

Technical Requirements

1. SDB CableCARD Resource Definition

This section defines the SDB Resource as an extension to [SCTE-28]. The iDCR Host shall implement a SDB Support resource. The SDB Support application in the CableCARD shall create a session to the SDB Support resource in the iDCR Host whenever there are SDB channels in the VCT.

Resource	Class	Type	Version	Resource Identifier
SDB Support	TBD	1	1	0x TBD

The SDB Support resource shall support the following APDUs:

Apdu_tag	Tag Value	Resource	Direction Host ↔ Card
sdb_tune_req()	0x TBD	SDB Support	→
sdb_tune_cnf()	0x TBD	SDB Support	←
sdb_tune_cancel()	0x TBD	SDB Support	→
sdb_tune_query()	0x TBD	SDB Support	←
sdb_tune_query_cnf()	0x TBD	SDB Support	→

1.1. SDB_tune_req and SDB_tune_cnf APDU Syntax

The Host shall issue a sdb_tune_req() APDU when any of its tuners needs to tune to a channel that has been identified as potentially being a SDB channel.

Table 1 sdb_tune_req() APDU Syntax

Syntax	No. of Bits	Mnemonic
Sdb_tune_req() { Sdb_tune_req_tag length_field() channel_num }	24	uimsbf
	12	uimsbf range 0-4095

sdb_tune_req_tag

0xTBD

channel_num

The virtual channel number of the channel that the tuner wants to tune to

The CableCARD shall respond to the sdb_tune_req() APDU with a sdb_tune_cnf() APDU containing the information needed by the Host to tune to the channel. The CableCARD shall also issue a new sdb_tune_cnf() APDU if any of the SDB channel information changes before the Host has issued a sdb_tune_cancel() APDU. Therefore the session will remain open during this time.

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

If the CableCARD issues a sdb_tune_cnf() APDU with request_status = 0x01 "Please Wait", it must eventually issue another sdb_tune_cnf() APDU with request_status other than 0x01.

The CableCARD may issue a sdb_tune_cnf() APDU with request_status = 0x03 "Request denied" to indicate that the channel is no longer available and has been removed from the plant.

Table 2 sdb_tune_cnf() APDU Syntax

Syntax	No. of Bits	Mnemonic
Sdb_tune_cnf() {		
Sdb_tune_cnf_tag	24	uimsbf
length_field()		
channel_num	12	uimsbf range 0-4095
request_status	8	uimsbf
transport_type	1	bit
if (transport_type == MPEG2) {		
source_ID	16	
frequency	16	uimsbf
program_number	16	
transmission_system	4	Uimsbf
inner_coding_mode	4	Uimsbf
split_bitstream_mode	1	Bslbf {no, yes}
modulation_format	5	uimsbf
symbol_rate	28	Uimsbf units: symbols per sec.
} else { /* non-MPEG-2 */		
Frequency	16	Uimsbf
video_standard	4	Uimsbf
}		
if (descriptors_included) {		
descriptors_count	8	Uimsbf
for (i=0;		
i<descriptors_count; i++) {		
descriptor()	*	
}		
}		
}		

sdb_tune_cnf_tag

0xTBD

channel_num

The virtual channel number of the channel information requested, used to match the confirmation with the tune_req APDU.

request_status

The status of the request:

0x00 - Success. Valid tuning information contained.

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

	0x01 - Please wait. Another tune_cnf() APDU shall follow.
	0x02 - Tuning information has changed since last tune_cnf()
	0x03 - Channel no longer available.
	0x04 - Invalid or unrecognized virtual channel number.
	0x05 - Tuning request permanently denied (max system resource usage reached and not expected to change).
	0x06 - Tuning request temporarily denied (max system resource usage reached but expected to be released later on)
transport_type	0 - MPEG2 1 - Analog
source_ID	A 16-bit unsigned integer number, in the range 0x0000 to 0xFFFF, that identifies the programming source ID associated with the virtual channel on a system-wide basis,
frequency	Contains the frequency for the Host to tune. The frequency is calculated by multiplying frequency by 0x0.05 MHz (50 kHz resolution).
program_number	A 16-bit unsigned integer number that associates the virtual channel number being defined with services defined in the Program Association and TS Program Map Table sections.
transmission_system	A 4-bit field that identifies the transmission standard employed for the waveform. Table 5.7 in SCTE65 defines the coding for transmission_system.
inner_coding_mode	A 4-bit field that indicates the coding mode for the inner code associated with the waveform. The following values are currently defined: 5/11, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, and 7/8. Coding of the inner_coding_mode field is shown in Table 5.8 in SCTE65.
modulation_format	A 5-bit field that defines the basic modulation format for the carrier. Table 5.9 in SCTE65 defines the parameter.
symbol_rate	A 28-bit unsigned integer field that indicates the symbol rate in symbols per second associated with the waveform.
video_standard	A 4-bit field that indicates the video standard associated with this non-Standard virtual channel. Table 5.21 in SCTE65 defines video_standard.
descriptor()	The structure may include, at its end, one or more structures of the form tag, length, data. The number of descriptors present is determined indirectly by processing the length field. Descriptors are defined in Section 6 in SCTE65.

1.2. SDB_tune_cancel APDU

The Host shall issue a sdb_tune_cancel() APDU when a tuner no longer needs a SDB channel. The Host must issue this APDU in all of the following cases:

- That tuner wishes to tune to another channel
- The Host has determined that the channel is no longer needed.

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

Table 3 sdb_tune_cancel() APDU Syntax

Syntax	No. of Bits	Mnemonic
Sdb_tune_cancel() { Sdb_tune_cancel_tag	24	uimsbf
length_field() channel_num }	12	uimsbf range 0-4095

sdb_tune_cancel_tag 0xTBD

channel_num The virtual channel number of the channel information requested, used to match the confirmation with the tune_req APDU.

1.3. SDB_tune_query and SDB_tune_query_cnf() APDU

The CableCARD may issue a sdb_tune_query() APDU to actively request the viewing status of a stream. The Host may automatically respond if manual user activity (e.g., remote control input) has occurred within the past four hours (240 minutes), or if the channel is being recorded by user request. The Host may present a message on the screen to the user asking for confirmation that they are still actively watching the tuner.

The Host may always respond automatically but the matching sdb_tune_query_cnf() response APDU shall be send within sixty (60) seconds of receiving the sdb_tune_query APDU. A lack of response after that time interval can be interpreted by the CableCARD as assuming the Host no longer needs the SDB channel.

Table 4 sdb_tune_query() APDU Syntax

Syntax	No. of Bits	Mnemonic
Sdb_tune_query() { Sdb_tune_query_tag	24	uimsbf
length_field() channel_num }	12	uimsbf range 0-4095

sdb_tune_query_tag 0xTBD

channel_num The virtual channel number of the channel information requested, used to match the confirmation with the tune_req APDU.

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

Table 5 sdb_tune_query_cnf() APDU Syntax

Syntax	No. of Bits	Mnemonic
Sdb_tune_query_cnf() {		
Sdb_tune_query_cnf_tag	24	uimsbf
length_field()		
channel_num	12	uimsbf range 0-4095
tune_status	8	uimsbf
}		

sdb_query_cnf_tag 0xTBD

channel_num The virtual channel number of the channel information requested, used to match the confirmation with the tune_req APDU.

tune_status 0x00 - channel no longer needed
 0x01 - channel still actively being viewed
 0x02 - channel being recorded
 0x04-0xFF - reserved

2. VOD CableCARD Resource Definition

This section defines the VOD Resource as an extension to SCTE-28.

Resource	Class	Type	Version	Resource Identifier
VOD Support	TBD	1	1	0x TBD

The iDCR Host may implement a VOD Support resource. The VOD Support application in the CableCARD shall create a session to the VOD resource in the iDCR Host if the Host supports it.

2.1. VOD Resource Overview

Video-On-Demand takes advantage of the bi-directional communication path between the Host and the cable plant to allow the user to browse a catalog of video titles, and select them for immediate streaming to the Host.

The Host shall use the `vod_init()` APDU to indicate to the CableCARD that the user wants to use the VOD service. The CableCARD shall indicate to the Host if the service is currently available in the response.

The Host shall use the `vod_getfolder_req()` APDU to retrieve the catalog of VOD titles. This catalog is contained in a hierarchical folder structure. The Host shall use repeated calls to `vod_getfolder_req()` to obtain the elements of the catalog that the user wants to see.

Once the user has picked a non-free VOD title, the user can attempt to purchase the title. The Host shall use the `vod_purchase_req()` APDU to make a purchase request. This may initiate a `vod_pin_req()` challenge from the CableCARD. If the user has chosen a free VOD title, the host shall use the `vod_purchase_req()` APDU to make a purchase request, but in this case there shall not be a `vod_pin_req()` challenge from the CableCARD.

After purchasing the VOD title, the Host shall attempt to create a VOD session with the VOD server in the cable system. After successfully creating a session, a purchased VOD title can be viewed. The playback speed and position within the VOD title can be adjusted by the Host via the `vod_setspeedpos_req()` APDU.

The VOD Support resource shall support the following APDUs:

Apdu_tag	Tag Value	Resource	Direction Host ↔ Card
<code>vod_init()</code>	0x TBD	VOD Support	→
<code>vod_init_cnf()</code>	0x TBD	VOD Support	←
<code>vod_getfolder_req()</code>	0x TBD	VOD Support	→
<code>vod_getfolder_cnf()</code>	0x TBD	VOD Support	←
<code>vod_getitem_req()</code>	0x TBD	VOD Support	→
<code>vod_getitem_cnf()</code>	0x TBD	VOD Support	←

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

vod_purchase_req()	0x TBD	VOD Support	→
vod_pin_req()	0x TBD	VOD Support	←
vod_pin_cnf()	0x TBD	VOD Support	→
vod_purchase_cnf()	0x TBD	VOD Support	←
vod_session_init()	0x TBD	VOD Support	→
vod_session_init_cnf()	0x TBD	VOD Support	←
vod_setspeedpos_req()	0x TBD	VOD Support	→
vod_setspeedpos_cnf()	0x TBD	VOD Support	←
vod_queryspeedpos_req()	0x TBD	VOD Support	→
vod_queryspeedpos_cnf()	0x TBD	VOD Support	←

2.2. VOD_init and VOD_init_cnf() APDU Syntax

The Host shall issue a vod_init() APDU after a session to the VOD resource has been created. The CableCARD can respond with information on a barker channel: a channel used to advertise VOD content available on the system. If the barker channel information is provided, the Host may tune to it and present it to the user. A barker channel may be full screen, or have active video only on a rectilinear portion of the screen. The CableCARD shall provide information on the format of the barker channel in the vod_init_cnf() APDU.

Table 6 vod_init() APDU Syntax

Syntax	No. of Bits	Mnemonic
vod_init() { vod_init_tag length_field() }	24 Always 0	uimsbf

vod_init_tag 0xTBD

The CableCARD will reply with a vod_init_cnf() APDU containing information on a barker channel, if present.

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

Table 7 vod_init_cnf() APDU Syntax

Syntax	No. of Bits	Mnemonic
Vod_init_cnf() {		
Vod_init_cnf_tag	24	uimsbf
length_field()	X	
vod_status	8	uimsbf
root_folder_providerID	20*8	uimsbf
root_folder_assetID	20*8	uimsbf
barker_channel	8	uimsbf
width	12	uimsbf
height	12	uimsbf
xpos	12	uimsbf
ypos	12	uimsbf
transport_type	1	bit
if (transport_type == MPEG2) {		
source_ID	16	
frequency	16	uimsbf
program_number	16	
transmission_system	4	Uimsbf
inner_coding_mode	4	Uimsbf
split_bitstream_mode	1	Bslbf {no, yes}
modulation_format	5	uimsbf
symbol_rate	28	Uimsbf units: symbols per sec.
} else { /* non-MPEG-2 */		
Frequency	16	Uimsbf
video_standard	4	Uimsbf
}		
if (descriptors_included) {		
descriptors_count	8	Uimsbf
for (i=0; i<descriptors_count;		
i++) {		
descriptor()	*	
}		
}		
}		

vod_init_cnf_tag 0xTBD

vod_status 0x00 - VOD service is available.
 0x01 - VOD service temporarily not available.
 0x02 - Device not authorized for VOD service.
 0x03-0xFF - Reserved

root_provider_id The providerID of the root VOD folder. Must be 20 ASCII characters.

root_asset_id The assetID of the root VOD folder. Must be 20 ASCII characters.

barker_channel 0x00 - no barker channel available. The tuning information below can be ignored.

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

	0x01 - barker channel available.
source_ID	A 16-bit unsigned integer number, in the range 0x0000 to 0xFFFF, that identifies the programming source ID associated with the barker channel on a system-wide basis,
frequency	Contains the frequency of the barker channel for the Host to tune. The frequency is calculated by multiplying frequency by 0x0.05 MHz (50 kHz resolution).
program_number	A 16-bit unsigned integer number that associates the barker channel with services defined in the Program Association and TS Program Map Table sections.
modulation	The type of modulation for the barker channel. 0x00 Analog ATSC 0x01 64QAM 0x02 256QAM 0x03-0xFF Reserved.
width	The width of the active video portion of the barker channel given as the fraction of the total screen width divided by 4095. A value of 0xFFFF indicates the barker channel is full screen horizontally.
height	The height of the active video portion of the barker channel given as the fraction of the total screen height divided by 4095. A value of 0xFFFF indicates the barker channel is full screen vertically.
xpos	The horizontal position of the upper left corner of the active video portion of the barker channel given as the fraction of the total screen width divided by 4095. A value of zero is the left side of the screen.
ypos	The vertical position of the upper left corner of the active video portion of the barker channel given as the fraction of the total screen height divided by 4095. A value of zero is the top of the screen.
transport_type	0 - MPEG2 1 - Analog
source_ID	A 16-bit unsigned integer number, in the range 0x0000 to 0xFFFF, that identifies the programming source ID associated with the virtual channel on a system-wide basis,
frequency	Contains the frequency for the Host to tune. The frequency is calculated by multiplying frequency by 0x0.05 MHz (50 kHz resolution).
program_number	A 16-bit unsigned integer number that associates the virtual channel number being defined with services defined in the Program Association and TS Program Map Table sections.
transmission_system	A 4-bit field that identifies the transmission standard employed for the waveform. Table 5.7 in SCTE65 defines the coding for transmission_system.
inner_coding_mode	A 4-bit field that indicates the coding mode for the inner code associated with the waveform. The following values are currently

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

	defined: 5/11, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, and 7/8. Coding of the inner_coding_mode field is shown in Table 5.8 in SCTE65.
modulation_format	A 5-bit field that defines the basic modulation format for the carrier. Table 5.9 in SCTE65 defines the parameter.
symbol_rate	A 28-bit unsigned integer field that indicates the symbol rate in symbols per second associated with the waveform.
video_standard	A 4-bit field that indicates the video standard associated with this non-Standard virtual channel. Table 5.21 in SCTE65 defines video_standard.
descriptor()	The structure may include, at its end, one or more structures of the form tag, length, data. The number of descriptors present is determined indirectly by processing the length field. Descriptors are defined in Section 6 in SCTE65.

2.3. The VOD MetaData Format

The descriptive metadata for the VOD content is represented by a series of hierarchical folders in XML format. A folder can contain descriptive data about other folders, vod assets, or both. An asset is a VOD video that the user can purchase (may be free) and begin streaming to the Host. The Host retrieves the descriptive metadata through a series of calls to the get_folder_req() APDU, starting at the root folder. Each folder shall have a unique folder identifier, just as each asset shall have a unique asset identifier. The Host uses the folder identifier in the vod_getfolder_cnf() APDU to indicate which folder it wants to retrieve. The descriptive metadata returned by the CableCARD in the vod_getfolder_cnf() APDU contains the identifiers for the folder's content.

As defined in the CableLabs Video-On-Demand Content Specification Version 2.0, each folder and asset is uniquely identified by a combination of its providerID and assetID. providerID is a unique identifier for the provider of the Asset. The providerID shall be set to a registered Internet domain name restricted to at most 20 lower-case characters and belonging to the provider. For example a valid providerID for CableLabs is "cablelabs-films.com". Throughout the VOD resource the providerID shall be exactly 20 ASCII characters long. Domain names shorter than 20 characters shall be zero padded. The assetID is an identifier for the asset that is unique within a provider's assetID space. The assetID shall be an ASCII String of exactly 20 characters.

The vod_getfolder_req() APDU allows for windowing of the folder contents. This allows the Host to request only a finite set of items contained in the folder instead of receiving the entire folder all at once.

The VOD system may preserve a Folder containing the titles that the user has already purchased and are still available for viewing. This 'Saved Programs' folder would be presented in the folder hierarchy like any other folder.

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

Table 8 The VOD Folder Metadata Format returned in the vod_getfolder_cnf() APDU

```
<FolderContents windowOffset="20" totalItems="50">
  <Folder providerId="foo.com" assetId="BARR0000000000000001">
    <!-- Metadata details from CalbeLabs ADI and VOD Content
specs... -->
  </Folder>
  <Folder providerId="foo.com" assetId="BARR0000000000000002" >
    <!-- Metadata details from CalbeLabs ADI and VOD Content
specs... -->
  </Folder>
  <Asset providerId="foo.com" assetId="BARR0000000000000003">
    <!-- Metadata details from CalbeLabs ADI and VOD Content
specs... -->
  </Asset>
  <!-- ... additional Folder and Asset elements ... -->
</FolderContents>
```

Once the Host has the folder information, it shall request details on folder items via the vod_getitem_req() APDU. The Host passes the unique identifiers for the folder or asset.

Table 9 XML Format returned in the vod_getitem_cnf() APDU

Response for an Asset:

```
<ItemDetails>
  <Asset providerId="foo.com" assetId="BARR0000000000000002">
    <!-- Metadata details from CalbeLabs ADI and VOD Content
specs... -->
  </Asset>
</ItemDetails/>
```

Response for a Folder:

```
<ItemDetails>
  <Folder providerId="foo.com" assetId="BARR0000000000000000">
    <!-- Metadata details from CalbeLabs ADI and VOD Content
specs... -->
  </Folder>
</ItemDetails/>
```

The asset metadata format returned by the CableCARD in the vod_getitem_cnf() APDU shall conform to the OpenCable Video-On-Demand Content Specification Version 2.0. In particular, a VOD asset shall include the following metadata elements

- Section 6.4 Title Asset is Mandatory, including at a minimum the Title, Genre, and Rating elements.
- Section 6.5 Terms Asset is Mandatory, with Suggested Price replaced with a mandatory Price element.

The Host can also indicate the amount of detail it wants in the descriptive metadata associated with assets. The VOD system may support several different detail levels of

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

metadata. Both features are designed to optimize the amount of bandwidth usage required for VOD metadata exchange.

2.4. vod_getfolder_req() & vod_getfolder_cnf() APDU Syntax

Table 10 VOD getfolder Request Object Syntax

Syntax	# of bits	Mnemonic
vod_getfolder_req() {		
vod_getfolder_tag	24	uimsbf
length_field()		
request_id	12	uimsbf
item_index	12	uimsbf
num_items	12	uimsbf
provider_id	20*8	uimsbf
asset_id	20*8	uimsbf
}		

vod_getfolder_req_tag

0xTBD

request_id

A unique number generated by the iDCR to identify a getfolder request. The associated *getfolder_cnf()* will include this **request_ID** value. iDCRs shall maintain a **transaction_ID** counter and increment it by 1 (mod 4096) for each new transaction.

item_index

The zero-based index to the first item in the folder that should be returned in the confirmation APDU. An index of 0x00 represents the first item in the folder.

num_items

The number of items to return, starting at item_index. The value 0x00 shall indicate that all items in the folder should be returned.

provider_id

The providerID of the folder to retrieve.

asset_id

The assetID of the folder to retrieve.

The CableCARD shall respond to the getfolder_req() APDU with a getfolder_cnf() APDU.

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

Table 11 VOD getfolder Confirmation Object Syntax

Syntax	No. of Bits	Mnemonic
vod_getfolder_cnf() { vod_getfolder_cnf_tag length_field() request_id status	24	Uimsbf
xml_length for (i=0; i <= xml_length; i++) { xml_byte}	16	Uimsbf
}	8	Uimsbf

vod_getfolder_cnf_tag 0xTBD

request_id The unique request number from the corresponding request APDU.

status The status of the request.
 0x00 - Success. XML data follows.
 0x01 - Unknown folder_id
 0x02 - Item index larger than number of items in folder

xml_lgnth, xml_byte The XML text in the response.

2.5. vod_getitem_req() & vod_getitem_cnf() APDU Syntax

Table 12 VOD getitem Request Object Syntax

Syntax	# of bits	Mnemonic
vod_getitem_req() { vod_getitem_tag length_field() request_id detail_level	24	uimsbf
provider_id asset_id	20*8 20*8	uimsbf uimsbf

vod_getitem_req_tag 0xTBD

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

request_id	A unique number generated by the iDCR to identify a getfolder request. The associated <i>getfolder_cnf()</i> will include this request_ID value. iDCRs shall maintain a transaction_ID counter and increment it by 1 (mod 4096) for each new transaction.
detail_level	The amount of descriptive detail to include in the response. The VOD system may or may not support different detail level. 0x00 - Low detail 0x01 - Medium detail 0x02 - High detail
provider_id	The providerID of the item to retrieve.
asset_id	The assetID of the folder to retrieve.

The CableCARD shall respond to the vod_getitem_req() APDU with a vod_getitem_cnf() APDU.

Table 13 VOD getitem Confirmation Object Syntax

Syntax	No. of Bits	Mnemonic
vod_getitem_cnf() { vod_getfolder_cnf_tag length_field() request_id status detail_level xml_length for (i=0; i <= xml_length; i++) { xml_byte} }	24 12 8 16 8	Uimsbf Uimsbf Uimsbf Uimsbf Uimsbf

vod_getitem_cnf_tag	0xTBD
request_id	The unique request number from the corresponding request APDU.
status	The status of the request. 0x00 - Success. XML data follows. 0x01 - Unknown item
detail_level	The detail level used in the response, which may differ from the detail level requested.
xml_lgnth, xml_byte	The XML text in the response.

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

2.6. vod_Purchase_req() & vod_Purchase_cnf() APDU Syntax

The iDCR's navigation application uses vod_*purchase_req()* to request a purchase of a VOD program. The program information is obtained by the metadata provided by the VOD server. The CableCARD responds with vod_*purchase_cnf()*.

Table 14 VOD Purchase Request Object Syntax

Syntax	# of bits	Mnemonic
vod_purchase_req() { vod_purchase_req_tag length_field() transaction_id provider_id asset_id }	24 12 20*8 20*8	uimsbf uimsbf uimsbf uimsbf

- vod_purchase_req_tag** 0xTBD
- transaction_id** A unique number generated by the iDCR to identify a transaction. The associated *program_cnf()* will include this *transaction_ID* value. iDCRs shall maintain a *transaction_ID* counter and increment it by 1 (mod 4096) for each new transaction.
- provider_id** The provider identifier for the asset.
- asset_id** The asset identifier for the asset.

The CableCARD shall respond to the vod_purchase_req() APDU with a vod_purchase_cnf() APDU. The CableCARD may issue a vod_pin_req() APDU first.

Table 15 VOD Purchase Confirmation Object Syntax

Syntax	No. of Bits	Mnemonic
vod_purchase_cnf() { vod_purchase_cnf_tag length_field() transaction_id status_field comment_length for (i=0; i <= comment_length; i++) {	24 12 8 16	uimsbf uimsbf uimsbf uimsbf

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

Syntax	No. of Bits	Mnemonic
comment_txt}	8	uimsbf
provider_id	20*8	uimsbf
asset_ID	20*8	uimsbf
}		

vod_purchase_cnf_tag 0xTBD

transaction_id The unique transaction number.

status_field This field returns the status of the *vod_purchase_req()*. If the CableCARD has validated the purchase, then the *status_field* is set to 0x00. Otherwise, the *status_field* is set to one of the following values. When there is more than one reason to deny the purchase, the *status_field* is set to the lowest applicable value.

provider_ID The provider ID of the requested VOD program, used to match the confirmation with the request APDU.

asset_ID The asset ID of the VOD program requested, used to match the confirmation with the request APDU.

Table 16 Status Field Values for VOD Purchase Confirm

Status_field	Value (hex)
Purchase Granted	00
Purchase Denied – Unknown Asset ID	01
Purchase Denied – Unknown Transaction_ID	02
Purchase Denied – Invalid PIN Code	03
Purchase Denied – Program Already Purchased	04
Purchase Denied – Blackout is Active	05
Purchase Denied – Credit Limit is Exceeded	06
Purchase Denied – VOD Slot Limit is Exceeded	07
Purchase Denied – Spending Limit is Exceeded	08
Purchase Denied – Rating Limit is Exceeded	09
Purchase Denied – Check Comments	0A
Reserved	0B-FF

comment_length, comment_txt These fields allow the CableCARD to explain, using plain text, why the purchase request has been granted or denied. The CableCARD shall use these fields only if an explanation is not covered by one of the other status values.

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

2.7. vod_pin_req() & vod_pin_cnf() APDU Syntax

In response to a vod_purchase_req() APDU, the CableCARD may issue a vod_pin_req() APDU requesting a valid personal identification number (PIN) in order to make the purchase.

Table 17 VOD PIN Request Syntax

Syntax	# of bits	Mnemonic
vod_pin_req() { vod_pin_req_tag length_field() transaction_id }	24	uimsbf
	12	uimsbf

vod_pin_req_tag 0xTBD
transaction_id The unique transaction ID associated with this purchase request and subsequent PIN challenge.

In response to a vod_pin_req() APDU, the Host shall respond with a vod_pin_cnf() APDU. The Host shall send the PIN code entered by the user, or it may indicate a desire to cancel the purchase request by setting the cancel_flag to 0x01.

Table 18 VOD PIN Confirmation Syntax

Syntax	# of bits	Mnemonic
vod_pin_cnf() { vod_pin_cnf_tag length_field() transaction_id cancel_flag PINcode_len for (i=0; i < PINcode_len; i++) { PINcode_byte; } }	24	uimsbf
	12	uimsbf
	8	uimsbf
	8	uimsbf
	8	Uimsbf

vod_pin_cnf_tag 0xTBD
transaction_id The unique transaction ID associated with this purchase request and subsequent PIN challenge.
cancel_flag If set to anything but 0x00, indicates that the user wishes to cancel the purchase request.
PINcode_len, PINcode_byte If cancel_flag == 0, these fields allow the iDCR navigation application to pass the requested PIN code to the CableCARD. The PINcode_len shall be zero otherwise.

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

2.8. vod_session_init() and vod_session_init_cnf() APDU Syntax

To start a VOD stream, the Host sends a vod_session_init() APDU to the CableCARD. The CableCARD interacts with the VOD system on the cable plant to initialize a session for the Host. The session is torn down by the VOD system when the VOD stream is stopped; typically, because the program has ended, or the user has requested the stream to be stopped.

The VOD server may time out a Paused stream and end the VOD session. In this case, the CableCARD shall issue a vod_session_init_cnf() APDU updating the status of the session.

Table 19 vod_session_init() APDU Syntax

Syntax	No. of Bits	Mnemonic
Vod_session_init() { vod_session_init_tag	24	uimsbf
length_field() provider_id	20*8	uimsbf
asset_id }	20*8	uimsbf

- vod_session_init_tag** 0xTBD
- provider_ID** The provider ID of the requested VOD program.
- asset_id** The program's assetID.

The CableCARD will reply with a vod_session_init_cnf() APDU.

Table 20 VOD Session Init Confirmation Object Syntax

Syntax	No. of Bits	Mnemonic
vod_session_init_cnf() { vod_session_init_cnf_tag	24	uimsbf
length_field() session_id	12	uimsbf
provider_id	20*8	uimsbf
asset_id	20*8	uimsbf
 status_field	8	uimsbf
duration	32	uimsbf

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

Syntax	No. of Bits	Mnemonic
trickplay_flag	8	uimsbf
transport_type	1	bit
if (transport_type == MPEG2) {		
source_ID	16	
frequency	16	uimsbf
program_number	16	
transmission_system	4	Uimsbf
inner_coding_mode	4	Uimsbf
split_bitstream_mode	1	Bslbf {no, yes}
modulation_format	5	uimsbf
symbol_rate	28	Uimsbf units: symbols per sec.
} else { /* non-MPEG-2 */		
Frequency	16	Uimsbf
video_standard	4	Uimsbf
}		
if (descriptors_included)		
{		
descriptors_count	8	Uimsbf
for (i=0;		
i<descriptors_count; i++)		
{		
descriptor()	*	
}		
}		
}		

vod_session_init_cnf_tag 0xTBD

session_id The unique session identifier created by the VOD system.

provider_ID The provider ID of the requested VOD program.

asset_id The program's assetID.

status_field This field returns the status of the *vod_session_init()*. If the VOD system established a VOD session, then the *status_field* is set to 0x00. Otherwise, the *status_field* is set to one of the following values.

Table 21 Status Field Values for VOD Purchase Confirm

Status_field	Value (hex)
Success	00
System Busy - Try Again	01
Unknown Asset ID	02
Asset has not been purchased	03
Maximum number of simultaneous VOD Sessions exceeded	04
Reserved	05-FF

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

duration	The duration of the VOD program, in milliseconds. This value is used in the set_speedpos APDUs.
trickplay_flag	Set to 0x00 if no part of the VOD stream supports non-realtime playback speed. Pause may be supported. Set to 0x01 if non-realtime playback (trickplay) of any part of the VOD stream is possible.
transport_type	0 - MPEG2 1 - Analog
source_ID	A 16-bit unsigned integer number, in the range 0x0000 to 0xFFFF, that identifies the programming source ID associated with the virtual channel on a system-wide basis,
frequency	Contains the frequency for the Host to tune. The frequency is calculated by multiplying frequency by 0x0.05 MHz (50 kHz resolution).
program_number	A 16-bit unsigned integer number that associates the virtual channel number being defined with services defined in the Program Association and TS Program Map Table sections.
transmission_system	A 4-bit field that identifies the transmission standard employed for the waveform. Table 5.7 in SCTE65 defines the coding for transmission_system.
inner_coding_mode	A 4-bit field that indicates the coding mode for the inner code associated with the waveform. The following values are currently defined: 5/11, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, and 7/8. Coding of the inner_coding_mode field is shown in Table 5.8 in SCTE65.
modulation_format	A 5-bit field that defines the basic modulation format for the carrier. Table 5.9 in SCTE65 defines the parameter.
symbol_rate	A 28-bit unsigned integer field that indicates the symbol rate in symbols per second associated with the waveform.
video_standard	A 4-bit field that indicates the video standard associated with this non-Standard virtual channel. Table 5.21 in SCTE65 defines video_standard.
descriptor()	The structure may include, at its end, one or more structures of the form tag, length, data. The number of descriptors present is determined indirectly by processing the length field. Descriptors are defined in Section 6 in SCTE65.

2.9. vod_setspeedpos_req() and vod_setspeedpos_cnf() APDU Syntax

The Host can request that the VOD server change the play speed or position of the VOD video stream. The position within the video stream is given by a 32-bit unsigned integer representing time in milliseconds. The value of the position ranges from 0, indicating the starting point of the stream up to the value of the **duration** of the stream, provided in the session confirmation.

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

The speed of the video playback is represented by a floating-point number, with 1.0 representing normal play speed. A positive value indicates forward play and a negative value indicates reverse play. Non-normal play speed can include fast or slow, forward or reverse play directions.

In the vod_setspeedpos() APDU, the requested direction and speed of play is determined by a speed parameter. It consists of a 16-bit signed numerator and 16-bit unsigned denominator. A value of 1/1 = speed of 1.0 represents normal viewing rate. A denominator of 0 is invalid and should not be used. A negative number represents reverse speeds. The VOD system may not support all trickplay speeds. For example, it may only support fast forward speeds of 3.0 and 10.0. If the Host requests a speed that the VOD system does not support, the VOD system must set the speed to the closest speed it does support. The vod_setspeedpos_cnf() APDU shall return the actual speed that the VOD system will use.

The Host must issue a setspeed_req() APDU with a speed of 1.0 after a successful VOD purchase and session initiation. The Host may indicate a position other than the beginning of the stream (0).

If trickplay of the VOD stream is supported, the Host may issue additional vod_setspeedpos_req APDUs to request a change in the stream play status. A play speed of 0.0 is a request to pause the stream. A resume of the stream after Pause is accomplished by requesting any play speed other than 0.0.

Table 22 VOD setspeedpos Request Syntax

Syntax	# of bits	Mnemonic
vod_setspeedpos_req() {		
vod_setspeedpos_req_tag	24	uimsbf
length_field()		
session_id	12	uimsbf
speed_num	16	simsbf
speed_denom	16	uimsbf
position_flag	1	bit
position	32	uimsbf
}		

vod_setspeedpos_req_tag	0xTBD
session_id	The unique session ID associated with this VOD program
speed_num	Numerator of the speed value
speed_denom	Denominator of the speed value. Shall never be set to 0.
position_flag	0x00 - Use the stream's current play position 0x01 - Use the play position given in this APDU
position	The position to execute the speed request at. 0x00 shall represent the beginning of the stream. A position that goes beyond the

Appendix C -- CEA's Technical Standards and Specifications, Access to Basic Interactive Services

duration of the stream will stop the stream as if it had reached the end of the stream.

In response to a vod_setspeedpos_req() APDU, the CableCARD shall respond with a vod_setspeedpos_cnf() APDU.

Table 23 VOD setspeedposConfirmation Syntax

Syntax	# of bits	Mnemonic
vod_setspeedpos_cnf() { vod_setspeedpos_cnf_tag length_field() session_id status_field speed_num speed_denom position }	24	uimsbf
	12	uimsbf
	8	uimsbf
	16	simsbf
	16	usisbf
	32	uimsbf

vod_setspeedpos_cnf_tag 0xTBD

session_id The unique transaction ID associated with this purchase request and subsequent PIN challenge.

status_field The status of the trickplay request.

position The current play position of the stream.

Table 24 setspeedposconfirmation status field definitions

status_field	Value (hex)
request granted	0x00
trickplay at this point not supported	0x01
unrecognized request	0x02
bad speed parameter	0x03
end of stream	0x04
unknown error	0x05

speed_num Numerator of the actual speed value

speed_denom Denominator of the actual speed value. Shall never be set to 0.

Table 25 VOD queriespeedpos Request Syntax

Syntax	# of bits	Mnemonic
vod_queriespeedpos_req() { vod_queryspeedpos_req_tag length_field() session_id }	24	uimsbf
	12	uimsbf

