

OpenCable™ Specifications

Common Download 2.0

OC-SP-CDL2.0-I09-090904

ISSUED

Notice

This OpenCable specification is the result of a cooperative effort undertaken at the direction of Cable Television Laboratories, Inc. for the benefit of the cable industry and its customers. This document may contain references to other documents not owned or controlled by CableLabs. Use and understanding of this document may require access to such other documents. Designing, manufacturing, distributing, using, selling, or servicing products, or providing services, based on this document may require intellectual property licenses from third parties for technology referenced in this document.

Neither CableLabs nor any member company is responsible to any party for any liability of any nature whatsoever resulting from or arising out of use or reliance upon this document, or any document referenced herein. This document is furnished on an "AS IS" basis and neither CableLabs nor its members provides any representation or warranty, express or implied, regarding the accuracy, completeness, noninfringement, or fitness for a particular purpose of this document, or any document referenced herein.

© Copyright 2006-2009 Cable Television Laboratories, Inc.
All rights reserved.

Document Status Sheet

Document Control Number:	OC-SP-CDL2.0-I09-090904		
Document Title:	Common Download 2.0		
Revision History:	I01 – Released 10/31/06 I02 – Released 1/5/07 I03 – Released 3/23/07 I04 – Released 6/15/07 I05 – Released 11/13/07 I06 – Released 1/18/08 I07 – Released 11/14/08 I08 – Released 2/6/09 I09 – Released 9/4/09		
Date:	September 4, 2009		
Status:	Work in Progress	Draft	Issued
Distribution Restrictions:	Author Only	GL/Member	GL/Member/Vendor Public

Key to Document Status Codes:

- Work in Progress** An incomplete document, designed to guide discussion and generate feedback that may include several alternative requirements for consideration.

- Draft** A document in specification format considered largely complete, but lacking review by Members and vendors. Drafts are susceptible to substantial change during the review process.

- Issued** A stable document, which has undergone rigorous member and vendor review and is suitable for product design and development, cross-vendor interoperability, and for certification testing.

- Closed** A static document, reviewed, tested, validated, and closed to further engineering change requests to the specification through CableLabs.

Trademarks:

CableLabs®, DOCSIS®, EuroDOCSIS™, eDOCSIS™, M-CMTS™, PacketCable™, EuroPacketCable™, PCMM™, CableHome®, CableOffice™, OpenCable™, OCAP™, CableCARD™, M-Card™, DCAS™, tru2way™ and CablePC™ are trademarks of Cable Television Laboratories, Inc.

Contents

1	SCOPE	1
1.1	Introduction and Overview	1
1.2	Purpose of document	1
1.3	Requirements	1
2	REFERENCES	2
2.1	Normative References	2
2.2	Informative References.....	2
2.3	Reference Acquisition	3
3	TERMS AND DEFINITIONS	4
4	ABBREVIATIONS AND ACRONYMS	6
5	OPERATIONAL DETAILS	8
5.1	Code Image Download Signaling Methods	8
5.1.1	<i>Embedded Cable Modem (eCM) Signaling</i>	8
5.1.2	<i>Embedded Set-top Box (eSTB) Signaling</i>	8
5.1.3	<i>Trigger Validation</i>	10
5.1.4	<i>Trigger Contention</i>	11
5.2	Code Image Download Delivery Methods	11
5.2.1	<i>eSTB DSM-CC Carousels</i>	11
5.2.2	<i>eSTB TFTP File Transfer</i>	12
6	SYSTEM CONTROL RESOURCE	13
6.1	Resource Identifier	15
6.2	Application Objects (APDUs).....	15
6.2.1	<i>host_info_request</i>	16
6.2.2	<i>host_info_response</i>	17
6.2.3	<i>code_version_table</i>	17
6.2.4	<i>code_version_table_reply</i>	19
6.2.5	<i>code_version_table2</i>	19
6.2.6	<i>host_download_control</i>	25
7	SECURE SOFTWARE DOWNLOAD	28
7.1	Initialization of Download.....	28
7.1.1	<i>CVC Delivery</i>	28
7.1.2	<i>Time Varying Controls</i>	32
7.1.3	<i>eCM Configuration File Initiated Download</i>	32
7.1.4	<i>SNMP Initiated Download</i>	33
7.1.5	<i>CVT Initiated Download (DSG Tunnel/In-Band upgrade)</i>	33
7.2	Image File Structure	33
7.2.1	<i>Signed Data</i>	34
7.2.2	<i>Signed Content</i>	35
7.2.3	<i>Code Signing Keys</i>	35
7.3	Delivery of the CVT via DSG Tunnel.....	35
7.3.1	<i>Signed Data</i>	36
7.3.2	<i>Signed Content</i>	37
7.3.3	<i>CVT File Validation</i>	37
7.4	Code Image Validation	37

ANNEX A DOWNLOADINFOINDICATOR MESSAGE DETAIL FOR COMMON DOWNLOAD (NORMATIVE)40

ANNEX B NON-MSO-AFFILIATED CODE IMAGE DOWNLOAD43

APPENDIX I REVISION HISTORY44

Figures

Figure 1 - CA System Flow Example.....9

Figure 2 - DSG CVT Delivery Example10

Figure 3 - Flow Chart Summarizing Code File Download and Verification.....27

Tables

Table 1 - Code Version Table MPEG Section Header10

Table 2 - Resource Identifier15

Table 3 - Application Protocol Data Units15

Table 4 - host_info_request (Type 1 Version 1).....16

Table 5 - host_info_request (Type 2 Version 1).....17

Table 6 - host_info_response.....17

Table 7 - code_version_table (Type 1 Version1)18

Table 8 - code_version_table_reply19

Table 9 - code_version_table2 (Type 2 Version 1)20

Table 10 - host_download_control.....26

Table 11 - Code File Structure33

Table 12 - PKCS #7 Signed Data34

Table 13 - CVT File Structure36

Table 14 - PKCS #7 Signed Data36

Table 15 - DownloadInfoIndicator Message Detail40

1 SCOPE

1.1 Introduction and Overview

This document specifies a common download protocol for CableCARD™-enabled OpenCable devices [OCHD2] with Out-of-Band (OOB) FDC (Forward Data Channel) channels, either [SCTE 55-1] or [SCTE 55-2], In-Band (IB) Forward Application Transport (FAT) channel, eDOCSIS support [eDOCSIS], and DOCSIS Set-top gateway [DSG] support. It is intended for OpenCable-certified Hosts for the purpose of updating any of the firmware objects used in the Host device. An object downloaded to the Host device is a monolithic build that contains one or more types of objects bundled together.

1.2 Purpose of document

The purpose of this document is to detail how firmware objects are downloaded to Open Cable Hosts. There are different methods for downloading the firmware objects with different triggering methods. These methods are covered in different sections detailed in the document.

1.3 Requirements

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

"SHALL"	This word means that the item is an absolute requirement of this specification.
"SHALL NOT"	This phrase means that the item is an absolute prohibition of this specification.
"SHOULD"	This word means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
"SHOULD NOT"	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
"MAY"	This word means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

2 REFERENCES

2.1 Normative References

In order to claim compliance with this specification, it is necessary to conform to the following standards and other works as indicated, in addition to the other requirements of this specification. Notwithstanding, intellectual property rights may be required to use or implement such normative references.

- [13818-1] ISO/IEC 18318-1, Generic Coding of Moving Pictures and Associate Audio System.
- [13818-6] ISO/IEC 13818-6, Extensions for DSM-CC.
- [BPI+] Data-Over-Cable Service Interface Specifications, Baseline Privacy Plus Interface Specification, CM-SP-BPI+-C01-081104, November 4, 2008, Cable Television Laboratories, Inc.
- [CCCP2.0] OpenCable CableCARD Copy Protection 2.0 Specification, OC-SP-CCCP2.0-I10-090904, September 4, 2009, Cable Television Laboratories, Inc.
- [CCIF2.0] OpenCable CableCARD Interface 2.0 Specification, OC-SP-CCIF2.0-I19-090904, September 4, 2009, Cable Television Laboratories, Inc.
- [CEA 679] CEA-679-C Part B, "National Renewable Security Standard" (March 2005).
- [DOCSIS] Data-Over-Cable Service Interface Specifications, DOCSIS RFIv2.0, CM-SP-RFIv2.0-C02-090422, April 22, 2009, Cable Television Laboratories, Inc.
- [DSG] DOCSIS Set-top Gateway (DSG) Interface Specification, CM-SP-DSG-I14-090529, May 29, 2009, Cable Television Laboratories, Inc.
- [eDOCSIS] Data-Over-Cable Service Interface Specifications, eDOCSIS specification, CM-SP-eDOCSIS-I18-090529, May 29, 2009, Cable Television Laboratories, Inc.
- [OCAP] OpenCable Application Platform 1.1 Specification, OC-SP-OCAP1.1.1-090612, June 12, 2009, Cable Television Laboratories, Inc.
- [OCHD2] OpenCable Host Device 2.1 Core Functional Requirements, OC-SP-HOST2.1-CFR-I09-090904, September 4, 2009, Cable Television Laboratories, Inc.
- [OCSEC] OpenCable System Security Specification, OC-SP-SEC-I07-061031, October 31, 2006, Cable Television Laboratories, Inc.
- [PKCS #7] RSA Laboratories, PKCS #7: Cryptographic Message Syntax Standard, An RSA Laboratories Technical Note, Version 1.5, Revised November 1, 1993.
- [SCTE 55-1] ANSI/SCTE 55-1 2002, (formerly DVS 178) "Digital Broadband Delivery System: Out of Band Transport Part 1: Mode A."
- [SCTE 55-2] ANSI/SCTE 55-2 2002, (formerly DVS 167), 2002, "Digital Broadband Delivery System: Out of Band Transport Part 2: Mode B", Society of Cable Telecommunications Engineers.
- [SCTE 65] ANSI/SCTE 65 2002, (formerly DVS 234) Service Information Delivered Out-of-Band for Digital Cable Television.
- [SCTE 7] ANSI/SCTE 07 2000 (formerly DVS/031): Digital Video Transmission Standard for Cable Television.
- [X509] IETF RFC 3280, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", (R. Housley, W. Ford, W. Polk, D. Solo), January 2002.

2.2 Informative References

This specification does not use any informative references.

2.3 Reference Acquisition

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027; Phone 303-661-9100; Fax 303-661-9199; Internet: <http://www.cablelabs.com/>
- Consumer Electronics Association (CEA), Internet: <http://global.ihs.com/>
- Internet Engineering Task Force (IETF), Internet: <http://www.ietf.org>
- International Organization for Standardization (ISO/IEC) Internet: <http://www.iso.org/iso/en/prods-services/ISOstore/store.html>
- RSA Laboratories of RSA, The Security Division of EMC, 174 Middlesex Turnpike, Bedford, MA 01730, Internet: <http://www.rsa.com/rsalabs/>
- Society of Cable Telecommunications Engineers (SCTE), Internet: <http://www.scte.org/standards>

3 TERMS AND DEFINITIONS

This specification uses the following terms:

Application Protocol Data Unit	A common structure to send application data between the Card and Host.
Application Program Interface	The software interface to system services or software libraries. An API can consist of classes, function calls, subroutine calls, descriptive tags, etc.
CableCARD™ device	A PCMCIA card distributed by cable providers and inserted into a Host device to enable premium services in compliance with the OpenCable specifications, also called "Card." It was formerly known as "Point of Deployment" (POD) module.
Card	CableCARD Device.
DOCSIS Set-top Gateway	A method of using DOCSIS protocols to support a one-way out-of-band communication path.
Downstream	Transmission from headend to Host.
DSG Advanced Mode	Operation with the DCD message. Address assignment is dynamic. The DSG Tunnel Address is determined by the DSG Agent and learned by the DSG Client through the DSG Address Table in the DCD message.
DSG Tunnel	A stream of packets sent from the CMTS to the eCM. In DSG Advanced Mode, a DSG Tunnel might be identified solely by its MAC Address, or it might be identified by a combination of the MAC Address along with other DSG Rule parameters: UCID range, Classifier IP addresses, and TCP port numbers.
Forward Application Transport	A data channel carried from the headend to the set-top or Host device in a modulated channel at a rate of 27 or 36 Mbps. MPEG-2 transport is used to multiplex video, audio, and data into the FAT channel.
Forward Data Channel	An out-of-band ("OOB") data channel from the headend to the Host.
Headend	The control center of a cable television system, where incoming signals are amplified, converted, processed and combined into a common cable along with any original cable casting, for transmission to subscribers. The System usually includes antennas, preamplifiers, frequency converters, demodulators, modulators, processors and, other related equipment.
Inband	The term means, "within the main Forward Applications Transport channel."
Internet Protocol	The internet protocol provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses.

Model ID	The model as it is marketed and appears on the label of the Certified Device, and as reported in the Digital Certificate Usage Report (see OpenCable Host 2.0 Digital Certificate authorization Agreement).
MPEG-2 Video	ISO-IEC 13818-2, international standard for the compression of video.
MPEG-2 Transport	ISO-IEC 13818-1, international standard for the transport of compressed digital media.
Quadrature Amplitude Modulation	A digital modulation method in which the value of a symbol consisting of multiple bits is represented by amplitude and phase states of a carrier. Typical types of QAM include 16 QAM (four bits per symbol), 32 QAM (five bits), 64 QAM (six bits), and 256 QAM (eight bits).
Quadrature Phase Shift Keying	A digital modulation method in which the state of a two-bit symbol is represented by one of four possible phase states.
Resource	A message exchange protocol identified by a unique Identifier and consists of one or more APDUs or objects, whose syntax and usage is well defined.
Return Data Channel	An out-of-band data communication channel running upstream from home to the headend.
Uniform Resource Locator	A standard method of specifying the location of an object or file.
Upstream	Transmission from host to headend.
User Datagram Protocol	A protocol on top of IP that is used for end-to-end transmission of user messages. Unlike TCP, UDP is an unreliable protocol, which means that it does not contain any retransmission mechanisms.

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations and acronyms:

APDU	Application Protocol Data Unit
API	Application Program Interface
ASCII	American Standard Code for Information Interchange
bslbf	Bit String (serial) – Left Most Bit First
CA	Conditional Access
CAS	Conditional Access System
CRC	Cyclic Redundancy Check
CL	CableLabs
CSI	Cable System Information
CSR	Customer Service Representative
CVC	Code Verification Certificate
CVS	Code Verification Signature
CVT	Code Version Table
DCD	Downstream Channel Descriptor
DII	Download Info Indicator
DOCSIS®	Data-Over-Cable Service Interface Specifications
DSG	DOCSIS Set-top Gateway
DSM-CC	Digital Storage Medium – Command and Control
eCM	Embedded Cable Modem
FAT	Forward Application Transport
FDC	Forward Data Channel
IB	In-Band
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol

kb	kilobyte
MAC	Media Access Control
MHz	MegaHertz
MIB	Management Information Base
MPEG	Moving Pictures Experts Group.
MSO	Multiple System Operator
OCAP	OpenCable Application Platform
OOB	Out-of-Band
OUI	Organizationally Unique Identifier
PID	Packet Identifier
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RDC	Return Data Channel
UDP	User Datagram Protocol
SNMP	Simple Network Management Protocol
TFTP	Trivial File Transfer Protocol
uimsbf	Unsigned Integer Most Significant Bit First
URL	Uniform Resource Locator
VCT	Virtual Channel Table
X.509	ITU-T Recommendation X.509

5 OPERATIONAL DETAILS

5.1 Code Image Download Signaling Methods

The availability of a code image download can be signaled to either the eCM or the eSTB component of a Host device. Each of these methods is used to trigger download of a code object that is used to update all or some software components of the Host. Downloads can be triggered through the eCM for upgrading an individual Host or triggered through the eSTB via the Code Version Table (CVT) message for global upgrade of a group of similar Hosts or for targeting an individual host.

5.1.1 Embedded Cable Modem (eCM) Signaling

Code image download signaling to the eCM can occur in two ways. One method is through the use of the configuration file downloaded to the eCM at the time of initialization. The other method is through the use of SNMP SET commands to applicable MIB objects. Regardless of which signaling method is used, the code image delivery is accomplished using Trivial File Transfer Protocol (TFTP) using a provided TFTP server IP address and code image name. Both of these methods are used to target individual Host devices for code image upgrade.

5.1.1.1 Configuration File Triggering

Download signaling via the eCM can be done through the configuration file during initialization. In this method software download is initiated by sending the Software Upgrade Filename in the configuration file (TLV 9). If the Software Upgrade Filename (TLV 9) in the eCM configuration file does not match the current software image of the device, the Host eCM SHALL request the specified file via TFTP.

When used to signal code image download for a Host device, the eCM configuration file must contain either an Mfg CVC (TLV 32) or a Co-signer CVC (TLV 33), but may contain both.

The Host SHALL support eCM configuration-file-initiated code image download.

5.1.1.2 SNMP Triggering

Download signaling via the eCM can be done at any time after registration using SNMP. This trigger method will only work if the eCM has been enabled for code download by inclusion of the Mfg or Co-signer CVC in the configuration file or the CVT message. The details of this method are described in section 7.1.4. The Host SHALL support SNMP-initiated code image download.

5.1.2 Embedded Set-top Box (eSTB) Signaling

Download signaling via the eSTB is done using the Code Version Table (CVT). The CVT message can be delivered using two methods. These methods are (1) CA system delivery to the CableCARD (OOB FDC or DSG CA tunnel), where the CVT is passed from the CableCARD to the eSTB via the Card-Host Interface utilizing the code version table APDU, and (2) delivery to the eSTB via DSG Broadcast Tunnel. The Host SHALL support CVT-initiated code image download.

5.1.2.1 CVT Delivery Methods

Code Version Tables (CVT), specific to each type of Host device on the network, provide one of: (1) an In-Band FAT channel locator (source ID; frequency, modulation mode and MPEG program number; or frequency, modulation mode and PID), (2) DSG Application Tunnel Application ID (Advanced DSG mode), (3) TFTP server address for the code image file, or (4) DSG Tunnel Parameters.

5.1.2.1.1 CA System CVT Delivery

This method delivers the CVT message to the Card in a network proprietary way via the OOB FDC or DSG CA tunnel. The Card performs filtering on the CVT based on information received from the Host device and only passes the CVT to the Host if parameters defined in the CVT are equivalent to the information provided by the Host. The following is an example of the CA System flow:

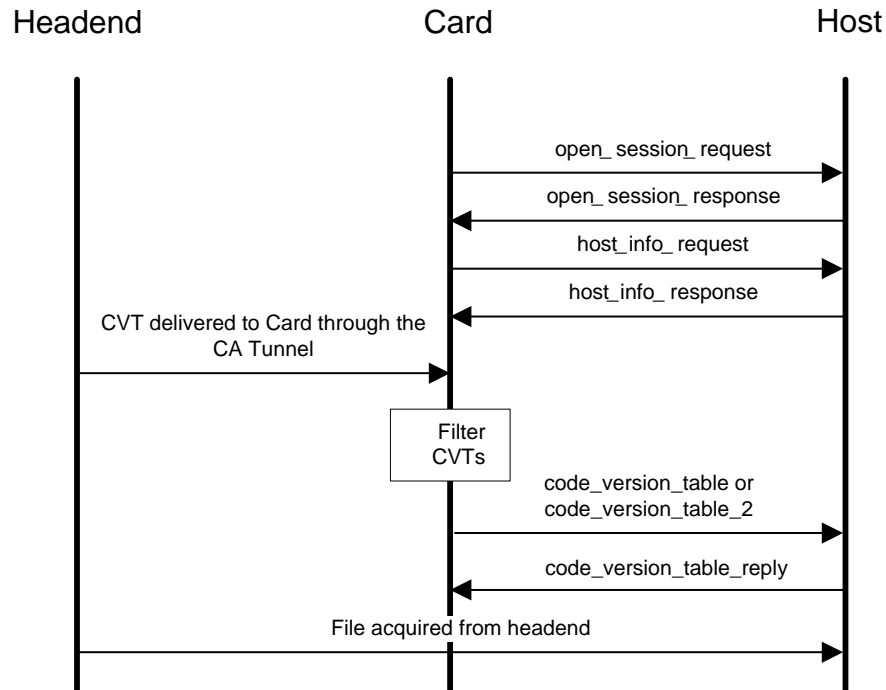


Figure 1 - CA System Flow Example

After a session is opened between the Card and the Host, the Card requests identification information from the Host by sending `host_info_request()` APDU message. The Host responds with the `vendor_id` (OUI of the Host eSTB MAC address) and `hardware_version_id` contained in the `host_info_response()` APDU. The Card uses these identifiers to filter CVTs that are received via the OOB FDC channel or DSG channel. If the CVT has defined either a `host_MAC_addr` or `host_id` field, the Card will obtain them from the Host and use these identifiers for additional filtering. Each CVT corresponds to a different hardware and software version for a given Host. The Card transmits a CVT to the Host only if it finds a match between identifiers specified in the CVT and those provided by the Host. The Host determines if a download is required by comparing the code file name in the CVT to the current code file name stored in the Host. If the code file names are different, then the Host begins the download process via the delivery method indicated in the CVT.

5.1.2.1.2 DSG Broadcast Tunnel CVT Delivery

In Advanced DSG mode, the CVT, `code_version_table2()` APDU, can be placed in a Broadcast Tunnel with a DSG Broadcast ID Value = 5 [DSG] and delivered directly to the Host. In this method the CVT is part of a secure PKCS#7 data structure signed with the MSO Code Verification Certificate (CVC). The PKCS#7 structure containing the CVT is embedded within an MPEG-2 section. This structure is shown in Table 1. The MPEG-2 section is encapsulated in a UDP datagram utilizing the BT header as defined in [DSG]. The Host processes this tunnel directly and is responsible for filtering the CVT. The Card is not used to filter the CVT in this scenario.

Note: The DSG Broadcast tunnel with Broadcast ID Value = 5 is also used to deliver the XAIT MPEG-2 section defined in [OCAP].

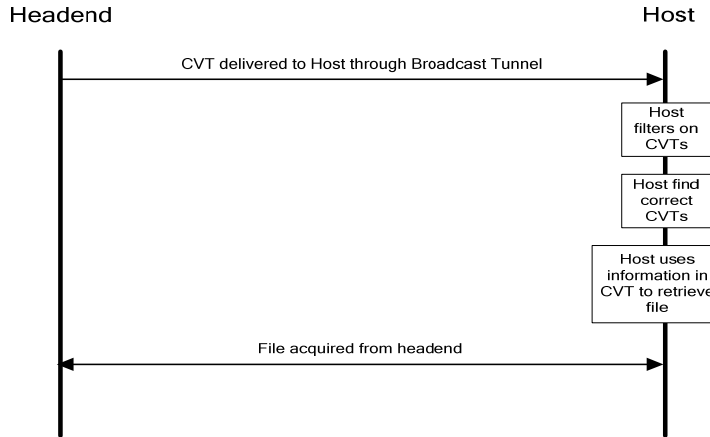


Figure 2 - DSG CVT Delivery Example

5.1.2.1.3 MPEG Section wrapper for DSG Delivery of the CVT

The following describes the MPEG Section wrapper that is used when a CVT is delivered directly to the Host via a DSG Broadcast tunnel with Broadcast ID Value = 5.

Table 1 - Code Version Table MPEG Section Header

	Bits	Format
<code>code_version_table_header(){</code>		
<table_id< td=""> <td>8</td> <td>uimsbf value 0xD9</td> </table_id<>	8	uimsbf value 0xD9
section_syntax_indicator	1	'0'
private_indicator	1	'0'
reserved	2	'11'
private_section_length	12	uimsbf
for(i=0;i<private_section_length;i++){		
PKCS#7 CVT File()		
}		
}		

- table_ID** 0xD9 This 8-bit field, the value of which identifies the Private Table this section belongs to.
- section_syntax_indicator** 0b When set to zero, it indicates that the private_data_bytes immediately follow the private_section_length field.
- private_indicator** 0b This is a 1-bit user-definable flag that is not used at this time and is set to zero.
- private_section_length** A 12-bit field. It specifies the number of bytes in the private section immediately following the private_section_length field up to the end of the private_section. The value in this field cannot exceed 4093 (0xFFD).
- PKCS#7 CVT File()** The PKCS#7 secure structure containing the CVT formatted as *code_version_table2()* APDU and defined in Section 7.3.

5.1.3 Trigger Validation

Once a code image download trigger is received via eCM configuration file, SNMP, or CVT, the triggering mechanism must be validated by the Host device before the actual code file transfer can be initiated. The Host

SHALL consider a download trigger to be valid only if the triggering mechanism has an associated Mfg CVC and/or Co-signed CVC that have been verified.

Triggers received via eCM configuration file, SNMP or CVT "Download Now" are considered immediate triggers and SHALL be immediately validated.

In the case of CVT "Deferred Download", the trigger validation SHALL be postponed until the deferral time has expired, as determined by the OCAP monitor application.

5.1.4 Trigger Contention

Once the Host has validated the download trigger and initiated the code file download process, it SHALL ignore all other triggers delivered to it by any signaling mechanism until the download process has completed or has timed out based on a failure to download. The start of the download process will be considered to be the time at which the initial trigger is validated. The process will be considered to be complete when the downloaded code image has been validated and becomes the active operational code or after the download terminates because of errors. In the case of CVT-Initiated "Deferred Download", the Host SHALL continue to process configuration file, SNMP and CVT triggers during the deferral interval. In the case of a CVT-Initiated "Deferred Download", the Host MAY download a new image into a cache during the deferred interval. If the Host device receives a valid trigger from another source during the deferral interval, the device SHOULD cancel the impending deferred download.

5.2 Code Image Download Delivery Methods

There are currently two different methods of code image download delivery. One method uses DSM-CC data carousels carried in the FAT channel or a DSG Tunnel, and the other method utilizes TFTP file transfer. DSM-CC data carousels can be used when large populations of Hosts need to be upgraded at the same time without causing a contention for system server resources. The data carousel is available to multiple Hosts at the same time. With TFTP file transfer, each Host must retrieve its image upgrade file, either from the software upgrade TFTP server or the eCM configuration file TFTP, independent of any other device on the network. The Host SHALL support a single secured firmware image download that is used for the entire device.

5.2.1 eSTB DSM-CC Carousels

The download protocol described in this section is based on the DSM-CC data carousel as defined in [13818-6]. A DSM-CC data carousel can be placed in an In-band FAT channel or a DSG Tunnel [DSG] for delivery to the eSTB. Data carousels delivered via DSG Tunnels can only be used by devices that support Type 2, Version 1 of the System Control Resource operating in Advanced DSG mode.

All code objects delivered over an In-Band FAT channel or a DSG Tunnel will be transported in a DSM-CC data carousel according to the DSM-CC specification [13818-6]. The Download Information Indication message used in these carousels is defined in Annex A of this document.

5.2.1.1 In-Band FAT Channel Data Carousel

A FAT Channel Data Carousel is a DSM-CC data carousel that is transported on an In-Band FAT Channel MPEG transport stream that may also contain video, audio, and data elementary streams. The carousel server resides in the headend and creates DSM-CC compliant data carousels containing monolithic code images using MPEG2 elementary streams. These images can be mixed together within the same carousel or sent on a separate carousel. Each data carousel is carried in its own elementary stream with an assigned PID. Elementary streams are multiplexed together in an MPEG transport stream, which is then modulated onto a specific carrier frequency within the cable plant channel line-up.

Locators for this type of download carousel include the following:

- **Source_id** – When `source_id` is contained in the CVT, it is used as an index into the Virtual Channel Table (VCT) defined in [SCTE 65]. From the VCT, the frequency, modulation mode, and MPEG program number associated with the elementary stream carrying the data carousel can be determined. This type of locator can only be used if the device has acquired all of the SCTE 65 SI tables, as the `source_id` is a pointer into these tables.
- **Frequency, modulation and PID** – The frequency, modulation mode and PID parameters in the CVT are used to directly determine the MPEG stream where the carousel is located. To use these locators the Host does not need to have acquired any SCTE 65 SI tables, as all the information needed to locate the carousel is contained in the CVT.
- **Frequency, modulation, and MPEG program number** – The frequency, modulation mode, and MPEG program number parameters in the CVT are used to directly determine the MPEG stream where the carousel is located. To use these locators, the Host does not need to have acquired any [SCTE 65] SI tables, as all the information needed to locate the carousel is contained in the CVT.

5.2.1.2 DSG Tunnel Data Carousel

A DSG Data Carousel is a DSM-CC data carousel that is transported on an Advanced mode DSG Tunnel. The data carousel server resides in the headend and creates a DSM-CC compliant data carousel composed of MPEG-2 sections.

When operating in Advanced DSG mode, the Host uses the `application_id` sent in the *code_version_table2()* APDU to find the corresponding Application Tunnel defined in the *DSG_directory()* APDU (DSG Resource Version 1) or may obtain the DSG Tunnel parameters directly from the CVT. Otherwise, the Host locates the DSG Tunnel via the *configure_advanced_dsg()* APDU (Extended Channel Resource Version 2, 3 or 4) in order to open the data carousel. Each of the carousel MPEG2 sections is sent in the DSG Tunnel within UDP over IPv4 utilizing the `DSG_Carousel_Header` as defined in [DSG].

Acquisition of a DSG Tunnel and parsing and processing of the DCD message are not within the scope of this document and are defined within [DSG] and [CCIF2.0].

Type 1, Version 1 of the System Control Resource does not support code download via DSG Tunnels.

5.2.2 eSTB TFTP File Transfer

Download via TFTP file transfer can be signaled to the eSTB using Type 2, Version 1 of the System Control Resource. In this case the *code_version_table2()* APDU is used to provide the Host with a code image file name and an IP address of the TFTP server containing the code image. When code download via TFTP is signaled in the CVT, it is the responsibility of the eCM to initiate and control the TFTP file transfer as defined in [DOCSIS].

The Host eSTB entity SHALL NOT initiate any CVT-triggered TFTP downloads (i.e., the source address of the Ethernet frame associated with a code image TFTP download must not be that of the eSTB and the source address of the IP packet associated with a code image TFTP download must not be that of the eSTB). If the eCM is not able to contact the defined TFTP server utilizing the provided server IP address, then download via this method is not possible.

6 SYSTEM CONTROL RESOURCE

The System Control resource is used for code image download operations triggered by CA System CVT delivery. This section provides details of the Card/Host Interface messages.

When the Card sends the *host_info_request()* APDU, the Host replies with the *host_info_response()* APDU containing the Host-specific identifiers to the Card. After receiving the *host_info_response()* APDU, the Card SHALL start filtering all CVTs it receives from the network and passing them to the Host using the *code_version_table()* APDU or *code_version_table2()* APDU.

The Card SHALL send a CVT using the *code_version_table()* APDU to the Host only if the following criteria are met:

- The resource identifier value negotiated with the Host is 0x002B0041.
- CVT vendor_id matches Host vendor_id, and CVT hardware_version_id matches Host hardware_version_id.

The Card SHALL send a CVT using the *code_version_table2()* APDU to the Host only if the following criteria are met:

- The resource identifier value negotiated with the Host is 0x002B0081.
- CVT protocol version is supported by the Card.
- CVT vendor_id matches Host vendor_id, and CVT hardware_version_id matches Host hardware_version_id.

If the CVT meets the conditions for sending the *code_version_table2()* APDU, the Card SHALL apply additional filtering when any of the optional identifier fields are present:

- CVT host_MAC_addr is non-zero AND matches the Host MAC address of the Host, or
- CVT Host_id is non-zero AND matches the Host ID of the Host.

The Card is not required to parse any of the information within the number_of_objects loop of the *code_version_table2()* with a protocol_version = 02 message. It is the responsibility of the Host to determine which if any of the objects defined within the loop are applicable and download object(s) as needed.

The Host SHALL respond to the receipt of the *code_version_table()* APDU or *code_version_table2()* APDU, either with an Acknowledgement or an appropriate error code message in the *code_version_table_reply()*. The Card SHALL continue to transmit the *code_version_table()* APDU or *code_version_table2()* APDU until it receives the *code_version_table_reply()* message.

The Host SHALL compare the group_id stored in persistent memory with the group_ID in the *code_version_table2()* APDU when descriptor_tag == 0x04 is present.

- The Host SHALL use the default value of 0x0000 for comparison if no group_id has been assigned via the Generic Feature resource [CCIF2.0].
- The Host SHALL operate as though a match has occurred with the group_id value stored in persistent memory (either default value or received via the Generic Feature resource [CCIF2.0]) when descriptor_tag == 0x04 is absent from the *code_version_table2()* APDU.

When *code_version_table2()* with a protocol_version = 01 is utilized, the Host SHALL initiate a download via the indicated delivery method in the *code_version_table2()* APDU if both of the following conditions are met:

- The code file names or code file name lengths in the CVT do not match the current code file name or name length stored in non-volatile memory.
- The group_id value stored or assumed by the Host matches the group_ID in the *code_version_table2()* APDU.

When *code_version_table2()* with a protocol_version = 02 is utilized, the Host SHALL only initiate a download of an individual code object defined in the CVT via the indicated delivery method if both of the following conditions are met:

- The code file name or code file name length does not match any code image stored in non-volatile memory.
- The group_id value stored by the Host matches the group_ID in the *code_version_table2()* APDU.

The Host SHALL initiate a download via the indicated delivery method in the *code_version_table()* APDU if the code file name or code file name length in the CVT does not match the current code file name or name length stored in non-volatile memory.

If System Control resource identifier value 0x002B0081 (Type 2, Version 1) has been negotiated and the host_MAC_addr in the *code_version_table2()* APDU is present and is non-zero, the Card SHALL determine the actual Host MAC address by sending the *diagnostic_req()* APDU with diagnostic_id = 0x03 of the Generic Diagnostic resource [CCIF2.0].

If the Host MAC address obtained in the *diagnostic_cnf()* APDU, the vendor_id, the hardware_version_id and any other optional identifiers match the corresponding parameters in the CVT, the Card SHALL send the *code_version_table2()* APDU to the Host.

If the host_MAC_addr is not present in the CVT or has the value zero, then the Card SHALL NOT use this parameter for filtering.

If System Control resource identifier value 0x002B0081 (Type 2, Version 1) has been negotiated and the host_ID in the CVT is present and non-zero, the Card SHALL determine the Host_ID by extracting it from the authenticated Host Device Certificate as described in [CCCP2.0].

If the Host_ID extracted from the Authenticated Device Certificate, the vendor_id, the hardware_version_id and any other optional identifiers match the corresponding parameters in the CVT, the Card SHALL send the *code_version_table2()* APDU to the Host.

If the host_ID is not present in the CVT or has the value zero, then the Card SHALL NOT use this parameter for filtering.

When operating in Advanced DSG Mode, where sessions to the DSG Resource and Extended Channel version 5 or 6 have been opened, if the Host is provided with an Application ID in the *code_version_table2()* APDU (location_type = 0x04), it SHALL parse the *DSG_directory()* APDU to determine the correct parameters and open the indicated DSG Application tunnel. It is the responsibility of the Host to parse all packets received on the DSG Application Tunnel and determine the packets that are applicable for the download. If the provided Application ID is not contained in the DCD message, then the download for the Host is not applicable. The Card SHOULD parse the DCD message to determine if the defined Application tunnel exists before passing the CVT to the Host.

When operating in Advanced DSG Mode, where a session to the Extended Channel Resource version 2, 3 or 4 has been opened, if the Host is provided with an Application ID in the *code_version_table2()* APDU (location_type = 0x04), then it SHALL issue a *DSG_message()* APDU to the Card requesting the DSG Application Tunnel parameters associated with the Application ID. The Card then parses the DCD, acquires the applicable Application Tunnel parameters and responds to the Host with a *configure_advanced_dsg()* APDU. On receipt of the *configure_advanced_dsg()* APDU from the Card, the Host parses the message to acquire the correct parameters and

open the indicated DSG Application tunnel. It is the responsibility of the Host to parse all packets received on the DSG Application Tunnel and determine the packets that are applicable for the download. If the provided Application ID is not contained in the DCD message, then the download for the Host is not applicable. The Card SHOULD parse the DCD message to determine if the defined Application tunnel exists before passing the CVT to the Host.

When operating in Advanced DSG Mode, if the Host is provided with a DSG Tunnel Address in the *code_version_table2()* APDU (location_type = 0x03), it SHALL use these parameters to open the indicated DSG Tunnel. It is the responsibility of the Host to parse all packets received on the DSG Tunnel and determine the packets that are applicable for the download.

If the Host is operating in OOB FDC mode, then the Card SHOULD NOT pass a CVT to the Host indicating a DSG Tunnel download (location_type = 0x03 or 0x04). If the Host receives a CVT that defines a DSG Tunnel download and is not operating in DSG Advanced mode, it SHALL send a *code_version_table_reply()* APDU message indicating an error (0x02 – Other parameter error) and not initiate a download.

In TFTP downloads, the Host is provided with a code image file name and the IP address of the TFTP server containing the code image. The Host uses the IP address and the file name to initiate the file download using the eCM. The eCM performs a TFTP download as defined by [BPI+], utilizing the signing and validation methods defined in Section 7 of this document.

6.1 Resource Identifier

The Host SHALL provide the System Control Resource using the identifier(s) defined in Table 2. The Card SHALL open a session to the System Control Resource using the identifier(s) defined in Table 2. The Card SHALL NOT close the session to the System Control Resource once it is established. The Host SHALL support only one session to the System Control Resource.

Table 2 - Resource Identifier

Resource	Mode	Class	Type	Version	Identifier (hex)
System Control	S-Mode/M-Mode	43	1	1	0x002B0041
System Control	S-Mode/M-Mode	43	2	1	0x002B0081
Notes: Type 1 Version 2, defined in OC-SP-CCIF2.0-I03-051117, has been deprecated. Type 1 Version 3, defined in OC-SP-CCIF2.0-I05-060413, has been deprecated.					

6.2 Application Objects (APDUs)

The following table is a list of the APDUs that are required for the System Control Resource.

Table 3 - Application Protocol Data Units

APDU_tag	Type, Version Info	Tag value (hex)	Resource	Direction Host <-> Card
Host_info_request	T1V1, T2V1	9F9C00	System Control	←
Host_info_response	T1V1, T2V1	9F9C01	System Control	→
Code_version_table	T1V1	9F9C02	System Control	←
Code_version_table_reply	T1V1, T2V1	9F9C03	System Control	→

APDU_tag	Type, Version Info	Tag value (hex)	Resource	Direction Host <-> Card
Host_download_control	T1V1, T2V1	9F9C04	System Control	→
Code_version_table2	T2V1	9F9C05	System Control	←

Notes: The *code_version_table()*, *code_version_table2()*, and *code_version_table_reply()* APDUs are utilized to support Host firmware download. The Code Version Table (*code_version_table*) is now defined as two unique APDUs.

A *code_version_table()* APDU tag value of 0x9F9C02 is utilized only with System Control resource identifier values of 0x002B0041.

A *code_version_table2()* APDU tag value of 0x9F9C05 is utilized only with a System Control resource identifier value of 0x002B0081.

A *code_version_table()* APDU with a tag value equal to 0x9F9C02 SHALL be supported by S-Cards and M-Cards operating in S-Mode/M-Mode.

A *code_version_table2()* APDU with a tag value equal to 0x9F9C05 MAY be used by S-Cards or M-Cards operating in S-Mode.

A *code_version_table2()* APDU with a tag value equal to 0x9F9C05 SHALL be supported by M-Cards operating in M-Mode.

The Host SHALL advertise and support both Type 1 and Type 2 of the System Control Resource.

The Card SHALL open a session to either Type 1 or Type 2 of the System Control Resource based on its knowledge of the network.

System Control resource identifier values 0x002B0042 and 0x002B0043 have been deprecated.

6.2.1 host_info_request

The Card SHALL send the *host_info_request()* APDU to the Host to determine its vendor ID and hardware version ID, after a session to the System Control Resource has been opened. Cards implemented per SCTE28/CCIF 1.0 will set the length field to 1.

Table 4 - host_info_request (Type 1 Version 1)

Syntax	# of bits	Mnemonic
<i>host_info_request()</i> {		
<i>host_info_request_tag</i>	24	uimsbf
<i>length_field()</i>		
<i>supported_download_type</i>	8	uimsbf
}		

host_info_request_tag 0x9F9C00

supported_download_type Defines the type of Common Download method utilized by the Headend.

- 0x00 OOB Forward Data Channel method
- 0x01 Reserved
- 0x02 DOCSIS only
- 0x03 – 0xFF Reserved

Table 5 - host_info_request (Type 2 Version 1)

Syntax	# of bits	Mnemonic
host_info_request() { host_info_request_tag	24	uimsbf
length_field() reserved }	8	uimsbf

host_info_request_tag 0x9F9C00

6.2.2 host_info_response

The Host SHALL respond to the *host_info_request()* APDU with its vendor ID, hardware version ID, and any optionally requested information using the *host_info_response()* APDU.

Table 6 - host_info_response

Syntax	# of bits	Mnemonic
host_info_response() { host_info_response_tag	24	uimsbf
length_field() vendor_id	24	uimsbf
hardware_version_id	32	uimsbf
number_of_descriptors	8	uimsbf
for(i=0;i<number_of_descriptors;i++){		
descriptor_tag	8	uimsbf
descriptor_len	8	uimsbf
descriptor_data() } }		

host_info_response_tag 0x9F9C01

vendor_id Organizationally Unique Identifier (OUI) assigned to the Host device vendor by the IEEE. A value of 0x000000 is not valid.

hardware_version_id Unique hardware identifier assigned to each type of hardware from a particular vendor. The hardware_version_id assigned by the vendor must be a unique number (for a given vendor_id) and correspond on a one-to-one basis with the Model ID of the Host.

number_of_descriptors Indicates the number of descriptors defined in the following fields.

descriptor_tag 0x00 descriptor_data is Host proprietary data. The maximum value for descriptor_len is 128.
 0x01 – 0x7F Reserved for future standardization.
 0x80 – 0xFF Optional, for use by Card-Host pairs, where both Card and Host support the same implementation of the Specific Application Resource. Other Card-Host pairs will skip these descriptors using descriptor_len value.

descriptor_len Length of descriptor_data field.

descriptor_data() Variable length field for the descriptor_tag data.

6.2.3 code_version_table

This APDU is used for Single Stream Hosts and is applicable only for In-Band FAT Channel downloads.

Table 7 - code_version_table (Type 1 Version1)

Syntax	# of bits	Mnemonic
code_version_table() {		
code_version_table tag	24	uimsbf
length_field()		
number of descriptors	8	uimsbf
for(i=0;i<number of descriptors;i++){		
descriptor_tag	8	uimsbf
descriptor_len	8	uimsbf
descriptor_data()		
}		
download_type	4	uimsbf
download_command	4	uimsbf
frequency_vector	16	uimsbf
transport_value	8	uimsbf
reserved	3	uimsbf
PID	13	uimsbf
code_file_name_length	8	uimsbf
for(i=0;i<software_filename_length;i++){		
code_file_name_byte	8	uimsbf
}		
code_verification_certificates()		
}		

code_version_table_tag 0x9F9C02

number_of_descriptors A minimum of two descriptors are required; mandatory descriptors are vendor_id and hardware_version_id.

descriptor_tag

- 0x00 vendor_id (mandatory, descriptor_len = 3). Unique Identifier (the vendor's OUID) assigned to each vendor.
- 0x01 hardware_version_id (mandatory, descriptor_len = 4). Unique Hardware identifier assigned to each type of hardware from a particular vendor.
- 0x02 Host proprietary data. The maximum value for descriptor_len is 16.
- 0x03 – 0x7F Reserved for future standardization.
- 0x80 – 0xFF Optional, for use by Card-Host pairs where both the Card and Host support the same implementation of the Specific Application Resource. Other Card-Host pairs will skip these descriptors using descriptor_len value.

descriptor_len Length of descriptor_data field (bytes).

descriptor_data() Variable length field for the descriptor_tag data.

download_type

- 0x00 One-way, broadcast
- 0x01 Always On Demand
- 0x02 Reserved
- 0x03 Download unsupported – no code object available. In this case, the code_file_name_length, frequency_vector and PID parameters must be equal to zero.
- 0x04 – 0xFF Reserved

download_command

- 0x00 Download now
- 0x01 Deferred download
- 0x02 – 0xFF Reserved

frequency_vector Frequency of the download carousel. The frequency is coded as the number of 0.25 MHz intervals. If download_type parameter = 0x02, this parameter is set to 0.

transport_value

- 0x00 Reserved
- 0x01 FAT channel/QAM 64

	0x02 FAT channel/QAM 256
	0x03 – 0xFF Reserved
PID	MPEG Transport Stream packet identifier for the code file. If download_type parameter = 0x02, this parameter is set to 0.
code_file_name_length	Length of code file name.
code_file_name_byte	Name of software upgrade file. This is the name of the code file that is on the broadcast carousel as well as in Host NVM. For download_type = 0x00 and 0x01, in the <i>code_version_table()</i> APDU, the DSM-CC data carousel will carry the Code File Name in the Download Info Indication message, module_info_byte loop. All bytes in the code_file_name_byte loop, in the <i>code_version_table()</i> APDU and the associated byte in the DownloadInfoIndication message module Info Byte loop must be the same.
code_verification_certificates	Code Verification Certificates per [OCSEC]. May contain Mfg CVC, Cosigner CVC or both.

6.2.4 code_version_table_reply

When the Host receives a *code_version_table()* or *code_version_table2()* APDU from the Card, it SHALL respond with the *code_version_table_reply()* APDU. This response serves as an acknowledgement to the receipt of the CVT and an error code if necessary. When the Card receives the *code_version_table_reply()* APDU, it SHALL stop sending *code_version_table()* or *code_version_table2()* APDU to the Host until a different version of the CVT is received from the headend or the Host or Card are rebooted.

Table 8 - code_version_table_reply

Syntax	# of bits	Mnemonic
code_version_table_reply() { code_version_table_reply_tag	24	uimsbf
length_field() host_response }	8	uimsbf

code_version_table_reply_tag	0x9F9C03
host_response	0x00 Acknowledgement, no error
	0x01 Invalid vendor ID or hardware version ID
	0x02 Other parameter error
	0x03 Invalid Host MAC address (Valid only for Type 2 Version 1)
	0x04 Invalid Host_ID (Valid only for Type 2 Version 1)
	0x050xFF Reserved

6.2.5 code_version_table2

The APDU consists of two versions, Protocol Version 1 and Protocol Version 2, as defined in Table 9. Protocol Version 1 and 2 of this APDU is used for M-Cards operating in either S-mode or M-mode and can also be used by S-Cards.

Table 9 - code_version_table2 (Type 2 Version 1)

Syntax	# of bits	Mnemonic
code_version_table2() {		
code_version_table2_tag	24	uimsbf
length_field()		
protocol_version	8	uimsbf
configuration_count_change	8	uimsbf
number of descriptors	8	uimsbf
for(i=0;i<number of descriptors;i++){		
descriptor_tag	8	uimsbf
descriptor_len	8	uimsbf
descriptor_data()		
}		
if (protocol_version == 01) {		
download_type	4	uimsbf
download_command	4	uimsbf
If (download_type == 00) {		
location_type	8	uimsbf
if (location_type == 0) {		
source_ID	16	uimsbf
}		
if (location_type == 1) {		
frequency_vector	16	uimsbf
modulation_type	8	uimsbf
reserved	3	uimsbf
PID	13	uimsbf
}		
if (location_type == 2) {		
frequency_vector	16	uimsbf
modulation_type	8	uimsbf
program_number	16	uimsbf
}		
}		
if download_type == 01) {		
location_type	8	uimsbf
if (location_type == 3) {		
/* DSG Tunnel Address */		
DSG_Tunnel_address	48	uimsbf
source_ip_address	128	uimsbf
destination_ip_address	128	uimsbf
source_port_number	16	uimsbf
destination_port_number	16	uimsbf
}		
if (location_type == 4) {		
/* DSG Application ID*/		
application_id	16	uimsbf
}		
}		
if download_type == 02) {		
tftp_server_address	128	uimsbf
}		
code_file_name_length	8	uimsbf
for(i=0;i<code_file_name_length;i++){		
code_file_name_byte	8	uimsbf
}		
}		
if (protocol_version == 02) {		
number_of_objects	8	uimsbf
for(i=0;i<number_of_objects;i++){		
download_type	4	uimsbf
download_command	4	uimsbf
If (download_type == 00) {		
location_type	8	uimsbf
if (location_type == 0) {		
source_ID	16	uimsbf
}		
}		
}		

Note: In some events (for example, a failover or hot swap at the headend), discontinuities in the value of configuration change count may occur. After any event that can cause a discontinuity in the configuration change count, the headend must ensure that the configuration change count is incremented (modulo the field size) between two subsequent CVT messages (even if the CVT message does not change). This is done to ensure that, after a failover or hot swap in the headend, the new configuration change count does not match the configuration change count used before the failover event. The Card SHALL send a *code_version_table2()* APDU to the Host for verification if download is required whenever a CVT is received with an incremented (modulo the field size) configuration_count_change parameter.

number_of_descriptors	A minimum of two descriptors are required; mandatory descriptors are vendor_id and hardware_version_id.
descriptor_tag	<p>0x00 vendor_id (mandatory; descriptor_len = 3). Unique identifier (OUI of the Host eSTB MAC address) assigned to each vendor. Host sends the vendor ID to the Card to allow the Card to filter the CVT. A value of 0x000000 is not valid. It is expected that there will be a single vendor_id descriptor per CVT.</p> <p>0x01 hardware_version_id (mandatory; descriptor_len = 4). Unique Hardware identifier assigned to each type of hardware from a particular vendor. Host sends the hardware version ID to the Card to allow the Card to filter the CVT. This can be transmitted to the headend by the Card. A value of 0x00000000 is not valid. It is expected that there will be a single hardware_version_id descriptor per CVT.</p> <p>0x02 host_MAC_addr (optional; descriptor_len = 6). Host MAC address used for additional filtering if present and non-zero. (Deprecated)</p> <p>0x03 host_ID (optional; descriptor_len = 5). Host device's unique identification number used for additional filtering if present and non-zero. (Deprecated)</p> <p>0x04 group_ID (optional: descriptor_len = 2). Group ID assigned to the Host using the Generic Feature resource (see [CCIF2.0]). NOTE: Absence of this descriptor_tag in the APDU is the same as having a matching Group ID. It is expected that there will be no more than a single group_ID descriptor.</p> <p>0x05 – 0x7F Reserved for future standardization.</p> <p>0x80 – 0xFF Optional, for use by Card-Host pairs where both the Card and Host support the same implementation of the Specific Application Resource. Other Card-Host pairs will skip these descriptors using descriptor_len value.</p>
descriptor_len	Length of descriptor_data field (bytes).
descriptor_data()	variable length field for the descriptor_tag data.
number_of_objects	Identifies the total number of objects defined in the CVT (only applicable if protocol version = 02).
download_type	Code file delivery method <ul style="list-style-type: none"> 0x00 In-Band FAT Channel DSM-CC data carousel 0x01 DSG Tunnel DSM-CC data carousel 0x02 DOCSIS TFTP 0x03 – 0x0F Reserved
download_command	<ul style="list-style-type: none"> 0x00 Download Now 0x01 Deferred Download 0x02 – 0x0F Reserved

When **download_command** = 0x00 in the *code_version_table2()* APDU, the Host SHALL initiate download if the vendor_id and hardware_version_id and optionally either a host_MAC_addr, host_id, or group_id in the descriptor_data matches that of the Host and the code_file_name does not match that stored in NVM.

The Host SHALL NOT initiate download if there is a match of the vendor_id and hardware_version_id, but the code_file_name matches that stored in NVM.

When **download_command** = 0x01 in the *code_version_table2()* APDU, the Host SHALL defer initiation of download according to policies set in an OCAP Monitor Application as defined in [OCAP].

In the event that a Monitor Application is not available or no policies have been set, the Download Now scenario SHALL apply.

Before the initiation of the actual download, the esafeDevServiceIntImpact MIB object SHALL be set to a value other than significant (1). The Host device continues to defer the code download while the esafeDevServiceIntImpact MIB object is set to significant (1).

location_type	Determines nature of the locator for DSM-CC data carousel carrying code file. 0x00 Carousel located by source_id. 0x01 Carousel located by frequency vector and PID. 0x02 Carousel located by frequency and program number. 0x03 Carousel located by DSG Tunnel parameters (DSG Tunnel Address). 0x04 Carousel located by DSG Application Tunnel ID (DSG Application ID). 0x05 – 0xFF Reserved
source_ID	The VCT source ID that is associated with each program source. The source ID is utilized to locate the frequency on which the DSM-CC data carousel is multiplexed.
frequency_vector	Frequency of the download carousel. The frequency is coded as the number of 0.25 MHz intervals.
modulation_type	0x00 Reserved 0x01 FAT Channel/QAM 64 0x02 FAT Channel/QAM 256 0x03 – 0xFF Reserved
PID	Transport Stream packet identifier for the code file.
program_number	Defines the program number in the transport stream that identifies the DSM-CC data carousel.
DSG_Tunnel_address	MAC address of the DSG tunnel.
source_ip_address	The source IP address associated with the download applicable to the Host. This is utilized to allow the Host to better filter packets in the tunnel.

When **source_IP_address** = 0 in the *code_version_table2()* APDU, the Host SHALL ignore the source IP address of the packet and provide an-additional filter at either the destination IP address (if defined) or a layer above the IP layer (e.g., Port and/or MPEG section filtering).

destination_ip_address The destination IP address associated with the download applicable to the Host. This is utilized to allow the Host to better filter packets in the tunnel.

When **destination_IP_address** = 0 in the *code_version_table2()* APDU, the Host SHALL ignore the destination IP address of the packet and provide an additional filter at either the source IP address (if defined) or a layer above the IP layer (e.g., Port and/or MPEG section filtering).

source_port_number	The UDP source port number associated with the download applicable to the Host. This is utilized to allow the Host to better filter packets in the tunnel.
When source_port_number = 0 in the <i>code_version_table2()</i> APDU, then the Host SHALL NOT apply any source port layer filtering.	
destination_port_number	The UDP destination port number, associated with the download applicable to the Host. This value is utilized to allow the Host to better filter packets in the tunnel.
When destination_port_number = 0 in the <i>code_version_table2()</i> APDU, then the Host SHALL NOT apply destination port layer filtering.	
application_id	A DSG data stream identifier, associated with Application Tunnel information. Applicable only for the Host operating in DSG Advanced mode. This is utilized to allow the Host to build a DSG data stream routing table from information acquired from the DCD.
tftp_server_address	The IP address of the TFTP server where the code image resides. The code file name, as defined via the <i>code_file_name_byte</i> field, contains the complete directory path and name of the file to download. The address is 128 bits in length to support IPv6. IPv4 addresses will be placed in the lower 32 bits of this field with all other bits set to zero.
object_type	Identifies the type of object signaled in the download as one of the following (only applicable if protocol version = 02): <ul style="list-style-type: none"> 0x00 – Firmware Object 0x01 – Application Object 0x02 – Data Object 0x03 – Other 0x04 – 0x7FFF – Reserved for future use 0x8000-0xFFFF – Vendor defined
object_data_length	Length of object data. Number of bytes that compose the object data that follows (only applicable if protocol version = 02).
object_data_byte	Object data, when present, may define other parameters associated with the code file object. The format and contents of the object data is vendor-specific (only applicable if protocol version = 02).
code_file_name_length	Length of code file name. A length of zero indicates that the CVT is being used for CVC delivery only.
code_file_name_byte	Name of software upgrade file. This is the name of the code file that is on the broadcast carousel or TFTP server as well as in Host NVM. For <i>download_type</i> = 0x00 and 0x01, the DSM-CC data carousel will carry the code file name in the Download Info Indication message, <i>module_info_byte</i> loop. All bytes in the <i>code_file_name_byte</i> loop and the associated byte in the Download Info Indication message <i>module Info Byte</i> loop must be the same.
number_of_cv_certificates	The number of code verification certificates.
certificate_type	Type of CVC <ul style="list-style-type: none"> 0x00 Manufacturer CVC 0x01 Co-Signer CVC 0x02 – 0xFF Reserved
code_verification_certificate	Code Verification Certificate per [OCSEC]

6.2.6 host_download_control

The Host utilizes the *host_download_control()* APDU to inform the Card of the status of the download process.

The Host SHALL send the *host_download_control()* APDU with **host_command** = 0x02 (notify headend) if it does not find the code image file on the In-Band Fat Channel, DSG DSM-CC carousel, or TFTP server, or fails to connect to the TFTP server. The Card MAY send a notification to the headend along with the **vendor_id** and **hardware_version_id** and/or **host_id**. The headend should verify that the proper code file is available and then send the CVT with appropriate information, e.g., code version and locator data.

When the Host has successfully downloaded and authenticated the code file, it SHALL send the *host_download_control()* APDU with **host_command** = 0x01 (download completed). The Card MAY send a download complete message to the headend along with the **vendor_id** and **hardware_version_id** and/or **host_id** so that the code file can be unloaded from the carousel.

If the Host cannot successfully authenticate the code file, it SHALL send the *host_download_control()* APDU with **host_command** = 0x05 (certificate failure). The Card MAY indicate this condition to the headend along with the **vendor_id** and **hardware_version_id** and/or **host_id** to inform the headend that the code download has been rejected due to certificate authentication failure.

In the event that the Host determines that the image is damaged or corrupted, it SHALL reject the newly downloaded image and send the *host_download_control()* APDU with **host_command** = 0x04 (image damaged). The Card MAY indicate this condition to the headend along with the **vendor_id** and **hardware_version_id** and/or **host_id** to inform the headend that the downloaded code file is corrupted or incomplete.

The Host MAY re-attempt to download the new code image if the maximum number of download retries (Download max retries = 3) has not been reached.

On the third consecutive failed retry of the software download attempt, the Host SHALL revert to the last known working image and proceed to an operational state.

In this case, the Host SHALL send the *host_download_control()* APDU with **host_command** = 0x03 (download max retry).

The Card MAY indicate this condition to the headend along with the **vendor_id** and **hardware_version_id** and/or **host_id** to inform the headend that a given Host had a problem with downloading a code file.

The Host SHALL verify that the downloaded image is appropriate for its hardware.

If the image is appropriate, the Host SHALL write the new software image to non-volatile storage.

After the Host has written the code file to non-volatile storage, it SHALL reboot itself and proceed to an operational state using the new code image.

In case of a problem with running a new code image, the Host MAY re-attempt to reboot itself if the maximum number of reboot retries (Reboot max retries = 3) has not been reached. On the third consecutive failed retry of the reboot, the Host SHOULD revert back to the last known working image and send the *host_download_control()* APDU with a **host_command** = 0x06 (reboot max retry). This process SHOULD NOT be executed if the Host can not revert one or more software components included into a single monolithic code object received within Host downloaded image. The Card MAY indicate this condition to the headend along with the **vendor_id** and **hardware_version_id** and/or **host_id** to inform the headend that a given Host had a problem with running a newly downloaded code file.

The Host SHALL remain capable of accepting new software downloads (without operator or user interaction) if it is unable to complete the code file transfer for any reason, even if power or network connectivity is interrupted between attempts. In the event of a failed software download attempt, the Host SHALL log the nature of the failure.

In the event of a failed software download attempt, the Card MAY report it asynchronously to the network manager.

If the Host suffers a loss of power or resets during a CVT-initiated upgrade, it SHALL resume the upgrade process without requiring manual intervention.

If the Host suffers a loss of power or resets during a CVT-initiated upgrade, it SHALL ignore the fact that a previous upgrade was in progress and restart the download per receipt of the next CVT upon reboot.

If the Host suffers a loss of power or resets during an SNMP-initiated upgrade, it SHALL be able to resume the upgrade without requiring manual intervention upon reboot.

If the Host suffers a loss of power or resets during an eCM configuration file-initiated upgrade, it SHALL ignore the fact that a previous upgrade was in progress and following action:

Do not perform an upgrade if no upgrade TLVs are present in the config file.

If upgrade TLVs are present, take the action described in the requirements in Section 7.1.3, at the time of the reboot.

Table 10 - host_download_control

Syntax	# of bits	Mnemonic
host_download_control() { host_download_control_tag length_field() host_command }	24	uimsbf
	8	uimsbf

host_download_control_tag 0x9F9C04

host_command

- 0x00 Download Started – sent when Host receives a valid trigger and initiates the download process.
- 0x01 Download Completed – sent when download complete.
- 0x02 Notify headend – the Card can send an optional message indicating that the Host could not find the download image.
- 0x03 Download max retry – sent when max retry has been reached during code file download.
- 0x04 Image damaged – sent when downloaded code file is damaged or corrupted.
- 0x05 Certificate failure – sent when CVC authentication failed.
- 0x06 Reboot max retry – sent when max retry has been reached during device rebooting.
- 0x07-0xFF Reserved

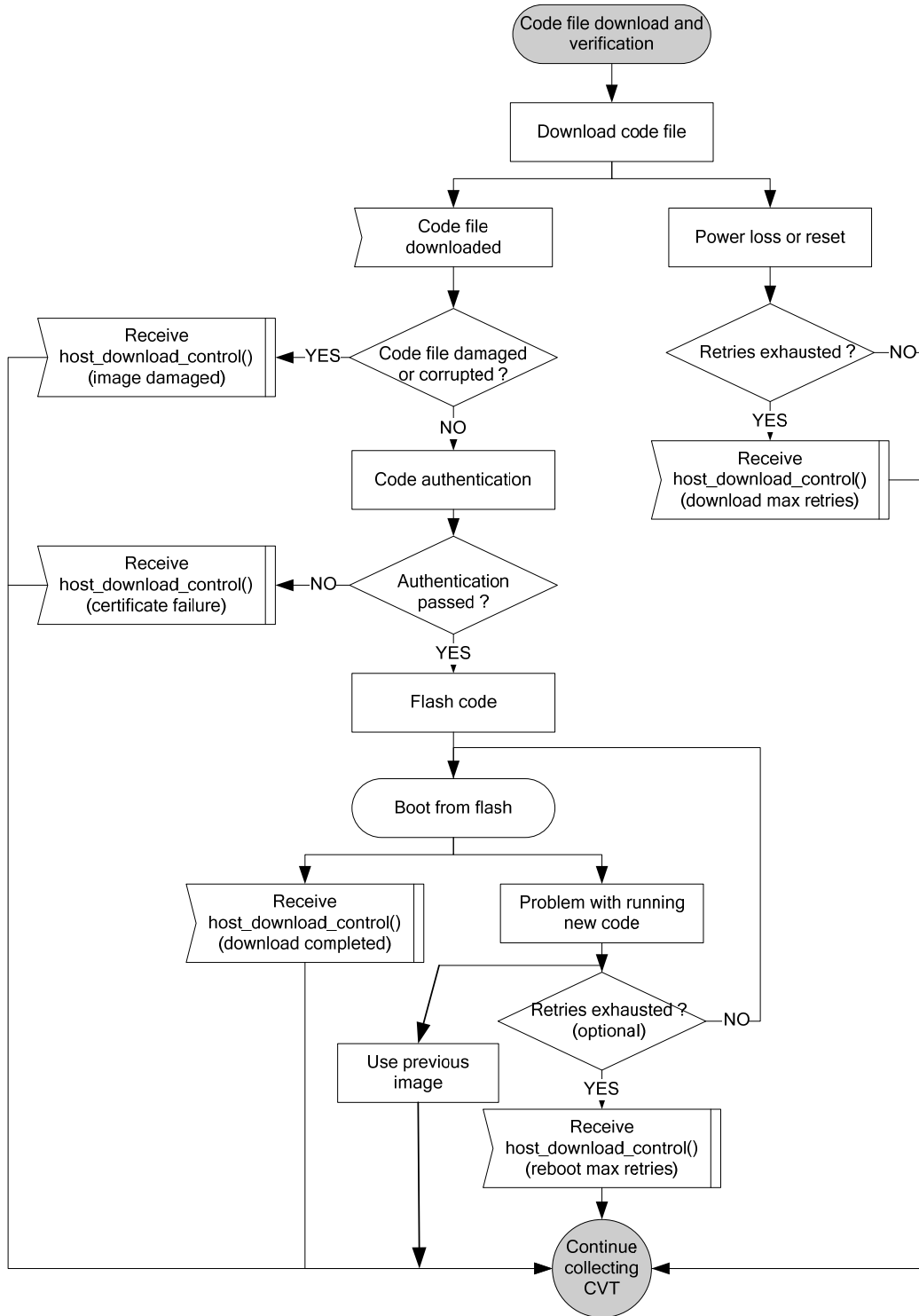


Figure 3 - Flow Chart Summarizing Code File Download and Verification

7 SECURE SOFTWARE DOWNLOAD

The Host SHALL implement the secure software download methods defined in [DOCSIS] and [BPI+] except for any changes noted in the Common Download 2.0 specification.

The Host SHALL perform code certificate verification against the CableLabs CVC Root CA and not the DOCSIS Root CA when upgrading software via the eCM or the eSTB.

7.1 Initialization of Download

In support of code verification, the eCM configuration file or the CVT is used as an authenticated means by which to initialize the code verification process. In the eCM configuration file or the CVT, the Host device receives configuration settings relevant to code upgrade verification.

7.1.1 CVC Delivery

The eCM configuration file and the CVT SHOULD always include the most up-to-date CVC applicable for the target Host device. When the eCM configuration file or CVT is used to initiate a code upgrade, it must include a CVC to initialize the Host device for accepting code files.

The Host SHALL always process a CVC in an eCM configuration file or a CVT, regardless of whether a code upgrade is required. If a code upgrade is not required, delivery of the CVC has the effect of updating the time-varying controls.

The Host SHALL discard any CVC that cannot be validated, regardless of whether a code upgrade is required.

The Host SHALL receive a valid CVC, either in an eCM configuration file or a CVT, before it will enable its ability to accept code files using any delivery method. The presence of any valid CVC is sufficient to enable further code file download operation.

The Host SHALL only store the most recent CVC values.

If older CVCs are received by whatever means, the Host SHALL ignore these CVCs and continue to allow downloads to occur.

If a download is triggered, the Host SHALL validate the code image file against the most recent CVC.

An eCM configuration file MAY contain:

- No CVC
- A Manufacturer's CVC only then.
- A Cosigner's (CableLabs or MSO) CVC only
- Both a Manufacturer's CVC and a Cosigner's CVC

When the eCM configuration file only contains a Manufacturer's CVC, the Host SHALL verify that the manufacturer's CVC chains up to the CableLabs CVC Root before accepting a code file.

When the eCM configuration file only contains a Manufacturer's CVC, the Host will only require a manufacturer signature on the code files and SHALL NOT accept code files that have been cosigned.

When the eCM configuration file contains a cosigner's CVC, the Host SHALL verify that the Cosigner CVC chains up to the CableLabs CVC Root CA before accepting a code file.

When the eCM configuration file contains a valid cosigner's CVC, it is used to initialize the device with a cosigner. Once validated, the name of the CVC's subject organizationName will become the code cosigner assigned to the Host device. In order for Host device to subsequently accept a code image, the cosigner, in addition to the Host device manufacturer, must have signed the code file.

When the eCM configuration file contains both a Manufacturer's CVC and a Cosigner's CVC, the Host SHALL verify that both CVCs chain up to the CableLabs CVC Root before accepting a code file.

A CVT delivered to the eSTB MAY contain:

- No CVC
- A Manufacturer's CVC only
- A Cosigner's (CableLabs or MSO) CVC only
- Both a Manufacturer's CVC and a Cosigner's CVC

When the CVT only contains a Manufacturer's CVC, the Host SHALL verify that the manufacturer's CVC chains up to the CableLabs CVC Root before accepting a code file.

When the CVT only contains a Manufacturer's CVC, the Host device will only require a manufacturer signature on the code files and SHALL NOT accept code files that have been cosigned.

When the CVT contains a cosigner's CVC, the Host SHALL verify that the Cosigner CVC chains up to the CableLabs CVC Root CA before accepting a code file.

When the CVT contains a valid cosigner's CVC, it is used to initialize the device with a cosigner. Once validated, the name of the CVC's subject organizationName will become the code cosigner assigned to the Host device. In order for Host device to subsequently accept a code image, the cosigner, in addition to the Host device manufacturer, must have signed the code file.

When the CVT contains both a Manufacturer's CVC and a Cosigner's CVC, the Host SHALL verify that both CVCs chain up to the CableLabs CVC Root before accepting a code file.

7.1.1.1 CVC Processing

To allow delivery of an updated CVC without requiring the Host device to process a code image file, the CVC MAY be delivered in either the eCM configuration file, as an SNMP MIB object or thru the CVT. The format of the CVC is the same whether it is in a code image file, an eCM configuration file, an SNMP MIB object or the CVT.

7.1.1.1.1 Processing the eCM Configuration File CVC

The Host device processes eCM configuration file-delivered CVCs to upgrade operating code images. When a CVC is included in the eCM configuration file, the Host device verifies the CVC before accepting any of the code upgrade settings it contains.

If the eCM configuration file includes a CVC that does not validate properly, the Host SHALL NOT download upgrade code files, whether triggered by the eCM configuration file, or via an SNMP MIB object. If the eCM configuration file includes a CVC that does not validate properly, the Host is not required to process CVCs subsequently delivered via an SNMP MIB object and SHALL NOT accept information from a CVC subsequently delivered via an SNMP MIB object.

Upon receipt of the CVC in the eCM configuration file, and after the eCM has successfully registered with the CMTS, the Host SHALL perform the following validation and procedural steps:

1. Verify that the extendedKeyUsage extension is in the CVC and includes an OID id-kp-codeSigning.

2. Check the CVC subject organization name.
 - a) If the CVC is a Manufacturer's CVC (TLV type 32), then:
 - i. If the organizationName is identical to the Host's manufacturer name, then this is the manufacturer's CVC. In this case, the Host verifies that the manufacturer's CVC validity start time is greater-than or equal-to the manufacturer's cvcAccessStart value currently held in the Host device.
 - ii. If the organizationName is not identical to the Host's manufacturer name, then this CVC is rejected, the error logged as applicable, and the CVC validation process terminated.
 - b) If the CVC is a Cosigner's CVC (TLV type 33), then:
 - i. If the organizationName is identical to the Host's current code cosigner, then this is the current cosigner's CVC and the Host verifies that the validity start time is greater-than or equal-to the cosigner's cvcAccessStart value currently held in the Host device.
 - ii. If the organizationName is not identical to the current code cosigner name, then after the CVC has been validated (and registration is complete), this subject organization name will become the Host's new code cosigner. The Host rejects a code file unless it has been signed by the manufacturer, and cosigned by this code cosigner.
3. Validate the CVC issuer signature using the CL CVC CA Public Key held by the Host.
4. Update the Host's current value of cvcAccessStart corresponding to the CVC's subject organizationName (i.e., manufacturer or cosigner) with the validity start time value from the validated CVC. If the CVC validity start time value is greater than the current value of codeAccessStart, the Host updates the codeAccessStart value with the validity start time value. All certificate parameters EXCEPT for the validity start time are no longer needed and SHOULD be discarded.
5. If any of the verification checks fail, the Host SHALL immediately halt the CVC verification process, log the error if applicable, and remove all remnants of the process up to that step.

7.1.1.1.2 Processing the SNMP CVC

The Host device processes SNMP-delivered CVCs when enabled to upgrade operating code images; otherwise, all CVCs delivered via SNMP are rejected.

Upon receipt of the CVC via SNMP, the Host SHALL perform the following validation and procedural steps:

1. Verify that the extendedKeyUsage extension is in the CVC and includes an OID id-kp-codeSigning.
2. Check the CVC subject organization name.
 - a) If the CVC is a Manufacturer's CVC, then:
 - i. If the organizationName is identical to the Host's manufacturer name, then this is the manufacturer's CVC. In this case, the Host verifies that the manufacturer's CVC validity start time is greater-than the manufacturer's cvcAccessStart value currently held in the Host device.
 - ii. If the organizationName is not identical to the Host's manufacturer name, then this CVC is rejected, the error logged as applicable, and the CVC validation process terminated.
 - b) If the CVC is a Cosigner's CVC, then:
 - i. If the organizationName is identical to the Host's current code cosigner, then this is the current cosigner's CVC, and the Host verifies that the validity start time is greater-than the cosigner's cvcAccessStart value currently held in the Host device.
 - ii. If the organizationName is not identical to the current code cosigner name, then after the CVC has been validated (and registration is complete), this subject organization name will become the Host's new code cosigner. The Host rejects a code file unless it has been signed by the manufacturer, and cosigned by this code cosigner.

3. Validate the CVC issuer signature using the CL CVC CA Public Key held by the Host.
4. Update the Host's current value of `cvcAccessStart` corresponding to the CVC's subject organizationName (i.e., manufacturer or cosigner) with the validity start time value from the validated CVC. If the CVC validity start time value is greater than the current value of `codeAccessStart`, the Host updates the `codeAccessStart` value with the validity start time value. All certificate parameters, EXCEPT for the validity start time, are no longer needed and SHOULD be discarded.
5. If any of the verification checks fail, the Host device SHALL immediately halt the CVC verification process, log the error if applicable, and remove all remnants of the process up to that step.

7.1.1.1.3 Processing the CVT CVC

The Host device processes CVT-delivered CVCs to upgrade operating code images. The Host verifies the CVC included in the CVT before accepting any of the code upgrade settings it contains.

The Host device SHALL NOT download code files when triggered by the CVT if the received CVT does not include a CVC that validates properly.

If the CVT includes a CVC that does not validate properly, the Host is not required to process CVCs subsequently delivered via an SNMP MIB object and SHALL NOT accept information from a CVC subsequently delivered via an SNMP MIB object.

Upon receipt of the CVC in a CVT, the Host SHALL perform the following validation and procedural steps:

1. Verify that the `extendedKeyUsage` extension is in the CVC and includes an OID `id-kp-codeSigning`.
2. Check the CVC subject organization name.
 - a) If the CVC is a Manufacturer's CVC (certificate type `0x00`), then:
 - i. If the organizationName is identical to the Host's manufacturer name, then this is the manufacturer's CVC. In this case, the Host verifies that the manufacturer's CVC validity start time is greater-than or equal-to the manufacturer's `cvcAccessStart` value currently held in memory.
 - ii. If the organizationName is not identical to the Host's manufacturer name, then this CVC is rejected, the error logged as applicable and the CVC validation process terminated.
 - b) If the CVC is a Co-signer's CVC (certificate type `0x01`), then:
 - i. If the organizationName is identical to the Host's current code cosigner, then this is the current cosigner's CVC and the Host verifies that the validity start time is greater-than or equal-to the cosigner's `cvcAccessStart` value currently held in memory.
 - ii. If the organizationName is not identical to the current code cosigner name, then after the CVC has been validated (and registration is complete), this subject organization name will become the Host's new code cosigner. The Host rejects a code file unless it has been signed by the manufacturer, and cosigned by this code cosigner.
3. Validate the CVC issuer signature by using the CL CVC CA Public Key held by the Host device.
4. Update the Host's current value of `cvcAccessStart` corresponding to the CVC's subject organizationName (i.e., manufacturer or cosigner) with the validity start time value from the validated CVC. If the CVC validity start time value is greater than the current value of `codeAccessStart`, the Host device updates the `codeAccessStart` value with the validity start time value. All certificate parameters, EXCEPT for the validity start time, are no longer needed and SHOULD be discarded.
5. If any of the verification checks fail, the Host device SHALL immediately halt the CVC verification process, log the error if applicable, and remove all remnants of the process up to that step.

7.1.2 Time Varying Controls

To mitigate the possibility of a Host device receiving a previous code file via a replay attack, the code image file includes a signing-time value in the PKCS #7 structure that can be used to indicate the time the code image file was signed.

The Host SHALL securely store and maintain two UTC time values, `codeAccessStart` and `cvcAccessStart`, associated with the Host manufacturer code-signing agent.

In addition to storing time-varying values for the Host device manufacturer, the Host SHALL also store and maintain a separate set of time values for the cosigner if delivered in the CVC.

The first time-varying value, `codeAccessStart`, controls validity of the CVS (digital signature on the PKCS#7 code file). The second time-varying value, `cvcAccessStart`, controls validity of the CVC.

The Host SHALL make the time-varying values available to both the eCM and the eSTB by an implementation-specific mechanism.

These values SHALL be updated whether the CVC or code image is delivered to the eCM or the eSTB.

For the purposes of this specification, the device is treated as a single integrated unit, i.e., Common Download does not distinguish between integrated or separate eCM / eSTB implementations.

The time-varying values are used to control code file access to the Host device and are defined as:

- `codeAccessStart` – a 12-byte UTC time value referenced to Greenwich Mean Time (GMT).
- `cvcAccessStart` – a 12-byte UTC time value referenced to GMT.

UTC time values in the CVC must be expressed as GMT and must include seconds. That is, they must be expressed in the following form: YYMMDDHHMMSSZ. The year field (YY) must be interpreted as 20YY.

These values will always be referenced to GMT, so the final ASCII character (Z) can be removed when stored by the Host device as `codeAccessStart` and `cvcAccessStart`.

The Host device SHALL maintain each of the time-varying values, `codeAccessStart` and `cvcAccessStart`, in a format that contains equivalent time information and accuracy to the 12-character UTC format (i.e., YYMMDDHHMMSS).

The values of `codeAccessStart` and `cvcAccessStart` corresponding to the Host device's manufacturer must NOT decrease. The value of `codeAccessStart` and `cvcAccessStart` corresponding to the cosigner must NOT decrease as long as the cosigner does not change and the Host device maintains that cosigner's time-varying control values.

7.1.3 eCM Configuration File Initiated Download

The eCM configuration file software download is initiated by defining the Software Upgrade File Name TLV and Software Upgrade TFTP Server TLV in the eCM's configuration file. If the Software Upgrade File Name in the eCM configuration file does not match the current code image of the Host device, then the Host device requests the specified file via TFTP from the Software Upgrade TFTP Server.

When a firmware download is initiated via eCM Configuration File, the Host SHALL follow the Operations Support System (OSS) requirements defined in section 5.2.7.1.1 of [eDOCSIS].

7.1.4 SNMP Initiated Download

SNMP-initiated firmware download requires that the network management station perform the following steps on objects in the docsDev MIB:

- Set docsDevSwServer to the address of the TFTP server for software upgrades.
- Set docsDevSwFilename to the file pathname of the software upgrade image.
- Set docsDevSwAdminStatus to Upgrade-from-mgt.

When a firmware download is initiated via SNMP, the Host SHALL follow the Operations Support System (OSS) requirements defined in section 5.2.7.1.1 of [eDOCSIS].

7.1.5 CVT Initiated Download (DSG Tunnel/In-Band upgrade)

When a firmware download is initiated by delivery of a CVT to the eSTB, the Host SHALL follow the Operations Support System (OSS) requirements defined in section 5.2.7.1.2 of [eDOCSIS].

7.1.5.1 CVT Initiated TFTP upgrade

In the event that a code image upgrade via TFTP is signaled in the CVT, the Host SHALL deliver the TFTP server IP address and code file name to the eCM and proceed according to the SNMP Initiated Download as defined in section 7.1.4.

7.2 Image File Structure

For secure software download, the code download file is a file built using a PKCS #7 compliant structure that has been defined in a specific format for use with a Host device and is the same regardless of download delivery method. The code file must comply with [X509] and must be DER encoded. The code file must match the structure shown in Table 11.

When downloading the CA Certificates (e.g., a CableLabs CVC CA Certificate and/or a CableLabs Device CA Certificate) as a part of the Code File, the certificates MAY be contained in the fields specified in Table 11, separated from the actual code image contained in the CodeImage field.

Table 11 - Code File Structure

Code File	Description
PKCS #7 Digital Signature {	
ContentInfo	
ContentType	SignedData
SignedData()	EXPLICIT signed-data content value: includes CVS and X.509 compliant CVCs
} end PKCS #7 Digital Signature	
SignedContent {	
DownloadParameters {	Mandatory TLV format (Type 28). (Length is zero if there is no sub-TLVs.)
MfgCACerts ()	Optional TLV for one or more DER-encoded CableLabs Manufacturer CA Certificate(s) each formatted according to CableLabs Device CA-Certificate TLV format (Type 17) Refer to Section 7.2.2.1.
clabCVCRootCACert()	Optional TLV for one DER-encoded certificate formatted according to the CableLabs CVC Root CA CA-Certificate TLV format (Type 51). Refer to Section 7.2.2.2.
clabCVCCACertificate()	Optional TLV for one DER-encoded certificate formatted according to the CableLabs CVC CA-Certificate TLV format (Type 52). Refer to Section 7.2.2.3.
}	
CodeImage()	Upgrade code image.
} end SignedContent	

7.2.1 Signed Data

The code download file will contain the information in a PKCS #7 Signed Data content type as shown below in Table 12. Though maintaining compliance to [X509], the structure used has been restricted in format to ease the processing performed by the Host device to validate the signature. The PKCS #7 SignedData must be DER encoded and exactly match the structure shown below except for any change required for DER encoding (e.g., the ordering of SET OF attributes). The Host SHALL reject the PKCS #7 code file if the PKCS #7 Signed Data does not match the DER encoded structure defined in Table 12 or signature validation fails.

Table 12 - PKCS #7 Signed Data

PKCS #7 Field	Description
SignedData {	
Version	version = 1
DigestAlgorithmIdentifiers	SHA-1
ContentInfo	
ContentType	data (SignedContent is concatenated at the end of the PKCS #7 structure)
certificates {	CableLabs Code Verification Certificates
MfgCVC	(REQUIRED for all code files)
CosignerCVC	(OPTIONAL; required for cosignatures)
} end certificates	
SignerInfos {	
MfgSignerInfo {	(REQUIRED for all code images)
Version	version = 1
IssuerAndSerialNumber	
IssuerName	
CountryName	US
OrganizationName	CableLabs
CommonName	CableLabs CVC CA
CertificateSerialNumber	<CableLabs CVC CA serial number>
DigestAlgorithm	SHA-1
AuthenticatedAttributes	
ContentType	data (contentType of signedContent)
SigningTime	UTC Time (GMT), YYMMDDHHMMSSZ
MessageDigest	Digest of the content as defined in PKCS #7
DigestEncryptionAlgorithm	RsaEncryption
EncryptedDigest	
} end mfg signer info	
CoSignerInfo {	(OPTIONAL; required for cosignatures)
Version	version = 1
IssuerAndSerialNumber	
IssuerName	
CountryName	US
OrganizationName	CableLabs
CommonName	CableLabs CVC CA
CertificateSerialNumber	<CableLabs CVC CA serial number>
DigestAlgorithm	SHA-1
AuthenticatedAttributes	
ContentType	data (contentType of signedContent)
SigningTime	UTC Time (GMT), YYMMDDHHMMSSZ
MessageDigest	Digest of the content as defined in PKCS #7
DigestEncryptionAlgorithm	RsaEncryption
EncryptedDigest	
} end CoSignerInfo	
} end SignerInfos	
} end SignedData	

7.2.2 Signed Content

The signed content field of the code file contains the code image and the download parameters field, which possibly contains additional optional items - CA Certificates (e.g., a CableLabs CVC CA Certificate and/or a CableLabs Device CA Certificate).

The final code image is in a format compatible with the destination Host device. In support of the PKCS #7 signature requirements, the code content is typed as data, i.e., a simple octet string. The format of the final code image is not specified here and will be defined by each manufacturer according to its requirements.

Each manufacturer SHOULD build its code with additional mechanisms that verify an upgrade code image is compatible with the destination Host device.

More than one CA certificate may optionally be included in the MfgCACerts field of the signed content field. For example, one CA certificate may be included to replace the CableLabs Manufacturer CA Certificate that is used for CableCard-Host authentication. Another CA certificate may be included to replace the CA certificate that is used for BPI+ authentication of the embedded DOCSIS cable modem. This CA certificate can be either a DOCSIS Manufacturer CA Certificate that is issued under the distributed CA or a CableLabs Manufacturer CA Certificate that is issued under the centralized CA.

If included in the signed content field, the certificate fields are intended to replace the CableLabs CVC Root CA or the CableLabs CVC CA Certificate(s) currently stored in the Host device. If the code download and installation are successful, the Host SHALL update its currently stored certificates with the ones contained in the MfgCACerts, clabCVCRootCACert and clabCVCCACertificate fields of the signed content field in the PKCS #7 code file.

7.2.2.1 CableLabs Device CA-Certificate TLV Format

The attribute is a string attribute containing an X.509 CA Certificate, as defined in [X509].

Type	Length	Value
17	Variable	X.509 CableLabs Device CA Certificate (DER-encoded ASN.1)

7.2.2.2 CVC Root CA CA-Certificate TLV Format

The attribute is a string attribute containing an X.509 CA Certificate, as defined in [X509].

Type	Length	Value
51	Variable	X.509 CableLabs CVC Root CA Certificate (DER-encoded ASN.1)

7.2.2.3 CVC CA-Certificate TLV Format

The attribute is a string attribute containing an X.509 CA Certificate, as defined in [X509].

Type	Length	Value
52	Variable	X.509 CableLabs CVC CA Certificate (DER-encoded ASN.1)

7.2.3 Code Signing Keys

The PKCS #7 digital signature uses the RSA Encryption Algorithm with SHA-1 [X509].

7.3 Delivery of the CVT via DSG Tunnel

For delivery of the CVT in a DSG Broadcast Tunnel, the CVT is secured by a file built using a PKCS #7 compliant structure that has been defined in a specific format for use with a Host device. The CVT file must comply with [X509] and must be DER encoded and match the structure shown in Table 13.

Table 13 - CVT File Structure

CVT File	Description
PKCS #7 Digital Signature {	
ContentInfo	
ContentType	SignedData
SignedData()	EXPLICIT signed-data content value: includes CVS and X.509 compliant CVCs
} end PKCS #7 Digital Signature	
SignedContent {	
DownloadParameters {	Mandatory TLV format (Type 28). (Length is zero)
}	
CVT()	<i>code_version_table2()</i> APDU
} end SignedContent	

7.3.1 Signed Data

The CVT file will contain the information in a PKCS #7 Signed Data content type as shown below in Table 14. Though maintaining compliance to [X509], the structure used has been restricted in format to ease the processing performed by the Host device to validate the signature. The PKCS #7 SignedData must be DER encoded and exactly match the structure shown below, except for any change required for DER encoding (e.g., the ordering of SET OF attributes). The Host SHALL reject the PKCS #7 CVT file if the PKCS #7 Signed Data does not match the DER encoded structure or signature validation fails.

Table 14 - PKCS #7 Signed Data

PKCS #7 Field	Description
SignedData {	
Version	version = 1
DigestAlgorithmIdentifiers	SHA-1
ContentInfo	
ContentType	data (SignedContent is concatenated at the end of the PKCS #7 structure)
certificates {	CableLabs Code Verification Certificates
MSOCVC or CLCVC	(REQUIRED)
} end certificates	
SignerInfos {	
SignerInfo {	(REQUIRED)
Version	version = 1
IssuerAndSerialNumber	
IssuerName	
CountryName	US
OrganizationName	CableLabs
CommonName	CableLabs CVC CA
CertificateSerialNumber	<CableLabs CVC CA serial number>
DigestAlgorithm	SHA-1
AuthenticatedAttributes	
ContentType	data (contentType of signedContent)
SigningTime	UTC Time (GMT), YYMMDDHHMMSSZ
MessageDigest	Digest of the content as defined in PKCS #7
DigestEncryptionAlgorithm	RsaEncryption
EncryptedDigest	
} end SignerInfo	
} end SignerInfos	
} end SignedData	

7.3.2 Signed Content

The signed content field of the CVT File contains the CVT.

The CVT is in a format of the *code_version_table2()* APDU. In support of the PKCS #7 signature requirements, the CVT content is typed as data, i.e., a simple octet string. The format of the final code image is not specified here and will be defined by each manufacturer according to its requirements.

7.3.3 CVT File Validation

The Host SHALL perform the following steps to validate the CVT File:

1. The Host validates the CL/MSO CVC by verifying that the:
 - a) CVC subject organizationName is identical to CableLabs/MSO.
 - b) extendedKeyUsage extension is in the CVC and includes an OID id-kp-codeSigning.
2. The Host validates the certificate signature of the CL/MSO CVC contained in the Signed Data using the CableLabs CVC CA Public Key. Verification of the signature will authenticate the source of the public code verification key (CVK) and confirm trust in the key. Once trust has been established in the CVK, the remaining certificate parameters are no longer needed and SHOULD be discarded.
3. The Host verifies the CableLabs/MSO CVT file signature.
 - a) The Host performs a new SHA-1 hash over the SignedContent. If the value of the messageDigest doesn't match the new hash, the Host rejects the signature on the CVT file as invalid.
 - b) If the signature does not verify, all components of the CVT file are to be rejected and SHOULD be immediately discarded.

Note: The time-varying controls are not used in the CVT File validation process.

7.4 Code Image Validation

The Host performs the code image file verification checks numbered below irrespective of download signaling method or download delivery method. The verification checks can be made in any order, as long as all of the applicable checks presented in this section are made.

The Host SHALL perform the following steps to verify the code image file:

1. The Host validates the manufacturer's signature information by verifying that the PKCS #7 signingTime value is:
 - a) greater-than or equal-to the manufacturer's codeAccessStart value currently held in the Host.
 - b) greater-than or equal-to the manufacturer's CVC validity start time.
 - c) less-than or equal-to the manufacturer's CVC validity end time.
2. The Host validates the manufacturer's CVC by verifying that the:
 - a) CVC subject organizationName is identical to the manufacturer name currently stored in the Host device's memory.
 - b) CVC validity start time is greater-than or equal-to the manufacturer's cvcAccessStart value currently held in the Host's memory.
 - c) extendedKeyUsage extension is in the CVC and includes an OID id-kp-codeSigning.
3. The Host validates the certificate signature of each CVC contained in the Signed Data using the CableLabs CVC CA Public Key. Verification of the signature will authenticate the source of the public code verification

- key (CVK) and confirm trust in the key. Once trust has been established in the manufacturer's and cosigner's CVK, the remaining certificate parameters, except for the validity start time, are no longer needed and SHOULD be discarded.
4. The Host verifies the manufacturer's code file signature.
 - a) The Host performs a new SHA-1 hash over the SignedContent. If the value of the messageDigest doesn't match the new hash, the Host rejects the signature on the code file as invalid.
 - b) If the signature does not verify, all components of the code file (including the code image), and any values derived from the verification process are to be rejected and SHOULD be immediately discarded.
 5. If the manufacturer signature verifies and a cosigning agent signature is required:
 - a) The Host validates the cosigner's signature information by verifying that the:
 - i. cosigner's signature information is included in the code file.
 - ii. PKCS #7 signingTime value is greater-than or equal-to the corresponding codeAccessStart value currently held in the Host device.
 - iii. PKCS #7 signingTime value is greater-than or equal-to the corresponding CVC validity start time.
 - iv. PKCS #7 signingTime value is less-than or equal-to the corresponding CVC validity end time.
 - b) The Host validates the cosigner's CVC, by verifying that the:
 - i. CVC subject organizationName is identical to the cosigner's organization name currently stored in the Host's memory.
 - ii. CVC validity start time is greater-than or equal-to the cvcAccessStart value currently held in the Host for the corresponding subject organizationName.
 - iii. extendedKeyUsage extension is in the CVC.
 - c) The Host validates the certificate signature using the CL CVC CA certificate held by the Host. Verification of the signature will authenticate the source of the co-signer's public code verification key (CVK) and confirm trust in the key. Once trust has been established in the cosigner's CVK, the remaining certificate parameters, except for the validity start time, are no longer needed and SHOULD be discarded.
 - d) The Host verifies the cosigner's code file signature.
 - e) The Host performs a new SHA-1 hash over the SignedContent. If the value of the messageDigest doesn't match the new hash, the Host device rejects the signature on the code file as invalid.
 - f) If the signature does not verify, all components of the code file (including the code image), and any values derived from the verification process are to be rejected and SHOULD be immediately discarded.
 6. The Host verifies that the downloaded software image is appropriate for its hardware version using an implementation-specific method.
 7. If the manufacturer's, and optionally the cosigner's, signature has been verified, the code image can be trusted and installation may proceed. Before installing the code image, all other components of the code file, and any values derived from the verification process, except the PKCS #7 signingTime values and the CVC validity start values, SHOULD be immediately discarded.
 8. If the code installation is unsuccessful, the Host rejects the PKCS #7 signingTime values and CVC validity start values it just received in the code file.
 9. When the code installation is successful, the Host updates the manufacturer's time varying controls with the values from the manufacturer's signature information and CVC:
 - a) Update the current value of codeAccessStart with the PKCS #7 signingTime value.
 - b) Update the current value cvcAccessStart with the CVC validity start value.

10. When the code installation is successful and if the code file was cosigned, the Host updates the cosigner's time-varying controls with the values from the cosigner's signature information and CVC:
 - a) Update the current value of codeAccessStart with the PKCS #7 signingTime value.
 - b) Update the current value of cvcAccessStart with the CVC validity start value.
11. If any of the verification checks fail, or if any portion of the code file is rejected due to invalid formatting, the Host SHALL immediately halt the download process, log the error if applicable, remove all remnants of the process to that step, and continue to operate with its existing code.

Annex A DownloadInfoIndicator Message Detail for Common Download (Normative)

The information in this section is given here to facilitate Common Download. This information is to clarify ambiguities in the DSM-CC specification for use for Common Download. Since the code objects are carried in a DSM-CC data carousel, a standard format for the DownloadInfoIndicator (DII) message should be defined so that Host device can interpret this message properly.

The DownloadInfoIndication (DII) message is defined as identified in Table 15.

Table 15 - DownloadInfoIndicator Message Detail

Syntax	# of bits	Mnemonic
DSMCC_section() {		
Table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved	3	bslbf
dsmcc_section_length	12	uimsbf
table_id_extension	16	uimsbf
reserved	2	bslbf
version_number	5	bslbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
dsmccMessageHeader() {		
protocolDiscriminator	8	uimsbf
DsmccType	8	uimsbf
MessageId	16	uimsbf
TransactionId() {		
Originator	2	uimsbf
Version	14	uimsbf
Identification	15	uimsbf
Update_Flag	1	bslbf
}		
Reserved	8	
adaptionLength	8	uimsbf
messageLength	16	uimsbf
}		
DownloadInfoIndication() {		
DownloadId	32	uimsbf
blockSize	16	uimsbf
windowSize	8	uimsbf
ackPeriod	8	uimsbf
tCDownloadWindow	32	uimsbf
tCDownloadScenario	32	uimsbf
compatibilityDescriptor(){		
length	16	uimsbf
compatibilityDescriptorInfo	16	uimsbf
}		
numberOfModules	16	uimsbf
for(I=0;i<numberOfModules;i++){		
ModuleID	16	uimsbf
ModuleSize	32	uimsbf
ModuleVersion	8	uimsbf
ModuleInfoLength	8	uimsbf
for(i=0;i<ModuleInfoLength;i++){		
ModuleInfoByte	8	uimsbf
}		
}		
privateDataLength	16	uimsbf
for(i=0;i<privateDataLength;i++){		
privateDataByte	8	uimsbf
}		

Syntax	# of bits	Mnemonic
<pre> } } CRC_32 </pre>	32	uimsbf

Table_ID	0x3B for the DII table
section_syntax_indicator	1b
dsmcc_section_length	The number of bytes from the table_ID extension through the last byte of the checksum/CRC32 field.
table_id_extension	The table_id_extension field is populated from the bottom two bytes of the transaction_id field from the dsmccMessageHeader section of the message.
version_number	00000b
current_next_indicator	1b
section_number	0x00
last_section_number	0x00
protocolDiscriminator	0x11
DsmccType	0x03
MessageId	0x1002, to denote a DII message.
Originator	10b
Version	Incremented/changed each time any message in the Download scenario is updated.
Identification	All zeros for a 1 layer data carousel.
Update_Flag	Toggled each time the DII message is updated.
adaptionLength	0x00
messageLength	Total length in bytes of the message following this field.
Download_ID	Used to associate the download data message and the download control messages of a single instance of a download scenario.
blockSize	The length in bytes of the data in every block of the DownloadDataBlock message, except for the last block of each module, which may be smaller than blockSize.
windowSize	0x00
ackPeriod	0x00
tCDownloadWindow	0x00
tCDownloadScenario	Indicates the timeout period in microseconds for the entire download scenario in progress.
compatibilityDescriptor	As defined in the DSM-CC specifications. It may be used by the data carousel server to provide additional information to the Host, which may then be used to further determine applicability of the code image.
numberOfModules	The number of modules described in the loop following this field. The loop includes information for all of the modules to be downloaded by the Client. For

	the Data Carousel scenario, the loop describes a subset of all the modules associated with this Data Carousel, although it may describe all of them.
moduleID	An identifier for the module that is described by the moduleSize, moduleVersion and moduleInfoByte field. The moduleID is unique within the scope of the downloadID.
moduleSize	The length in bytes of the described module.
moduleVersion	The version of the described module.
moduleInfoLength	The length in bytes of the moduleInfo field.
moduleInfoByte	This field contains the ASCII representation of each letter of the code_file_name as defined in the CVT.
privateDataLength	Not used within the scope of these messages at this time.
privateDataByte	Not used within the scope of these messages at this time.

Annex B Non-MSO-Affiliated Code Image Download

B.1 Non-MSO-Affiliated Signaling Method

Signaling for a code update MAY be delivered over an In-Band FAT Channel as part of a non-MSO-affiliated code update scheme, which MAY be proprietary, under the following conditions:

- A CableLabs issued Code Verification Certificate (CVC) SHALL be included in the signaling, AND
- A valid PKCS#7 code file, as defined by Section 7.2 of this specification and [BPI+], which is signaled by this method, SHALL be delivered separately from the signaling, AND
- The HOST 2.0 implementation SHALL adhere to authentication procedures defined in Sections 7.1 and 7.4 of this specification AND [BPI+], except for the requirement of the CVT in Section 7.1.

The signaling and operation of this method of code image upgrade is outside the scope of this specification.

B.2 Non-MSO-Affiliated Code Image Download Mechanisms

A Host MAY be upgraded by other means not defined within this document (e.g., code image transfer via a USB flash drive, HTTP-based code image download, etc.). If a Host can be upgraded by means other than those defined within this document, the upgrade SHALL be done using a mechanism that is equivalent in security to Secure Software Download, described in Section 7 of this specification. The Host SHALL NOT be upgraded in any manner that circumvents the security incorporated on the Host.

B.3 Non-MSO-Affiliated Signaled Download MIB Object Settings

In the event that a code image upgrade is signaled via a non-MSO-affiliated code update scheme, the Host device SHALL follow the procedure defined in section 5.2.7.1 of [eDOCSIS] for setting MIB objects.

Appendix I Revision History

The following ECN was incorporated into version I02 of this specification:

ECN Identifier	Date Accepted	Description
CDL2.0-N-06.0952-5	12/22/06	Corrections to CDL 2.0

The following ECNs were incorporated into version I03 of this specification:

ECN Identifier	Date Accepted	Description
CDL2.0-N-07.0975-3	3/9/07	System Control Resource descriptor_len Definition Correction
CDL2.0-N-07.0988-1	3/9/07	System Control Resource

The following ECN was incorporated into version I04 of this specification:

ECN Identifier	Date Accepted	Description
CDL2.0-N-07.1034-1	6/1/07	Editorial removal of duplicate OSS requirements

The following ECN was incorporated into version I05 of this specification:

ECN Identifier	Date Accepted	Description
CDL2.0-N-07.1094-2	10/23/07	Modifications to Support Group Download

The following ECN was incorporated into version I06 of this specification:

ECN Identifier	Date Accepted	Description
CDL2.0-N-07.1139-1	12/21/07	Corrections to PKCS #7 Signed Data

The following ECNs were incorporated into version I07 of this specification:

ECN Identifier	Date Accepted	Description
CDL2.0-N-08.1266-1	10/3/08	Clarification for Manufacturer CA Certificates than can be carried in the signed content field of the code file
CDL2.0-N-08.1294-3	10/17/08	Deprecate DSG Basic Mode

The following ECNs were incorporated into version I08 of this specification:

ECN Identifier	Date Accepted	Description
CDL2.0-N-08.1359-2	1/16/09	CDL Transition Support
CDL2.0-N-08.1355-2	1/16/09	Common Download ReqPro Omnibus

The following ECNs were incorporated into version I09 of this specification:

ECN Identifier	Date Accepted	Description
CDL2.0-N-08.1356-4	8/7/09	Addition of CVT Type 2 Version 2
CDL2.0-N-09.1395-1	8/7/09	Deprecate CVT2 MAC address descriptor
CDL2.0-N-09.1396-1	8/7/09	Allow download of image in deferred mode
CDL2.0-N-09.1399-2	8/7/09	Clarification of Descriptor Operation

