Universal Serial Bus Communications Class Subclass Specification for PSTN Devices

Revision 1.2

February 9, 2007

Revision History

Rev	Date	Filename	Comments
1.2	2/9/07		Final update as per February 2007 CDC meeting.

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

Contributors, v1.1

Paul E. Berg	Moore Computer Consultants, Inc.
Chuck Brabenac	Intel Corporation
Shelagh Callahan	Intel Corporation
Paul Chehowski	Mitel Corporation
Joe Decuir	Microsoft Corporation
Stefan Eder	Siemens Semiconductors
Ed Endejan	3Com Corporation
Randy Fehr	Northern Telecom
Diego Friedel	AVM
John Howard	Intel Corporation
Ken Lauffenburger	Efficient Networks, Inc.
Ron Lewis	Rockwell Semiconductors
Dan Moore	Diamond Multimedia Systems
Terry Moore	Moore Computer Consultants, Inc.
Andy Nicholson	Microsoft Corporation
Nathan Peacock	Northern Telecom
Dave Perry	Mitel Corporation
Kenny Richards	Microsoft Corporation
Charlie Tai	Intel Corporation
Mats Webjörn	Universal Access

Contributors, V1.2

Bruce Balden	Belcarra
Jun Guo	Broadcom
CC Hung	Mentor Graphics
Dale Self	Symbian
Janne Rand	Nokia
Joe Decuir	MCCI
Joel Silverman	K-Micro
John Turner	Symbian
Ken Taylor	Motorola
Morten Christiansen	Ericsson AB
Peter FitzRandolph	MCCI
Richard Petrie	Nokia
Saleem Mohammed	Synopsys
Tero Soukko	Nokia

Table of Contents

1 Introduction 1
1.1 Purpose 1
1.2 Scope 1
1.3 Related Documents 1
1.4 Terms and Abbreviations 2
2 Management Overview 4
3 Functional Overview 5
3.1 Function Models 5
3.2 USB PSTN Device Models 5
3.2.1 Direct Line Control Model 5
3.2.1.1 DL Model 5
3.2.1.2 Datapump Model 6
3.2.2 Abstract Control Model 7
3.2.2.1 Abstract Control Model Serial Emulation 8
3.2.3 USB Telephone Model 8
3.2.3.1 Telephone Control Model 8
4 Class-Specific Codes 10
4.1 Communications Class Code 10
4.2 Communications Class Subclass Codes 10
4.3 Communications Class Protocol Codes 10
5 Descriptors 11
5.1 Standard USB Descriptor Definitions 11
5.2 Class-Specific Descriptors 11
5.3 Functional Descriptors 11
5.3.1 Call Management Functional Descriptor 11
5.3.2 Abstract Control Management Functional Descriptor 12
5.3.3 Direct Line Management Functional Descriptor 13
5.3.4 Telephone Ringer Functional Descriptor 13
5.3.5 Telephone Operational Modes Functional Descriptor 14
5.3.6 Telephone Call and Line State Reporting Capabilities Descriptor 15
5.4 Sample Class-Specific Functional Descriptors 16
6 Communications Class Specific Messages 18
6.1 Overview 18
6.2 Management Element Requests 18
6.2.1 Direct Line Control Model Requests 18
6.2.2 Abstract Control Model Requests 19
6.2.3 Telephone Control Model Requests 19
6.3 PSTN Subclass Specific Requests 20
6.3.1 SetCommFeature 21
6.3.2 GetCommFeature 21
6.3.3 ClearCommFeature 22

6.3.4 SetAuxLineState ... 22
6.3.5 SetHookState ... 22
6.3.6 PulseSetup ... 23
6.3.7 SendPulse ... 23
6.3.8 SetPulseTime ... 23
6.3.9 RingAuxJack ... 24
6.3.10 SetLineCoding ... 24
6.3.11 GetLineCoding ... 24
6.3.12 SetControlLineState ... 25
6.3.13 SendBreak ... 25
6.3.14 SetRingerParms ... 26
6.3.15 GetRingerParms ... 26
6.3.16 SetOperationParms ... 26
6.3.17 GetOperationParms ... 27
6.3.18 SetLineParms ... 27
6.3.19 GetLineParms ... 28
6.3.20 DialDigits ... 29
6.4 Management Element Notifications ... 30
6.4.1 Direct Line Control Model Notifications ... 30
6.4.2 Abstract Control Model Notifications ... 30
6.4.3 Telephone Control Model Notifications ... 31
6.5 PSTN Subclass Specific Notifications ... 31
6.5.1 ResponseAvailable ... 31
6.5.2 AuxJackHookState ... 32
6.5.3 RingDetect ... 32
6.5.4 SerialState ... 32
6.5.5 CallStateChange ... 33
6.5.6 LineStateChange ... 34
Appendix A: Communications Device Class Examples ... 36
A.1 Basic Telephone ... 36
A.2 Modem ... 36
Appendix B: Sample Configurations ... 38
B.1 Basic Telephony Configurations ... 38
B.2 Modem Configurations ... 38

List of Tables

Table 1 Class Subclass Code ... 10
Table 2 Class Protocol Code ... 10
Table 3: Call Management Functional Descriptor ... 11
Table 4: Abstract Control Management Functional Descriptor ... 12

Table 5: Direct Line Management Functional Descriptor 13

Table 6: Telephone Ringer Functional Descriptor....................13

Table 7: Telephone Operational Modes Functional Descriptor....................14

Table 8: Telephone Call State Reporting Capabilities Descriptor 16

Table 9: Sample Communications Class Specific Interface Descriptor*....................16

Table 10: Requests — Direct Line Control Model*....................18

Table 11: Requests — Abstract Control Model* 19

Table 12: Requests — Telephone Control Model*....................19

Table 13: Class-Specific Request Codes for PSTN subclasses....................20

Table 14: Communications Feature Selector Codes 21

Table 15: Feature Status Returned for ABSTRACT_STATE Selector....................21

Table 16: PSTN Relay Configuration Values 23

Table 17: Line Coding Structure 24

Table 18: Control Signal Bitmap Values for SetControlLineState 25

Table 19: Ringer Configuration Bitmap Values....................26

Table 20: Operation Mode Values 27

Table 21: Line State Change Value Definitions....................28

Table 22: Line Status Information Structure 28

Table 23: Line State Bitmap 28

Table 24: Call State Bitmap....................29

Table 25: Call State Value Definitions....................29

Table 26: Characters in a Dialing Command 30

Table 27: Notifications — Direct Line Control Model*....................30

Table 28: Notifications — Abstract Control Model* 31

Table 29: Notifications — Telephone Control Model*....................31

Table 30: Class-Specific Notification Codes for PSTN subclasses....................31

Table 31: UART State Bitmap Values....33

Table 32: Call State Change Value Definitions....33

Table 33: Line State Change Values....34

Table 34: Telephone Configurations....38

Table 35: Example Modem Configurations....38

List of Figures

Figure 1: DL Model....6

Figure 2: Datapump Model....7

Figure 3: Abstract Control Model....7

Figure 4: Telephone Control Model....9

1 Introduction

1.1 Purpose

The USB Communications Device Class Specification 1.1 contains general Communications Class specifications and details for seven device subclasses. That specification has been editorially reorganized into a common USB Communications Class Specification [USBCDC1.2] and a set of subclass specifications. This should help implementers understand what is necessary for each device subclass, and facilitate specification maintenance by the USB Device Working Group.

This document is one of those subclass specifications. It contains material technically identical to that contained in USB CDC 1.1. It is intended to guide implementers of USB-connected devices of the types listed in the following section.

1.2 Scope

This document specifies three device subclasses intended for use with Communications devices, based on [USBCDC1.2]. These device subclasses are:

- Direct Line Control Model
- Abstract Control Model
- Telephony Control Model

The intention of this specification is that all material presented here is technically compatible with the previous version of the USB CDC 1.1 Specification, from which it is derived. Numeric codes are defined for subclass codes, protocol codes, management elements, and notification elements.

In some cases material from [USBCDC1.2] is repeated for clarity. In such cases, [USBCDC1.2] shall be treated as the controlling document.

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

1.3 Related Documents

Reference	**Description**
[USB2.0]	Universal Serial Bus Specification, revision 2.0 (also referred to as the USB Specification). **http://www.usb.org**
[USBCCS1.0]	Universal Serial Bus Common Class Specification, revision 1.0. **http://www.usb.org**
[USBCDC1.2]	Universal Serial Bus Class Definitions for Communications Devices, Version 1.2. **http://www.usb.org**.
[USBWMC1.1]	Universal Serial Bus Subclass Specification for Wireless Mobile Communications, Version 1.1.**http://www.usb.org**
[USBAUD2.0]	Universal Serial Bus Device Class Definition for Audio Devices, Version 2.0. http://www.usb.org
[USBHID1.11]	Universal Serial Bus Device Class Definition for Human Interface Devices (HID), Version 1.11

[USBVID1.1]	Universal Serial Bus Device Class Definition for Video Devices, Version 1.1
[V24]	ITU-T V.24 (1996), List of definitions for interchange circuits between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE).
[V250]	ITU-T V.250 (1999) (renumbered from V.25ter) , Serial Asynchronous Automatic Dialing and Control. http://www.itu.ch
[V42]	ITU-T V.42 (1993), Error correction procedures for DCE using asynchronous to synchronous conversion. http://www.itu.ch
[V42b]	ITU-T V.42bis (1990), Data compression procedures for data circuit terminating equipment (DCE) using error correction procedure. http://www.itu.ch
[V34]	ITU-T V.34 (1998) A modem operating at data signalling rates of up to 33 600 bit/s for use on the general switched telephone network and on leased point-to-point telephone-type circuits. http://www.itu.ch
[V90]	ITU-T V.90 (1998) A digital and analogue modem pair for use on the Public Switched Telephone Network (PSTN) at data signalling rates of up to 56 000 bit/s downstream and up to 33 600 bit/s upstream. http://www.itu.ch
[ISO3166]	ISO 3166:1993, Codes for the representation of the names of countries. http://www.iso.org

1.4 Terms and Abbreviations

TERM	Description
ASVD	Analog Simultaneous Voice and Data, signaling method mixes data and voice. Example: ITU-T V.61
AT COMMAND SET	A telecommunications device control protocol. For details, see V.250.
BYTE	For the purposes of this document, the definition of a byte is 8 bits.
CALL MANAGEMENT	Refers to a process that is responsible for the setting up and tearing down of calls. This same process also controls the operational parameters of the call. The term “call,” and therefore “call management,” describes processes which refer to a higher level of call control, rather than those processes responsible for the physical connection.
COMMUNICATIONS INTERFACE	Refers to a USB interface that identifies itself as using the Communications Class definition.
DATA INTERFACE	Refers to a USB interface that identifies itself as using the Data Class definition.
DCE	Data Circuit Terminating Equipment; for example, a modem or ISDN Terminal Adaptor.
DEVICE MANAGEMENT	Refers to requests and responses that control and configure the operational state of the device. Device management requires the use of a Communications Class interface.
DESCRIPTOR	Data structure used to describe a USB device capability or characteristic
DEVICE	A logical or physical entity that performs a function. The actual entity described depends on the context of the reference. At the lowest level, device may refer to a single hardware component, as in a memory device. At a higher level, it may refer to a collection of hardware components that perform a particular function, such as a USB interface device. At an even higher level, device may refer to the function performed by an entity attached to the USB; for example, a data/FAX modem device. Devices may be physical, electrical, addressable, and logical. When used as a non-specific reference, a USB device is either a hub or a function.
DSVD	Digital Simultaneous Voice and Data, signaling method mixes data and digitized voice. Example: ITU-T V.70.
DTE	Data Terminal Equipment; for example, a Personal Computer.
FUNCTION	Device capabilities exposed over the USB cable.
ITU	International Telecommunications Union (formerly CCITT).
MANAGEMENT ELEMENT	Refers to a type of USB pipe that manages the communications device and its interfaces. Currently, only the Default Pipe is used for this purpose.
MASTER INTERFACE	A Communications Class interface, which has been designated the master of zero or more interfaces that implement a complete function in a USB Communications device. This interface will accept

	management requests for the union.
MULTIFUNCTION DEVICE	A device of peripheral that exposes one or more functions or services to an end user. Exposed services can but do have to be exposed as USB functions.
NOTIFICATION ELEMENT	Refers to a type of USB pipe. Although a notification element is not required to be an interrupt pipe, a notification element is typically defined in this way.
NOTIFICATION ELEMENT	Refers to a type of USB pipe. Although a notification element is not required to be an interrupt pipe, a notification element is typically defined in this way.
PSTN	Public Switched Telephone Network.
UNION	A relationship between a collection of one or more interfaces that can be considered to form a functional unit.
UNIT	Entity that provides the basic building blocks to describe a protocol stack.
VIDEO PHONE	A device which simultaneously sends voice and video with optional data. For example: ITU-T H.324

2 Management Overview

This subclass specification includes specifications for two kinds of devices that connect to the Public Switched Telephone Network (PSTN):

- Voiceband modems
- Telephones

There are two common types of voiceband modems:

- Host dependent modems, which contain essential hardware for connecting to the PSTN, and may contain a modem signal processor, but which also depend on the host system for control logic. These are supported by the Direct Line Control Model.
- 'Smart modems' These contain a command processor that executes serial AT commands, such as those from ITU-T V.250 or its extensions. These are supported by the Abstract Control Model.

This specification addresses three control models:

- Direct Line Control Model, for modem devices directly controlled by the USB host
- Abstract Control Model, for modem devices controlled through a serial command set
- Telephone Control Model, for voice telephony devices.

Devices of these subclasses must conform to:

- USB Specification [USB2.0]
- Device Framework
- Communications Device Class 1.2 [USBCDC1.2]

3 Functional Overview

3.1 Function Models

[USB2.0] defines “function” as a “USB device that provides a capability to the host, such as an ISDN connection, a digital microphone, or speakers”. Further, in section 5.2.3, it says “Multiple functions may be packaged into what appears to be a single physical device…. A device that has multiple interfaces controlled independently of each other is referred to as a composite device.” We therefore adopt the term “function” to describe a set of one or more interfaces which taken together provide a capability to the host.

Functions defined in this specification may be used as part of multifunction devices (e.g. [USBWMC1.1]) or as single-function devices.

3.2 USB PSTN Device Models

A USB telephony device used on a PSTN line has several types of interfaces that could be presented to the host. The arrangement and use of those different interfaces depends upon the type of PSTN telephony device and the basic model used to build the device.

The difference between the various models of telephony devices can be divided according to the amount of processing the device performs on the analog signal before presenting it to the host. To help illustrate how the different types of interfaces could be put together to build a USB PSTN telephony device, three example models are presented in the following sections.

Note: In many cases a Data Class interface might not be used to present data to the host. Where the USB device is constructed with minimal intelligence, some analog class-specific interface control codes are required.

3.2.1 Direct Line Control Model

The Direct Line Control Model contains two examples: the Direct Line Model (or DL Model) and the Datapump Model.

A Communications Class interface of type Direct Line Control Model will consist of a minimum of two pipes; one is used to implement the management element and the other to implement a notification element. In addition, the device can use two or more pipes to implement channels over which to carry vendor-specific data.

3.2.1.1 DL Model

The DL Model is the simplest type of connection to a PSTN line. At this level, the USB device is only converting the analog PSTN line signal to digital data and presenting it to the USB bus. The modem modulation protocol (e.g. [V34], [V90]) is implemented in the host. Instead of using the Data Class, the Audio Class is used to present the digitally converted data to the host. This type of connection could also be useful for a voice-only device, such as an answering machine.

Because the DL Model is the simplest, it provides a perfect example of why a device requires the Direct Line Control Model control codes. The key feature of a DL Model device is low cost, so reducing the processing power requirements on the USB device is essential. The DL Model uses a Direct Line Control Model SubClass code in the descriptor definition of its Communications Class interface.

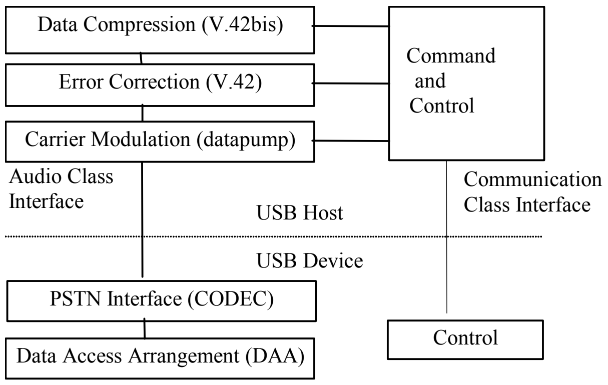
Data Compression (V.42bis)
Error Correction (V.42)
Carrier Modulation (datapump)
Command and Control
Audio Class Interface
USB Host
Communication Class Interface
USB Device
PSTN Interface (CODEC)
Data Access Arrangement (DAA)
Control

Figure 1: DL Model

These requests for controlling the interface between the USB device and the PSTN line are presented in Table 10. There are also some additional signals that fall outside the analog phone signal which shall go back to the host as notifications, which are represented in Table 27. These requests and notifications are transported via the Communications Class interface for the device.

3.2.1.2 Datapump Model

The Datapump view of the device is the next logical break and is similar to the DL Model. In the Datapump view, the USB device handles the carrier modulation instead of the host. Because there are no standard interfaces for Datapumps, and it would be difficult to generalize the I/O space and registers required, it is assumed a vendor-specific interface is employed based on the specifics of the Datapump being used.

The PSTN line interface requests and notifications required for the Datapump USB device are the same as used for the DL Model as described in Table 10 and Table 27, so the Direct Line Control Model SubClass code shall be used.

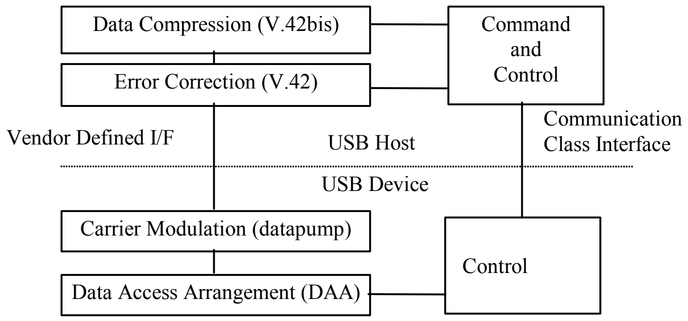
Data Compression (V.42bis)
Error Correction (V.42)
Command and Control
Vendor Defined I/F
USB Host
Communication Class Interface
USB Device
Carrier Modulation (datapump)
Data Access Arrangement (DAA)
Control

Figure 2: Datapump Model

3.2.2 Abstract Control Model

With an Abstract Control Model, the USB device understands standard V.250 (AT) commands. The device contains a Datapump and micro-controller that handles the AT commands and relay controls. The device uses both a Data Class interface and a Communications Class interface. For an illustration of the use of both interfaces, see Figure 3. The device can also, at times, make use of other class interfaces; for example a device could use an Audio Class interface for the audio functions in a speakerphone.

A Communications Class interface of type Abstract Control Model will consist of a minimum of two pipes; one is used to implement the management element and the other to implement a notification element. In addition, the device can use two pipes to implement channels over which to carry unspecified data, typically over a Data Class interface.

For PSTN line control, an Abstract Control Model shall either support [V250] commands embedded in the data stream or [V250] commands sent down the Communications Class interface. When [V250] commands are multiplexed in the data stream, the Heatherington Escape Sequence or the TIES method would define the only supported escape sequences.

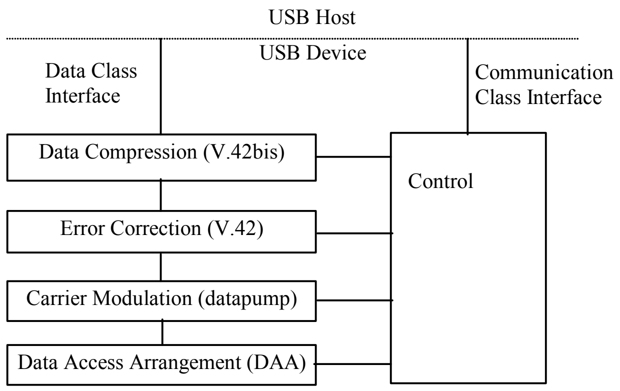
USB Host
USB Device
Data Class Interface
Communication Class Interface
Data Compression (V.42bis)
Control
Error Correction (V.42)
Carrier Modulation (datapump)
Data Access Arrangement (DAA)

Figure 3: Abstract Control Model

Error correction and data compression could be implemented on the host, and not necessarily on the device. This type of device differs from the Direct Line Control Model, because the data from the USB device is presented to the host via a native class-defined interface rather than a vendor-specific

Datapump interface. Also, [V250] commands are used to control the PSTN line interface. V.80 defines one way that the host can control the DCE data stream to accomplish this, but there are also proprietary methods.

3.2.2.1 Abstract Control Model Serial Emulation

The Abstract Control Model can bridge the gap between legacy modem devices and USB devices. To support certain types of legacy applications, two problems need to be addressed. The first is supporting specific legacy control signals and state variables, which are addressed directly by the various carrier modulation standards. Because of these dependencies, they are important for developing an analog modem, which presents an Abstract Control Model type Communications Class interface to the host. To support these requirements additional requests (Table 11) and notifications (Table 28) have been created.

The second significant item, which is needed to bridge the gap between legacy modem designs and the Abstract Control Model, is a means to multiplex call control (AT commands) on the Data Class interface. Legacy modem designs are limited by only supporting one channel for both "AT" commands and the actual data. To allow this type of functionality, the device must have a means to specify this limitation to the host.

When describing this type of device, the Communications Class interface would still specify an Abstract Control Model, but call control would actually occur over the Data Class interface. To describe this particular characteristic, the Call Management Functional Descriptor (Section 5.3.1) would have bit D1 of *bmCapabilities* set.

For devices that support both modes, call control over the Communications Class interface and call control over a Data Class interface, and need to switch between them, then the *GetCommFeature* (Section 6.3.2) request is used to switch between modes.

3.2.3 USB Telephone Model

A USB telephone device has a type of Communications Class interface that will be presented to the host, and it has the SubClass code of Telephone Control Model. A telephone device will not typically present a Data Class interface.

3.2.3.1 Telephone Control Model

Telephone devices with multiple lines will have a separate Communications Class interface for each physical line connected to the device. Each individual interface will correspond to a different physical line representing a network connection to the device.

Functional descriptors will be used to describe the various capabilities of a USB telephone device. These functional descriptors are defined in Section 5.3, “Functional Descriptors.”

A Communications Class interface of SubClass code Telephone Control Model will consist of a minimum of two pipes: one to implement the management element and the other to implement the notification element. This model describes the simplest version of a USB telephone device using only a Communications Class interface; other, more complicated implementations are possible.

To create more complicated implementations of a USB telephone device for example, use an Audio Class interface to provide the audio capabilities of a telephone and a Human Interface Device Class interface to provide the keypad capabilities of a telephone.

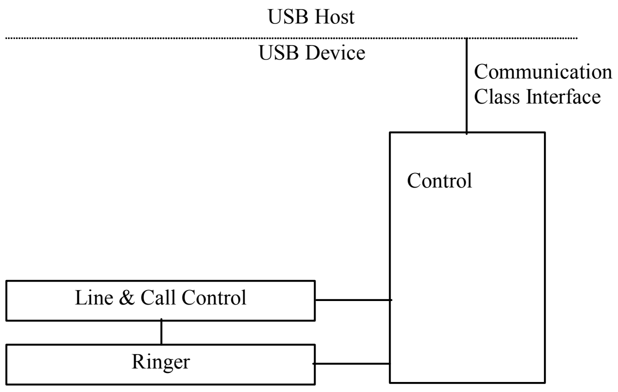

Figure 4: Telephone Control Model

The requests for controlling the USB telephone device via its Communications Class interface are presented in Table 12. Unsolicited messages from the USB telephone device to the host are sent using the notification element messages that are presented in Table 29. These requests and notifications are transported via the Communications Class interface for the device.

4 Class-Specific Codes

This section lists the codes for the Communications Device Class, Communications Interface Class and Data Interface Class, including subclasses and protocols. These values are used in the *DeviceClass*, *bInterfaceClass*, *bInterfaceSubClass*, and *bInterfaceProtocol* fields of the standard device descriptors as defined in chapter 9 of the [USB2.0].

4.1 Communications Class Code

This is defined in [USBCDC1.2].

4.2 Communications Class Subclass Codes

The following table defines the Communications Subclass codes:

Table 1 Class Subclass Code

Code	Subclass
01h	Direct Line Control Model
02h	Abstract Control Model
03h	Telephone Control Model

4.3 Communications Class Protocol Codes

[USBCDC1.2] section 4.4 defines Communications Class Protocols.

The following table lists the Protocol codes used in this subclass specification:

Table 2 Class Protocol Code

Code	Protocol
00h	No class specific protocol required
01h	AT Commands: [V250] etc

If a Communications Class interface appears with multiple alternate settings, all alternate settings for that interface shall have the same bInterfaceClass, bInterfaceSubclass and bInterfaceProtocol codes.

5 Descriptors

5.1 Standard USB Descriptor Definitions

Devices that conform to this subclass specification need to implement the standard USB descriptors for the Communications Device Class, Communications Interface Class and Data Interface Class. These are defined in [USBCDC1.2].

5.2 Class-Specific Descriptors

Devices that conform to this subclass specification may need to implement class-specific descriptors for the Communications Interface Class and Data Interface Class. These are defined in [USBCDC1.2].

5.3 Functional Descriptors

Functional descriptors describe the content of the class-specific information within an Interface descriptor. Functional descriptors all start with a common header descriptor, which allows host software to easily parse the contents of class-specific descriptors. Each class-specific descriptor consists of one or more functional descriptors. Although the Communications Class currently defines class specific descriptor information, the Data Class does not.

The [USBCDC1.2] specification describes functional descriptors that may be used in all Communications Subclasses. These include:

- Header Functional Descriptor
- Union Functional Descriptor
- Country Selection Functional Descriptor

The following Functional Descriptors are specific to PSTN subclass devices.

5.3.1 Call Management Functional Descriptor

The Call Management functional descriptor describes the processing of calls for the Communications Class interface. It can only occur within the class-specific portion of an Interface descriptor.

Table 3: Call Management Functional Descriptor

Offset	Field	Size	Value	Description
0	*bFunctionLength*	1	Number	Size of this functional descriptor, in bytes.
1	*bDescriptorType*	1	Constant	CS_INTERFACE
2	*bDescriptorSubtype*	1	Constant	Call Management functional descriptor subtype, as defined in [USBCDC1.2].

Offset	Field	Size	Value	Description
3	*bmCapabilities*	1	Bitmap	The capabilities that this configuration supports: D7..D2: RESERVED (Reset to zero) D1: 0 - Device sends/receives call management information only over the Communications Class interface. 1 - Device can send/receive call management information over a Data Class interface. D0: 0 - Device does not handle call management itself. 1 - Device handles call management itself. The previous bits, in combination, identify which call management scenario is used. If bit D0 is reset to 0, then the value of bit D1 is ignored. In this case, bit D1 is reset to zero for future compatibility.
4	*bDataInterface*	1	Number	Interface number of Data Class interface optionally used for call management. *

* Zero based index of the interface in this configuration (*bInterfaceNum*).

5.3.2 Abstract Control Management Functional Descriptor

The Abstract Control Management functional descriptor describes the commands supported by the Communications Class interface, as defined in Section 3.2.2, with the SubClass code of Abstract Control Model. It can only occur within the class-specific portion of an Interface descriptor.

Table 4: Abstract Control Management Functional Descriptor

Offset	Field	Size	Value	Description
0	*bFunctionLength*	1	Number	Size of this functional descriptor, in bytes.
1	*bDescriptorType*	1	Constant	CS_INTERFACE
2	*bDescriptorSubtype*	1	Constant	Abstract Control Management functional descriptor subtype as defined in [USBCDC1.2].
3	*bmCapabilities*	1	Bitmap	The capabilities that this configuration supports. (A bit value of zero means that the request is not supported.) D7..D4: RESERVED (Reset to zero) D3: 1 - Device supports the notification Network_Connection. D2: 1 - Device supports the request Send_Break D1: 1 - Device supports the request combination of Set_Line_Coding, Set_Control_Line_State, Get_Line_Coding, and the notification Serial_State. D0: 1 - Device supports the request combination of Set_Comm_Feature, Clear_Comm_Feature, and Get_Comm_Feature. The previous bits, in combination, identify which requests/notifications are supported by a CommunicationsClass interface with the SubClass code of Abstract Control Model.

5.3.3 Direct Line Management Functional Descriptor

The Direct Line Management functional descriptor describes the commands supported by the Communications Class interface, as defined in Section 3.2.1, with the SubClass code of Direct Line Control Model. It can only occur within the class-specific portion of an Interface descriptor.

Table 5: Direct Line Management Functional Descriptor

Offset	Field	Size	Value	Description
0	*bFunctionLength*	1	Number	Size of this functional descriptor, in bytes.
1	*bDescriptorType*	1	Constant	CS_INTERFACE
2	*bDescriptorSubtype*	1	Constant	Direct Line Management functional descriptor subtype, as defined in [USBCDC1.2].
3	*bmCapabilities*	1	Bitmap	The capabilities that this configuration supports. (A value of zero means that the request or notification is not supported.) D7..D3: RESERVED (Reset to zero) D2: 1 - Device requires extra Pulse_Setup request during pulse dialing sequence to disengage holding circuit. (see Section 6.3.6) D1: 1 - Device supports the request combination of Set_Aux_Line_State, Ring_Aux_Jack, and notification Aux_Jack_Hook_State. D0: 1 - Device supports the request combination of Pulse_Setup, Send_Pulse, and Set_Pulse_Time. The previous bits, in combination, identify which requests/notifications are supported by a Communications Class interface with the SubClass code of DL Control Modem.

5.3.4 Telephone Ringer Functional Descriptor

The Telephone Ringer functional descriptor describes the ringer capabilities supported by the Communications Class interface, as defined in Section 3.2.3.1, with the SubClass code of Telephone Control. It can only occur within the class-specific portion of an Interface descriptor.

For a multiple line phone device, where separate Communications Class interfaces would exist for each line supported by the phone, typically one interface would be designated via a Union functional descriptor to be the controlling interface for the device. If only one ringer existed for all the lines, the Telephone Ringer Functional descriptor would only be needed for the descriptor of this controlling interface.

Table 6: Telephone Ringer Functional Descriptor

Offset	Field	Size	Value	Description
0	*bFunctionLength*	1	Number	Size of this functional descriptor, in bytes.
1	*bDescriptorType*	1	Constant	CS_INTERFACE
2	*bDescriptorSubtype*	1	Constant	Telephone Ringer functional descriptor subtype as defined in [USBCDC1.2]

Offset	Field	Size	Value	Description
3	*bRingerVolSteps*	1	Number	Number of discrete steps in volume supported by the ringer, values are: 0: 256 discrete volume steps. 1: 1 discrete volume step (i.e., fixed volume). Value 0 will be ringer off setting and Value 1 to 255 will result in the same ringer volume level. 2: 2 discrete volume steps Value 0 will be ringer off setting and Values 1 to 127 will result in the first volume level setting. Values 128 to 255 will result in the 2nd volume level setting 3: 3 discrete volume steps. Value 0 will be ringer off setting and Values 1 to 84 will result in the first volume level setting. Value 85 to 170 will result in the 2nd volume level setting. Value 171 to 255 will result in the 3rd volume level setting. As a general rule, the range of volume settings is broken up into a number of equal steps, the number of steps defined by the bRingerVolSteps value. A general formula for defining ranges, based on X=bRingerVolSteps and values [1 to Y] defining the first volume range is: Y = (256/X) - 1, where X<>0 and X<=128 Second volume range is: [(Y+1) to (Y+Y)] Note: that the maximum value in the last range must always be 255
4	*bNumRingerPatterns*	1	Number	Number of ringer patterns supported, values of 1 to 255 with a value of 0 being reserved for future use.

5.3.5 Telephone Operational Modes Functional Descriptor

The Telephone Operational Modes functional descriptor describes the operational modes supported by the Communications Class interface, as defined in Section 3.2.3.1, with the SubClass code of Telephone Control. It can only occur within the class-specific portion of an Interface descriptor. The modes supported are Simple, Standalone, and Computer Centric. See Section 6.3.16 for a definition of the various operational modes and Table 20 for the definition of the operational mode values.

For a multiple line phone device, where separate Communications Class interfaces would exist for each line supported by the phone, typically one interface would be designated via a Union functional descriptor to be the controlling interface for the device. In this case, the Telephone Operational Modes descriptor would only be needed for the descriptor of this controlling interface.

Table 7: Telephone Operational Modes Functional Descriptor

Offset	Field	Size	Value	Description
0	*bFunctionLength*	1	Number	Size of this functional descriptor, in bytes.
1	*bDescriptorType*	1	Constant	CS_INTERFACE

Offset	Field	Size	Value	Description
2	*bDescriptorSubtype*	1	Constant	Telephone Operational Modes functional descriptor subtype as defined in [USBCDC1.2]
3	*bmCapabilities*	1	Bitmap	This configuration supports the following operational modes: D7..D3: RESERVED (Reset to zero) D2: 0 - Does not support Computer Centric mode. 1 - Supports Computer Centric mode. D1: 0 - Does not support Standalone mode. 1 - Supports Standalone mode. D0: 0 - Does not support Simple mode. 1 - Supports Simple mode.

5.3.6 Telephone Call and Line State Reporting Capabilities Descriptor

The Telephone Call and Line State Reporting Capabilities functional descriptor describes the abilities of a telephone device to report optional call and line states. All telephone devices, as a minimum, shall be capable of reporting the following call states:

- Idle
- Dialtone
- Dialing
- Connected
- Ringing
- Answered

Call state reports that are optional and will be described by this descriptor are states such as:

- Interrupted dialtone
- Ringback
- Busy
- Fast busy (also known as equipment busy or reorder tone)
- Caller ID
- Distinctive ringing decoding

Line state reports are optional and will be described by this descriptor.

The Telephone Call State Reporting Capabilities functional descriptor can exist in the class-specific portion of a Communications Class interface, as defined in Section 3.2.3.1, with the SubClass code of Telephone Control. For a multiple line phone device, where separate Communications Class interfaces would exist for the each line supported by the phone, typically one interface would be designated via a Union functional descriptor, to be the controlling interface for the device. In this case, the Telephone Call State Reporting Capabilities Functional descriptor would only be needed for the descriptor of this controlling interface, if each of the Communications Class interfaces supported the same call state reporting capabilities.

Table 8: Telephone Call State Reporting Capabilities Descriptor

Offset	Field	Size	Value	Description
0	*bFunctionLength*	1	Number	Size of this functional descriptor, in bytes.
1	*bDescriptorType*	1	Constant	CS_INTERFACE
2	*bDescriptorSubtype*	1	Constant	Telephone Call State Reporting Capabilities descriptor subtype, as defined in [USBCDC1.2].
3	*bmCapabilities*	4	Bitmap	Call and line state reporting capabilities of the device when the following bits are set: D31-D6: RESERVED (Reset to zero) D5: 0 – Does not support line state change notification. 1 – Does support line state change notification. D4: 0 – Cannot report dual tone multi-frequency (DTMF) digits input remotely over the telephone line. 1 – Can report DTMF digits input remotely over the telephone line. D3: 0 – Reports only incoming ringing. 1 – Reports incoming distinctive ringing patterns. D2: 0 – Does not report caller ID. 1 – Reports caller ID information. D1: 0 – Reports only dialing state. 1 – Reports ringback, busy, and fast busy states. D0: 0 – Reports only dialtone (does not differentiate between normal and interrupted dialtone). 1 – Reports interrupted dialtone in addition to normal dialtone.

5.4 Sample Class-Specific Functional Descriptors

Table 9 presents an example of the Communications Class Functional Descriptors for a simple Abstract Control Model device.

Table 9: Sample Communications Class Specific Interface Descriptor*

Offset	Field	Size	Value	Description
0	*bFunctionLength*	1	05h	Size of this functional descriptor, in bytes.
1	*bDescriptorType*	1	24h	CS_INTERFACE
2	*bDescriptorSubtype*	1	00h	Header. This is defined in [USBCDC1.2], which defines this as a header.
3	*bcdCDC*	2	0110h	USB Class Definitions for Communications Devices Specification release number in binary-coded decimal.
5	*bFunctionLength*	1	04h	Size of this functional descriptor, in bytes.

Offset	Field	Size	Value	Description
6	*bDescriptorType*	1	24h	CS_INTERFACE
7	*bDescriptorSubtype*	1	02h	Abstract Control Management functional descriptor subtype as defined in [USBCDC1.2].
8	*bmCapabilities*	1	0Fh	This field contains the value 0Fh, because the device supports all the corresponding commands for the Abstract Control Model interface.
9	*bFunctionLength*	1	05h	Size of this functional descriptor, in bytes
10	*bDescriptorType*	1	24h	CS_INTERFACE
11	*bDescriptorSubtype*	1	06h	Union Descriptor Functional Descriptor subtype as defined in [USBCDC1.2].
12	*bControlInterface*	1	00h	Interface number of the control (Communications Class) interface
13	*bSubordinateInterface0*	1	01h	Interface number of the subordinate (Data Class) interface
14	*bFunctionLength*	1	05h	Size of this functional descriptor, in bytes
15	*bDescriptorType*	1	24h	CS_INTERFACE
16	*bDescriptorSubtype*	1	01h	Call Management Functional Descriptor subtype as defined in [USBCDC1.2].
17	*bmCapabilities*	1	03h	Indicate that the device handles call management itself (bit D0 is set), and will process commands multiplexed over the data interface in addition to commands sent using SEND_ENCAPSULATED_COMMAND (bit D1 is set).
18	*bDataInterface*	1	01h	Indicates that multiplexed commands are handled via data interface 01h (same value as used in the UNION Functional Descriptor)

* This descriptor is specific to the Communications Class.

6 Communications Class Specific Messages

6.1 Overview

The Communications Interface Class supports the standard requests defined in chapter 9 of [USB2.0]. In addition, the Communications Interface Class has some class-specific requests and notifications. These are used for device and call management.

Requests for controlling the interface between a USB PSTN device are presented in Section 6.2. There are also some additional signals that shall go back to the host as notifications, which are represented in Section 6.3. These requests and notifications are transported via the Communications Class interface for the device.

6.2 Management Element Requests

The following requests are used by devices conforming to this subclass specification. The only class-specific request codes that are valid for a Communications Class interface with a Communications Class SubClass codes of DLM, ACM or TCM are listed in the following tables.

6.2.1 Direct Line Control Model Requests

A *Direct Line Control Model* Communications Device uses a Communications Class interface for device management. with a Communications Subclass code of Direct Line. The only class-specific request codes that are valid for a Communications Class interface with a Communications Class SubClass code of Direct Line Control Model are listed in Table 10. The other class-specific requests not listed in the previous table, such as *SendEncapsulatedCommand*, are inappropriate for a Direct Line Control Model and shall generate a STALL condition if sent to such an interface. For example, hanging up the line would be accomplished by using *SetHookState*, rather than by sending "ATH" via *SendEncapsulatedCommand*.

Table 10: Requests — Direct Line Control Model*

Request	Summary	Req'd/Opt	reference
SetAuxLineState	Request to connect or disconnect secondary jack from PSTN circuit or CODEC, depending on hook state.	Optional	6.3.4
SetHookState	Select relay setting for on-hook, off-hook, and caller ID.	Required	6.3.5
PulseSetup	Initiate pulse dialing preparation.	Optional	6.3.6
SendPulse	Request number of make/break cycles to generate.	Optional	6.3.7
SetPulseTime	Setup value for time of make and break periods when pulse dialing.	Optional	6.3.8
RingAuxJack	Request for a ring signal to be generated on secondary phone jack.	Optional	6.3.9

*These requests are specific to the Communications Class.

6.2.2 Abstract Control Model Requests

An Abstract Control Model Communications Device uses a Communications Class interface for device management. With a Communications Subclass code of Abstract Control. The only class-specific request codes that are valid for a Communications Class interface with a Communications Class SubClass code of Abstract Control Model are listed in the following Table 11. The other class-specific requests not listed in that table, such as *SetHookState*, are inappropriate for an Abstract Control Model and would generate a STALL condition if sent to such an interface. For example, hanging up the line would be accomplished by sending "ATH" via *SendEncapsulatedComand*, rather than by using *SetHookState*.

Table 11: Requests — Abstract Control Model*

Request	Summary	Req'd/Opt	reference
SendEncapsulatedCommand	Issues a command in the format of the supported control protocol.	Required	[USBCDC1.2]
GetEncapsulatedResponse	Requests a response in the format of the supported control protocol.	Required	[USBCDC1.2]
SetCommFeature	Controls the settings for a particular communications feature.	Optional	6.3.1
GetCommFeature	Returns the current settings for the communications feature.	Optional	6.3.2
ClearCommFeature	Clears the settings for a particular communications feature.	Optional	6.3.3
SetLineCoding	Configures DTE rate, stop-bits, parity, and number-of-character bits.	Optional +	6.3.10
GetLineCoding	Requests current DTE rate, stop-bits, parity, and number-of-character bits.	Optional +	6.3.11
SetControlLineState	[V24] signal used to tell the DCE device the DTE device is now present.	Optional	6.3.12
SendBreak	Sends special carrier modulation used to specify [V24] style break.	Optional	6.3.13

* These requests are specific to the Communications Class.
+ An analog modem is strongly recommended to support these requests.

6.2.3 Telephone Control Model Requests

A Telephone Control Model Communications Device uses a Communications Class interface for device management. with a Communications Subclass code of Telephone Control. The only class-specific request codes that are valid for a Communications Class interface with a Communications Class SubClass code of Telephone Control Model are listed in the following Table 12.

Table 12: Requests — Telephone Control Model*

Request	Summary	Req'd/Opt	reference
SetCommFeature	Used to set a unique communications feature, which is normally specific to a particular device.	Optional	6.3.1
GetCommFeature	Returns the current settings for the communications feature.	Optional	6.3.2
ClearCommFeature	Clears the settings for a particular communications feature.	Optional	6.3.3

Request	Summary	Req'd/Opt	reference
SetRingerParms	Configures the ringer for a telephone device.	Optional	6.3.14
GetRingerParms	Gets the current ringer configuration for a telephone device.	Required	6.3.15
SetOperationParms	Configures the operational mode of the telephone.	Optional	6.3.16
GetOperationParms	Gets the current operational mode of the telephone.	Optional	6.3.17
SetLineParms	Allows changing the current state of the line associated with the interface, providing basic call capabilities, such as dialing and answering calls.	Required	6.3.18
GetLineParms	Gets current status of the line.	Required	6.3.19
DialDigits	Dials digits on the network connection.	Required	6.3.20

* These requests are specific to the Communications Class.

6.3 PSTN Subclass Specific Requests

The Communications Interface Class supports the following class-specific requests. This section describes the requests that are specific to the Communications Interface Class PSTN Subclass. These requests are sent over the management element and can apply to different device views as defined by the Communications Class interface codes.

Table 13: Class-Specific Request Codes for PSTN subclasses

Request Code	Value
SET_COMM_FEATURE	02h
GET_COMM_FEATURE	03h
CLEAR_COMM_FEATURE	04h
SET_AUX_LINE_STATE	10h
SET_HOOK_STATE	11h
PULSE_SETUP	12h
SEND_PULSE	13h
SET_PULSE_TIME	14h
RING_AUX_JACK	15h
RESERVED (future use)	16h-1Fh
SET_LINE_CODING	20h
GET_LINE_CODING	21h
SET_CONTROL_LINE_STATE	22h
SEND_BREAK	23h
RESERVED (future use)	24h-2Fh
SET_RINGER_PARMS	30h
GET_RINGER_PARMS	31h
SET_OPERATION_PARMS	32h
GET_OPERATION_PARMS	33h
SET_LINE_PARMS	34h

Request Code	Value
GET_LINE_PARMS	35h
DIAL_DIGITS	36h

6.3.1 SetCommFeature

This request controls the settings for a particular communications feature of a particular target

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	SET_COMM_FEATURE	Feature Selector	Interface	Length of State Data	State

For more information about the defined list of feature selectors per target, see Section 6.3.2, "*GetCommFeature*."

6.3.2 GetCommFeature

This request returns the current settings for the communications feature as selected.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
10100001B	GET_COMM_FEATURE	Feature Selector	Interface	Length of Status Data	Status

Table 14: Communications Feature Selector Codes

Feature selector	Code	Targets	Length of Data	Description
RESERVED	00h	None	None	Reserved for future use
ABSTRACT_STATE	01h	Interface	2	Two bytes of data describing multiplexed state and idle state for this Abstract Model communications device. This selector is only valid for Abstract Control Model.
COUNTRY_SETTING	02h	Interface	2	Country code in hexadecimal format as defined in [ISO3166], release date as specified in offset 3 of the Country Selection Functional Descriptor. This selector is only valid for devices that provide a Country Selection Functional Descriptor, and the value supplied shall appear as supported country in the Country Selection Functional Descriptor

For the ABSTRACT_STATE selector, the following two bytes of data are defined:

Table 15: Feature Status Returned for ABSTRACT_STATE Selector

Bit position	Description

Bit position	Description
D15..D2	RESERVED (Reset to zero)
D1	Data Multiplexed State 1: Enables the multiplexing of call management commands on a Data Class. 0: Disables multiplexing.
D0	Idle Setting 1: All of the endpoints in this interface will not accept data from the host or offer data to the host. This allows the host call management software to synchronize the call management element with other media stream interfaces and endpoints, particularly those associated with a different host entity (such as a voice stream configured as a USB Audio Class device). 0: The endpoints in this interface will continue to accept/offer data.

6.3.3 *ClearCommFeature*

This request controls the settings for a particular communications feature of a particular target, setting the selected feature to its default state. The validity of the feature selectors depends upon the target type of the request.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	CLEAR_ COMM_FEATURE	Feature Selector	Interface	Zero	None

For more information about for the defined list of feature selectors per target, see Section 6.3.2, "*GetCommFeature*."

6.3.4 *SetAuxLineState*

This request is used to connect or disconnect a secondary jack to POTS circuit or CODEC, depending on hook state.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	SET_AUX_LINE_STATE	0 - Disconnect 1 - Connect	Interface	Zero	None

State selector values in the *wValue* field are used to instruct the device to connect or disconnect the secondary phone jack from the POTS circuit or CODEC, depending on hook state. Device will acknowledge the status change.

6.3.5 *SetHookState*

This request is used to set the necessary PSTN line relay code for on-hook, off-hook, and caller ID states.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	SET_HOOK_STATE	Relay Config.	Interface	Zero	None

The *wValue* will instruct the device to configure the necessary relays for going off-hook, on-hook, or into a snooping state for receiving caller ID data.

Table 16: PSTN Relay Configuration Values

Code	Value
ON_HOOK	0000h
OFF_HOOK	0001h
SNOOPING	0002h

6.3.6 *PulseSetup*

This request is used to prepare for a pulse-dialing cycle.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	PULSE_SETUP	Enable/ Disable	Interface	Zero	None

If *wValue* field contains the value FFFFh, the request is being sent to disengage the holding circuit after the dialing sequence has been completed. Any other value in the *wValue* field is meant to prepare the device for a pulse-dialing cycle.

Not all devices require a **PulseSetup** request to disengage the holding circuit after a pulse dialing cycle. The extra request in the dialing cycle is generally required for devices designed to be usable in multiple countries. The device indicates whether the extra request is required or not by setting bit D2 of Direct Line Management Functional Descriptor, in Section 5.3.3.

6.3.7 *SendPulse*

This request is used to generate a specified number of make/break pulse cycles.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	SEND_PULSE	Cycles	Interface	Zero	None

The *wValue* field contains the number of make/break pulse cycles to generate.

6.3.8 *SetPulseTime*

This request sets the timing of the make and break periods for pulse dialing.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	SET_PULSE_TIME	Timing	Interface	Zero	None

The *wValue* field specifies the break time period in the high byte and the make time period in the low byte. The time periods are specified in milliseconds.

6.3.9 RingAuxJack

This request is used to generate a ring signal on a secondary phone jack.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	RING_AUX_JACK	Number of Rings	Interface	Zero	None

The *wValue* field contains the number of ring signals to generate on a secondary phone jack of the device.

6.3.10 SetLineCoding

This request allows the host to specify typical asynchronous line-character formatting properties, which may be required by some applications. This request applies to asynchronous byte stream data class interfaces and endpoints; it also applies to data transfers both from the host to the device and from the device to the host.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	SET_LINE_CODING	Zero	Interface	Size of Structure	Line Coding Structure

For the definition of valid properties, see Table 17, Section 6.3.111.1.1.

6.3.11 GetLineCoding

This request allows the host to find out the currently configured line coding.

bmRequestType	bRequestCode	wValue	wIndex	Wlength	Data
10100001B	GET_LINE_CODING	Zero	Interface	Size of Structure	Line Coding Structure

The line coding properties are defined in the following table:

Table 17: Line Coding Structure

Offset	Field	Size	Value	Description
0	*dwDTERate*	4	Number	Data terminal rate, in bits per second.

Offset	Field	Size	Value	Description
4	*bCharFormat*	1	Number	Stop bits 0 - 1 Stop bit 1 - 1.5 Stop bits 2 - 2 Stop bits
5	*bParityType*	1	Number	Parity 0 - None 1 - Odd 2 - Even 3 - Mark 4 - Space
6	*bDataBits*	1	Number	Data bits (5, 6, 7, 8 or 16).

6.3.12 SetControlLineState

This request generates RS-232/V.24 style control signals.

bmRequestType	bRequestCode	wValue	wIndex	WLength	Data
00100001B	SET_CONTROL_LINE _STATE	Control Signal Bitmap	Interface	Zero	None

Table 18: Control Signal Bitmap Values for SetControlLineState

Bit position	Description
D15..D2	RESERVED (Reset to zero)
D1	Carrier control for half duplex modems. This signal corresponds to V.24 signal 105 and RS-232 signal RTS. 0 - Deactivate carrier 1 - Activate carrier The device ignores the value of this bit when operating in full duplex mode.
D0	Indicates to DCE if DTE is present or not. This signal corresponds to V.24 signal 108/2 and RS-232 signal DTR. 0 - Not Present 1 - Present

6.3.13 SendBreak

This request sends special carrier modulation that generates an RS-232 style break.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	SEND_BREAK	Duration of Break	Interface	Zero	None

The *wValue* field contains the length of time, in milliseconds, of the break signal. If *wValue* contains a value of FFFFh, then the device will send a break until another **SendBreak** request is received with the *wValue* of 0000h.

6.3.14 SetRingerParms

This request configures the ringer for the communications device, either on a global basis (master interface of the union), or on a per-line basis for multiple line devices.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	SET_RINGER_PARMS	Zero	Interface	4	Ringer Configuration bitmap

The command sets up the ringer characteristics for the communications device or for the line. The Ringer Configuration bitmap is defined in the following table:

Table 19: Ringer Configuration Bitmap Values

Bit position	Description
D31	0=A ringer does not exist. 1=A ringer exists. When using the GetRingerParms request to return the Ringer Configuration bitmap, a value of zero for this bit means a ringer does not exist for the addressed element (i.e. device or line).
D30..D16	RESERVED (Reset to zero)
D15..D8	Ringer Volume Setting 0 - Ringer Volume Off 255 - Maximum Ringer Volume
D7..D0	Ringer Pattern Type Selection This corresponds to an internal ringer pattern or sound supported within the device, which could be a distinctive ringing type pattern or a sound effect type ring like a chirping sound, siren sound, etc.

6.3.15 GetRingerParms

This request returns the ringer capabilities of the device and the current status of the device's ringer, including its enabled state and current selection.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
10100001B	GET_RINGER_PARMS	Zero	Interface	4	Ringer Configuration bitmap

This command is typically sent to the master interface of the union. If the ringer for each line can be configured independently, then sending the command to the interface representing a line gets the ringer information for that line. For a description of the returned Ringer Configuration bitmap values, see Table 19.

6.3.16 SetOperationParms

Sets the operational mode for the device, between a simple mode, standalone mode and a host centric mode. Standalone mode means no control from the host; host centric mode means all control is performed from the host.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	SET_OPERATION_ PARMS	Operation Mode	Interface	Zero	None

The *wValue* field is used to specify the mode of operation to be used. Current supported modes of operation are defined in the following table:

Table 20: Operation Mode Values

Operation mode	Description
0	Simple Mode Communications device operates in standalone fashion, and sends no status information to the host and accepts only SetOperationMode commands from host. The device is capable of independent operation..
1	Standalone Mode Communications device operates in standalone fashion, but sends complete status information to the host and will accept any command from the host.
2	Host Centric Mode Communications device is completely controlled by computer but will not perform any communications functions without host control.

In the case of dialing on a phone device, mode 0 would correspond to operating as a typical phone, where the phone would dial out the digits over the phone line. Mode 1 would be the same, except each of the digits dialed by the phone would be reported to the host. In mode 2, the phone would simply report which digits were pushed on the phone keypad to the host, and the host would be responsible for dialing the digits over the phone line.

6.3.17 GetOperationParms

This request gets the current operational mode for the device.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
10100001B	GET_OPERATION_ PARMS	Zero	Interface	2	Operation mode

The returned operation mode value describes the current operational mode of the device, as specified in Table 20.

6.3.18 SetLineParms

This request is used to change the state of the line, corresponding to the interface or master interface of a union to which the command was sent.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	SET_LINE_PARMS	Line State Change	Interface	Length of Data	Data

Some of the commands will require extra data, which will be provided in a packet transmitted during the Data phase. Current line state change values supported are defined in the following table:

Table 21: Line State Change Value Definitions

Line State change value	Description
0000h	Drop the active call on the line.
0001h	Start a new call on the line.
0002h	Apply ringing to the line.
0003h	Remove ringing from the line.
0004h	Switch to a specific call on the line. Data is used to pass a 1-byte call index that identifies the call.

6.3.19 GetLineParms

This request is used to report the state of the line that corresponds to the interface or master interface of a union to which the command was sent.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
10100001B	GET_LINE_PARMS	Zero	Interface	Size of Structure	Line Status Information Structure

This command is issued to the interface or master interface of a union representing a specific line. The returned Line Status Information structure is defined in the following table:

Table 22: Line Status Information Structure

Offset	Field	Size	Value	Description
0	*wLength*	2	Number	Size of this structure, in bytes.
2	*dwRingerBitmap*	4	Bitmap	Ringer Configuration bitmap for this line. For the format of this field, see Table 19.
6	*dwLineState*	4	Bitmap	Defines current state of the line.
10	*dwCallState0*	4	Bitmap	Defines current state of first call on the line.
…	…	…	…	
6 + N*4	*dwCallStateN-1*	4	Bitmap	Defines current state of call N on the line.

The Line State bitmap format provided within the line status information is defined in the following table:

Table 23: Line State Bitmap

Bit position	Description

Bit position	Description
D31	Active flag 0 - No activity on the line. 1 - Line is active (i.e. not idle).
D30..D8	RESERVED (Reset to zero)
D7..D0	Index of active call on this line. Equals 255 if no call exists on the line.

The Call State bitmap format provided within the line status information is defined in the following table:

Table 24: Call State Bitmap

Bit position	Description
D31	Active flag 0 - No active call. 1 - Call is active (i.e., not idle).
D30..D16	RESERVED (Reset to zero)
D15..D8	Call state change value. (For definitions of call state change values, see Table 32.)
D7..D0	Call state value. (For definitions of call state values, see Table 25.)

Table 25: Call State Value Definitions

Call state value	Description
00h	Call is idle.
01h	Typical dial tone.
02h	Interrupted dial tone.
03h	Dialing is in progress.
04h	Ringback. Call state additional data, D15..D8, contains extra information, as defined in Table 32.
05h	Connected. Call state additional data, D15..D8, contains extra information, as defined in Table 32.
06h	Incoming call. Call state additional data, D15..D8, contains extra information, as defined in Table 32.

6.3.20 DialDigits

This request dials the DTMF digits over the specified line.

bmRequestType	bRequestCode	wValue	wIndex	wLength	Data
00100001B	DIAL_ DIGITS	Zero	Interface	Length of Dialing String	Dialing string

The data packet consists of a dialing command, with only the following characters in V.4 supported as being part of the command:

Table 26: Characters in a Dialing Command

Characters	Action
0-9	Dial the specified digit.
* #	Dial the specified DTMF key.
P p	Use pulse dialing for dialing all subsequent digits.
T t	Use tone dialing for dialing all subsequent digits.
!	Insert a hook switch flash into the dialing string.
,	(Comma) Pause the dialing for a fixed period of time defined by the device (usually 2 seconds).
;	(Semicolon) Indicates that more digits will be provided later.
W w	Wait for dial tone or interrupted dial tone before continuing to dial digits.
D d	Hold tone on. All subsequent dialing tones are left on until hold tone off is received.
U u	Hold tone off. All held dialing tones are turned off.

6.4 Management Element Notifications

[USBCDC1.2] defines the common Communications Interface Class notifications that the device uses to notify the host of interface, or endpoint events. This section describes the notifications that are specific to the Communications Interface Class PSTN Subclasses. These requests are sent over the management element and can apply to different device views as defined by the Communications Class interface codes.

The only class-specific request codes that are valid for a Communications Class interface with a Communications Class SubClass codes of DLM, ACM or TCM are listed in the following tables.

6.4.1 Direct Line Control Model Notifications

The following table lists the only notifications which are valid for a Communications Class interface with a Communications Class SubClass code of Direct Line Control Model. The other class-specific notifications not listed in that table, such as RESPONSE_AVAILABLE, are inappropriate for a Direct Line Control Model and shall not be sent by such a device.

Table 27: Notifications — Direct Line Control Model*

Notification	Summary	Req'd/Opt	reference
AuxJackHookState	Indicates hook state of secondary device plugged into the auxiliary phone jack.	Optional	6.5.2
RingDetect	Message to notify host that ring voltage was detected on PSTN interface.	Required	6.5.3

* These notifications are specific to the Communications Class.

6.4.2 Abstract Control Model Notifications

The following table lists the only notifications which are valid for a Communications Class interface with a Communications Class SubClass code of Abstract Control Model. The other class-specific notifications not listed in that table, such as *RingDetect*, are inappropriate for an Abstract Control Model and shall not be sent by such a device.

Table 28: Notifications — Abstract Control Model*

Notification	Summary	Req'd/Opt	reference
NetworkConnection	Notification to host of network connection status.	Optional +	[USBCDC1.2]
ResponseAvailable	Notification to host to issue a GET_ENCAPSULATED_RESPONSE request.	Required	6.5.1
SerialState	Returns the current state of the carrier detect, DSR, break, and ring signal.	Optional +	6.5.4

* These notifications are specific to the Communications Class.
+ An analog modem is strongly recommended to support these requests.

6.4.3 Telephone Control Model Notifications

The following table lists the only notifications which are valid for a Communications Class interface with a Communications Class SubClass code of Telephone Control Model.

Table 29: Notifications — Telephone Control Model*

Notification	Summary	Req'd/Opt	reference
CallStateChange	Reports a state change on a call.	Required	6.5.5
LineStateChange	Reports a state change on a line.	Optional	6.5.6

* These notifications are specific to the Communications Class.

6.5 PSTN Subclass Specific Notifications

Table 30: Class-Specific Notification Codes for PSTN subclasses

Notification Code	Value
NETWORK_CONNECTION	00h
RESPONSE_AVAILABLE	01h
AUX_JACK_HOOK_STATE	08h
RING_DETECT	09h
SERIAL_STATE	20h
CALL_STATE_CHANGE	28h
LINE_STATE_CHANGE	23h

6.5.1 ResponseAvailable

This notification allows the device to notify the host that a response is available. This response can be retrieved with a subsequent *GetEncapsulatedResponse* request.

bmRequestType	bNotification	wValue	wIndex	wLength	Data
10100001B	RESPONSE_AVAILABLE	Zero	Interface	Zero	None

6.5.2 *AuxJackHookState*

This notification indicates the loop has changed on the auxiliary phone interface of the USB device. The secondary or downstream device, which is connected to the auxiliary phone interface, has changed hook states.

bmRequestType	bNotificationCode	wValue	wIndex	wLength	Data
10100001B	AUX_JACK_ HOOK_STATE	0 – On hook 1 – Off hook	Interface	Zero	None

On devices that provide separate control of the auxiliary or downstream phone interface, this notification provides a means of announcing hook state changes of devices plugged into that interface. When the USB device has separate control of this phone interface, it is helpful to know when the secondary device, which is plugged into the auxiliary phone interface, switches between the on-hook/off-hook states.

The *wValue* field returns whether loop current was detected or not detected. Notification is only sent when the state changes.

6.5.3 *RingDetect*

This notification indicates ring voltage on the POTS line interface of the USB device.

bmRequestType	bNotification	wValue	wIndex	wLength	Data
10100001B	RING_DETECT	Zero	Interface	Zero	None

6.5.4 *SerialState*

This notification sends asynchronous notification of UART status.

bmRequestType	bNotification	wValue	wIndex	wLength	Data
10100001B	SERIAL_STATE	Zero	Interface	2	UART State bitmap

The *Data* field is a bitmapped value that contains the current state of carrier detect, transmission carrier, break, ring signal, and device overrun error. These signals are typically found on a UART and are used for communication status reporting. A state is considered enabled if its respective bit is set to 1.

SerialState is used like a real interrupt status register. Once a notification has been sent, the device will reset and re-evaluate the different signals. For the consistent signals like carrier detect or transmission carrier, this will mean another notification will not be generated until there is a state change. For the irregular signals like break, the incoming ring signal, or the overrun error state, this will reset their values to zero and again will not send another notification until their state changes.

Table 31: UART State Bitmap Values

Bits	Field	Description
D15..D7		RESERVED (future use)
D6	*bOverRun*	Received data has been discarded due to overrun in the device.
D5	*bParity*	A parity error has occurred.
D4	*bFraming*	A framing error has occurred.
D3	*bRingSignal*	State of ring signal detection of the device.
D2	*bBreak*	State of break detection mechanism of the device.
D1	*bTxCarrier*	State of transmission carrier. This signal corresponds to V.24 signal 106 and RS-232 signal DSR.
D0	*bRxCarrier*	State of receiver carrier detection mechanism of device. This signal corresponds to V.24 signal 109 and RS-232 signal DCD.

6.5.5 *CallStateChange*

This notification identifies that a change has occurred to the state of a call on the line corresponding to the interface or union for the line.

bmRequestType	**bNotification**	**wValue**	**wIndex**	**wLength**	**Data**
10100001B	CALL_STATE_CHANGE	Call Index and Call State Change Value.	Interface	Length of Data	Variable length structure containing additional information for call state change.

The high-order byte D15-D8 of the *wValue* field will contain the call index, and the low-order byte D7-D0 will contain the call state change value. Not all devices may be capable of reporting all changes of the call state, which should not cause any problems to the higher-level software. All extra data associated with a call state change (*i.e.,* Caller ID data) is returned within the data field. Currently, defined call state values are listed in the following table:

Table 32: Call State Change Value Definitions

Call state change	**Description**
00h	RESERVED
01h	Call has become idle.
02h	Dialing.
03h	Ringback, with an extra byte of data provided to describe the type of ringback signaling 0 = normal 1 = busy 2 = fast busy 3-254 = reserved for future use 255=unknown ringback type

Call state change	Description
04h	Connected, with an extra byte of data provided to describe the type of connection 0 = voice connection 1 = answering machine connection 2 = fax machine connection 3 = data modem connection 4-254 = reserved for future use 255 = unknown connection type
05h	Incoming Call, with the following extra bytes of data (minimum of 4 extra bytes): Extra data byte 0 - Indicates the ringing pattern present as: 0 = ringing pattern 1 (default or normal pattern) 1 = ringing pattern 2 2 = ringing pattern 3 3 – ringing pattern 4 4-255 = reserved for future use Extra data byte 1 - Size of the string (next n bytes) which contains the time (in displayable format) of the incoming call as delivered via Caller ID. The string is not null terminated and is encoded using one character per byte. It is not a UNICODE string. If time is not available then a size of 0 is required as a place setter. Next data byte following number - Size of string (next n bytes) which contains the phone number of calling party as delivered via Caller ID. The string is not null terminated and is encoded using one character per byte. It is not a UNICODE string. If no number is available then a size of 0 is required as a place-setter. Next data byte following name - Size of string (next n bytes) which contains the name of the calling party as delivered via Caller ID. The string is not null terminated and is encoded using one character per byte. It is not a UNICODE string. If no name is available then a size of 0 is required as a place-setter.

6.5.6 *LineStateChange*

This notification identifies that a change has occurred to the state of the line corresponding to the interface or master interface of a union sending the notification message.

bmRequestType	bNotification	wValue	wIndex	wLength	Data
10100001B	LINE_STATE_CHANGE	Value	Interface	Length of Data	Variable Length Line State structure.

Some line state changes may provide extra information, and this information would be provided in the attached extra Line State data structure. Current line state change information are defined in the following table:

Table 33: Line State Change Values

Line State change	Description
0000h	Line has become idle.

Line State change	Description
0001h	Line connected to hold position.
0002h	Hook-switch has gone off hook.
0003h	Hook-switch has gone on hook.

Appendix A: Communications Device Class Examples

This appendix highlights some examples of typical communications device classes. Detailed examples are provided in separate white papers that are not a part of this specification. The latest copies of the white papers can be found at **http://www.usb.org**.

A.1 Basic Telephone

A basic telephone is defined as the household/desktop type phone common to most users. This phone has a handset, keypad, and a 2-wire connection to a local telephone company. In this example, a USB port is added for connecting the phone to the host.

By connecting the phone to a host via the USB, the following functions can be supported:

1. Host monitoring of incoming and outgoing calls.
2. Host-originated dialing of a call.
3. Host recording and playback of voice over the phone line.

This example is not intended to define the computer telephony application features or user interface. The example demonstrates how the USB Communications Interface Class protocol can be used to identify, control, and monitor a telephony device.

A.2 Modem

For compatibility with legacy computer software and to facilitate the development of generic drivers, a USB modem should conform to the ITU-T V.250 standard. For common extended functions, the following standards are recommended:

- Modem identification: ITU V.250 +G commands
- Data modems: ITU V.250 (modulation, error control, data compression)
- Data modems: ITU V.80 In-band DCE control and synchronous data modes for asynchronous DTE
- Fax modems: ITU T.31 or T.32 +F commands (or TIA equivalents)
- Voice modems: TIA IS-101 +V commands
- General wireless modems: PCCA STD-101 +W commands (TIA 678)
- Analog cellular modems: PCCA STD-101 Annex I (TIA 678 Annex C)
- Digital cellular modems: TIA IS-707/CS-10017-0, TIA IS-135, GSM 7.07 or 3GPP 27.07 +C commands.
- Text phone modems: V.250 +MV18 commands.

For a complete list of standard modem command sets, see the ITU Supplement to V.250.

Note: A USB modem may provide means to accommodate common functions performed on a 16550 UART. For more information, see Section 3.2.2.1, "Abstract Control Model Serial Emulation."

Appendix B: Sample Configurations

B.1 Basic Telephony Configurations

This section defines three examples of telephony configurations: a basic telephone, a telephone with keypad, and a combination telephone with analog modem. The minimum requirement for this type of device is a configuration with a single Communications Class interface. If you wish to support a standard telephone keypad, you would require an additional Human Interface Device Class interface to support the keypad. The most basic audio-capable telephone is constructed by adding an Audio interface for audio transmission and reception. A more advanced configuration could optionally have local Audio interfaces connected to the handset and microphone/speaker, and one Data interface. In this case, the Data interface could be the raw linear data as sampled from the network. The responsibility for demodulation and interpretation of this data would lie within the host at the application level (i.e., processor-based modem).

Table 34: Telephone Configurations

Example configuration	Interface (class code)	Reference section	Description
Basic telephone	Communications Class	[USBCDC1.2]	Device management and call management. Consisting of a management element and a notification element.
Telephone with keypad	Communications Class	[USBCDC1.2]	Device management and call management. Consisting of a management element and a notification element.
	HID Class	[USBHID1.11]	I/O for a keypad interface.
Audio/data telephone	Communications Class	[USBCDC1.2]	Device management and call management. Consisting of a management element and a notification element.
	Audio Class	[USBAUD2.0]	I/O for uncompressed audio.
	Data Class	[USBCDC1.2]	Demodulated modem data.

A communications device that supports audio type media streams over its interfaces can use the selected Audio interface to indicate which voice or audio coding formats it supports (for example, IS-101 for voice modems).

B.2 Modem Configurations

This section defines three examples of modem configurations: legacy modem, DSVD modem, and multimedia modem. The first configuration covers legacy modems for data, fax, and voice. The second configuration covers SVD modems, such as ASVD (ITU V.61) and DSVD (ITU V.70). The third configuration covers multimedia modems that would be used in ITU H.324 situation.

Table 35: Example Modem Configurations

Example configuration	Interface (class code)	Reference section	Description

Example configuration	Interface (class code)	Reference section	Description
Legacy modem	Communications Class	[USBCDC1.2]	Device management and call management. Consisting of a management element and a notification element.
	Data Class	[USBCDC1.2]	Demodulated modem data.
SVD modem	Communications Class	[USBCDC1.2]	Device management and call management. Consisting of a management element and a notification element.
	Data Class	[USBCDC1.2]	Demodulated modem data.
	Audio Class	[USBAUD2.0]	I/O for uncompressed audio.
Multimedia modem	Communications Class	[USBCDC1.2]	Device management and call management. Consisting of a management element and a notification element.
	Data Class	[USBCDC1.2]	Demodulated modem data.
	Audio Class	[USBAUD2.0]	I/O for uncompressed audio.
	Video Class	[USBVID1.1]	I/O for video (for example, H.263).

Most of today’s modem type devices — single-media or multimedia — contain various types of media processing resources such as compression engines (for example, V.42bis) or audio/video CODECs. Given the projections of increased processing power for future host systems and the availability of appropriate media transport to and from the host (i.e., the USB), it is likely that various models of media processing will emerge that do not rely solely on the device for these resources. In this case, where media processing resources are located arbitrarily within the system (for example, V.42bis on the host and V.34 on the device), interface choices for media types could vary. For example, if a device developer chose to include an MPEG2 CODEC in a device, a bi-directional isochronous interface may be more appropriate for transport of the video stream. Conversely, if the CODEC is not in the USB device, a bi-directional bulk interface would be more appropriate.

The processing required for some types of media streams is asymmetrical in nature. For example, MPEG2 decompression is trivial by today’s standards, although compression requires substantial processing resources. In light of this fact, it may be appropriate to configure an interface with the appropriate asymmetry. Continuing the MPEG example, a device that relies on host-based decompression and device-based compression would choose an interface that consists of an isochronous endpoint for video in the host-to-device direction and a bulk endpoint for the device-to-host direction.

An example of the bandwidth implications of device in contrast with host-based media stream processing is outlined in the following list. USB bandwidth is expressed in bytes per millisecond (B/ms). For example, typical performance of V.42bis is 3-4:1 on data streams, 33.6 kb/s V.34 data could unpack to 16.8 B/ms.

Similar bandwidth issues are relevant to audio and video, but they will be handled by the video or audio interfaces rather than Communications Class interfaces.

- G.723 voice CODEC (5.5 - 6.5 Kb/s) could be unpacked to 11 kHz audio by the modem (22 B/ms).
- H.263 compressed video is in the 2.5 B/ms range; but if H.263 is decompressed, a typical bandwidth is approximately 96 B/ms.