The function must do an automatic registration when the Radio goes to ON state and the event notification must be sent.

Set and Query:

MBIM_REGISTRATION_STATE_INFO is returned from both **Set** and **Query** complete messages in the InformationBuffer.

For **Query**, the InformationBuffer is empty. For **Set**, it contains an MBIM_SET_REGISTRATION_STATE structure.

Unsolicited Event:

The Event InformationBuffer contains an MBIM_REGISTRATION_STATE_INFO structure.

10.5.9.2 PARAMETERS

Table 10-43: Parameters

	Set	Query	Notification
Command	MBIM_SET_REGISTRATION_STATE	Empty	NA
Response	MBIM_REGISTRATION_STATE_INFO	MBIM_REGISTRATION_STATE_INFO	MBIM_REGISTRATION_STATE_INFO

10.5.9.3 DATA STRUCTURES

The following table shows registration, packet-attach, and packet-detach cause code failure values that are defined in the 3GPP TS 24.008 Specification for GSM-based network [3GPP24008]. These are to be reported back in NwError as appropriate, and as detailed in the NwError usage guidance throughout this specification.

Table 10-44: 3GPP TS 24.008 Cause codes for NwError

3GPP 24.008 Cause code	Interpretation of cause code
2 - International Mobile Subscriber Identity (IMSI) unknown in HLR	Either the SIM or the device is not activated, or the subscription has expired, which caused a network deactivation.
4 - IMSI unknown in VLR	Roaming feature is not subscribed to.
6 - Illegal ME	MS blocked by network due to stolen report.
7 - GPRS services not allowed	User does not have a GPRS subscription. User has only a voice connection subscription.
8 - GPRS and non-GPRS services not allowed	GPRS and non-GPRS services are not allowed.

11 - PLMN not allowed	Service is blocked by the network due to an expired subscription or another cause.
12 - Location area not allowed	User subscription does not allow access in the present location area.
13 - Roaming not allowed in this location area	The subscription permits roaming, but roaming is not allowed in the present location area.
14 - GPRS services not allowed in this PLMN	Selected network provider does not provide GPRS service to the MS.
15 - No suitable cells in location area	No subscription for the service.
17 - Network failure	Registration failed.
22 – Congestion	Registration failed due to network congestion.

For example, if the network initiates a deactivate context event because roaming is not allowed in the location area, functions should set the NwError member to 13 per the 3GPP TS 24.008 Cause codes for GSM-based networks [3GPP24008].

Similar logic should be applied to CDMA-based networks as well. However, there is no standard for CDMA-based network error codes. CDMA-based devices should use the network -specific or device-specific error codes.

Table 10-45: MBIM_REGISTER_ACTION

Types	Value
MBIMRegisterActionAutomatic	0
MBIMRegisterActionManual	1

Table 10-46: MBIM_REGISTER_STATE

Types	Value
MBIMRegisterStateUnknown	0

MBIMRegisterStateDeregistered	1
MBIMRegisterStateSearching	2
MBIMRegisterStateHome	3
MBIMRegisterStateRoaming	4
MBIMRegisterStatePartner	5
MBIMRegisterStateDenied	6

If the device is registered with a roaming provider, the device shall set RegisterState as MBIMRegisterStatePartner if the provider is a preferred roaming partner or just MBIMRegisterStateRoaming for a roaming partner, respectively. If the device does not distinguish between the two, it shall set the value to MBIMRegisterStateRoaming.

Table 10-47: MBIM_REGISTER_MODE

Types	Value
MBIMRegisterModeUnknown	0
MBIMRegisterModeAutomatic	1
MBIMRegisterModeManual	2

Table 10-48: MBIM_REGISTRATION_FLAGS is a bitmap that indicates network conditions related to auto-attach and manual network selection capabilities.

Table 10-48: MBIM_REGISTRATION_FLAGS

Types	Mask	Description
MBIM_REGISTRATION_NONE	0	No bits are set.
MBIM_REGISTRATION_MANUAL_SELECTION_NOT_AVAILABLE	1	If set, this bit indicates that the network currently does not support manual network selection. This can vary on a per-cell basis.

MBIM_REGISTRATION_PACKET_SERVICE_AUTOMATIC_ATTACH	2	If set, this bit indicates that the function will auto-attach to the network after registration without involvement from the host.

If a function sets the MBIM_REGISTRATION_PACKET_SERVICE_AUTOMATIC_ATTACH bit, it is expected to auto-attach to the provider network after registration without involvement by the host. The function will indicate the result to the host using an unsolicited Packet Service notification following the registration. Intermediate queries by the host will be answered by the device using the MBIMPacketServiceStateAttaching status. If the auto-attach fails, the function might not retry the attach procedure, in which case it will report the Packet Service status as MBIMPacketServiceStateDetached. Upon success, the function must report MBIMPacketServiceStateAttached. Prior to sending a MBIM_CID_CONNECT (MBIMActivationCommandActivate), the host will send a MBIM_CID_PACKET_SERVICE (MBIMPacketServiceActionAttach) set request if the function has previously reported MBIMPacketServiceStateDetached. Conversely, after sending a MBIM_CID_CONNECT (MBIMActivationCommandDeactivate) the host does not send a set MBIM_CID_PACKET_SERVICE (MBIMPacketServiceActionDetach). This mechanism allows the function to manage its own packet service detach state.

If a function does not set the MBIM_REGISTRATION_PACKET_SERVICE_AUTOMATIC_ATTACH flag, then (1) prior to MBIM_CID_CONNECT (MBIMActivationCommandActivate) the host sends a set MBIM_CID_PACKET_SERVICE (MBIMPacketServiceActionAttach), as long as the function is not already attached or attaching; and (2), the host issues an MBIMPacketServiceActionDetach following the MBIM_CID_CONNECT (MBIMActivationCommandDeactivate).

10.5.9.4 SET

The following structure shall be used in the InformationBuffer

Table 10-49: MBIM_SET_REGISTRATION_STATE

Offset	Size	Field	Type	Description

0	4	ProviderIdOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a numeric (0-9) string ProviderId that represents the network provider identity. For GSM-based networks, this string is a concatenation of a three-digit Mobile Country Code (MCC) and a two or three-digit Mobile Network Code (MNC). GSM-based carriers may have more than one MNC, and hence more than one ProviderId . For CDMA-based networks, this string is a five-digit System ID (SID). Generally, a CDMA-based carrier has more than one SID. Typically, the carrier has one SID for each market, which is usually divided geographically within a nation by regulations, such as Metropolitan Statistical Areas (MSA) in the United States of America. Devices of CDMA-based devices must specify 0 if this information is not available.
4	4	ProviderIdSize	SIZE (0..12)	Size used for the ProviderId
8	4	RegisterAction	MBIM_REGISTER_ACTION	The registration action that the device is requested to perform. If this member is set to MBIMRegisterActionAutomatic, the ProviderId member should be ignored See Table 10-45: MBIM_REGISTER_ACTION

12	4	DataClass	MBIM_DATA_CLASS	A bitmap that represents the data access technologies that are preferred for a connection. For a detailed list of values, see Table 10-11: MBIM_DATA_CLASS. When multiple data-classes are set as preferred, devices are expected register to the highest available data-class technology that is currently visible. Devices should attempt to register the best available data-class as requested. If the device cannot register the data-class specified in this member, it should register the best available data-class.
16		DataBuffer	DATABUFFER	ProviderId

10.5.9.5 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.9.6 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-50: MBIM_REGISTRATION_STATE_INFO

Offset	Size	Field	Type	Description
0	4	NwError	UINT32	A network-specific error. Table 10-44: 3GPP TS 24.008 Cause codes for NwError shows the connection failure values as documented in the Cause values in the appendixes of the 3GPP TS 24.008 Specification [3GPP24008].
4	4	RegisterState	MBIM_REGISTER_STATE	See Table 10-46: MBIM_REGISTER_STATE
8	4	RegisterMode	MBIM_REGISTER_MODE	See Table 10-47:

				MBIM_REGISTER_MODE
12	4	AvailableDataClasses	MBIM_DATA_CLASS	A bitmap of the values in Table 10-11: MBIM_DATA_CLASS that represent the supported data classes in the registered network, for the cell the device is registered in. The value is set to MBIMDataClassNone as long as the RegisterState is not MBIMRegisterStateHome, MBIMRegisterStateRoaming or MBIMRegisterStatePartner.
16	4	CurrentCellularClass	MBIM_CELLULAR_CLASS	Indicates the current cellular class in use for a multi-mode function. See Table 10-8: MBIM_CELLULAR_CLASS. For a single-mode function this is the same as the cellular class reported in MBIM_CID_DEVICE_CAPS. For multi-mode functions, a transition from CDMA to GSM or vice versa is indicated with an updated CurrentCellularClass
20	4	ProviderIdOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a numeric (0-9) string ProviderId that represents the network provider identity. For GSM-based networks, this string is a concatenation of a three-digit Mobile Country Code (MCC) and a two or three-digit Mobile Network Code (MNC). GSM-based carriers may have more than one MNC, and hence more than one ProviderId .

				For CDMA-based networks, this string is a five-digit System ID (SID). Generally, a CDMA-based carrier has more than one SID. Typically, a carrier has one SID for each market, which is usually divided geographically within a nation by regulations, such as Metropolitan Statistical Areas (MSA) in the United States of America. Devices of CDMA-based devices must specify MBIM_CDMA_DEFAULT_PROVIDER_ID if this information is not available. When processing a query request, and the registration state is in automatic register mode, this member contains the provider ID that the device is currently associated with (if applicable). When the registration state is in manual register mode, this member contains the provider ID that the device is requested to register with (even if the provider is unavailable). When processing a set request and the registration state is in manual mode, this contains the provider ID selected by the Host for the device to register with. When the registration state is in automatic register mode, this parameter is ignored. CDMA 1xRTT providers must be set to MBIM_CDMA_DEFAULT_PROVIDER_ID if the provider ID is not available

24	4	ProviderIdSize	SIZE (0..12)	Size used for ProviderId
28	4	ProviderNameOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string ProviderName that represents the network provider's name. This member is limited to, at most, MBIM_PROVIDERNAME_LEN characters. For GSM-based networks, if the Preferred Presentation of Country Initials and Mobile Network Name (PPCI&N) is longer than twenty characters, the device should abbreviate the network name. This member is ignored when the Host sets the preferred provider list. Devices should specify a NULL string for devices that do not have this information.
32	4	ProviderNameSize	SIZE (0..40)	Size used for ProviderName
36	4	RoamingTextOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string RoamingText to inform the user that the device is roaming. This member is limited to at most 63 characters. This text should provide additional information to the user when the registration state is either MBIMRegisterStatePartner or MBIMRegisterStateRoaming. This member is optional.

40	4	RoamingTextSize	SIZE (0..126)	Size used for RoamingText
44	4	RegistrationFlag	MBIM_REGISTRATION_FLAGS	Flag set per Table 10-48: MBIM_REGISTRATION_FLAGS.
48		DataBuffer	DATABUFFER	ProviderId ProviderName RoamingText

10.5.9.7 NOTIFICATION

See Table 10-50: MBIM_REGISTRATION_STATE_INFO

10.5.9.8 STATUS CODES

The following table shows possible error status codes.

Status code	Description
MBIM_STATUS_PIN_REQUIRED	Device requires PIN value input.
MBIM_STATUS_FAILURE	Unable to get registration state.
MBIM_STATUS_NOT_INITIALIZED	The operation failed because the function is in the process of initializing. Retry the operation after the ready-state of the device changes to MBIMReadyStateInitialized.
MBIM_STATUS_BAD_SIM	The operation failed because a bad SIM card was detected.
MBIM_STATUS_SIM_NOT_INSERTED	The operation failed because the SIM card was not inserted fully into the device.

Functions can return the following error codes (in addition to the above listed) only in the event a session activation set operation fails.

Status code	Description
MBIM_STATUS_FAILURE	Unable to set registration state. More information is specified in NwError (see Table 10-44: 3GPP TS 24.008 Cause codes for NwError). For other error codes, NwError should

	be set to 0.
MBIM_STATUS_NO_DEVICE_SUPPORT	CDMA-based devices must return this error code, if a set registration request is for manual registration.
MBIM_STATUS_SERVICE_NOT_ACTIVATED	Service activation failed. Subscription expired. Device does not allow setting registration state.
MBIM_STATUS_PROVIDER_NOT_VISIBLE	Provider is not visible for registration. This is for manual registration.
MBIM_STATUS_INVALID_PARAMETERS	If the request is manual registration to a forbidden provider.
MBIM_STATUS_BUSY	The device is busy and unable to change the registration mode. This scenario can occur if the device does not permit changing the registration mode when a session is activated.

10.5.9.9 STATUS CODES FOR SET OPERATION

Status code	Description
MBIM_STATUS_SERVICE_NOT_ACTIVATED	Service activation failed. Subscription expired. Device does not allow setting registration state.
MBIM_STATUS_PROVIDER_NOT_VISIBLE	Provider is not visible for registration. This is for manual registration.
MBIM_STATUS_INVALID_PARAMETERS	If the request is manual registration to a forbidden provider.
MBIM_STATUS_BUSY	The device is busy and unable to change the registration mode. This scenario can occur if the device does not permit changing the registration mode when a PDP context is activated.

10.5.10 MBIM_CID_PACKET_SERVICE

10.5.10.1 DESCRIPTION

This command is used to instruct devices to perform packet service attach or detach actions on the current registered provider's network for both GSM-based and CDMA-based MB devices. In addition to the packet service attach/detach status, this CID is used to determine data class availability, the currently used data class information, and the uplink and downlink speeds.

Note: This CID is not about IP (v4 or v6) connectivity. The host may use this CID to determine if it will be able to obtain IP service, or to prepare for it.

CDMA-based devices should use this as an opportunity to release the network resource allocation if possible.

Some SIM cards enable the MB device to register only on the packet domain and not the circuit-switched domain. Once a data call ends, the host sends an MBIM_CID_PACKET_SERVICE set request to detach packet service. This causes the MB device to detach from the packet domain. The MB device unregisters from the network and goes into a power save mode.

Devices should send MBIM_STATUS_PACKET_SERVICE unsolicited events. Devices should fail PDP activation if the device packet service state is not set to MBIMPacketServiceStateAttached.

The function can expect the host not to proceed with context activation until the packet service state has reached MBIMPacketServiceStateAttached.

The values for UplinkSpeed and DownlinkSpeed are expected to be nominal speeds, and are not expected to reflect actual throughput, goodput or any other such precise values. These speed values are analogous to the nominal speeds in other networking technologies, like "11 Mbps" in IEEE 802.11 or "100 Mbps" in IEEE 802.3. Nevertheless, these values may change. If so, the function must use the unsolicited event to notify the host.

Note: Uplink and downlink are the 3gpp definitions which not is the same as the upstream/downstream in the [USB20] and [USB30] specifications.

Functions use the MBIMPacketServiceStateAttaching and MBIMPacketServiceStateDetaching transient states only when responding to query requests or in unsolicited events. Functions must not return these transient states in MBIM_COMMAND_DONE for Set request. Functions must only send MBIM_COMMAND_DONE for Set request after they have successfully completed a packet service attach or detach, or detected an error.

Set and Query:

MBIM_PACKET_SERVICE_INFO is returned from both **Set** and **Query** complete messages in the InformationBuffer.

For **Query**, the InformationBuffer is empty. For **Set**, it contains an MBIM_SET_PACKET_SERVICE.

Unsolicited Event:

The Event InformationBuffer contains an MBIM_PACKET_SERVICE_INFO structure.

10.5.10.2 PARAMETERS

Table 10-51: Parameters

	Set	Query	Notification
Command	MBIM_SET_PACKET_SERVICE	Empty	NA
Response	MBIM_PACKET_SERVICE_INFO	MBIM_PACKET_SERVICE_INFO	MBIM_PACKET_SERVICE_INFO

10.5.10.3 DATA STRUCTURES

Table 10-52: MBIM_PACKET_SERVICE_ACTION

Types	Value
MBIMPacketServiceActionAttach	0
MBIMPacketServiceActionDetach	1

Table 10-53: MBIM_PACKET_SERVICE_STATE

Types	Value
MBIMPacketServiceStateUnknown	0
MBIMPacketServiceStateAttaching	1
MBIMPacketServiceStateAttached	2
MBIMPacketServiceStateDetaching	3
MBIMPacketServiceStateDetached	4

10.5.10.4 SET

The following structure shall be used in the InformationBuffer

Table 10-54: MBIM_SET_PACKET_SERVICE

Offset	Size	Field	Type	Description
0	4	PacketServiceAction	MBIM_PACKET_SERVICE_ACTION	See Table 10-52: MBIM_PACKET_SERVICE_ ACTION

10.5.10.5 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.10.6 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-55: MBIM_PACKET_SERVICE_INFO

Offset	Size	Field	Type	Description
0	4	NwError	UINT32	A network-specific error. Table 10-44: 3GPP TS 24.008 Cause codes for NwError shows the connection failure values as documented in the Cause values in the appendixes of the 3GPP TS 24.008 Specification [3GPP24008].
4	4	PacketServiceState	MBIM_PACKET_SERVICE_STA TE	See Table 10-53: MBIM_PACKET_SERVICE_STA TE

8	4	HighestAvailableDataClass	MBIM_DATA_CLASS	The highest available data class in the current cell, specified according to Table 10-11: MBIM_DATA_CLASS. Functions must set this member to MBIMDataClassNone as long as the function is not in packet service state attached. Except for HSPA (i.e., HSUPA and HSDPA), the function sets this member to a single MBIM_DATA_CLASS value. For HSPA data services, functions specify a bit-wise OR of MBIM_DATA_CLASS_HSDPA and MBIM_DATA_CLASS_HSUPA. For cells that support HSDPA but not HSUPA, only HSDPA is indicated (implying UMTS data class for uplink data). The highest available data class may not be indicated by the cell at packet attach, but may become known first when a context is activated. Whenever the highest available data class changes, functions shall send a notification indicating the new value of HighestAvailableDataClass.

12	8	UplinkSpeed	UINT64	Contains the uplink bit rate, in bits per second.
20	8	DownlinkSpeed	UINT64	Contains the downlink bit rate, in bits per second.

10.5.10.7 NOTIFICATION

See Table 10-55: MBIM_PACKET_SERVICE_INFO

10.5.10.8 STATUS CODES

Status code	Description
MBIM_STATUS_FAILURE	Packet-attach or packet-detach has failed. More information is set at NwError member of MBIM_PACKET_SERVICE_INFO structure (see Table 10-44: 3GPP TS 24.008 Cause codes for NwError). For other MBIM_STATUS_ERROR values, NwError should be set to zero.
MBIM_STATUS_SERVICE_NOT_ACTIVATED	The device does not allow setting packet service state because of service activation failure or expired subscription.
MBIM_STATUS_PROVIDER_NOT_VISIBLE	Provider is not visible for packet service operations.
MBIM_STATUS_NOT_REGISTERED	The device is not in registered state to perform a packet-attach operation.
MBIM_STATUS_NO_DEVICE_SUPPORT	SET packet service is not supported by this CDMA-based device.
MBIM_STATUS_RADIO_POWER_OFF	Unable to packet-attach because the radio is turned off.

10.5.11 MBIM_CID_SIGNAL_STATE

10.5.11.1 DESCRIPTION

This command returns or sets the current signal state.

On a set request from the Host, devices should:

- Return the current values for Rssi and ErrorRate in the MBIM_SIGNAL_STATE_INFO structure in addition to reporting the absolute values for SignalStrengthInterval, RssiThreshold and ErrorRateThreshold that has been set in the device.
- Internally cache the SignalStrengthInterval and/or RssiThreshold values even if the device is not currently registered with any operator and that any restriction imposed by device in setting parameters can only be possible post-registration state. The device should try to apply these settings in the next immediate available situation.
- Complete the request successfully, if the hardware and/or software radio switch state is currently OFF. Device cache the request data and start reporting the signal strength after the switch is turned ON.
- Can fail this request with the appropriate Status error code set.

Devices can do the following when processing query requests from the Host:

- Return the current values for Rssi and ErrorRate in the MBIM_SIGNAL_STATE_INFO structure in addition to reporting the absolute values for SignalStrengthInterval and RssiThreshold that has been set in the device.
- Fail this request with the appropriate Status error code set.

Query and Set:

MBIM_SIGNAL_STATE_INFO is returned from both **Set** and **Query** complete messages in the InformationBuffer.

For **Query**, the InformationBuffer is empty. For **Set**, it contains an MBIM_SET_SIGNAL_STATE.

Unsolicited Event:

The Event InformationBuffer contains an MBIM_SIGNAL_STATE_INFO structure.

Devices must notify the HOST if the Rssi value changes by at least +/-5 decibels from the last reported value, or at a maximum frequency of one indication every 5 seconds. The threshold value is specified in the RssiThreshold field of Table 10-58: MBIM_SIGNAL_STATE_INFO; while the maximum frequency value is specified in the SignalStrengthInterval field of Table 10-58: MBIM_SIGNAL_STATE_INFO.

The CellularClass field of the Table 10-14: MBIM_DEVICE_CAPS_INFO structure controls how the Rssi value will be interpreted by the Host. If CellularClass is MBIMCellularClassGSM, Rssi is reported as decibels above the device's sensitivity noise floor. If CellularClass is MBIMCellularClassCDMA, Rssi is reported as compensated RSSI (accounts for noise). In the case of multi-mode devices, Rssi uses GSM reporting format with appropriate translation.

10.5.11.2 PARAMETERS

Table 10-56: Parameters

	Set	Query	Notification
Command	MBIM_SET_SIGNAL_STATE	Empty	NA
Response	MBIM_SIGNAL_STATE_INFO	MBIM_SIGNAL_STATE_INFO	MBIM_SIGNAL_STATE_INFO

10.5.11.3 SET

The following structure shall be used in the InformationBuffer

Table 10-57: MBIM_SET_SIGNAL_STATE

Offset	Size	Field	Type	Description
0	4	SignalStrengthInterval	UINT32	The reporting interval, in seconds. If set to zero, use default behavior in the function
4	4	RssiThreshold	UINT32	The difference in RSSI coded values (Table 10-58: MBIM_SIGNAL_STATE_INFO) that trigger a report. If set to zero, use default behavior in the function
8	4	ErrorRateThreshold	UINT32	The difference in ErrorRate coded values (Table 10-58: MBIM_SIGNAL_STATE_INFO) that trigger a report. Use 0xffffffff to not receive any report based on changes in ErrorRateThreshold. If set to zero, use default behavior in the function

10.5.11.4 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.11.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-58: MBIM_SIGNAL_STATE_INFO

<table>
<tr><th>Offset</th><th>Size</th><th>Field</th><th>Type</th><th>Description</th></tr>
<tr><td>0</td><td>4</td><td>Rssi</td><td>UINT32</td><td>
<table>
<tr><th>Signal Strength (in dBm)</th><th>Coded Value (Minimum=0, Maximum=31)</th></tr>
<tr><td>-113 or less</td><td>0</td></tr>
<tr><td>-111</td><td>1</td></tr>
<tr><td>-109</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>-51 or greater</td><td>31</td></tr>
<tr><td>Unknown or undetectable</td><td>99</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>4</td><td>ErrorRate</td><td>UINT32</td><td>
<table>
<tr><th>Channel bit error rate (in %)</th><th>Frame error rate (in %)</th><th>Coded value (Min=0, Max=7)</th></tr>
<tr><td>< 0.2</td><td>< 0.01</td><td>0</td></tr>
<tr><td>0.2-0.4</td><td>0.01-0.1</td><td>1</td></tr>
<tr><td>0.4-0.8</td><td>0.1-0.5</td><td>2</td></tr>
<tr><td>0.8-1.6</td><td>0.5-1.0</td><td>3</td></tr>
<tr><td>- 3.2</td><td>1.0 - - 2.0</td><td>4</td></tr>
<tr><td>- 6.4</td><td>2.0-4.0</td><td>5</td></tr>
<tr><td>6.4-12.8</td><td>4.0-8.0</td><td>6</td></tr>
<tr><td>> 12.8</td><td>> 8.0</td><td>7</td></tr>
<tr><td colspan="2">Unknown or undetectable</td><td>99</td></tr>
</table>
</td></tr>
<tr><td>8</td><td>4</td><td>SignalStrengthInterval</td><td>UINT32</td><td>The reporting interval, in seconds</td></tr>
<tr><td>12</td><td>4</td><td>RssiThreshold</td><td>UINT32</td><td>The difference in RSSI coded values that trigger a report.</td></tr>
</table>

16	4	ErrorRateThreshold	UINT32	The difference in ErrorRate coded values that trigger a report. Use 0xffffffff for, do not care.

10.5.11.6 NOTIFICATION

See Table 10-58: MBIM_SIGNAL_STATE_INFO.

10.5.11.7 STATUS CODES

This CID only uses Generic Status Codes (see Use of the Status Codes section 9.4.5).

10.5.12 MBIM_CID_CONNECT

10.5.12.1 DESCRIPTION

This command activates or deactivates a particular IP data stream session and reads the activation state of a session. The set operation requests an IP data stream session (to a specific APN) to be made available to the host. The function may already have an IP data stream session to the APN on the radio interface. In that case the function shall make this IP data stream available to the host. If no IP data stream exists on the radio interface, the function shall try to establish it to the requested APN on the radio interface and make it available to the host.

The device must send appropriate event notifications whenever the activation state changes.

The **Set** command can succeed only if the device is in a register state of MBIMRegisterStateHome, MBIMRegisterStatePartner, or MBIMRegisterStateRoaming. When packet service is active, the device must also be in an attach state of MBIMPacketServiceStateAttached.

Functions make IP data stream sessions available to the host only as instructed by it. Once a function has indicated an IP data stream session as unavailable to the host, the function must not automatically make the IP data stream session available again to the host. If the access string is not provided in the MBIM_CID_CONNECT request, the function shall try to set up an IP data stream session to the network's default APN. If the function already has an active session to the network, it may choose to make it available to the host. A multi-mode single-carrier function should use the ActivationCommand parameters for GSM even when connecting to the network in CDMA coverage area. IP connectivity must be maintained across the handoff if IP configuration doesn't change.

The function may deactivate an IP data stream session independently of the Host. This may occur when network connectivity has been lost for a period that exceeds the threshold of temporary loss of signal, or as part of a graceful shutdown or state cleanup. On Set requests, the Host Provides both SessionId and ActivationCommand parameters in the MBIM_CONNECT_INFO data structure. It instructs the device to activate or deactivate an IP data streamidentified by SessionId, based on the ActivationCommand parameter value MBIMActivationCommandActivate or MBIMActivationCommandDeactivate.

- If the service or subscription requires activation, the device should return error code MBIM_STATUS_SERVICE_NOT_ACTIVATED. The session activation may not happen until the service or subscription is activated. All the emergency services might be available subject to the support from the device and the operator. The operating system might call the MBIM_SET_SERVICE_ACTIVATION in response to this error code.
- If the device receives an IP data stream session activation request while its maximum number of IP data streams is currently activated, it returns error code MBIM_STATUS_MAX_ACTIVATED_CONTEXTS.
- If the device receives a context deactivation request but the context identified by SessionId is not currently activated, it returns error code MBIM_STATUS_CONTEXT_NOT_ACTIVATED.
- If the device receives an activation request for an additional IP data stream to the same APN the device may reject the activation with error code MBIM_STATUS_OPERATION_NOT_ALLOWED

The device uses the following logic to determine the validity of AccessString, UserName, and Password settings from a set request:

- If ActivationCommand is MBIMActivationCommandDeactivate, the device should ignore the settings of these three parameters. The rest of the cases only consider the case when ActivationCommand is MBIMActivationCommandActivate.

On a **Set** request, the Host may specify an IP type to activate. If MBIMContextIPTypeDefault is specified, the host does not care what IP type is activated. If a value other than MBIMContextIPTypeDefault is specified, the device must only activate that context. When activated, the device must return the appropriate IPType in response to a query request.

Functions use the MBIMActivationStateActivating and MBIMActivationStateDeactivating transient states only when responding to **Query** requests or in **Unsolicited Events**. Functions must not return these transient states in MBIM_COMMAND_DONE for **Set** request. Functions must only send MBIM_COMMAND_DONE for **Set** request after they have successfully activated or deactivated an IP data stream session, or detected an error.

Functions use the MBIMActivationStateActivated state to signal to the host that the link is up, and that it is now okay to use this IP data stream.

If the function receives a **Set** request for this CID for a given SessionId while processing another **Set** request for this CID for that same SessionId, the function shall fail the second **Set** request by returning MBIM_STATUS_BUSY, and continue processing the original **Set** request.

On **Query** requests, the Host uses this object to find out the activation state.

Query and Set:

For **Query**, the InformationBuffer contains an MBIM_CONNECT_INFO in which the only relevant field is the SessionId. The SessionId in a **Query** indicates which IP data stream's connect information is to be returned by the function. For **Set**, the InformationBuffer contains an MBIM_SET_CONNECT.

MBIM_CONNECT_INFO is returned from both **Set** and **Query** complete messages in the InformationBuffer.

Unsolicited Event:

The Event InformationBuffer contains an MBIM_CONNECT_INFO structure.

The function must also notify the host when changes in context state are caused unrelated to a **Set** request from the host. For example, the function must notify the host if the network deactivates a context. The function must issue an **Unsolicited Event** to notify the host about any such applicable context state changes (i.e., any changes that can be expressed via Table 10-68: MBIM_CONNECT_INFO).

Functions that support separate voice and data connections must follow these guidelines:

- At the time of initialization, the VoiceCallState must be set to MBIMVoiceCallStateNone.
- On the start of the voice call, send an event notification with VoiceCallState set to MBIMVoiceCallStateInProgress. All the other members must reflect their current state. In case of no active connection during the voice call, the SessionId should be set to "0".

Once the voice call is completed, send an event notification with VoiceCallState set to MBIMVoiceCallStateHangUp. All the other members must reflect their current state. In case of no active connection during the voice call hang up, the SessionId should be set to "0".

10.5.12.2 PARAMETERS

Table 10-59: Parameters

	Set	Query	Notification
Command	MBIM_SET_CONNECT	MBIM_CONNECT_INFO	NA
Response	MBIM_CONNECT_INFO	MBIM_CONNECT_INFO	MBIM_CONNECT_INFO

10.5.12.3 DATA STRUCTURES

Table 10-60: MBIM_ACTIVATION_COMMAND

Types	Value
MBIMActivationCommandDeactivate	0

MBIMActivationCommandActivate	1

Table 10-61: MBIM_COMPRESSION

Types	Value
MBIMCompressionNone	0
MBIMCompressionEnable	1

Table 10-62: MBIM_AUTH_PROTOCOL

Types	Value
MBIMAuthProtocolNone	0
MBIMAuthProtocolPap	1
MBIMAuthProtocolChap	2
MBIMAuthProtocolMsChapV2	3

Table 10-63: MBIM_CONTEXT_IP_TYPE

Types	Value	Description
MBIMContextIPTypeDefault	0	It is up to the function to decide, the host does not care. This type is mandatory
MBIMContextIPTypeIPv4	1	IPv4 context.
MBIMContextIPTypeIPv6	2	IPv6 context.
MBIMContextIPTypeIPv4v6	3	Corresponds to the IPv4v6 context type specified in 3gpp. See TS 23.06 and 24.008
MBIMContextIPTypeIPv4AndIPv6	4	Combining two PDP contexts (one for IPv4 and one for IPv6) into one

		data stream

Table 10-64: MBIM_ACTIVATION_STATE

Types	Value
MBIMActivationStateUnknown	0
MBIMActivationStateActivated	1
MBIMActivationStateActivating	2
MBIMActivationStateDeactivated	3
MBIMActivationStateDeactivating	4

Table 10-65: MBIM_VOICE_CALL_STATE

Types	Value
MBIMVoiceCallStateNone	0
MBIMVoiceCallStateInProgress	1
MBIMVoiceCallStateHangUp	2

Table 10-66: MBIM_CONTEXT_TYPES

Types	UUID Value	Description
MBIMContextTypeNone	B43F758C-A560-4B46-B35E-C5869641FB54	The context is not yet provisioned
MBIMContextTypeInternet	7E5E2A7E-4E6F-7272-736B-656E7E5E2A7E	The context represents a connection to the Internet This context type is mandatory.
MBIMContextTypeVpn	9B9F7BBE-8952-44B7-83AC-CA41318DF7A0	The context represents a connection to virtual private network (VPN to a corporate network).
MBIMContextTypeVoice	88918294-0EF4-4396-8CCA-A8588FBC02B2	The context represents a connection to a Voice-over-IP (VOIP) service

MBIMContextTypeVideoShare	05A2A716-7C34-4B4D-9A91-C5EF0C7AAACC	The context represents a connection to a video sharing service
MBIMContextTypePurchase	B3272496-AC6C-422B-A8C0-ACF687A27217	The context represents a connection to an over-the-air activation site.
MBIMContextTypeIMS	21610D01-3074-4BCE-9425-B53A07D697D6	The context represents a connection to IMS.
MBIMContextTypeMMS	46726664-7269-6BC6-9624-D1D35389ACA9	The context represents a connection to MMS.
MBIMContextTypeLocal	A57A9AFC-B09F-45D7-BB40-033C39F60DB9	The context represents a local connection which is terminated at the device. Hence, the IP traffic will not be sent over the air.

The intention with MBIMContextTypes is to have a possibility to indicate what type of traffic is sent over the IP stream. It enables both the host and the device to make intelligent enhancements. This could be especially interesting for multiple PDP context scenarios.

10.5.12.4 SET

The following structure shall be used in the InformationBuffer

Table 10-67: MBIM_SET_CONNECT

Offset	Size	Field	Type	Description
0	4	SessionId	UINT32	Host specifies this member to uniquely identify the session for the IP data stream and its corresponding state. Device must use the value in this member when completing set requests. The Host uses the value in this member in subsequent query requests as well as disconnect requests to the device. The value shall also be used as an identifier on

				the data transport. The function shall reject the request if the host uses a SessionId that exceeds MaxSessions-1, where MaxSessions is in Table 10-14: MBIM_DEVICE_CAPS_INFO Informative note: It is expected that hosts will use 0 for the Internet connection. Valid values range from 0 to MaxSessions-1.
4	4	ActivationCommand	MBIM_ACTIVATION_COMMAND	See Table 10-60: MBIM_ACTIVATION_COMMAND
8	4	AccessStringOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string AccessString to access the network. For GSM-based networks, this would be an Access Point Name (APN) string such as "data.thephone-company.com". For CDMA-based networks, this might be a special dial code such as "#777" or a Network Access Identifier (NAI) such as "foo@thephone-company.com". This member can be NULL, to request that the network assign the default APN. NOTE: Not all networks support this NULL APN convention, so a connect failure due to an invalid APN

				is a possible outcome. The size of the string should not exceed 100 characters.
12	4	AccessStringSize	SIZE (0..200)	Size used for AccessString
16	4	UserNameOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string UserName that represents the username to authenticate. This member can be NULL.
20	4	UserNameSize	SIZE (0..510)	Size used for UserName
24	4	PasswordOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string Password that represents the username's password. This member can be NULL.
28	4	PasswordSize	SIZE (0..510)	Size used for Password
32	4	Compression	MBIM_COMPRESSION	See Table 10-61: MBIM_COMPRESSION
36	4	AuthProtocol	MBIM_AUTH_PROTOCOL	See Table 10-62: MBIM_AUTH_PROTOCOL
40	4	IPType	MBIM_CONTEXT_IP_TYPE	See Table 10-63: MBIM_CONTEXT_IP_TYPE
44	16	ContextType	MBIM_CONTEXT_TYPES	Specifies the type of context being represented, for example, Internet connectivity, VPN (a connection to a corporate network), or Voice-over-IP (VOIP). Devices should specify MBIMContextTypeNone for empty or unprovisioned

				contexts. See Table 10-66: MBIM_CONTEXT_TYPES
60		DataBuffer	DATABUFFER	AccessString UserName Password

10.5.12.5 QUERY

See Table 10-68: MBIM_CONNECT_INFO and **Query** description.

10.5.12.6 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-68: MBIM_CONNECT_INFO

Offset	Size	Field	Type	Description
0	4	SessionId	UINT32	The Host specifies a value for this member at the time of the connect request by using MBIM_CID_CONNECT. Devices must copy this value and use it when they notify the MB Service on subsequent connection state changes.
4	4	ActivationState	MBIM_ACTIVATION_STATE	See Table 10-64: MBIM_ACTIVATION_STATE
8	4	VoiceCallState	MBIM_VOICE_CALL_STATE	See Table 10-65: MBIM_VOICE_CALL_STATE
12	4	IPType	MBIM_CONTEXT_IP_TYPE	See Table 10-63: MBIM_CONTEXT_IP_TYPE

16	16	ContextType	MBIM_CONTEXT_TYPES	Specifies the type of context being represented, for example, Internet connectivity, VPN (a connection to a corporate network), or Voice-over-IP (VOIP). Devices should specify MBIMContextTypeNone for empty or unprovisioned contexts. See Table 10-66: MBIM_CONTEXT_TYPES
32	4	NwError	UINT32	A network-specific error. Table 10-44: 3GPP TS 24.008 Cause codes for NwError shows the connection failure values as documented in the Cause values in the appendixes of the 3GPP TS 24.008 Specification [3GPP24008].

10.5.12.7 NOTIFICATION

See Table 10-68: MBIM_CONNECT_INFO

10.5.12.8 STATUS CODES

Status code	Description
MBIM_STATUS_SUCCESS	The operation succeeded. Functions can return this value if the context has already been activated.
MBIM_STATUS_RADIO_POWER_OFF	The operation failed because the radio is currently turned off. This error code should be returned only in response to a MBIM_CID_CONNECT set request. If the radio state is off then the function should respond to MBIM_CID_CONNECT query requests with MBIM_STATUS_SUCCESS and specify the current context state as MBIMActivationStateDeactivated.
MBIM_STATUS_SERVICE_NOT_ACTIVATED	The operation failed because either the subscription has expired, or the device does not allow PDP activation.

MBIM_STATUS_PROVIDER_NOT_VISIBLE	The operation failed because the service provider is not currently visible.
MBIM_STATUS_MAX_ACTIVATED_CONTEXTS	The operation failed because the maximum number of activated contexts has been reached.
MBIM_STATUS_INVALID_ACCESS_STRING	The operation failed because the access string is invalid.
MBIM_STATUS_INVALID_USER_NAME_PWD	The operation failed because the user name and/or password supplied are invalid. Network specific error code may be available in NwError (see Table 10-44: 3GPP TS 24.008 Cause codes for NwError).
MBIM_STATUS_PACKET_SERVICE_DETACHED	The operation failed because packet service is detached.
MBIM_STATUS_NOT_REGISTERED	The operation failed because the device is not in the registered state to perform PDP activation.
MBIM_STATUS_VOICE_CALL_IN_PROGRESS	The operation failed and cannot proceed with PDP activation because a voice call is currently in progress. This value applies only to devices with voice class is set to MBIMVoiceClassSeparateVoiceData.
MBIM_STATUS_CONTEXT_NOT_ACTIVATED	The operation failed because the context identified by SessionId is not the currently activated context.
MBIM_STATUS_CONTEXT_NOT_SUPPORTED	The operation failed because it could not support the type of context identified by ContextType.
MBIM_STATUS_OPERATION_NOT_ALLOWED	If the device receives an activation request for an additional IP data stream to the same APN the device may reject the activation with this error code.

10.5.13 MBIM_CID_PROVISIONED_CONTEXTS

10.5.13.1 DESCRIPTION

This command reads or updates the provisioned context entries stored on the MB device or the Subscriber Identity Module (SIM).

Functions should return MBIM_STATUS_NOT_SUPPORTED if the MB device they support does not support retrieval of provisioned contexts.

GSM-based functions must support query and set operations. CDMA-based devices must support query operations reporting Simple IP (MBIM_CTRL_CAPS_CDMA_SIMPLE_IP).

The provisioned context entries stored on the MB device or the SIM are local to the device

Provisioned contexts are the connectivity parameters (AccessString, UserName, and Password) that are either pre-provisioned by the Operators or OTA provisioned by the device and can be stored either in the device memory or SIM. The connectivity parameters returned by the Provisioned contexts will be used by the Host for PDP activation.

The list of provisioned contexts must be restricted only to the home provider network even though the device may have the capability to store multiple network provider contexts. The context list must always be the home provider network specific even in case of roaming.

The SET MBIM_CID_PROVISIONED_CONTEXT operation should associate the context with the network provider that is specified in the set request in the ProviderId field of Table 10-71: MBIM_SET_PROVISIONED_CONTEXT. Provisioned context stored through set MBIM_CID_PROVISIONED_CONTEXT requests must persist across system restarts and device power cycles.

CDMA devices that are configured for SimpleIP, reporting MBIM_CTRL_CAPS_CDMA_SIMPLE_IP in ControlCaps must always return at least one provisioned context filled with the correct AccessString , UserName, and Password members for the query request from Host.

Provisioned context list should be pre-provisioned in the device. Host-initiated updates are accomplished via MBIM_CID_PROVISIONED_CONTEXT operations; the device shall not heed any updates attempted via the MBIM_CID_CONNECT operation. Updates, initiated by either the device or the operator, are also possible (for example, using SMS or OTA).

Query and Set:

MBIM_PROVISIONED_CONTEXTS_INFO is returned from both **Query** and **Set** complete messages in the InformationBuffer.

For **Query**, the InformationBuffer is empty. For **Set**, it contains an MBIM_SET_PROVISIONED_CONTEXT.

Unsolicited Event:

The Event InformationBuffer contains an MBIM_PROVISIONED_CONTEXTS_INFO structure.

In some cases, the list of provisioned contexts is updated by the network either Over-The-Air (OTA) or by Short Message Service (SMS). The function must update the list of provisioned contexts accordingly. Thereafter, functions must notify the Host about the updates using this event with the updated list.

10.5.13.2 PARAMETERS

Table 10-69: Parameters

	Set	Query	Notification
Command	MBIM_SET_PROVISIONED_CONTEXT	Empty	NA
Response	MBIM_PROVISIONED_CONTEXTS_INFO	MBIM_PROVISIONED_CONTEXTS_INFO	MBIM_PROVISIONED_CONTEXTS_INFO

10.5.13.3 DATA STRUCTURES

Table 10-70: MBIM_CONTEXT

Offset	Size	Field	Type	Description
0	4	ContextId	UINT32	A unique ID for this context. For set MBIM_CID_PROVISIONED_CONTEXT requests, the Host can set the value to 0xffffffff. If this value is used, the device should decide the index for storing the context information. 0xffffffff shall never be returned in response to query MBIM_CID_PROVISIONED_CONTEXT requests.
4	16	ContextType	MBIM_CONTEXT_TYPES	Specifies the type of context being represented, for example, Internet connectivity, VPN (a connection to a corporate network), or Voice-over-IP (VOIP). Devices should specify MBIMContextTypeNone for empty or unprovisioned contexts. See Table 10-66: MBIM_CONTEXT_TYPES

20	4	AccessStringOffset	OFFSET	Offset in data buffer to a string AccessString to access the network. For GSM-based networks, this would be an Access Point Name (APN) string such as "data.thephone-company.com". For CDMA-based networks, this might be a special dial code such as "#777" or a Network Access Identifier (NAI) such as "foo@thephone-company.com". This member can be NULL, to request that the network assign the default APN. NOTE: Not all networks support this NULL APN convention, so a connect failure due to an invalid APN is a possible outcome. The size of the string should not exceed 100 characters.
24	4	AccessStringSize	SIZE (0..200)	Size used for AccessString
28	4	UserNameOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string UserName that represents the username to authenticate. This member can be NULL.
32	4	UserNameSize	SIZE (0..510)	Size used for UserName
36	4	PasswordOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string Password that represents the username's password. This member can be NULL.
40	4	PasswordSize	SIZE (0..510)	Size used for Password

44	4	Compression	MBIM_COMPRESSION	Specifies the compression to be used in the data connection for header and data. This member applies only to GSM-based devices. The Host sets this member to MBIMCompressionNone for CDMA-based devices. See Table 10-61: MBIM_COMPRESSION
48	4	AuthProtocol	MBIM_AUTH_PROTOCOL	Authentication type to use for the PDP activation. See Table 10-62: MBIM_AUTH_PROTOCOL
52		DataBuffer	DATABUFFER	AccessString UserName Password

10.5.13.4 SET

The following structure shall be used in the InformationBuffer

Table 10-71: MBIM_SET_PROVISIONED_CONTEXT

Offset	Size	Field	Type	Description
0	4	ContextId	UINT32	A unique ID for this context. For set MBIM_CID_PROVISIONED_CONTEXT requests, the Host can set the value to 0xffffffff. If this value is used, the device should decide the index for storing the context information. 0xffffffff shall never be returned in response to query MBIM_CID_PROVISIONED_CONTEXT requests.

4	16	ContextType	MBIM_CONTEXT_TYPES	Specifies the type of context being represented, for example, Internet connectivity, VPN (a connection to a corporate network), or Voice-over-IP (VOIP), etc. Devices should specify MBIMContextTypeNone for empty or unprovisioned contexts. See Table 10-66: MBIM_CONTEXT_TYPES
20	4	AccessStringOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string AccessString to access the network. For GSM-based networks, this would be an Access Point Name (APN) string such as "data.thephone-company.com". For CDMA-based networks, this might be a special dial code such as "#777" or a Network Access Identifier (NAI) such as "foo@thephone-company.com". This member can be NULL, to request that the network assign the default APN. NOTE: Not all networks support this NULL APN convention, so a connect failure due to an invalid APN is a possible outcome. The size of the string should not exceed 100 characters.
24	4	AccessStringSize	SIZE (0..200)	Size used for AccessString
28	4	UserNameOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string UserName that represents the username to authenticate. This member can be NULL.
32	4	UserNameSize	SIZE (0..510)	Size used for UserName

36	4	PasswordOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string Password that represents the username's password. This member can be NULL.
40	4	PasswordSize	SIZE (0..510)	Size used for Password
44	4	Compression	MBIM_COMPRESSION	Specifies the compression to be used in the data connection for header and data. This member applies only to GSM-based devices. The Host sets this member to MBIMCompressionNone for CDMA-based devices. See Table 10-61: MBIM_COMPRESSION
48	4	AuthProtocol	MBIM_AUTH_PROTOCOL	Authentication type to use for the PDP activation. See Table 10-62: MBIM_AUTH_PROTOCOL
52	4	ProviderIdOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string ProviderId that represents the network provider identification for which the provisioned context should be stored in set MBIM_CID_PROVISIONED_CONTEXT requests. Devices should return the added provisioned context in response to subsequent query operations when a Subscriber Identity Module (SIM card) with this home provider ID is in the device.
56	4	ProviderIdSize	SIZE (0..12)	Size used for ProviderId
60		DataBuffer	DATABUFFER	AccessString UserName Password ProviderId

10.5.13.5 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.13.6 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-72: MBIM_PROVISIONED_CONTEXTS_INFO

Offset	Size	Field	Type	Description
0	4	ElementCount (EC)	UINT32	Count of MBIM_CONTEXT structures that follow in the DataBuffer
4	8*EC	ProvisionedContextRefList	OL_PAIR_LIST	The first element of the pair is a 4 byte Offset in bytes, calculated from the beginning (offset 0) of this MBIM_PROVISIONED_CONTEXTS_INFO structure, to an MBIM_CONTEXT structure (see Table 10-70: MBIM_CONTEXT). The second element of the pair is a 4 byte size of the corresponding MBIM_CONTEXT structure.
4+8*EC		DataBuffer	DATABUFFER	Array of MBIM_CONTEXT structures

10.5.13.7 NOTIFICATION

See Table 10-72: MBIM_PROVISIONED_CONTEXTS_INFO

10.5.13.8 STATUS CODES

For Query and Set Operations

Status code	Description
MBIM_STATUS_INVALID_PARAMETERS	The operation failed because of invalid parameters.

MBIM_STATUS_READ_FAILURE	The operation failed because the device was unable to get provisioned contexts.

For Set Operations Only

Status code	Description
MBIM_STATUS_WRITE_FAILURE	The operation failed because the update request was unsuccessful.

10.5.14 MBIM_CID_SERVICE_ACTIVATION

10.5.14.1 DESCRIPTION

This command instructs devices to initiate service activation in order to gain access to the provider's network.

Set only:

InformationBuffer of MBIM_COMMAND_MSG contains an MBIM_SET_SERVICE_ACTIVATION.

The InformationBuffer of MBIM_COMMAND_DONE contains an MBIM_SERVICE_ACTIVATION_INFO.

10.5.14.1.1 SERVICE ACTIVATION DETECTION

Hosts can determine whether they must perform service activation in a couple of ways:

- For CDMA-based devices, in North America or other places where U-RIM is not used, there should be a flag on the device to indicate activation status. CDMA Devices should be able to detect the activation status during initialization without contacting the provider network. Devices should perform service activation automatically when the device first connects over-the-air to the home network. After activation has been completed, devices should clear the flag so that they will not need to perform service activation again.
- The function can inform the host about service activation progress by sending MBIM_CID_SUBSCRIBER_READY_STATUS notifications during MB device initialization. Alternatively, to determine service activation status, the host may send an MBIM_CID_SUBSCRIBER_READY_STATUS query request to the function. In both cases, the initial ready-state should be MBIMSubscriberReadyStateNotActivated. After service has been activated, devices shall resume the initialization process and notify the host as the device ready-state changes.

- For GSM-based devices, there is no general method to detect whether a device must have its service activated. Devices can implement their own proprietary method, specific to its carrier, to perform service detection and activation.

10.5.14.1.2 SERVICE ACTIVATION

Service activation refers to the process of activating the MB service subscription so that the device can gain access to the provider's network. Service activation can be automatic, or manual, or a combination of both. Hosts need to perform service activation only once for each new subscription. For a multi-mode capable function, service activation is performed only once for the subscription across all modes.

10.5.14.2 PARAMETERS

Table 10-73: Parameters

	Set	Query	Notification
Command	MBIM_SET_SERVICE_ACTIVATION	NA	NA
Response	MBIM_SERVICE_ACTIVATION_INFO	NA	NA

10.5.14.3 SET

Since the service activation data varies across carriers, VendorSpecificBuffer is a placeholder for sending the carrier-specific data to activate the service.

Table 10-74: MBIM_SET_SERVICE_ACTIVATION

Offset	Size	Field	Type	Description
0	X	DataBuffer	DATABUFFER	VendorSpecificBuffer

10.5.14.4 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-75: MBIM_SERVICE_ACTIVATION_INFO

Offset	Size	Field	Type	Description
0	4	NwError	UINT32	A network-specific error. Table 10-44: 3GPP TS 24.008 Cause codes for NwError shows the connection failure values as documented in the Cause

				values in the appendixes of the 3GPP TS 24.008 Specification [3GPP24008].
4	X	DataBuffer	DATABUFFER	VendorSpecificBuffer

10.5.14.5 STATUS CODES

Status code	Description
MBIM_STATUS_INVALID_PARAMETERS	Service activation failed because of invalid parameters.
MBIM_STATUS_FAILURE	Service activation failed. Device can return this value if service has already been activated.
MBIM_STATUS_PROVIDER_NOT_VISIBLE	Service activation failed because the service provider is not currently visible.

10.5.15 MBIM_CID_SMS_CONFIGURATION

10.5.15.1 DESCRIPTION

This command indicates the state of the SMS storage and also sets or returns a MB device's SMS text message configuration.

Query and Set:

GSM-based functions should support both **Query** and **Set** operations. CDMA-based functions should support only **Query** operations. CDMA-based functions should return a valid value in the MaxMessages member of the MBIM_SMS_CONFIGURATION_INFO structure for **Query** requests and can ignore the other members.

MBIM_SMS_CONFIGURATION_INFO is returned from both **Query** and **Set** complete messages in the InformationBuffer. If the SmsStorageState is MBIMSmsStorageNotInitialized, the fields other than SmsStorageState are not valid and may be set to arbitrary values by the function.

For **Query**, the InformationBuffer is empty. For **Set**, it contains an MBIM_SET_SMS_CONFIGURATION.

Unsolicited Event:

The **Unsolicited Event** is sent at any time the state of the SMS storage changes. The Event InformationBuffer contains an MBIM_SMS_CONFIGURATION_INFO structure.

The function must return MBIM_STATUS_NOT_INITIALIZED for **Set** requests if the device is initialized but the SMS subsystem is not yet initialized.

The function must return MBIM_STATUS_NOT_SUPPORTED if it does not support configuring SMS text messages.

10.5.15.2 PARAMETERS

Table 10-76: Parameters

	Set	Query	Notification
Command	MBIM_SET_SMS_CONFIGURATION	Empty	NA
Response	MBIM_SMS_CONFIGURATION_INFO	MBIM_SMS_CONFIGURATION_INFO	MBIM_SMS_CONFIGURATION_INFO

10.5.15.3 DATA STRUCTURES

Table 10-77: MBIM_SMS_STORAGE_STATE

Types	Value
MBIMSmsStorageNotInitialized	0
MBIMSmsStorageInitialized	1

A multi-mode single-carrier function must use only GSM PDU format (MBIMSmsFormatPdu) even if the underlying cellular technology switches from GSM to CDMA, and must provide the appropriate transcoding in the send and receive paths. This applies to both PDU and CDMA text formats.

Table 10-78: MBIM_SMS_FORMAT

Types	Value
MBIMSmsFormatPdu	0
MBIMSmsFormatCdma	1

10.5.15.4 SET

The following structure shall be used in the InformationBuffer

Table 10-79: MBIM_SET_SMS_CONFIGURATION

Offset	Size	Field	Type	Description
0	4	Format	MBIM_SMS_FORMAT	See Table 10-78: MBIM_SMS_FORMAT
4	4	ScAddressOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string ScAddress with a maximum length of 20 digits that represents the Service Center (SC) address. This member is used by all text messages for sending and receiving. For PDU-style SMS messages, this information is used if it is not available in PDU data. The number can be in any of the following formats: • "+ <International Country Code> <SMS Service Center Number>" • "<SMS Service Center Number>" For set requests, the Host can set this member to zero to indicate that the device does not need to update ScAddress. In this case, the ScAddressSize must also be set to zero.
8	4	ScAddressSize	SIZE (0..40)	Size of ScAddress. If ScAddressOffset is set to zero ScAddressSize must also be set to zero.
12		DataBuffer	DATABUFFER	ScAddress

10.5.15.5 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.15.6 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-80: MBIM_SMS_CONFIGURATION_INFO

Offset	Size	Field	Type	Description
0	4	SmsStorageState	MBIM_SMS_STORAGE_STATE	The current state of the SMS storage according to Table 10-77: MBIM_SMS_STORAGE_STATE
4	4	Format	MBIM_SMS_FORMAT	See Table 10-78: MBIM_SMS_FORMAT
8	4	MaxMessages	UINT32	The maximum number of messages that can be stored on the device
12	4	CdmaShortMessageSize	UINT32	The maximum SMS character length that is supported by the device, if the device is CDMA-based. This member does not apply to GSM-based devices. GSM-based devices shall specify 0. CDMA-based devices that support SMS should specify their carrier-specific maximum SMS character length to be greater than 0. The value could possibly be larger than 160 if Enhanced Messaging Service (EMS) is supported. Otherwise, it is expected to be less than 160. CDMA-based devices that do not support SMS should set this member to 0.

16	4	ScAddressOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string ScAddress with a maximum length of 20 digits that represents the Service Center (SC) address. This member is used by all text messages for sending and receiving. For PDU-style SMS messages, this information is used if it is not available in PDU data. The number can be in any of the following formats: "+ <International Country Code> <SMS Service Center Number>" "<SMS Service Center Number>"
20	4	ScAddressSize	SIZE (0..40)	Size of ScAddress
24		DataBuffer	DATABUFFER	ScAddress

10.5.15.7 NOTIFICATION

See Table 10-80: MBIM_SMS_CONFIGURATION_INFO

10.5.15.8 STATUS CODES

Status code	Description
MBIM_STATUS_SMS_FORMAT_NOT_SUPPORTED	The operation failed because the SMS format specified in MBIM_SMS_CONFIGURATION_INFO is not supported.
MBIM_STATUS_SMS_UNKNOWN_SMSC_ADDRESS	The operation failed because the SMSC address is unknown or invalid.

10.5.16 MBIM_CID_SMS_READ

10.5.16.1 DESCRIPTION

This command reads SMS text messages stored in the MB device, or Subscriber Identity Module (SIM card), or any other auxiliary non-volatile memory or memories.

MBIM_CID_SMS_READ supports reading both PDU-mode and CDMA-mode SMS text messages, depending on the capabilities of the function.

Functions may receive requests to read SMS text messages based on an index, or to read all SMS text messages. Read requests may consist of any one of the basic filters such as new (unread) messages, old (read) messages, draft messages, or sent messages.

Functions that implement SMS text message functionality must support the reading of all messages using the basic filter for MBIMSmsFlagAll and all new messages using the basic filter for MBIMSmsFlagNew. All other filter types are optional to support.

Functions must logically project a single SMS text message store across all available physically different SMS text message stores.

Query:

The InformationBuffer of the MBIM_COMMAND_MSG contains an MBIM_SMS_READ_REQ structure.

The InformationBuffer of the MBIM_COMMAND_DONE message contains an MBIM_SMS_READ_INFO structure.

Unsolicited Event:

The Event InformationBuffer contains an MBIM_SMS_READ_INFO structure.

The function sends this event in the case of the arrival of a new class-0 (flash/alert) message from the network provider as an event notification.

10.5.16.2 PARAMETERS

Table 10-81: Parameters

	Set	Query	Notification
Command	NA	MBIM_SMS_READ_REQ	Empty
Response	NA	MBIM_SMS_READ_INFO	MBIM_SMS_READ_INFO

10.5.16.3 DATA STRUCTURES

Table 10-82: MBIM_SMS_FLAG

Types	Value
MBIMSmsFlagAll	0
MBIMSmsFlagIndex	1
MBIMSmsFlagNew	2
MBIMSmsFlagOld	3
MBIMSmsFlagSent	4
MBIMSmsFlagDraft	5

Table 10-83: MBIM_SMS_CDMA_LANG

Types	Value
MBIMSmsCdmaLangUnknown	0
MBIMSmsCdmaLangEnglish	1
MBIMSmsCdmaLangFrench	2
MBIMSmsCdmaLangSpanish	3
MBIMSmsCdmaLangJapanese	4
MBIMSmsCdmaLangKorean	5
MBIMSmsCdmaLangChinese	6
MBIMSmsCdmaLangHebrew	7

Table 10-84: MBIM_SMS_CDMA_ENCODING

Types	Value
MBIMSmsCdmaEncodingOctet	0
MBIMSmsCdmaEncodingEpm	1
MBIMSmsCdmaEncoding7BitAscii	2

MBIMSmsCdmaEncodingIa5	3
MBIMSmsCdmaEncodingUnicode	4
MBIMSmsCdmaEncodingShiftJis	5
MBIMSmsCdmaEncodingKorean	6
MBIMSmsCdmaEncodingLatinHebrew	7
MBIMSmsCdmaEncodingLatin	8
MBIMSmsCdmaEncodingGsm7Bit	9

Table 10-85: MBIM_SMS_MESSAGE_STATUS

Types	Value
MBIMSmsStatusNew	0
MBIMSmsStatusOld	1
MBIMSmsStatusDraft	2
MBIMSmsStatusSent	3

Table 10-86: MBIM_SMS_PDU_RECORD

Offset	Size	Field	Type	Description
0	4	MessageIndex	UINT32	An index into the virtual message store that is maintained by the device. This index is 1-based and the maximum index is MaxMessages returned in MBIM_SMS_CONFIGURATION_INFO. Be aware that the specification does not differentiate between physically available data stores. If the message is a Class 0 (flash/alert) message, this must be set to MBIM_MESSAGE_INDEX_NONE.

4	4	MessageStatus	MBIM_SMS_MESSAGE_STATUS	See Table 10-85: MBIM_SMS_MESSAGE_STATUS
8	4	PduDataOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to the PDU representation of the SMS message, PduData. For GSM-based devices, use the PDU representation of the SMS message per 3GPP TS 27.005 [3GPP27005] and 3GPP TS 23.040 [3GPP23040]. For CDMA-based devices that support reading SMS messages in binary this member contains the SMS message as a byte array, as defined in section 3.4.2.1 SMS Point-to-Point Message in 3GPP2 specification C.S0015-A "Short Message Service (SMS) for Wideband Spread Spectrum Systems". SMS will only support Wireless Messaging Teleservice (WMT) format It is not coded in hexadecimal string format.
12	4	PduDataSize	SIZE (0..255)	The size, in bytes, of the PduData. For CDMA PDU the maximum size of PduData is 255. For GSM PDU the maximum size of PduData is 183.
16		DataBuffer	DATABUFFER	PduData

Table 10-87: MBIM_SMS_CDMA_RECORD

Offset	Size	Field	Type	Description
0	4	MessageIndex	UINT32	An index into the virtual message

				store that is maintained by the device. This index is 1-based and the maximum index is MaxMessages returned in MBIM_SMS_CONFIGURATION_INFO. Be aware that the specification does not differentiate between physically available data stores. If the message is a Class 0 (flash/alert) message, this must be set to 0.
4	4	MessageStatus	MBIM_SMS_MESSAGE_STATUS	See Table 10-85: MBIM_SMS_MESSAGE_STATUS
8	4	AddressOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string Address with a maximum length of 20 digits that represents a mobile number. The number can be in any of the following formats: • "+ <International Country Code> <Mobile Number>" • "<Mobile Number>" If MsgStatus is MBIMSmsStatusDraft or MBIMSmsStatusSent, devices should specify the receiver's mobile number in the previous members. Otherwise, if MsgStatus is MBIMSmsStatusNew or MBIMSmsStatusOld, devices should specify the sender's mobile number.
12	4	AddressSize	SIZE (0..40)	Size used for Address
16	4	TimeStampOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to an ASCII string TimeStamp with a maximum length of 21 that represents the Service Center (SC)

				timestamp, in the following format: " YY/ MM/ DD, HH: mm: SS± ZZ" where: • YY represents the last two digits of the year. For example, 07 corresponds to 2007. Valid range is between 00 and 99. • MM represents the month index in double digits. For example, 01 for January and 12 corresponds to December. Valid range is between 01 and 12. • DD represents the day of the month in double digits. For example, 01 corresponds to the 1st day of the month, and 31 corresponds to the 31st day. Valid range is between 01 and 31. • HH represents the hours in 24-hour format. For example, 01 corresponds to 1 AM and 13 corresponds to 1PM. Valid range is between 00 and 23. • mm represents the minutes in double digits. For example, 01 corresponds to 1 minute and 30 corresponds to 30 minutes. Valid range is between 00 and 59. • SS represents the seconds in double digits. For example, 01 corresponds to 1 second and 30 corresponds to 30 seconds. Valid range is between 00 and 59. • ZZ represents the time zone with reference to Greenwich Mean Time (GMT). For example, 01 corresponds to 1 hour and 12 corresponds to 12 hours. Valid range is between 00 and 13 (-12 to +13 when combined with the ± symbol).

				For example, to represent October 2nd, 1996, 20:01:54 GMT+2 hours use the following string timestamp "96/10/02,20:01:54+02"
20	4	TimeStampSize	SIZE (0..21)	Size used for TimeStamp
24	4	EncodingId	MBIM_SMS_CDMA_ENCODING	See Table 10-84: MBIM_SMS_CDMA_ENCODING
28	4	LanguageId	MBIM_SMS_CDMA_LANG	See Table 10-83: MBIM_SMS_CDMA_LANG
32	4	EncodedMessage Offset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to the encoded content of the record that represents the SMS text message (EncodedMessage).
36	4	SizeInBytes	SIZE (0..160)	The size, in bytes, of EncodedMessage. Devices must specify a value for this member for all encoding types.
40	4	SizeInCharacters	UINT32	Size of EncodedMessage in number of characters represented by the encoded data. Devices should specify 0 for this member when EncodingId is set to MBIMSmsCdmaEncodingShiftJis or MBIMSmsCdmaEncodingKorean.
44		DataBuffer	DATABUFFER	Address TimeStamp EncodedMessage

10.5.16.4 QUERY

Table 10-88: MBIM_SMS_READ_REQ

Offset	Size	Field	Type	Description
0	4	SmsFormat	MBIM_SMS_FORMAT	See Table 10-78: MBIM_SMS_FORMAT
4	4	Flag	MBIM_SMS_FLAG	See Table 10-82: MBIM_SMS_FLAG
8	4	MessageIndex	UINT32	A value between 1 and MaxMessages that is an index into the device's message store. This value in this member is valid only if Flag is set to MBIMSmsFlagIndex. The Host sets this member if Flag is set to MBIMSmsFlagIndex. For all flags other than MBIMSmsFlagIndex, the Host sets this index to 0

10.5.16.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-89: MBIM_SMS_READ_INFO

Offset	Size	Field	Type	Description
0	4	Format	MBIM_SMS_FORMAT	See Table 10-78: MBIM_SMS_FORMAT
4	4	ElementCount (EC)	UINT32	Number of elements in array

8	8*EC	SmsRefList	OL_PAIR_LIST	First element of the pair is a 4 byte Offset in bytes, calculated from the beginning (offset 0) of this MBIM_SMS_READ_INFO structure, to a structure defined either per Table 10-86: MBIM_SMS_PDU_RECORD or per Table 10-87: MBIM_SMS_CDMA_RECORD. Second element of the pair is a 4 byte size of the record element.
8+8*EC		DataBuffer	DATABUFFER	Array of MBIM_SMS_PDU_RECORD or MBIM_SMS_CDMA_RECORD

10.5.16.6 NOTIFICATION

See Table 10-89: MBIM_SMS_READ_INFO

10.5.16.7 STATUS CODES

Status code	Description
MBIM_STATUS_SMS_FORMAT_NOT_SUPPORTED	The operation failed because the SMS format specified in MBIM_SMS_CONFIGURATION_INFO is not supported.
MBIM_STATUS_MEMORY_FAILURE	The operation failed because the because of device or SIM memory failure.
MBIM_STATUS_INVALID_MEMORY_INDEX	The operation failed because of an invalid memory index.
MBIM_STATUS_FILTER_NOT_SUPPORTED	The operation failed because the filter type is not supported.

10.5.17 MBIM_CID_SMS_SEND

10.5.17.1 DESCRIPTION

This command sends SMS text messages to another device capable of receiving SMS.

MBIM_CID_SMS_SEND supports sending both PDU-mode and CDMA-mode SMS text messages, depending on the capabilities of the function.

GSM-based functions are expected to support only PDU-mode SMS text messages. CDMA-based functions can support PDU-mode or CDMA-mode SMS text messages. Functions must be able to complete **Set** requests regardless of SMS text message mode.

Set only:

The InformationBuffer of the MBIM_COMMAND_MSG contains an MBIM_SET_SMS_SEND structure.

The InformationBuffer of the MBIM_COMMAND_DONE message contains an MBIM_SMS_SEND_INFO structure.

10.5.17.2 PARAMETERS

Table 10-90: Parameters

	Set	Query	Notification
Command	MBIM_SET_SMS_SEND	NA	NA
Response	MBIM_SMS_SEND_INFO	NA	NA

10.5.17.3 DATA STRUCTURES

Table 10-91: MBIM_SMS_SEND_PDU

Offset	Size	Field	Type	Description

0	4	PduDataOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to the PDU representation of the SMS message, PduData. For GSM-based devices, use the PDU representation of the SMS message per 3GPP TS 27.005 [3GPP27005] and 3GPP TS 23.040 [3GPP23040]. For CDMA-based devices that support reading SMS messages in binary this member contains the SMS message as a byte array, as defined in section 3.4.2.1 SMS Point-to-Point Message in 3GPP2 specification C.S0015-A "Short Message Service (SMS) for Wideband Spread Spectrum Systems". SMS will only support Wireless Messaging Teleservice (WMT) format. It is not coded in hexadecimal string format. In case a SC address is not specified in the PDU, the SC address shall be set to 00
4	4	PduDataSize	SIZE (0..255)	The size, in bytes, of the PduData. For CDMA PDU the maximum size of PduData is 255. For GSM PDU the maximum size of PduData is 183.
8		DataBuffer	DATABUFFER	PduData

Table 10-92: MBIM_SMS_SEND_CDMA

Offset	Size	Field	Type	Description
0	4	EncodingId	MBIM_SMS_CDMA_ENCODING	Table 10-84: MBIM_SMS_CDMA_ENCODING
4	4	LanguageId	MBIM_SMS_CDMA_LANG	Table 10-83: MBIM_SMS_CDMA_LANG
8	4	AddressOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string Address with a maximum length of 20 digits that represents a mobile number. The number can be in any of the following formats: "+ <International Country Code> <Mobile Number>" "<Mobile Number>0"
12	4	AddressSize	SIZE (0..40)	Size used for Address
16	4	EncodedMessageOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to the encoded message (EncodedMessage).
20	4	SizeInBytes	SIZE (0..160)	The size, in bytes, of EncodedMessage
24	4	SizeInCharacters	UINT32	Size of EncodedMessage in number of characters represented by the encoded data. Hosts should specify 0 for this member when EncodingId is set to MBIMmsCdmaEncodingShiftJis or MBIMSmsCdmaEncodingKorea

				n.
28		DataBuffer	DATABUFFER	Address EncodedMessage

10.5.17.4 SET

The following structure shall be used in the InformationBuffer

Table 10-93: MBIM_SET_SMS_SEND

Offset	Size	Field	Type	Description
0	4	SmsFormat	MBIM_SMS_FORMAT	See Table 10-78: MBIM_SMS_FORMAT
4		DataBuffer	DATABUFFER	Message: Depending on the value of SmsFormat this field is either MBIM_SMS_SEND_PDU (Table 10-91: MBIM_SMS_SEND_PDU) or MBIM_SMS_SEND_CDMA (Table 10-92: MBIM_SMS_SEND_CDMA)

10.5.17.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-94: MBIM_SMS_SEND_INFO

Offset	Size	Field	Type	Description

0	4	MessageReference	UINT32	A reference number that is returned upon successful delivery of the SMS text message. This number corresponds to the following entities: 3GPP/GSM: TP-Message-Reference maintained by the function per GSM TS 51.011 and 3GPP TS 31.102). In 3GPP/GSM, the MessageReference number may possess values in the range 0 to 255 (0ffh). 3GPP2/CDMA: Message Identifier per 3GPP2 SMS specification C.S0015-A_v1.0_111403. In 3GPP2/CDMA, the MessageReference number may possess values in the range 0 to 65535 (0ffffh).

10.5.17.6 STATUS CODES

Status code	Description
MBIM_STATUS_SMS_NETWORK_TIMEOUT	The SMS operation failed because of a network timeout.
MBIM_STATUS_SMS_LANG_NOT_SUPPORTED	The SMS operation failed because the SMS language is not supported.This applies to CDMA based devices only.
MBIM_STATUS_SMS_ENCODING_NOT_SUPPORTED	The SMS operation failed because the SMS encoding is not supported. This applies to CDMA based device only.
MBIM_STATUS_SMS_FORMAT_NOT_SUPPORTED	The operation failed because the SMS format specified in MBIM_SMS_CONFIGURATION_INFO is not supported.
MBIM_STATUS_MEMORY_FAILURE	The operation failed because the because of device or SIM memory failure.

MBIM_STATUS_SMS_UNKNOWN_SMSC_ADDRESS	The SMS operation failed because the service center address is either invalid or unknown.

10.5.18 MBIM_CID_SMS_DELETE

10.5.18.1 DESCRIPTION

This command deletes SMS text messages stored in the MB device, or Subscriber Identity Module (SIM card), or any other auxiliary non-volatile memory or memories.

Functions may receive requests to delete SMS text messages based on an index, or to delete all SMS text messages. Delete requests may consist of any one of the basic filters such as new (unread) messages, old (read) messages, draft messages, or sent messages.

Set only:

InformationBuffer for MBIM_COMMAND_MSG contains MBIM_SET_SMS_DELETE.

InformationBuffer for MBIM_COMMAND_DONE is not used.

10.5.18.2 PARAMETERS

Table 10-95: Parameters

	Set	Query	Notification
Command	MBIM_SET_SMS_DELETE	NA	NA
Response	Empty	NA	NA

10.5.18.3 DATA STRUCTURES

10.5.18.4 SET

The following structure shall be used in the InformationBuffer

Table 10-96: MBIM_SET_SMS_DELETE

Offset	Size	Field	Type	Description
0	4	Flags	MBIM_SMS_FLAG	See Table 10-82: MBIM_SMS_FLAG

4	4	MessageIndex	UINT32	A value between 1 and MaxMessages that is an index into the device's message store. This value in this member is valid only if Flag is set to MBIMSmsFlagIndex. The Host sets this member if Flag is set to MBIMSmsFlagIndex. For all flags other than MBIMSmsFlagIndex, the Host sets this index to 0.

10.5.18.5 RESPONSE

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.18.6 STATUS CODES

Status code	Description
MBIM_STATUS_MEMORY_FAILURE	The operation failed because the because of device or SIM memory failure.
MBIM_STATUS_INVALID_MEMORY_INDEX	The operation failed because of an invalid memory index.
MBIM_STATUS_FILTER_NOT_SUPPORTED	The operation failed because the filter type is not supported.

10.5.19 MBIM_CID_SMS_MESSAGE_STORE_STATUS

10.5.19.1 DESCRIPTION

This command reports the status of the MB device's message store.

Query:

InformationBuffer in MBIM_COMMAND_MSG is empty. InformationBuffer in MBIM_COMMAND_DONE contains MBIM_SMS_STATUS_INFO.

Unsolicited Event:

The Event InformationBuffer contains an MBIM_SMS_STATUS_INFO structure.

The function sends this event to inform the Host about the arrival of all non-class-0 (flash/alert) messages.

The MBIM_SMS_FLAG_NEW_MESSAGE flag is set as soon as there are new (unread) messages in the message store, even if the messages arrived before the MBIM session between host and function was opened.

Notifications of MBIM_CID_SMS_MESSAGE_STORE_STATUS are sent each time a new non-Class 0 SMS arrives even if there were already new (unread) messages. Notifications are not sent when all new messages have been read and the MBIM_SMS_FLAG_NEW_MESSAGE flag is unset.

Notification that the message store is full is sent as soon as this happens. No notification is sent when the store stops being full.

10.5.19.2 PARAMETERS

Table 10-97: Parameters

	Set	Query	Notification
Command	NA	Empty	NA
Response	NA	MBIM_SMS_STATUS_INFO	MBIM_SMS_STATUS_INFO

10.5.19.3 DATA STRUCTURES

This is a bitmap that represents possible status for the SMS message store.

Table 10-98: MBIM_SMS_STATUS_FLAGS

Types	Mask	Description
MBIM_SMS_FLAG_NONE	0	No status to report
MBIM_SMS_FLAG_MESSAGE_STORE_FULL	1	The message store is full.
MBIM_SMS_FLAG_NEW_MESSAGE	2	A new, non-Class 0 (flash/alert) message has arrived

10.5.19.4 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.19.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-99: MBIM_SMS_STATUS_INFO

Offset	Size	Field	Type	Description
0	4	Flag	MBIM_SMS_STATUS_FLAGS	See Table 10-98: MBIM_SMS_STATUS_FLAGS
4	4	MessageIndex	UINT32	This is the index of the newly arrived message or the recently arrived message in case of a query response.

10.5.19.6 NOTIFICATION

See Table 10-99:

10.5.19.7 STATUS CODES

Status code	Description
MBIM_STATUS_MEMORY_FAILURE	The operation failed because the because of device or SIM memory failure.

10.5.20 MBIM_CID_IP_CONFIGURATION

10.5.20.1 DESCRIPTION

Sometimes, some or all of the IP configuration information is obtained as part of the procedure in which the link-layer for data services is established. In such cases, the MBIM function acquires IP configuration information from the carrier network on behalf of the host. This CID is used to transfer such IP configuration information from an MBIM function to the host.

It is also possible for some or all of the IP configuration information to be acquired by the host's IP stack interacting directly at or above the IP layer with the carrier network. In this case, the MBIM function does not participate in the acquisition of IP configuration information other than enabling basic IP packet service.

Thus, the two alternatives are:

- The MBIM function obtains IP configuration information on behalf of the host from the carrier network (more common for IPv4).
- The host's IP stack obtains IP configuration information from the carrier network (more common for IPv6).

It is also possible to obtain some IP configuration information via the MBIM function, and some other via the host's IP stack. In particular, some deployments of IPv6 may still rely on the MBIM function to obtain the DNS information, while relying on the host's IP stack to acquire the other elements of IP configuration information.

An MBIM function indicates which IP configuration information it has obtained for IPv4 and IPv6 by setting appropriately these 2 bitmap fields in Table 10-105: MBIM_IP_CONFIGURATION_INFO:

- IPv4ConfigurationAvailable
- IPv6ConfigurationAvailable

For each of the IP configuration elements (IP address element, gateway, MTU, etc), the corresponding bit indicates the following:

- Bit value is 0 ("information not available"): The function does not participate in obtaining the corresponding IP configuration element.
    - The function sets the corresponding field to 0 so the host can ignore it.
    - Typically, this implies that the host invokes mechanisms such as routing advertisements (in IPv6) and DHCP (for v4 or v6) in order to obtain the IP configuration element in question.
- Bit value is 1 ("information available"): The function has obtained the corresponding IP configuration element. This may mean that the host is to either set or delete the corresponding IP configuration information.
    - The host does not ignore this information.
    - If the element has a valid (non-zero) value, the host uses this value. All values of the corresponding element are set (e.g., if a new DNS server is obtained, the complete list is sent rather than just the newly acquired DNS server info).
    - If the value of the element is zero, the host deletes the corresponding information. There is no selective deletion. All values of the corresponding element are deleted (e.g., all DNS servers if the bit in question corresponds to DNS server information).
    - In the case of both MBIM_IPV4_ELEMENTand MBIM_IPV6_ELEMENT, the value of the element being zero means that both fields in those structures (OnLinkPrefixLength and IPv4 or IPv6 Address) are set to 0. Both fields in these structures are treated atomically, that is, they are either both being set (in which case both have valid non-zero values) or both are being deleted (in which case both have values of zero).

Even though this mechanism allows each IP configuration element to be configured either by the MBIM function or by the host, in the interest of simplicity this flexibility is constrained. For the purposes of this constraint, the IP configuration elements for either IPv4 or IPv6 are of two types:

- DNS server information
- Basic IP information (address, gateway or MTU)

The values for the elements in the Basic IP information group must be obtained the same way: either all by the MBIM function or all by the host's IP stack. In other words, the bits for Basic IP information in the IPv4ConfigurationAvailable or IPv6ConfigurationAvailable bitmaps in Table 10-105: MBIM_IP_CONFIGURATION_INFO (bits 0, 1 and 3) must all be set to the same value. The bit for the DNS server information (bit 2) may be set independently. Furthermore, for the duration of an IP data stream session, the bits in the IPv4ConfigurationAvailable and IPv6ConfigurationAvailable bitmaps must not change. In other words, the values for any given IP Configuration element, are obtained either the MBIM function or via the host's IP stack and this does not change during the duration of an IP data stream session.

Query:

For **Query**, the InformationBuffer contains an MBIM_IP_CONFIGURATION_INFO in which the only relevant field is the SessionId. The SessionId in a **Query** indicates which IP data stream's configuration information is to be returned by the function. InformationBuffer of MBIM_COMMAND_DONE contains an MBIM_IP_CONFIGURATION_INFO.

Unsolicited Event:

Unsolicited Event InformationBuffer contains MBIM_IP_CONFIGURATION_INFO. Whenever the MBIM function obtains updated IP configuration information, it must issue an **Unsolicited Event** to inform the host about the new value(s).

10.5.20.2 PARAMETERS

Table 10-100: Parameters

	Set	Query	Notification
Command	NA	MBIM_IP_CONFIGURATION_INFO	NA
Response	NA	MBIM_IP_CONFIGURATION_INFO	MBIM_IP_CONFIGURATION_INFO

10.5.20.3 DATA STRUCTURES

Table 10-101: MBIM_IPV4_ADDRESS

Offset	Size	Field	Type	Description
4	4	IPv4Address	UINT8[4]	Unicast IPv4 Address

Table 10-102: MBIM_IPV4_ELEMENT

Offset	Size	Field	Type	Description
0	4	OnLinkPrefixLength	UINT32	The length, in bits, of the prefix or network part of the IP address. For a unicast IPv4 address, any value greater than 32 is an illegal value.
4	4	IPv4Address	MBIM_IPV4_ADDRESS	Unicast IPv4 Address

Table 10-103: MBIM_IPV6_ADDRESS

Offset	Size	Field	Type	Description
4	16	IPv6Address	UINT8[16]	Unicast IPv6 Address

Table 10-104: MBIM_IPV6_ELEMENT

Offset	Size	Field	Type	Description
0	4	OnLinkPrefixLength	UINT32	The length, in bits, of the prefix or network part of the IP address. For a unicast IPv6 address, any value greater than 128 is an illegal value.
4	16	IPv6Address	MBIM_IPV6_ADDRESS	Unicast IPv6 Address

10.5.20.4 QUERY

See Table 10-105: MBIM_IP_CONFIGURATION_INFO and **Query** description.

10.5.20.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-105: MBIM_IP_CONFIGURATION_INFO

Offset	Size	Field	Type	Description

0	4	SessionId	UINT32	The device uses the SessionId specified by the host in the MBIM_CID_CONNECT command to uniquely identify the session for the IP data stream and its corresponding state.
4	4	IPv4ConfigurationAvailable	UINT32	Bit 0: IPv4 Address info available Bit 1: IPv4 gateway info available Bit 2: IPv4 DNS server info available Bit 3: IPv4 MTU info available
8	4	IPv6ConfigurationAvailable	UINT32	Bit 0: IPv6 Address info available Bit 1: IPv6 gateway info available Bit 2: IPv6 DNS server info available Bit 3: IPv6 MTU info available
12	4	IPv4AddressCount	UINT32	Count of IPv4 Elements supplied
16	4	IPv4AddressOffset	OFFSET	Offset to the beginning of the IPv4 Elements relative to the beginning of the structure. See Table 10-102: MBIM_IPV4_ELEMENT
20	4	IPv6AddressCount	UINT32	Count of IPv6 Elements supplied

24	4	IPv6AddressOffset	OFFSET	Offset to the beginning of the addresses relative to the beginning of the structure. See Table 10-104: MBIM_IPV6_ELEMENT
28	4	IPv4GatewayOffset	OFFSET	Offset to the beginning of the address relative to the beginning of the structure. See Table 10-101: MBIM_IPV4_ADDRESS
32	4	IPv6GatewayOffset	OFFSET	Offset to the beginning of the addresses relative to the beginning of the structure. See Table 10-103: MBIM_IPV6_ADDRESS
36	4	IPv4DnsServerCount	UINT32	Count of address supplied
40	4	IPv4DnsServerOffset	OFFSET	Offset to the beginning of the addresses relative to the beginning of the structure. . See Table 10-101: MBIM_IPV4_ADDRESS
44	4	IPv6DnsServerCount	UINT32	Count of address supplied
48	4	IPv6DnsServerOffset	OFFSET	Offset to the beginning of the addresses relative to the beginning of the structure. See Table 10-103: MBIM_IPV6_ADDRESS
52	4	IPv4Mtu	UINT32	MTU for IPv4
56	4	IPv6Mtu	UINT32	MTU for IPv6
60		DataBuffer	DATABUFFER	Actual address data

For actual IP address data the IPv4 address will be 4 bytes in size and the IPv6 addresses will be 16 bytes in size. The addresses will be in network byte order (Big Endian).

10.5.20.6 NOTIFICATION

See Table 10-105: MBIM_IP_CONFIGURATION_INFO.

10.5.20.7 STATUS CODES

Status code	Description
MBIM_STATUS_CONTEXT_NOT_ACTIVATED	The operation failed because no context is currently activated.

10.5.21 MBIM_CID_USSD

10.5.21.1 DESCRIPTION

This CID is used to control the Unstructured Supplementary Service Data (USSD) according to 3GPP TS 22.090 [3GPP33402]. The command targets GSM-based devices and does not have to be supported by CDMA-based devices. Multi-mode functions currently operating in CDMA mode must fail USSD operations.

The command supports **Set** requests and **Unsolicited Events**. **Query** requests are not supported.

Set:

The command requires that SIM access and network access is available. The **Set** command shall include MBIM_SET_USSD as InformationBuffer. MBIM_SET_USSD includes MBIM_USSD_ACTION which can be used to initiate a new USSD session or continue the existing session, or to cancel an ongoing session. In case MBIM_SET_USSD is used to initiate or continue a session, the field USSDPayload in MBIM_SET_USSD includes the USSD data. This data is not a string as defined in section 10.4. Instead, it is data formatted per 3GPP TS 22.090 [3GPP33402]. If a session is canceled, the field USSDPayloadLength is set to 0 and the content of the field USSDPayload is undefined.

When a USSD command is completed the InformationBuffer of the response includes MBIM_USSD_INFO.

Unsolicited Event:

MBIM_USSD_INFO can also be sent unsolicited when a USSD session is initiated by the network. MBIM_USSD_INFO includes a response code from the network and also the USSD string from the network if applicable. When a USSD command to cancel a session is completed, the response is of type MBIMUSSDTerminated.

A USSD session is active until it is cancelled by sending a USSD command with action MBIMUSSDCancel, or until a complete command with MBIM_USSD_RESPONSE code not equal to MBIMUSSDActionRequired is received. Only one USSD session can exist at a time.

10.5.21.2 PARAMETERS

Table 10-106: Parameters

	Set	Query	Notification
Command	MBIM_SET_USSD	NA	NA
Response	MBIM_USSD_INFO	NA	MBIM_USSD_INFO

10.5.21.3 DATA STRUCTURES

Table 10-107: MBIM_USSD_ACTION

Types	Value
MBIMUSSDInitiate	0
MBIMUSSDContinue	1
MBIMUSSDCancel	2

Table 10-108: MBIM_USSD_RESPONSE

Types	Value
MBIMUSSDNoActionRequired	0
MBIMUSSDActionRequired	1
MBIMUSSDTerminatedByNW	2
MBIMUSSDOtherLocalClient	3
MBIMUSSDOperationNotSupported	4
MBIMUSSDNetworkTimeOut	5

Table 10-109: MBIM_USSD_SESSION_STATE

Types	Value
MBIMUSSDNewSession	0
MBIMUSSDExistingSession	1

10.5.21.4 SET

The following structure shall be used in the InformationBuffer

Table 10-110: MBIM_SET_USSD

Offset	Size	Field	Type	Description
0	4	USSDAction	MBIM_USSD_ACTION	See Table 10-107: MBIM_USSD_ACTION
4	4	USSDDataCodingScheme	UINT32	Value of the Data Coding Scheme as defined in 3GPP 23.038 [3GPP23038].
8	4	USSDPayloadOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to the USSDPayload in the DataBuffer. Set to 0 if MBIM_USSD_ACTION is MBIMUSSCancel
12	4	USSDPayloadLength	SIZE (0..160)	Length of USSD payload. This value is between 1 and 160 bytes. Set to 0 if MBIM_USSD_ACTION is MBIMUSSCancel
16		DataBuffer	DATABUFFER	USSDPayload

10.5.21.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-111: MBIM_USSD_INFO

Offset	Size	Field	Type	Description
0	4	USSDResponse	MBIM_USSD_RESPONSE	See Table 10-108: MBIM_USSD_RESPONSE
4	4	USSDSessionState	MBIM_USSD_SESSION_STATE	See Table 10-109: MBIM_USSD_SESSION_STATE. Set to MBIMUSSDNewSession for the first message of a network initiated session; MBIMUSSDExistingSession otherwise.
8	4	USSDDataCodingScheme	UINT32	Value of the Data Coding Scheme as defined in 3GPP 23.038 [3GPP23038].
12	4	USSDPayloadOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to the USSDPayload in the DataBuffer. Only applicable if USSDResponse is MBIMUSSDNoActionRequired or MBIMUSSDActionRequired; set to 0 otherwise.
16	4	USSDPayloadLength	SIZE (0..160)	Length of USSD payload. This value is between 1 and 160 bytes. Only applicable if USSDResponse is MBIMUSSDNoActionRequired or MBIMUSSDActionRequired; set to 0 otherwise.

20		DataBuffer	DATABUFFER	USSDPayload

10.5.21.6 NOTIFICATION

See Table 10-111: MBIM_USSD_INFO

10.5.21.7 STATUS CODES

Status code	Description
MBIM_STATUS_NOT_REGISTERED	The USSD operation failed because the device is not registered with any network.
MBIM_STATUS_INVALID_PARAMETERS	The USSD operation failed because of invalid parameters.

10.5.22 MBIM_CID_PHONEBOOK_CONFIGURATION

10.5.22.1 DESCRIPTION

This CID is used to retrieve information of the device phonebook. The command supports both GSM- and CDMA-based devices.

Query:

The InformationBuffer for the **Query** command is empty. The completed command returns the MBIM_PHONEBOOK_CONFIGURATION_INFO structure. A multi-mode single carrier function must provide access to the SIM phonebook even when operating in CDMA mode.

Unsolicited Notification:

Unsolicited Notifications contains the MBIM_PHONEBOOK_CONFIGURATION_INFO and are sent at two occasions; whenever there is a change in the phonebook state and when the phonebook becomes full, meaning that TotalNbrOfEntries equals UsedEntries in MBIM_PHONEBOOK_CONFIGURATION_INFO. An unsolicited event shall not be sent by the function after each successful write to the phonebook.

The MaxNumberLength and MaxNameLength fields of the MBIM_PHONEBOOK_CONFIGURATION_INFO structure report the maximum number of bytes for the phone number and for the contact name in a phonebook entry respectively. How many bytes are needed for a phonebook field does, however, depend on which characters are used. See GSM 11.11 specification [3GPP1111] for details of how the characters may be coded.

10.5.22.2 PARAMETERS

Table 10-112: Parameters

	Set	Query	Notification
Command	NA	Empty	NA
Response	NA	MBIM_PHONEBOOK_CONFIGURATION_INFO	MBIM_PHONEBOOK_CONFIGURATION_INFO

10.5.22.3 DATA STRUCTURES

Table 10-113: MBIM_PHONEBOOK_STATE

Types	Value
MBIMPhonebookNotInitialized	0
MBIMPhonebookInitialized	1

10.5.22.4 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.22.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-114: MBIM_PHONEBOOK_CONFIGURATION_INFO

Offset	Size	Field	Type	Description
0	4	PhonebookState	MBIM_PHONEBOOK_STATE	The current state of the phonebook, specified according to Table 10-113: MBIM_PHONEBOOK_STATE
4	4	TotalNbrOfEntries	UINT32	Total number of entries in the phonebook
8	4	UsedEntries	UINT32	Number of phonebook entries that are currently used
12	4	MaxNumberLength	UINT32	Maximum number of bytes for the phone number of a

				phonebook entry
16	4	MaxNameLength	UINT32	Maximum number of bytes for the name of a phonebook entry

10.5.22.6 NOTIFICATION

See Table 10-114: MBIM_PHONEBOOK_CONFIGURATION_INFO

10.5.22.7 STATUS CODES

Status code	Description
MBIM_STATUS_NO_PHONEBOOK	There is no phonebook storage currently available (e.g., no phonebook available in SIM card)
MBIM_STATUS_PIN_REQUIRED	A PIN code restricts access to information about the phonebook

10.5.23 MBIM_CID_PHONEBOOK_READ

10.5.23.1 DESCRIPTION

This CID is used to read one or all entries from the device phonebook.

Query only:

Neither **Set** command nor **Unsolicited Event** is supported. The **Query** command includes MBIM_PHONEBOOK_READ_REQ in InformationBuffer, which specifies if all phonebook entries or only a specific phonebook entry shall be read.

The complete command includes the MBIM_PHONEBOOK_READ_INFO structure as InformationBuffer. The MBIM_PHONEBOOK_READ_INFO includes an array of MBIM_PHONEBOOK_ENTRY structures, one for each entry being read.

10.5.23.2 PARAMETERS

Table 10-115: Parameters

	Set	Query	Notification
Command	NA	MBIM_PHONEBOOK_READ_REQ	NA
Response	NA	MBIM_PHONEBOOK_READ_INFO	NA

10.5.23.3 DATA STRUCTURES

Table 10-116: MBIM_PHONEBOOK_FLAG

Types	Value
MBIMPhonebookFlagAll	0
MBIMPhonebookFlagIndex	1

Table 10-117: MBIM_PHONEBOOK_ENTRY

Offset	Size	Field	Type	Description
0	4	EntryIndex	UINT32	Index for the phonebook entry
4	4	NumberOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to the PhoneNumber.
8	4	NumberLength	SIZE (0..MaxNumberLength)	Length of PhoneNumber for the phonebook entry (in bytes). MaxNumberLength is obtained from Table 10-114: MBIM_PHONEBOOK_CONFIGURATION_INFO.
12	4	NameOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to the Name of the phonebook entry
16	4	NameLength	SIZE (0..MaxNameLength)	Length of the name for the phonebook entry (in bytes). MaxNameLength is obtained from Table 10-114: MBIM_PHONEBOOK_CONFIGURATION_INFO.
20		DataBuffer	DATABUFFER	PhoneNumber Name

10.5.23.4 QUERY

The following structure shall be used in the InformationBuffer

Table 10-118: MBIM_PHONEBOOK_READ_REQ

Offset	Size	Field	Type	Description
0	4	FilterFlag	MBIM_PHONEBOOK_FLAG	See Table 10-116: MBIM_PHONEBOOK_FLAG
4	4	FilterMessageIndex	UINT32	A value between 1 and TotalNbrOfEntries (see Table 10-114: MBIM_PHONEBOOK_CONFIGURATION_INFO) that is an index into the device's phonebook store. This value in this member is valid only if FilterFlag is set to MBIMPhonebookFlagIndex. For all flags other than MBIMPhonebookFlagIndex, the Host sets this index to 0.

10.5.23.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-119: MBIM_PHONEBOOK_READ_INFO

Offset	Size	Field	Type	Description
0	4	ElementCount (EC)	UINT32	Number of phonebook entries
8	8*EC	PhoneBookRefList	OL_PAIR_LIST	First element of the pair is a 4 byte Offset in bytes, calculated from the beginning (offset 0) of this MBIM_PHONEBOOK_READ_INFO structure, to a Table 10-117: MBIM_PHONEBOOK_ENTRY. The second element of the pair is a 4 byte size of the record element.

4+8*EC		DataBuffer	DATABUFFER	Array of phonebook entries, each specified as in Table 10-117: MBIM_PHONEBOOK_ENTRY.

10.5.23.6 STATUS CODES

Status code	Description
MBIM_STATUS_NOT_INITIALIZED	The device has not finished initializing the phonebook storage and is not able to read the phonebook.
MBIM_STATUS_NO_PHONEBOOK	There is no phonebook storage currently available (e.g. no phonebook available in SIM card)
MBIM_STATUS_PIN_REQUIRED	A PIN code restricts access to information about the phonebook
MBIM_STATUS_INVALID_MEMORY_INDEX	The index specified in the query command is either empty or not available in the phonebook
MBIM_STATUS_FILTER_NOT_SUPPORTED	The filter selected in the query command is not supported by the device
MBIM_STATUS_MEMORY_FAILURE	The command failed due to an error in the phonebook storage memory

10.5.24 MBIM_CID_PHONEBOOK_DELETE

10.5.24.1 DESCRIPTION

This CID is used to delete one or all entries in the device phonebook.

Set only:

Neither **Query** command nor **Unsolicited Event** is supported. The **Set** command includes MBIM_SET_PHONEBOOK_DELETE in InformationBuffer, which specifies if all phonebook entries or only a specific phonebook entry shall be deleted.

The complete command has no data in InformationBuffer only success or failure information will be included.

10.5.24.2 PARAMETERS

Table 10-120: Parameters

	Set	Query	Notification
Command	MBIM_SET_PHONEBOOK_DELETE	NA	NA
Response	Empty	NA	NA

10.5.24.3 SET

The following structure shall be used in the InformationBuffer

Table 10-121: MBIM_SET_PHONEBOOK_DELETE

Offset	Size	Field	Type	Description
0	4	FilterFlag	MBIM_PHONEBOOK_FLAG	SeeTable 10-116: MBIM_PHONEBOOK_FLAG.
4	4	FilterMessageIndex	UINT32	A value between 1 and TotalNbrOfEntries (see Table 10-114: MBIM_PHONEBOOK_CONFIGURATION_INFO) that is an index into the device's phonebook store. This value in this member is valid only if FilterFlag is set to MBIMPhonebookFlagIndex. For all flags other than MBIMPhonebookFlagIndex, the Host sets this index to 0.

10.5.24.4 RESPONSE

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.24.5 STATUS CODES

Status code	Description
MBIM_STATUS_NOT_INITIALIZED	The device has not finished initializing the phonebook storage and is not able to delete phonebook entries.
MBIM_STATUS_NO_PHONEBOOK	There is no phonebook storage currently available (e.g. no phonebook available in SIM card)
MBIM_STATUS_PIN_REQUIRED	A PIN code needs to be entered before

	phonebook entries can be deleted.
MBIM_STATUS_INVALID_MEMORY_INDEX	The index specified in the query command is either empty or not available in the phonebook
MBIM_STATUS_FILTER_NOT_SUPPORTED	The filter selected in the query command is not supported by the device
MBIM_STATUS_MEMORY_FAILURE	The command failed due to an error in the phonebook storage memory

10.5.25 MBIM_CID_PHONEBOOK_WRITE

10.5.25.1 DESCRIPTION

This CID is used to write an entry to the device phonebook.

Set only:

Neither **Query** command nor **Unsolicited Event** is supported. The **Set** command includes MBIM_SET_PHONEBOOK_WRITE in InformationBuffer, which can be used to specify if the phonebook entry shall be saved to a specific location or if the first unused location shall be used.

The CID can also be used to change an existing phonebook entry by specifying an already used location to save the new entry.

The complete command has no data in InformationBuffer; only success or failure information is provided.

10.5.25.2 PARAMETERS

Table 10-122: Parameters

	Set	Query	Notification
Command	MBIM_SET_PHONEBOOK_WRITE	NA	NA
Response	Empty	NA	NA

10.5.25.3 DATA STRUCTURES

Table 10-123: MBIM_PHONEBOOK_WRITE_FLAG

Types	Value
MBIMPhonebookFlagSaveUnused	0
MBIMPhonebookFlagSaveIndex	1

10.5.25.4 SET

The following structure shall be used in the InformationBuffer

Table 10-124: MBIM_SET_PHONEBOOK_WRITE

Offset	Size	Field	Type	Description
0	4	SaveFlag	MBIM_PHONEBOOK_WRITE_FLAG	See Table 10-123: MBIM_PHONEBOOK_WRITE_FLAG
4	4	SaveIndex	UINT32	A value between 1 and TotalNbrOfEntries (see Table 10-114: MBIM_PHONEBOOK_CONFIGURATION_INFO) that is an index into the device's phonebook store. This value in this member is valid only if SaveFlag is set to MBIMPhonebookFlagSaveIndex. The Host sets this member if SaveFlag is set to MBIMSmsFlagSaveIndex. For all flags other than MBIMPhonebookFlagIndex, the Host sets this index to0.
8	4	NumberOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to the PhoneNumber.
12	4	NumberLength	SIZE (0..MaxNumberLength)	Length of the PhoneNumber for the phonebook entry (in bytes). MaxNumberLength is obtained from Table 10-114: MBIM_PHONEBOOK_CONFIGURATION_INFO.
16	4	NameOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to the Name of the phonebook entry

20	4	NameLength	SIZE (0..MaxNameLength)	Length of the Name for the phonebook entry (in bytes). MaxNameLength is obtained from Table 10-114: MBIM_PHONEBOOK_CONFIGURATION_INFO
24		DataBuffer	DATABUFFER	PhoneNumber, Name

10.5.25.5 RESPONSE

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.25.6 STATUS CODES

Status code	Description
MBIM_STATUS_NOT_INITIALIZED	The device has not finished initializing the phonebook storage and is not able to write to the phonebook
MBIM_STATUS_NO_PHONEBOOK	There is no phonebook storage currently available (e.g. no phonebook available in SIM card)
MBIM_STATUS_PIN_REQUIRED	A PIN code restricts the command to be performed
MBIM_STATUS_INVALID_MEMORY_INDEX	The index specified in the set command is not available in the phonebook
MBIM_STATUS_FILTER_NOT_SUPPORTED	The SaveFlag selected in the set command is not supported by the device
MBIM_STATUS_MEMORY_FAILURE	The command failed due to an error in the phonebook storage memory
MBIM_STATUS_MEMORY_FULL	The set command could not be completed since there is no unused phonebook entry

MBIM_STATUS_PARAMETER_TOO_LONG	One or several of the parameters in the set command was/were too long to be stored in the phonebook

10.5.26 MBIM_CID_STK_PAC

10.5.26.1 DESCRIPTION

This CID is used to propagate proactive commands from the SIM card to the host. A proactive command is always triggered from the SIM card. A proactive command may be handled in four different ways:

- The proactive command is handled by the function without notifying the host
- The proactive command is handled by the function with a notification sent to the host, e.g. to generate a message to the end-user
- The proactive command is propagated to the host for handling
- The proactive command is not handled by either host or function

Query:

The **Query** command can be used to read the current setting of which commands are handled where.

The **Query** command includes no InformationBuffer. The complete response to the **Query** command includes the MBIM_STK_PAC_INFO structure in the InformationBuffer.

Set:

The **Set** command can be used by the host to say which commands it would like to handle or receive notifications for.

The **Set** command includes the MBIM_SET_STK_PAC structure in the InformationBuffer, which allows the host to specify which commands it would like to handle or receive notifications for. The function shall let the host handle/receive notifications for all commands that the host requests and that the function allows the host to handle/receive notifications for. Proactive commands that the host does not request to handle shall not be propagated to the host. The complete response to the **Set** command contains the MBIM_STK_PAC_INFO structure, where the function specifies which proactive commands will be handled where after the set command has been processed.

Unsolicited Event:

Once a proactive command is sent from the SIM card that the function is configured to propagate to the host, an MBIM_CID_STK_PAC notification will be sent by the function. The notification will include MBIM_STK_PAC structure in the InformationBuffer.

Note that the length of the proactive command is the second parameter included for the proactive command and it is therefore not specified as a separate field in MBIM_STK_PAC.

Multi-mode devices currently operating in CDMA mode must fail STK operations.

10.5.26.2 PARAMETERS

Table 10-125: Parameters

	Set	Query	Notification
Command	MBIM_SET_STK_PAC	Empty	NA
Response	MBIM_STK_PAC_INFO	MBIM_STK_PAC_INFO	MBIM_STK_PAC

10.5.26.3 DATA STRUCTURES

Table 10-126: MBIM_STK_PAC_PROFILE

Types	Value	Description
MBIMSTKNotHandledByFunctionCannotBeHandledByHost	0	The command is not handled in the function and the function will not be able to forward the command to the host.
MBIMSTKNotHandledByFunctionMayBeHandledByHost	1	The command is not handled by function but may be handled by the host if the host request so with the set command
MBIMSTKHandledByFunctionOnlyTransparentToHost	2	The command is handled by function without informing the host. The function will not allow the host to handle the command or receive notifications of the command even if the host requests so.
MBIMSTKHandledByFunctionNotificationToHostPossible	3	The command is handled by function without informing the host. However, if the host requests to receive the command

		it will receive notifications.
MBIMSTKHandledByFunctionNotificationsToHostEnabled	4	The command is handled by function, but the function will also send a notification to the host when the SIM sends the command.
MBIMSTKHandledByFunctionCanBeOverriddenByHost	5	The command is currently handled by the function, but if the host requests to handle the command it will get full control of the command and no handling will be made in the function.
MBIMSTKHandledByHostFunctionNotAbleToHandle	6	The command will be forwarded to the host. If the host decides to not receive the command any longer the function will not handle the command.
MBIMSTKHandledByHostFunctionAbleToHandle	7	The command will be forwarded to the host. If the host decides to not receive the command any longer the function will handle the command.

Table 10-127: MBIM_STK_PAC_TYPE

Types	Value	Description
MBIMSTKProactiveCommand	0	The proactive command is a proactive command that the host is requested to handle
MBIMSTKNotification	1	The proactive command is handled by the function, but the host is notified that the SIM has sent the proactive command and may decide to e.g. indicate this to the end user

10.5.26.4 SET

The following structure shall be used in the InformationBuffer

Table 10-128: MBIM_SET_STK_PAC

Offset	Size	Field	Type	Description
0	32	PacHostControl	UINT8[32]	32 bytes where each bit indicates if the host likes to handle/receive a notification for a specific proactive command or not. Bit A in the string tells if the host would like to handle proactive command A or not. If the bit is 1 the host would like to handle the command; if it is 0, it does not. The numbering of the commands will be according to the values associated with each command in [ETSITS102223900] for Type of Command coding in BER-TLV tags. E.g. bit 1 means the REFRESH command and bit 21 means DISPLAY TEXT. Bytes that correspond to values that are not associated with any Type of Command shall be set to 0. The bits are numbered so the least-significant bit is number 0.

10.5.26.5 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.26.6 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-129: MBIM_STK_PAC_INFO

Offset	Size	Field	Type	Description
0	256	PacSupport	MBIM_STK_PAC_PROFILE[256]	256 bytes where each byte identifies the current support for a specific proactive command. Byte A in the string tells the current support for proactive command A. The numbering of the commands will be according to the values associated with

				each command in [ETSITS102223900] for Type of Command coding in BER-TLV tags. I.e., byte 1 means the REFRESH command and byte 21 means DISPLAY TEXT. Each byte can have one of the values defined in Table 10-126: MBIM_STK_PAC_PROFILE. Bytes that correspond to values that are not associated with any Type of Command shall be set to 0.

10.5.26.7 NOTIFICATION

The following structure shall be used in the InformationBuffer

Table 10-130: MBIM_STK_PAC

Offset	Size	Field	Type	Description
0	4	PacType	MBIM_STK_PAC_TYPE	The type of the proactive command, values in Table 10-127: MBIM_STK_PAC_TYPE possible to use.
4		DataBuffer	DATABUFFER	Proactive command in BER-TLV format, starting with the Proactive UICC Command Tag as defined in [ETSITS102223900].

10.5.26.8 STATUS CODES

Status code	Description
MBIM_STATUS_NOT_INITIALIZED	The function has not finished initializing the UICC and the command cannot be completed
MBIM_STATUS_PIN_REQUIRED	A PIN code restricts the command to be performed

10.5.27 MBIM_CID_STK_TERMINAL_RESPONSE

10.5.27.1 DESCRIPTION

This CID is used to send a terminal response to a proactive command.

Set only:

Neither **Query** command nor **Unsolicited Event** is supported. The **Set** command includes the MBIM_SET_STK_TERMINAL_RESPONSE structure.

Compared to the proactive command, the terminal response as defined in [ETSITS102223900] does not specify the length of the response and the response length is therefore included as a specific field in MBIM_SET_STK_TERMINAL_RESPONSE.

The complete response includes the Table 10-133: MBIM_STK_TERMINAL_RESPONSE_INFO structure in the InformationBuffer, which contains the response APDU structure to the terminal response. This information can be used to see if the UICC response included any warning or error messages. Note that MBIM_CID_STK_TERMINAL_RESPONSE will be completed with MBIM_STATUS_SUCCESS also in cases the response APDU structure includes errors or warnings. The MBIM_STATUS_SUCCESS is not used when it was not possible to send the terminal response to the UICC.

10.5.27.2 PARAMETERS

Table 10-131: Parameters

	Set	Query	Notification
Command	MBIM_SET_STK_TERMINAL_RESPONSE	NA	NA
Response	MBIM_STK_TERMINAL_RESPONSE_INFO	NA	NA

10.5.27.3 DATA STRUCTURES

10.5.27.4 SET

The following structure shall be used in the InformationBuffer

Table 10-132: MBIM_SET_STK_TERMINAL_RESPONSE

Offset	Size	Field	Type	Description
0	4	ResponseLength	SIZE	Length of the terminal response in bytes.

4	Response Length	DataBuffer	DATABUFFER	TerminalResponse: Terminal response in BER_TLV format, starting with command details as defined in [ETSITS102223900].

10.5.27.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-133: MBIM_STK_TERMINAL_RESPONSE_INFO

Offset	Size	Field	Type	Description
0	4	ResultDataStringOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a APDU response data string. See [ETSITS102223900].
4	4	ResultLength	SIZE	Length of the APDU response data string.
8	4	StatusWords	UINT32	Status words SW1 and SW2 in the APDU response structure. Coded as 0xAABB if SW1=AA and SW2=BB. See [ETSITS102223900].
12		DataBuffer	DATABUFFER	ResultDataString

10.5.27.6 STATUS CODES

Status code	Description
MBIM_STATUS_NOT_INITIALIZED	The function has not finished initializing the UICC and the command cannot be completed
MBIM_STATUS_PIN_REQUIRED	A PIN code restricts the command to be performed
MBIM_STATUS_OPERATION_NOT_ALLOWED	The function does not allow the terminal response to be sent to the UICC, e.g. due to that no terminal response is expected

10.5.28 MBIM_CID_STK_ENVELOPE

10.5.28.1 DESCRIPTION

This CID is used to send an envelope command from the host to the SIM card.

Query:

The **Query** command can be used to retrieve which envelope commands the function expects the host to send to the SIM card and which envelope commands the function will generate on its own.

The response to the **Query** command includes MBIM_STK_ENVELOPE_INFO as InformationBuffer.

Set:

The **Set** command includes as MBIM_SET_STK_ENVELOPE InformationBuffer. The response to the **Set** command does not include any InformationBuffer (the reaction to an envelope command can be a proactive command from the SIM card).

10.5.28.2 PARAMETERS

Table 10-134: Parameters

	Set	Query	Notification
Command	MBIM_SET_STK_ENVELOPE	Empty	NA
Response	Empty	MBIM_STK_ENVELOPE_INFO	NA

10.5.28.3 SET

The following structure shall be used in the InformationBuffer

Table 10-135: MBIM_SET_STK_ENVELOPE

Offset	Size	Field	Type	Description
0	…	DataBuffer	DATABUFFER	EnvelopeCommand: Envelope command in BER-TLV format, starting with a tag that specifies which envelope command is being sent. The tag values are defined in [ETSITS102220920]. The format of the envelope commands are

				defined in [ETSITS102223900].

10.5.28.4 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.28.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-136: MBIM_STK_ENVELOPE_INFO

Offset	Size	Field	Type	Description
0	32	EnvelopeSupport	UINT8[32]	32 bytes where each bit indicates if a specific envelope command is expected to be sent by the host. Hosts must not send envelope commands that the function does not expect it to send. A bit in the field indicates if the corresponding envelope command is expected to be handled by the host or not. If the bit is 1 the host is expected to send the command; if it is 0, it is not. The commands are numbered according to the values associated with each envelope command in [ETSITS102223900]. The numbers associated with each envelope command are specified in [ETSITS102220920]. For example, bit 0xD3 means the Menu Selection, and bit 0xD7 means Timer Expiration. The bits are numbered so the least-significant bit is numbered 0.

10.5.28.6 STATUS CODES

Status code	Description
MBIM_STATUS_NOT_INITIALIZED	The function has not finished initializing the UICC and the command cannot be completed
MBIM_STATUS_PIN_REQUIRED	A PIN code restricts the command to be performed
MBIM_STATUS_OPERATION_NOT_ALLOWED	The function does not allow the terminal response to be sent to the UICC, e.g. due to that function sends the used envelope command itself

10.5.29 MBIM_CID_DEVICE_SERVICES

10.5.29.1 DESCRIPTION

This CID is used to query the device services supported by the MBIM devices and their properties.

Query only:

The MBIM_DEVICE_SERVICES_INFO structure will be filled out by the device along with the appropriate number of MBIM_DEVICE_SERVICE_ELEMENTs and returned to the host in the InformationBuffer of the CID request.

10.5.29.2 PARAMETERS

Table 10-137: Parameters

	Set	Query	Notification
Command	NA	Empty	NA
Response	NA	MBIM_DEVICE_SERVICES_INFO	NA

10.5.29.3 DATA STRUCTURES

Table 10-138: MBIM_DEVICE_SERVICE_ELEMENT

Offset	Size	Field	Type	Description

0	16	DeviceServiceId	UUID	A 16 byte UUID that identifies the device service.
16	4	DssPayload	UINT32	Bit 0: Indicates that DSS payload is supported from the host to the device for this device service. Bit 1: Indicates that DSS payload is supported to the host from the device for this device service. Unset: DSS payload not supported for this device service.
20	4	MaxDssInstances	UINT32	Max amount of instances of this service, which device can handle (support).
24	4	CidCount	UINT32	Number of CIDs supported for this device service.
28		DataBuffer	DATABUFFER	CidList: List of CIDs supported for this device service. There must be CIDCount number of entries in this list. Each CID is of type UINT32 per Table 10-4: Defined CIDs and Message Formats for Commands and Results.

10.5.29.4 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.29.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-139: MBIM_DEVICE_SERVICES_INFO

Offset	Size	Field	Type	Description
0	4	DeviceServicesCount (DSC)	UINT32	Number of MBIM_DEVICE_SERVICE_ELEMENT structures that follow in the DataBuffer. Each device service supported by the device must have a separate MBIM_DEVICE_SERVICE_ELEMENT entry
4	4	MaxDssSessions	UINT32	The maximum number of activated Device Service Stream sessions that are supported by the function. The protocol allows up to 256 simultaneous DSS sessions. See Section 7 on Data Transport.

8	8*DSC	DeviceServicesRefList	OL_PAIR_LIST	The first element of the pair is a 4 byte Offset in bytes, calculated from the beginning (offset 0) of this MBIM_DEVICE_SERVICES_INFO structure, to an MBIM_DEVICE_SERVICE_ELEMENT structure (see Table 10-138: MBIM_DEVICE_SERVICE_ELEMENT). The second element of the pair is a 4 byte size of the corresponding MBIM_DEVICE_SERVICE_ELEMENT structure.
8+8*DSC		DataBuffer	DATABUFFER	Array of MBIM_DEVICE_SERVICE_ELEMENT structures.

10.5.29.6 STATUS CODES

This CID only uses Generic Status Codes (see Use of the Status Codes section 9.4.5).

10.5.30 MBIM_CID_DEVICE_SERVICE_SUBSCRIBE_LIST

10.5.30.1 DESCRIPTION

The host uses this CID to inform the function of the CIDs for which the host wishes to receive unsolicited events via MBIM_INDICATE_STATUS_MESSAGE. As a result, the function must only indicate notifications for CIDs which have been enabled via this CID. The host updates this list appropriately as its state changes. State changes at the host are possibly triggered by events received from the function (including "wake up" from dormant states). Upon re-enabling suppressed notifications the host is responsible for synchronizing to the latest device state by performing queries for those CIDs.

Upon OPEN_DONE completion, notifications for CIDs defined in this specification are enabled (aka subscribed to) by default, and vendor extension notifications are off by default.

Set only:

The **Set** command and its complete response include an MBIM_DEVICE_SERVICE_SUBSCRIBE_LIST structure.

10.5.30.2 PARAMETERS

Table 10-140: Parameters

	Set	Query	Notification
Command	MBIM_DEVICE_SERVICE_SUBSCRIBE_LIST	NA	NA
Response	MBIM_DEVICE_SERVICE_SUBSCRIBE_LIST	NA	NA

10.5.30.3 DATA STRUCTURES

Table 10-141: MBIM_EVENT_ENTRY

Offset	Size	Field	Type	Description
0	16	DeviceServiceId	UUID	A 16 byte UUID that identifies the device service that the CID belongs to
16	4	CidCount	UINT32	The number of CID values that follow. If CidCount is 0, the function must enable unsolicited events for all CIDs defined for this device service.
20	4 * CidCount	DataBuffer	DATABUFFER	CidList: The list of CID values (pertaining to this DeviceServiceId) for which the host wishes to receive events. There must be CidCount number of entries in this list

10.5.30.4 SET

The following structure shall be used in the InformationBuffer

Table 10-142: MBIM_DEVICE_SERVICE_SUBSCRIBE_LIST

Offset	Size	Field	Type	Description
0	4	ElementCount (EC)	UINT32	Number of structures that follow in the DataBuffer
4	8*EC	DeviceServiceSubscribeRefList	OL_PAIR_LIST	The first element of the pair is a 4 byte Offset in bytes, calculated from the beginning (offset 0) of this MBIM_DEVICE_SERVICE_SUBSCRIBE_LIST structure, to an MBIM_EVENT_ENTRY structure (see Table 10-141: MBIM_EVENT_ENTRY). The second element of the pair is a 4 byte size of the corresponding MBIM_EVENT_ENTRY structure.
4+8*EC		DataBuffer	DATABUFFER	Array of MBIM_EVENT_ENTRY structures

10.5.30.5 RESPONSE

SeeTable 10-142: MBIM_DEVICE_SERVICE_SUBSCRIBE_LIST

10.5.30.6 STATUS CODES

Status code	Description
MBIM_STATUS_INVALID_DEVICE_SERVICE_OPERATION	The operation is invalid

10.5.31 MBIM_CID_AKA_AUTH

10.5.31.1 DESCRIPTION

This CID is used to send an authentication challenge to the device. The device must use the AKA (as opposed to AKA') 3rd generation Authentication and Key Agreement mechanism, specified for Universal Mobile Telecommunications System (UMTS) in [TS33.102] and for CDMA2000 in [S.S0055-A] for generating and indicating a response.

Multi-mode functions must support access to the SIM for authentication when operating in CDMA mode.

Query only:

The structure for the **Query** is MBIM_AKA_AUTH_REQ and the structure for the response is MBIM_AKA_AUTH_INFO.

10.5.31.2 PARAMETERS

Table 10-143: Parameters

	Set	Query	Notification
Command	NA	MBIM_AKA_AUTH_REQ	NA
Response	NA	MBIM_AKA_AUTH_INFO	NA

10.5.31.3 QUERY

The following structure shall be used in the InformationBuffer

Table 10-144: MBIM_AKA_AUTH_REQ

Offset	Size	Field	Type	Description
0	16	Rand	UINT8[16]	Random number challenge of 128 bit. This represents a multi-byte value in little-endian format.
16	16	Autn	UINT8[16]	AUTN is an authentication value generated by the AuC, which, together with the Rand, authenticates the server to the peer, 128 bits. This represents a multi-byte value in little-endian format.

10.5.31.4 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-145: MBIM_AKA_AUTH_INFO

Offset	Size	Field	Type	Description

0	16	Res	UINT8[16]	Authentication result from the peer, which, together with the Rand, authenticates the peer to the server, 128 bits. This is valid only in case of success. This represents a multi-byte value in little-endian format.
16	4	ResLen	UINT32	Response Length
20	16	IK	UINT8[16]	Integrating key. This is valid only in case of success. This represents a multi-byte value in little-endian format.
36	16	CK	UINT8[16]	Ciphering key. This is valid only in case of success. This represents a multi-byte value in little-endian format.
52	14	Auts	UINT8[14]	A value generated by the peer upon experiencing a synchronization failure, 112 bits. This is valid only in case of failure. This represents a multi-byte value in little-endian format.

10.5.31.5 STATUS CODES

Status code	Description
MBIM_STATUS_AUTH_INCORRECT_AUTN	The device sets this error on an AKA or AKAPrime challenge response when the AKA or AKAPrime challenge sent has incorrect AUTN.
MBIM_STATUS_AUTH_SYNC_FAILURE	The device sets this error on an AKA or AKAPrime challenge response when the AKA or AKAPrime challenge sent has synchronization failure. When this error code is returned the AUTS field would be set.

10.5.32 MBIM_CID_AKAP_AUTH

10.5.32.1 DESCRIPTION

This CID is used to send an authentication challenge to the device. The device must use the AKA' (as opposed to AKA) 3rd generation Authentication and Key Agreement mechanism, specified for Universal Mobile Telecommunications System (UMTS) in [TS33.102] and for CDMA2000 in [S.S0055-A] for generating and indicating a response. For more information see [RFC 5448] and 3GPP 33.402 [3GPP33402]

Multi-mode functions must support access to the SIM for authentication when operating in CDMA mode.

Query only:

The structure for the **Query** is MBIM_AKAP_AUTH_REQ and the structure for the response is MBIM_AKAP_AUTH_INFO

10.5.32.2 PARAMETERS

Table 10-146: Parameters

	Set	Query	Notification
Command	NA	MBIM_AKAP_AUTH_REQ	NA
Response	NA	MBIM_AKAP_AUTH_INFO	NA

10.5.32.3 QUERY

The following structure shall be used in the InformationBuffer

Table 10-147: MBIM_AKAP_AUTH_REQ

Offset	Size	Field	Type	Description
0	16	Rand	UINT8[16]	Random number challenge of 128 bits. This represents a multi-byte value in little-endian format.

16	16	Autn	UINT8[16]	AUTN is an authentication value generated by the AuC, which, together with the Rand, authenticates the server to the peer, 128 bits. This represents a multi-byte value in little-endian format.
32	4	NetworkNameOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a string NetworkName that represents the Name of the Access Network. See [RFC 5448].
36	4	NetworkNameLength	SIZE	Length of Name of Access Network.
40		DataBuffer	DATABUFFER	NetworkName

10.5.32.4 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-148: MBIM_AKAP_AUTH_INFO

Offset	Size	Field	Type	Description
0	16	Res	UINT8[16]	Authentication result from the peer, which, together with the Rand, authenticates the peer to the server, 128 bits. This is valid only in case of success. This represents a multi-byte value in little-endian format.
16	4	ResLen	UINT32	Response Length
20	16	IK	UINT8[16]	Integrating key. This is valid only in case of success. This represents a multi-byte value in little-endian format.

36	16	CK	UINT8[16]	Ciphering key. This is valid only in case of success. This represents a multi-byte value in little-endian format.
52	14	Auts	UINT8[14]	A value generated by the peer upon experiencing a synchronization failure, 112 bits. This is valid only in case of failure. This represents a multi-byte value in little-endian format.

10.5.32.5 STATUS CODES

Status code	Description
MBIM_STATUS_AUTH_INCORRECT_AUTN	The device sets this error on an AKA or AKAPrime challenge response when the AKA or AKAPrime challenge sent has incorrect AUTN.
MBIM_STATUS_AUTH_SYNC_FAILURE	The device sets this error on an AKA or AKAPrime challenge response when the AKA or AKAPrime challenge sent has synchronization failure. When this error code is returned the AUTS field would be set.
MBIM_STATUS_AUTH_AMF_NOT_SET	The device sets this error on an AKAPrime challenge response when the AKAPrime challenge sent does not have the AMF bit set to 1.

10.5.33 MBIM_CID_SIM_AUTH

10.5.33.1 DESCRIPTION

This CID is used to send an authentication challenge to the device. The device must use the authentication mechanism that is based on the GSM authentication and key agreement primitives which is a 2nd generation mobile network standard

Multi-mode functions must support access to the SIM for authentication when operating in CDMA mode.

Query only:

The structure for the **Query** is MBIM_SIM_AUTH_REQ and the structure for the response is MBIM_SIM_AUTH_INFO.

10.5.33.2 PARAMETERS

Table 10-149: Parameters

	Set	Query	Notification
Command	NA	MBIM_SIM_AUTH_REQ	NA
Response	NA	MBIM_SIM_AUTH_INFO	NA

10.5.33.3 QUERY

The following structure shall be used in the InformationBuffer.

Table 10-150: MBIM_SIM_AUTH_REQ

Offset	Size	Field	Type	Description
0	16	Rand1	UINT8[16]	Random number challenge of 128 bit. This represents a multi-byte value in little-endian format.
16	16	Rand2	UINT8[16]	Random number challenge of 128 bit. This represents a multi-byte value in little-endian format.
32	16	Rand3	UINT8[16]	Random number challenge of 128 bit. This represents a multi-byte value in little-endian format.
48	4	n	UINT32	number of random number challenges. “n” can be 2 or 3 per RFC 4186 [RFC 4186]. If n= 2, Rand1 and Rand2 should be used and if n=3, Rand1, Rand2 and Rand3 should be used

10.5.33.4 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-151: MBIM_SIM_AUTH_INFO

Offset	Size	Field	Type	Description
0	4	Sres1	UINT32	Response1 of 32 bit.
4	8	Kc1	UINT64	Encryption key1 of 64 bit.
12	4	Sres2	UINT32	Response2 of 32 bit.
16	8	Kc2	UINT64	Encryption key2 of 64 bit.
24	4	Sres3	UINT32	Response3 of 32 bit.
28	8	Kc3	UINT64	Encryption key3 of 64 bit.
36	4	n	UINT32	Number of responses. “n” can be 2 or 3 per RFC 4186 [RFC 4186]. If n= 2, Sres1/Kc1, Sres2/Kc2 should be used and if n=3, Sres1/Kc1, Sres2/Kc2 and Sres3/Kc3 should be used

10.5.33.5 STATUS CODES

This CID only uses Generic Status Codes (see Use of the Status Codes section 9.4.5).

10.5.34 MBIM_CID_PACKET_STATISTICS

10.5.34.1 DESCRIPTION

This command returns the packet statistics for all the Raw IP Data Streams kept by the device. These are IP packet statistics, unrelated to the underlying NTB transport used to transfer the IP packets. The scope of the statistics is to get a picture on what have been transferred on the USB layer. These statistics must be clear at initialization performed in MBIM_OPEN_DONE as a response to MBIM_OPEN_MSG (section 9.4.1).

Statistics are kept in two directions:

- “In”: From the function to the host.

- "Out": From the host to the function.

Query only:

InformationBuffer of **Query** message is empty. InformationBuffer of MBIM_COMMAND_DONE contains MBIM_PACKET_STATISTICS_INFO.

10.5.34.2 PARAMETERS

Table 10-152: Parameters

	Set	Query	Notification
Command	NA	Empty	NA
Response	NA	MBIM_PACKET_STATISTICS_INFO	NA

10.5.34.3 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.34.4 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-153: MBIM_PACKET_STATISTICS_INFO

Offset	Size	Field	Type	Description
0	4	InDiscards	UINT32	The number of inbound IP packets that have been discarded to free up memory resources (for example, due to flow control) or for other reasons not related to any error that would prevent their being sent to the host.
4	4	InErrors	UINT32	The number of inbound IP packets that have not been sent to the host because of errors.
8	8	InOctets	UINT64	The total number of raw IP bytes successfully sent to the host (not including any padding in NTB).
16	8	InPackets	UINT64	The number of IP packets successfully sent to the host.

24	8	OutOctets	UINT64	The total number of raw IP bytes successfully transmitted out of the function (not including any padding in NTB).
32	8	OutPackets	UINT64	The total number of raw IP packets successfully received by the function from the host.
40	4	OutErrors	UINT32	The number of outbound IP packets received by the function from the host that could not be delivered out of the function because of errors.
44	4	OutDiscards	UINT32	The number of outbound IP packets that have been discarded to free up memory resources (for example, due to flow control) or for other reasons not related to any error.

10.5.34.5 STATUS CODES

This CID only uses Generic Status Codes (see Use of the Status Codes section 9.4.5).

10.5.35 MBIM_CID_NETWORK_IDLE_HINT

10.5.35.1 DESCRIPTION

This CID is a network idle mode hint from the host to the function. The MBIM function can use this hint in its heuristics to enable mechanisms such as 'Fast Dormancy', and to faster enter low power modes in its network operations. The tradeoff, of course, is potentially longer latency. The host also uses its own heuristics to determine when to send this hint to the function, and may typically happen when the host estimates that for a period of time there will be a reduction in network traffic via this MBIM function, or if the host is entering some idle state.

Set and Query:

The **Query** command contains no InformationBuffer. The **Set** command and the MBIM_COMMAND_DONE (for both **Query** and **Set** requests) contain MBIM_NETWORK_IDLE_HINT as InformationBuffer.

10.5.35.2 PARAMETERS

Table 10-154: Parameters

	Set	Query	Notification
Command	MBIM_NETWORK_IDLE_HINT	Empty	NA
Response	MBIM_NETWORK_IDLE_HINT	MBIM_NETWORK_IDLE_HINT	NA

10.5.35.3 DATA STRUCTURES

Table 10-155: MBIM_NETWORK_IDLE_HINT_STATES

Types	Value	Description
MBIMNetworkIdleHintDisabled	0	The current hint to the function is to not utilize mechanisms for entering lower power modes faster.
MBIMNetworkIdleHintEnabled	1	The current hint to the function is to utilize mechanisms for entering lower power modes faster.

10.5.35.4 SET

The following structure shall be used in the InformationBuffer

Table 10-156: MBIM_NETWORK_IDLE_HINT

Offset	Size	Field	Type	Description
0	4	NetworkIdleHintState	MBIM_NETWORK_IDLE_HINT_STATES	One of the values in Table 10-155: MBIM_NETWORK_IDLE_HINT_STATES. For Set requests the value represents the hint from the host to the function. For completions to Query and Set requests the value represents the hint the function is currently following.

10.5.35.5 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.35.6 RESPONSE

See Table 10-156: MBIM_NETWORK_IDLE_HINT.

10.5.35.7 STATUS CODES

This CID only uses Generic Status Codes (see Use of the Status Codes section 9.4.5).

10.5.36 MBIM_CID_EMERGENCY_MODE

10.5.36.1 DESCRIPTION

This CID returns information about an MBIM function's emergency mode state. This CID applies to functions that are capable of supporting emergency mode. Emergency mode could be over the traditional voice or over VoIP.

Query:

Query InformationBuffer contains MBIM_EMERGENCY_MODE_INFO.

Unsolicited Event:

Unsolicited Event InformationBuffer contains MBIM_EMERGENCY_MODE_INFO.

10.5.36.2 PARAMETERS

Table 10-157: Parameters

	Set	Query	Notification
Command	NA	Empty	NA
Response	NA	MBIM_EMERGENCY_MODE_INFO	MBIM_EMERGENCY_MODE_INFO

10.5.36.3 DATA STRUCTURES

Table 10-158: MBIM_EMERGENCY_MODE_STATES

Types	Value	Description
MBIMEmergencyModeOff	0	Emergency service is not available.
MBIMEmergencyModeOn	1	Emergency service is available. Examples of emergency mode functions are calls to 112 in Europe or 911 in the US.

10.5.36.4 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.36.5 RESPONSE

The following structure shall be used in the InformationBuffer

Table 10-159: MBIM_EMERGENCY_MODE_INFO

Offset	Size	Field	Type	Description
0	4	EmergencyMode	MBIM_EMERGENCY_MODE_STATES	One of the values in Table 10-158: MBIM_EMERGENCY_MODE_STATES. This is the state of the device in terms of Emergency mode. Applicable to devices that are capable of supporting emergency mode. Emergency mode could be over the traditional voice or over VoIP. It is possible for devices to specify that they support emergency call services even in scenarios where they detect that the SIM is no longer valid. SIMs may become invalid for any of several reasons, e.g., the subscription is unpaid, or service has been deactivated because the device has been reported stolen.

10.5.36.6 NOTIFICATION

See Table 10-159: MBIM_EMERGENCY_MODE_INFO

10.5.36.7 STATUS CODES

This CID only uses Generic Status Codes (see Use of the Status Codes section 9.4.5).

10.5.37 MBIM_CID_IP_PACKET_FILTERS

10.5.37.1 DESCRIPTION

This command returns information about the list of packet filters and allows that list to be set. These packet filters apply exclusively to IP data stream sessions.

Some hosts are able to conserve energy and stay quiet in a “sleeping” state while not being used. USB Networking devices may provide special pattern filtering hardware that enables it to wake up the attached host on demand upon an attempt to contact the host (e.g., an incoming TCP connection).

NOTE: To enable remote wake-up, additional steps must be completed as described in [USB2.0].

If the host simply wishes to clear (remove) any previous filter for the specified SessionId, the value of PacketFiltersCount is set to Zero and no DataBuffer (pattern filter structure) follows.

If the specified pattern is not able to fit into the device, any pattern previously loaded is considered destroyed, and the function must set MBIM_STATUS_SUCCESS and PacketFiltersCount to 0 in the MBIM_PACKET_FILTERS returned from the response.

Once the PacketFilters are set on a given SessionId, the function must match the Session’s incoming packets to the specified filters. If there is a match, the function lets the packet through. If a packet does not match, it is discarded. If there are no PacketFilters set for a given SessionId, then the function allows all packets for that Session through.

NOTE: The use of packet filters can have a heavy impact on the performance of the device. Accordingly, packet filters are only recommended for use in low traffic scenarios. For firewall functionality it is recommended to use the host instead.

The host is in control of the packet filters by using this CID. It is expected that as the host changes state, it updates the set of filters to—for example—avoid being woken up due to non-essential traffic.

Query and Set:

Both **Query** and **Set** contain an MBIM_PACKET_FILTERS in its InformationBuffer. For **Query**, however, the only relevant field is the SessionId. The SessionId in a **Query** indicates which IP data stream’s filters are to be returned by the device.

MBIM_PACKET_FILTERS is returned from both **Query** and **Set** complete messages in the InformationBuffer.

10.5.37.2 PARAMETERS

Table 10-160: Parameters

	Set	Query	Notification
Command	MBIM_PACKET_FILTERS	MBIM_PACKET_FILTERS	NA
Response	MBIM_PACKET_FILTERS	MBIM_PACKET_FILTERS	NA

10.5.37.3 DATA STRUCTURES

Table 10-161: MBIM_SINGLE_PACKET_FILTER

Offset	Size	Field	Type	Description
0	4	FilterSize	SIZE (0..bMaxFilterSize)	Size in bytes of the PacketFilter field as well as the PacketMask field at the end of this structure. This number must not exceed the value of bMaxFilterSize from Table 6-3: MBIM FUNCTIONAL DESCRIPTOR.
4	4	PacketFilterOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a PacketFilter of size FilterSize. This is a string of bytes to perform pattern matching on, starting from the beginning of the IP packet (the Version field in an IPv4 or IPv6 header).
8	4	PacketMaskOffset	OFFSET	Offset in bytes, calculated from the beginning of this structure, to a PacketMask of size FilterSize. Each byte of Mask contains 8 masking bits, where each one of them represents whether or not the associated bit (if it is equal to 1) of the PacketFilter should be compared with what is seen on the media by the function.

12		DataBuffer	DATABUFFER	PacketFilter PacketMask

10.5.37.4 SET

The following structure shall be used in the InformationBuffer

Table 10-162: MBIM_PACKET_FILTERS

Offset	Size	Field	Type	Description
0	4	SessionId	UINT32	Packet Filters are specified in a per-SessionId basis. Accordingly, filters must not be set prior to opening the relevant SessionId via MBIM_CID_CONNECT.
4	4	PacketFiltersCount (PFC)	UINT32	Number of Table 10-161: MBIM_SINGLE_PACKET_FILTER structures that follow in the DataBuffer. This number must not exceed the value of bNumberFilters from Table 6-3: MBIM FUNCTIONAL DESCRIPTOR.
8	8*PFC	PacketFilterRefList	OL_PAIR_LIST	The first element of the pair is a 4 byte Offset in bytes, calculated from the beginning (offset 0) of this MBIM_PACKET_FILTERS structure, to an MBIM_SINGLE_PACKET_FILTER structure (see Table 10-161: MBIM_SINGLE_PACKET_FILTER). The second element of the pair is a 4 byte size of the corresponding MBIM_SINGLE_PACKET_FILTER structure.
8+8*PFC		DataBuffer	DATABUFFER	Array of MBIM_SINGLE_PACKET_FILTER

				structures.

10.5.37.5 QUERY

See **Set** and Table 10-162: MBIM_PACKET_FILTERS.

10.5.37.6 RESPONSE

See **Set** and Table 10-162: MBIM_PACKET_FILTERS.

10.5.37.7 STATUS CODES

Status code	Description
MBIM_STATUS_SUCCESS	The operation succeeded. If the specified pattern is not able to fit into the device, any pattern previously loaded is considered destroyed, and the function must still set MBIM_STATUS_SUCCESS but set PacketFiltersCount to 0 in the MBIM_PACKET_FILTERS returned from the response
MBIM_STATUS_BUSY	The operation failed because the device is busy.
MBIM_STATUS_FAILURE	The operation failed (a generic failure).
MBIM_STATUS_CONTEXT_NOT_ACTIVATED	The function shall set this if the context identified by SessionId is not an activated context.

10.5.38 MBIM_CID_DSS_CONNECT

10.5.38.1 DESCRIPTION

This CID activates and deactivates a data stream channel over the bulk pipe for a non-IP based Device Service.

This command can be called after a successful result of MBIM_OPEN_MSG.

Set operations are supported for this command.

The host might open several Device Service Streams up to the maximum number of sessions as specified in MaxDssSessions in Table 10-139: MBIM_DEVICE_SERVICES_INFO (itself, in response to a MBIM_CID_DEVICE_SERVICES per section 10.5.29).

The function may allow opening multiple instances of the same Device Service, if the value of MaxDssInstances in Table 10-138: MBIM_DEVICE_SERVICE_ELEMENT is more than one.

If the function cannot open a new session due to instance limit, it shall reply with status code MBIM_STATUS_DSS_INSTANCE_LIMIT.

Set only:

Set operations are supported for this command. The InformationBuffer for the command contains an MBIM_SET_DSS_CONNECT structure. The InformationBuffer in the response is empty.

10.5.38.2 PARAMETERS

Table 10-163: Parameters

	Set	Query	Notification
Command	MBIM_SET_DSS_CONNECT	NA	NA
Response	Empty	NA	NA

10.5.38.3 DATA STRUCTURES

Table 10-164: MBIM_DSS_LINK_STATE

Types	Value
MBIMDSSLinkDeactivate	0
MBIMDSSLinkActivate	1

10.5.38.4 SET

The following structure shall be used in the InformationBuffer

Table 10-165: MBIM_SET_DSS_CONNECT

Offset	Size	Field	Type	Description
0	16	DeviceServiceId	UUID	A 16 byte UUID that identifies the Device Service for provided DssSessionId.

16	4	DssSessionId	UINT32	Host specifies this member to uniquely identify the session for the Device Service Stream. Device must use the value in this member when completing set requests. The Host uses the value in this member in subsequent query requests as well as close requests to the device. The value shall also be used as an identifier on the data transport. The function shall reject the request if the host uses a DssSessionId that exceeds MaxDssSessions-1, where MaxDssSessions is in Table 10-139: MBIM_DEVICE_SERVICES_INFO. The host is using bits 7-0 of DssSessionId 4-byte value. Bits 31-8 are reserved and shall be zero. Valid values range from 0 to MaxDssSessions -1.
20	4	DssLinkState	UINT32	See Table 10-164: MBIM_DSS_LINK_STATE

10.5.38.5 RESPONSE

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.38.6 STATUS CODES

Status code	Description
MBIM_STATUS_SUCCESS	The DSS is successfully opened
MBIM_STATUS_BUSY	If the function temporarily cannot open the DSS it shall return this error code
MBIM_STATUS_FAILURE	An unknown error where encountered during opening of the DSS.

MBIM_STATUS_DSS_INSTANCE_LIMIT	If the function cannot open a new session due to instance limit, it shall reply with this status code.

10.5.39 MBIM_CID_MULTICARRIER_PROVIDERS

10.5.39.1 DESCRIPTION

This command returns a list of the current and previously added preferred providers for a multi-carrier function. This CID is supported when the function report support for MBIMCtrlCapsMultiCarrier.

Set and Query:

MBIM_PROVIDERS (see Table 10-39: MBIM_PROVIDERS) is returned in the InformationBuffer from both **Set** and **Query** complete messages. When the hosts sets a multicarrier preferred provider it is not required that the provider be settable as home provider. But when the list is queried by the host, the function must only return multicarrier preferred providers that are settable as home providers.

Unsolicited Event:

The InformationBuffer in the **Unsolicited Event** contains an MBIM_PROVIDERS structure.

10.5.39.2 PARAMETERS

Table 10-166: Parameters

	Set	Query	Notification
Command	MBIM_PROVIDERS	Empty	NA
Response	MBIM_PROVIDERS	MBIM_PROVIDERS	MBIM_PROVIDERS

10.5.39.3 QUERY

The InformationBuffer shall be null and InformationBufferLength shall be zero

10.5.39.4 RESPONSE

See Table 10-39: MBIM_PROVIDERS

10.5.39.5 NOTIFICATION

See Table 10-39: MBIM_PROVIDERS

10.5.39.6 STATUS CODES

Status code	Description

MBIM_STATUS_READ_FAILURE	Reading information from the function failed.

10.6 MBIM SERVICE AND CID EXTENSIBILITY

This specification targets only the basic functionality for a connectivity service, but the services and CIDs can easily be extended by creating a new MBIM service and matching UUID, and then registering them at the MBIM Registry [MBIMRegistry].

11 MBIM LOOPBACK TESTMODE

11.1 OVERVIEW

Loopback testing ensures that the link between host and device is verified with no dependency on the mobile broadband network. A successful pass of this test by the device, assures correct behavior of the device firmware's basic functionality.

Note that loopback functionality is tested only for IP data traffic. The scope of this test does not include any other network traffic, such as SMS or USSD. Also, because this is a loopback test that terminates at the device firmware, there is no dependency on the network, SIM, or air interfaces.

11.2 GUIDANCE

- MB device firmware should implement "loopback" APN functionality as explained here. Note that the loopback mode is independent of SIM and PIN lock states.
- On getting MBIM_CID_CONNECT set request with an ActivationCommand of MBIMActivationCommandActivate and an access string loopback, the firmware should do the following :
    - If the device is already connected, it should respond with an MBIM_STATUS_MAX_ACTIVATED_CONTEXTS.
    - The device should be able to enter loopback mode without registering with a provider.
    - The device should be able to enter loopback mode when its packet service state is detached.
    - The device should respond with an MBIM_CID_CONNECT response using the SessionId, IPType, and ContextType specified in the MBIM_CID_CONNECT request. ActivationState should be MBIMActivationStateActivated, and VoiceCallState should be MBIMVoiceCallStateNone.
    - Enter into loopback mode.
    - The device should respond to additional MBIM_CID_CONNECT set requests with MBIM_STATUS_MAX_ACTIVATE_CONTEXTS until loopback mode is deactivated.
- While in loopback mode:
    - On getting an MBIM_CID_CONNECT query, the device should respond with an MBIM_CID_CONNECT response using the SessionId, IPType, and ContextType specified in the MBIM_CID_CONNECT request. ActivationState should be MBIMActivationStateActivated, and VoiceCallState should be MBIMVoiceCallStateNone.
    - On getting an MBIM_CID_IP_CONFIGURATION query, the device should respond with an MBIM_STATUS_SUCCESS but not specify any IP addresses.
    - Listen for NTBs (NCM Transfer Block) on the BULK OUT pipe from the host.
        - The device should unpack the datagrams from the NTB and send them back to the host on the MBIM Bulk-IN pipe.
        - The device should swap the Source and Destination addresses of IPv4 and IPv6 datagrams. The device should not need to modify the IPv4 checksum and vendor datagrams.

- NTBs sent on the MBIM Bulk-IN pipe should conform to the NTB parameters specified by the device in the NCM GetNtbParameters function. Datagrams can be sent on the Bulk-IN pipe in one or more NTBs as required by the NTB parameters for the Bulk-IN pipe.

- On getting an MBIM_CID_CONNECT request with an ActivationCommand of MBIMActivationCommandDeactivate:

  - Ensure that the SessionId matches the SessionId used to start loopback mode. If the SessionId's don't match, the device should respond with MBIM_STATUS_CONTEXT_NOT_ACTIVATED.

  - The device should respond with an MBIM_CID_CONNECT response using the SessionId, IPType and ContextType specified in the MBIM_CID_CONNECT request. ActivationState should be MBIMActivationStateDeactivated and VoiceCallState should be MBIMVoiceCallStateNone.

12 MBIM MANDATORY FUNCTIONALITY

12.1 MBIM MANDATORY BASIC MECHANISMS

Basic mechanisms are mandatory as specified in sections 4, 6, 7 and 9. Section 0 contains mandatory mechanisms as specified in Table 8-1: Networking Control Model Requests.

The mechanisms specified in section 3.2 are conditionally mandatory: they are mandatory only for NCM/MBIM functions.

12.2 MBIM MANDATORY COMMAND IDS

Not all of the Service and Command IDs are mandatory to implement. That bare minimum mandatory to implement functionality comprises the following CIDs from the Basic Connectivity Service:

- MBIM_CID_DEVICE_CAPS
- MBIM_CID_SUBSCRIBER_READY_STATUS
- MBIM_CID_RADIO_STATE
- MBIM_CID_PIN
- MBIM_CID_HOME_PROVIDER
- MBIM_CID_REGISTER_STATE
- MBIM_CID_SIGNAL_STATE
- MBIM_CID_CONNECT
- MBIM_CID_IP_CONFIGURATION
- MBIM_CID_DEVICE_SERVICES
- MBIM_CID_PACKET_SERVICE

All other services and their CIDs are optional.

13 APPENDIX: COMPATIBILITY WITH NCM 1.0 EXAMPLE

This is an example on how the descriptors could look like for a MBIM/NCM device. In this example there is also an additional ACM port (Abstract Control Model from [USBCDC12]) just to show how MBIM/NCM can work with other functions as well. (Note: Both instances of "XXh" in the diagram below are due to the fact that the protocol in those two cases is implementation-dependent.)