Universal Serial Bus
Device Class Definition
For
Content Security Devices

INTEL CORPORATION
MICROSOFT CORPORATION
PHILIPS ITCL-USA

Revision 1.0

22 August 2000

Revision History

Revision	Date	Filename	Author	Description
1.0	08/22/2000			Correct Copyright statement
1.0	03/15/2000			Promotion to 1.0 at USB DWG
.9a	01/26/2000			Added a CS_GENERAL Descriptor in order to incorporate CS version tracking. Renumber sections.
.9	01/25/2000			USB DWG promotion to .9
.8e	12/23/1999			Align notification description to USB practices
.8d	12/22/1999			Device can only change CSM by using notification to request change.
.8c	12/18/1999			No simultaneous CS coverage of any individual endpoint. Add logical unit addressing capabilities and alternate channel.
.8b	12/17/1999			Change from 12/10/1999 USB DWG minor typos, Major:drop GUID from USB documentation, add another descriptor for modified CSMs. Clarification and completions
.8	11/1/1999			Promoted to .8 at 10/22/1999 USB DWG meeting, Corrected descriptions of bmMaxPacketSize, IdVendor, IdProduct, and IdSerialNumber fields of table 5-1
0.7f	9/27/1999			Separated appendices into individual USB CSM specifications, corrected Get request to have data field, reformatted text in all CS specs,
0.7e	9/1/1999			Added statement that only one CS Interface should be declared per Configuration. Changed bNumendpoint in CS Interface descriptor to be CSM Defined. Changed low byte of wIndex (Table 6-1) to be CSM defined. Added CSM defined Request Codes to section A.3.
0.7d	8/25/99			Cleaned up Management Overview and Notifications sections.
0.7c	8/19/1999			Combine edits from contributors Mark Williams, Billy Brackenridge, Geert Knapen, and Michael Andre. Use consistent terminology CSM (Content Security

Revision	Date	Filename	Author	Description
				Method), Extend to use Full 128 bit GUIDs. Start CSM numbering with 1.
0.7b	8/4/99			Put in improvements to sections 5 and 6 suggested by Steve McGowan, fixed some inconsistent uses of constant label values in Appendix A, renamed “Baseline Authentication Protocol” to “Basic Authentication Protocol” in Appendix B, reinstated first-draft architecture diagram in section 4 until better diagram is provided.
0.7	7/29/99			Made changes agreed to at face-to-face meeting of authors on 7/29/99.
0.6bx	7/26/99			Added recommendation that Authentication devices be built with short, tethered cables; added definition of requests for baseline authentication protocol; added criteria that measure a good baseline protocol; did miscellaneous small edits to improve clarity; changed Content Protection Method term to Content Security Method (except in Appendix B and C)
0.6b	7/19/99			Eliminated dependency on Common Class Logical Devices. Limited scope to devices used as sinks, to authentication protocol and not data transfer. Incorporated concept of one native authentication protocol and an open-ended number of alternative authentication protocols in the device. Took first step towards the terminology definitions that will be used consistently in the specification. Established a Class Requirements section by which completeness of evolving versions of the specification can be judged.
0.6a	7/15/99			Merged Intel and Philips proposals, deleting nothing.

Contributors

Michael Andre	Intel
John Howard	Intel
Steve McGowan	Intel
Tom Jones	Microsoft
Mark Williams	Microsoft
Ervin Peretz	Microsoft
Billy Brackenridge	Microsoft
Geert Knapen	Philips

Please send comments via electronic mail to michael.andre@intel.com

Table of Contents

List of Tables

List of Figures

1 Introduction

The need for the protected and controlled distribution of digital content is the basis of the USB Content Security Interface (CSI) described in this specification.

1.1 Purpose

The purpose of the USB Security Class is to specify a common set of USB data transport requests and descriptors necessary to support the various Content Security Methods (CSM). CSMs may use some or all of the services detailed in this specification. A CSM defines the USB support given to a particular Content Protection Method (CPM). The intent is that each CSM will be detailed in a separate but associated USB CSM specification.

1.2 Scope

This specification details the USB CS functionality, requests, and descriptors that support the various CSMs. CSMs are listed in Appendix A along with the corresponding CSM identification number assignment. Each CSM is detailed in a separate but associated Content Security Method Specification that contains data detailing its USB implementation. CSMs are added to the CS list by vote of USB CSWG and are subject to the USB specification promotion process.

The basic services needed to support these various CSMs are shown in the following bulleted list.

- Activating/Deactivating a particular CSM.
- Associating/Disassociating a CSM to a data transport channel (Isochronous, Bulk, Control).
- A notification service that allows either the Host or Device or both to initiate asynchronous CSM related communication.
- Uniquely identify each CSM for Host driver support.

1.3 Related Documents

- *Universal Serial Bus Specification*, Version 1.1 (also referred to as the *USB Specification*). In particular, see Section 9, “USB Device Framework.”
- *Universal Serial Bus Common Class Specification, Revision 1.0*.

1.4 Terms and Abbreviations

Content Security Device	Any USB device that contains a Content Security Interface.
Channel	A logical path over which secure data can be transmitted or received.
Content provider	The owner of the content
CPM	Content Protect Method, refers to a content provider protection scheme
CSI	Content Security Interface
CSM	Content Security Method
CSMV	Content Security Method Variant
Sink	The target of secure data transfers
Source	The source of secure data transfers

2 Management Overview

Protected Content typically refers to copyrighted content. Content Protection Methods (CPM) have been developed for the controlled distribution of protect content. The Content Security Class was initiated to support CPM protocol exchange and protected content transport over the USB.

The CS class provides a common set of extendable USB services, descriptors, and requests that are detailed in this document. To support and provide for each CPM's specific USB needs a corresponding USB Content Security Method (CSM) maybe needed. Each CSM describes the use of the common CS services and defines additional CPM specific USB transport services, descriptors, and requests.

The Content Security Interface (CSI) is either one of many interfaces that comprise a complex USB or it maybe the only interface as in an authentication dongle.

The intended operation of the CSI when transferring Protected Content is for the CSI to process and transport protected content as prescribed by a CPM and its associated support CSM. There is no need for additional Content Security specific endpoints to transport the protected content as the CSI uses the existing device's data transport channels to move the protected content. This is done with minimal impact to existing classes and interfaces.

There are two common requests **Get_Channel_Settings** and **Set_Channel_Settings.** The Get request is used to determine the CSM assigned to a channel. The USB device may return a CSM value of zero indicating that there is no active CSM on the specified channel. The Set request is used to assign a CSM to a channel or to deactivate a CSM assigned to a channel.

The definition and use of additional USB requests and Interrupts is strictly CSM dependent and is detailed in the associated CSM specification. For example, a CSM might need additional USB requests to exchange commands and responses between Host and Device. These requests may in turn support authentication, public-key formation, and System Renewal Messages.

The CSM will also detail USB Class interaction, notification format values (if used), and the format of content transferred over the associated USB data channel (Isochronous, Bulk, or Control).

3 Functional Characteristics

Typically, the Content Security Interface is one part of a complex USB device. A complex USB device is one that has several interfaces. The Content Security Interface maybe the only interface as in the case of an authentication dongle.

3.1.1 Control Endpoint

The Content Security Interface is addressed through the control endpoint.

3.1.2 Bulk and Isochronous Endpoints

There is no need for additional Content Security specific endpoints to transport the protected content as the CSI uses the existing device's data transport pipes to move protected data.

3.1.3 Interrupt Endpoint

The optional Interrupt endpoints in the Content Security Interface provide asynchronous notification of CSM-related events from a device to the Host or allow a Host to provide timely control requests to a device.

3.2 Content Security Methods

Protected content is distributed with a requirement that it be protected by a particular CPM. This content protection methodology allows compliant devices to recognize and correctly process and/or transport the protected content. CSMs define the USB services, descriptors, and requests need to support a given CPM.

3.3 Channels

In this specification, a channel is a logic construct representing a relationship between an interface or endpoint and one or more CSMs of which only one can be active at anyone time.

4 Operational Model

4.1 Operation Model

An example of an audio application is used to describe an operational model in the illustration below. Say protected content is directed to a USB device (a pair of Audio speakers). In this case, the Host would use a **Set_Channel_Settings** request to the USB Device to activate the necessary CSM and prepare the USB Device to receive and correctly process the protected content. In the illustration, actual data flow is shown with black arrows and logical data flow is shown with gray arrows.

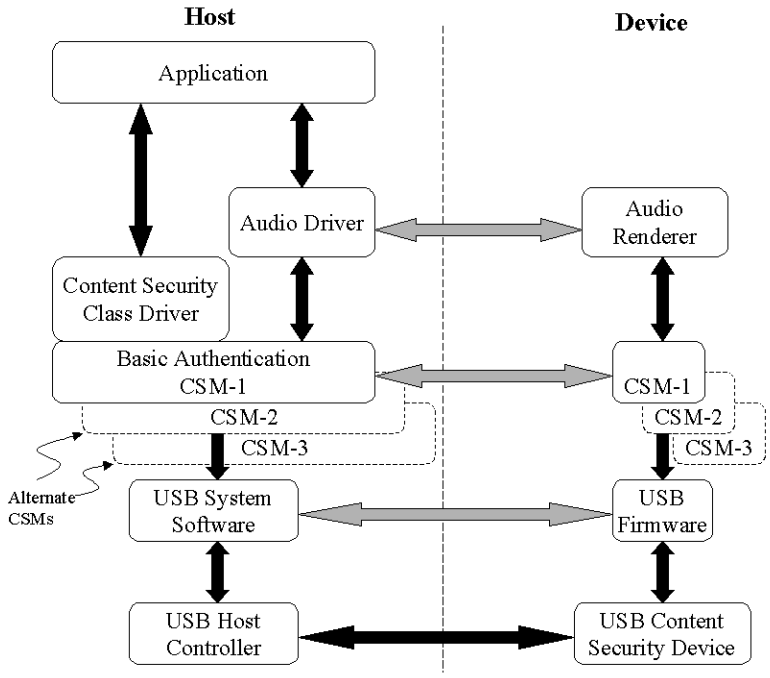

Figure 4-1 CS Audio Example

In the case where the USB Device is directed to send protected content to the Host, the USB Device would notify the Host and the Host would then respond with a **Get_Channel_Settings** request to the determine the CSM and make the corresponding preparations for protected content transport.

5 Descriptors

This section defines all the Content Security class-specific interface and endpoint descriptors. For completeness, the standard device, configuration, interface and endpoint descriptors are also included in this specification.

5.1 Device Descriptor

The device descriptor for a Content Security Device must indicate that class information is to be found at the interface level. Therefore, the **bDeviceClass** field of the device descriptor must contain zero to force the host enumeration software to search all the Interface descriptors on this device. This ensures that the Content Security Interface descriptor, with the **bInterfaceClass** field value set to CONTENT_SECURITY, is found by the host enumeration software, and that this device is known to the host to have a Content Security Interface. The **bDeviceSubClass** and **bDeviceProtocol** fields in the Device descriptor must also be set to zero.

There is no class-specific device descriptor for a Content Security device.

Table 5-1: Standard Device Descriptor on a Security Device

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor, in bytes
1	*bDescriptorType*	1	Constant	DEVICE descriptor type
2	*bcdUSB*	2	BCD	USB Specification Release Number in Binary-Coded Decimal (i.e., 2.10 is 210H). This field identifies the release of the USB Specification with which the device and its descriptors are compliant.
4	*bDeviceClass*	1	Number	Must be set to 0.
5	*bDeviceSubClass*	1	Number	Must be set to 0.
6	*bDeviceProtocol*	1	Number	Must be set to 0.
7	*bmMaxPacketSize0*	1	Number	Maximum packet size for endpoint 0 (only 8, 16, 32, or 64 are valid).
8	*idVendor*	2	ID	Vendor ID (assigned by USB). Must be set to the Vendor ID value assigned to the manufacturer of the Security Device.
10	*idProduct*	2	ID	Product ID. Must be set to a Product ID that uniquely identifies this type of device by the manufacturer of the Security Device.
12	*bcdDevice*	2	BCD	Device release number in binary-coded decimal.
14	*iManufacturer*	1	Index	Index of string descriptor describing manufacturer.
15	*iProduct*	1	Index	Index of string descriptor describing product.

Offset	Field	Size	Value	Description
16	*iSerialNumber*	1	Index	Index of string descriptor describing the device serial number.
17	*bNumConfigurations*	1	Number	Number of possible configurations

5.2 Configuration Descriptor

The configuration descriptor for a Content Security Device is identical to the standard configuration descriptor defined in Section 9.6.2, "Configuration" of the *USB Specification*. There is no class-specific configuration descriptor.

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	CONFIGURATION
2	*wTotalLength*	2	Number	Total length of data returned for this configuration. Includes the combined length of all descriptors (configuration, interface, endpoint, and class or vendor specific) returned for this configuration.
4	*bNumInterfaces*	1	Number	Number of interfaces supported by this configuration
5	*bConfigurationValue*	1	Number	Value to use as an argument to Set Configuration to select this configuration
6	*iConfiguration*	1	Index	Index of string descriptor describing this configuration
7	*bmAttributes*	1	Bitmap	Configuration characteristics D7 Bus Powered D6 Self Powered D5 Remote Wakeup D4..0 Reserved (reset to 0) A device configuration that uses power from the bus and a local source sets both D7 and D6. The actual power source at runtime may be determined using the Get Status device request. If a device configuration supports remote wakeup, D5 is set to 1.
Offset	**Field**	**Size**	**Value**	**Description**

8	*MaxPower*	1	mA	Maximum power consumption of USB device from the bus in this specific configuration when the device is fully operational. Expressed in 2 mA units (i.e., 50 = 100 mA). Note: A device configuration reports whether the configuration is bus-powered or self-powered. Device status reports whether the device is currently self-powered. If a device is disconnected from its external power source, it updates device status to indicate that it is no longer self-powered. A device may not increase its power draw from the bus, when it loses its external power source, beyond the amount reported by its configuration. If a device can continue to operate when disconnected from its external power source, it continues to do so. If the device cannot continue to operate, it fails operations it can no longer support. Host software may determine the cause of the failure by checking the status and noting the loss of the device's power source.

5.3 Content Security Interface Descriptors

The Content Security Interface descriptors fully characterize the security capabilities of the device. The standard Content Security Interface descriptor essentially characterizes the device as a Content Security Device and the class-specific interface descriptor fully describes the security capabilities of the particular device by, for example

- Identifying the CSM or CSMs implemented on the device
- Providing CSM implementation details
- Associating secure content transfer channels with endpoints on other interfaces on the device

5.3.1 Standard Content Security Interface Descriptor

The standard Content Security Interface descriptor is identical to the standard interface descriptor defined in Section 9.6.3, "Interface" of the *USB Specification*, with the **bInterface** field set to the Security Class code and the **bInterfaceSubClass** and **bInterfaceProtocol** fields set to 0. The value of the Security Class code is specified in Appendix A, "Content Security Device Class Codes".

Table 5-2: Standard Content Security Interface Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor, in bytes: 9
1	*bDescriptorType*	1	Constant	INTERFACE descriptor type
2	*bInterfaceNumber*	1	Number	Number of interface. A zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
3	*bAlternateSetting*	1	Number	Value used to select an alternate setting for the interface identified in the prior field.
4	*bNumEndpoints*	1	Number	Number of endpoints used by this interface (excluding endpoint 0) that are CSM dependent.
5	*bInterfaceClass*	1	Class	Content Security Interface Class code (assigned by the USB).
6	*bInterfaceSubClass*	1	Subclass	Not used. Must be set to 0
7	*bInterfaceProtocol*	1	Protocol	Not used. Must be set to 0.
8	*iInterface*	1	Index	Index of a string descriptor that describes this interface.

5.3.2 Class-Specific Interface Descriptors

The class-specific interface descriptors provide all the information that is needed to fully describe the characteristics and behavior of the Content Security Interface on a Security Device.

There are four types of class-specific interface descriptors for the Security class:

- **CS_GENERAL descriptor:** identifies the Content Security Interface version number.
- **Channel descriptor:** identifies one or more Content Security Methods the device can use on a channel. CS devices must contain at least one Channel Descriptor and each Channel Descriptor identifies at least one CSM.
- **Content Security Method descriptor:** describes one CSM implemented on a device. CS devices must contain at least one CSM descriptor.
- **Content Security Method Variant descriptor:** describes a variant of the associated CSM

5.3.2.1 CS_GENERAL Descriptor

This descriptor serves to identify the Content Security Interface Version number.

Table 5-3 CS_GENERAL Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Byte length of this descriptor.
1	*bDescriptorType*	1	Constant	CS_GENERAL (for value of this constant, see Appendix A)
2	*bcdVersion*	2	BCD	Content Security Interface Version number in Binary-Coded Decimal (i.e. version 2.10 is 0x0210).

5.3.2.2 Channel Descriptor

This section defines the class-specific Channel Descriptor for the Security Device class. In this specification, a channel is a logic construct representing a relationship between an interface or endpoint and one or more CSMs of which only one can be active at anyone time.

Each channel descriptor defines one channel on the device. The descriptor associates the channel with an interface number (defined in the **bInterfaceNumber** field of an Interface descriptor on the device) or to an endpoint address. An endpoint address indicates a specific pipe on the device that the Content Security Interface provides CS services over and is encoded in the Channel descriptor as an endpoint number along with its direction (IN or OUT).

Within a single configuration setting, two Channel Descriptors cannot overlap hardware resources that they protect. Specifically, two Channel Descriptors cannot reference the same interface or endpoint in their bmAttributes/bRecipient.

Table 5-4: Channel Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Byte length of this descriptor.
1	*bDescriptorType*	1	Constant	CHANNEL_DESCRIPTOR (for value of this constant, see Appendix A)
2	*bChannelID*	1	Number	Number of the Channel, must be a zero-based value that is unique across the device.
3	*bmAttributes*	1	Bitmap	D7..D5: Reserved and set to zero D4..D0: Recipient Type 0 = Not used 1 = Interface 2 = Endpoint 3..31 = Reserved

4	*bRecipient*	1	Number	Identifier of the target recipient. If the Recipient type field of bmAttributes = 1, then the value in the bRecipient field is the **bInterfaceNumber**. If the Recipient type field of bmAttributes = 2, then the value in the bRecipient field is an endpoint address, where: D7: Direction 0 = OUT 1 = IN D6..D4: Reserved and set to zero D3..D0: Endpoint number
5	*bRecipientAlt*	1	Number	If bmAttributes is set to interface then bRecipientAlt is the alternate setting for the interface to which this channel applies. Else, bRecipientAlt is set to 0.
6	*bRecipientLogicalUnit*	1	Number	If bmAttributes is set to Interface then bRecipientLogicalUnit is the logical unit within the protected interface to which this channel applies. Else, bRecipientLogicalUnit is set to 0. The definition of a LU is dependent upon the interface being protected. If the LU implemented then b is set to zero.
7	*bMethod[0]*	1	Number	Index of a class-specific CSM descriptor that describes one of the Content Security Methods offered by the device, must be a one-based value that is unique across the device. The value of 0 (zero) is reserved and must not be used in this field. See Table A-5.
8	*bMethodVariant[0]*	1	Number	Zero-based value that points to a CSM Variant descriptor. If there is no CSM Variant descriptor than it has the value of zero.
-	-	-		
7 + 2(N-1)	*bMethod[N-1]*	1	Number	
8 + 2(N-1)	*bMethodVariant[N-1]*	1	Number	

Each **bMethod** has an associated **bMethodVariant** field. If there is no variant the value of the **bMethodVariant** field is zero.

5.3.2.3 Content Security Method (CSM) Descriptor

This section defines the class-specific Content Security Method (CSM) descriptor. A Security device must report one CSM descriptor for each Content Security Method implemented by the device.

Table 5-5: CSM Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Byte length of this descriptor.
1	*bDescriptorType*	1	Constant	CSM_DESCRIPTOR (for value of this constant, see Appendix A)
2	*bMethodID*	1	Number	Index of a class-specific CSM descriptor that describes one of the Content Security Methods offered by the device, must be a one-based value that is unique across the device. The value of 0 (zero) is reserved and must not be used in this field. See Table A-5.
3	*iCSMDescriptor*	1	Index	Index of string descriptor that describes the Content Security Method.
4	*bcdVersion*	2	BCD	CSM Descriptor Version number in Binary-Coded Decimal (i.e. version 2.10 is 0x0210).
6	*CSMData*	N		Optional field(s) that provides device-specific implementation details for the CSM identified by the **bMethodID** field.

5.3.2.4 Content Security Method Variant Descriptor

The use of this descriptor is CSM dependent. The CSMV has at least two fields **bLength** and **bDescriptorType** and any subsequent fields are defined in the CSM that requires a CSMV descriptor.

Table 5-6 CSMV Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Byte length of this descriptor.
1	*bDescriptorType*	1	Constant	CSMV_DESCRIPTOR (for value of this constant, see Appendix A)
2	*CSMVData*	N		Optional field(s) that provides device-specific implementation details for the CSM Variant

5.3.3 Content Security Descriptor Topology

All the CS related descriptors on the device are available to the host after successful completion of a standard GET_DESCRIPTOR (configuration) request. All Channel, CSM, and CSMV descriptors follow the standard Content Security Interface descriptor and precede any Content Security Interface endpoint descriptors or any other interface descriptors.

5.4 Content Security Endpoint Descriptors

The Content Security Interface on a device is addressed through the default Control endpoint (endpoint 0), which every device must implement. However, Interrupt IN and OUT endpoints, which are optional for a Content Security device can be required for one or more of the CSMs implemented by the device.

If implemented, a Content Security Interface Interrupt Endpoint Descriptor is a standard Endpoint descriptor with the **bmAttributes** field set to Interrupt.

5.4.1 Content Security Interface Notification Format

To allow notifications from multiple channels on a device to share an interrupt endpoint, a standard Content Security Notification format is defined. The device initiates a Security notification by executing an Interrupt IN data transfer. The host may send a minimum latency Security notification to a device by executing an Interrupt OUT data transfer. The CS notification format is described in the following table.

Table 5-7: CS Notification Format

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Byte length of this notification.
1	*bChannel*	1	Number	ID of channel that generated the notification.
2	*bNotification*	1	Number	Identifies the type of notification. The value assignments are defined in appendix A.
3	*Data*	n	data	CSM-defined data for Notification

5.4.2 Change_Channel_Settings Notification

The **Change_Channel_Settings** notification is used to request that a CSM is activated and linked to the given channel. The Host upon receiving this notification will issue a **Change_Channel_Settings** request that actually causes the channel settings to be changed. Note a CS channel setting is only changed by a **Set_Channel_Settings** request.

If a device supports the **Change_Channel_Settings** notification, it must implement the CS notification Service. Otherwise the use of the notification service is optional and CSM dependent.

Table 5-8 Change_Channel_Settings Notification

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Byte length of this notification.
1	*bChannel*	1	Number	ID of channel that generated the notification.
2	*bNotification*	1	Number	Change_Channel_Settings (0x01)
3	*bData*	1	data	bData contains the CSM number that the USB device wants activated on the given bChannel.

6 Requests

This section specifies the requests a Content Security device can receive from the host at the Content Security Interface.

6.1 Standard Requests

The Content Security Device Class supports the standard requests described in Section 9, "USB Device Framework," of the *USB Specification*. The Content Security Device Class places no specific requirements on the values for the standard requests.

6.2 Class-Specific Requests

All Content Security class-specific requests are directed to the Content Security Interface. The basic Content Security class-specific request layout is the same as defined in Section 9.3 of the *USB Specification, Version 1.1*. The meaning of each request field is defined in the next paragraphs.

For most Content Security class-specific requests, the content of all the request fields except **bmRequestType** and **wIndex** is CSM specific and defined in the Appendices of this document. For more information, see the section "Class Specific Request Layout."

Two **bRequest** field values that all Content Security Class devices must accept, in addition to the requests specific to each of the Content Security Methods the device implements, are:

- **Get_Channel_Settings**
- **Set_Channel_Settings**

The host uses the **Get_Channel_Settings** request to determine the CSM associated to an interface or endpoint (channel) and **Set_Channel_Settings** request assigns a CSM to a channel.

Devices receiving unsupported requests must stall the control pipe upon receipt of an unsupported request.

6.2.1 Class Specific Request Layout

This section details the general structure of the Content Security Class-specific requests.

6.2.1.1 Content Security Method-Specific Requests

The basic Content Security class-specific request layout is the same as defined in Section 9.3 of the *USB Specification, Version 1.1*. The meaning of each request field for Content Security Method-specific requests is defined in Table 6-1.

Table 6-1: Content Security Method-Specific Request Field Definitions

Offset	Field	Size	Value	Description
0	*bmRequestType*	1	Bitmap	Characteristics of request: D7: Data transfer direction 0 = Host-to-device 1 = Device-to-host D6…5: Type 1 = Class D4…0: Recipient 1 = Interface
1	*bRequest*	1	Value	Specific request. See requests in this document. CSM's are allowed to define additional requests as needed.
2	*wValue*	2	Value	Word-sized field that is bRequest dependent.
4	*wIndex*	2	Value	Word-sized field that is bRequest dependent.
6	*wLength*	2	Count	Word-sized field that specifies byte length of associated data field.

6.2.1.2 Get Channel Settings

The **GET_CHANNEL_SETTINGS** request returns the ID of the Content Security Method (CSM) currently selected for a specified channel.

Table 6-2: Get_Channel_Settings Request Field Definitions

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1 01 00001B	*GET_CHANNEL_SETTINGS*	0	HByte: Channel ID LByte: the number of CSI	2	HByte: 0 LByte: Index to CSM currently running

The **bmRequestType** is divided into 3 sub fields. Bit D7 specifies direction of data transfer and is from device to host when (D7 = 0b1). This is a class specific request (D6..D5 = 1b01) directed to the Content Security Interface (D4..D0 = 0b00001).

The values for **bRequest** are defined in Appendix A.

6.2.1.3 Set Channel Setting

The **SET_CHANNEL_SETTINGS** request sets the current CSM for a channel and is the only method for assigning a CSM to an interface or endpoint.

Table 6-3: Set_Channel_Settings Request Field Definitions

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0 01 00001B	*SET_CHANNEL_SETTINGS*	HByte: 0 Reserved LByte – Index to CSM descriptor for CSM currently running on channel.	HByte: Channel ID LByte: : the number of CSI	0	None

The **bmRequestType** is divided into 3 sub fields. Bit D7 specifies direction of data transfer in data stage, if any, and is from host to device when (D7 = 0b0). That this is a class specific request (D6..D5 = 0b01) directed to either the Content Security Interface (D4..D0 = 0b00001).

The values for **bRequest** are defined in Appendix A.

The LByte of the **wValue** is set to zero to indicate that there is no active CSM operating on the specified interface or endpoint (Channel ID).

A. Content Security Value Assignments

A.1 Content Security Interface Class Code

Table A-1 Content Security Interface Class Code

Content Security Interface Class Code	Value
CONTENT SECURITY	0x0D

A.2 Content Security Class-Specific Descriptor Codes

Table A-2 Content Security Class-Specific Descriptor

Descriptor Code	Value
CS_GENERAL_DESCRIPTOR	0x21
CHANNEL_DESCRIPTOR	0x22
CSM_DESCRIPTOR	0x23
CSMV_DESCRIPTOR	0x24
Reserved for future extension	0x25..0x3F

A.3 Content Security Class-Specific Request Codes

Table A-3 Content Security Class-Specific Request Codes

Request Code	Value
Undefined	0x00
GET_CHANNEL_SETTINGS	0x01
SET_CHANNEL_SETTINGS	0x02
Reserved for future extension	0x03 – 0x7F
CSM Defined	0x80 –0xFF

A.4 Content Security Notification Values

Table A-4 Content Security Notification Value Assignments

Request Code	Value
Undefined	0x00
Change_Channel_Settings	0x01
Reserved for future extension	0x02 – 0x7F
CSM Defined	0x80 –0xFF

A.5 CSM bMethod Value Assignments

Table A-5 CSM bMethod Value Assignments

CSM Number	CONTENT SECURITY METHOD SPECIFICATION	bMethod
CSM-1	BASIC AUTHENTICATION PROTOCOL	0x01
CSM-2	USB DIGITAL TRANMISSION CONTENT PROTECTION IMPLEMENTATION	0X02
CSM-3	OPEN COPY PROTECTION SYSTEM	0X03
CSM-4	ELLIPTIC CURVE CONTENT PROTECTION PROTOCOL	0X04