# CableLabs® Specifications

# Software Management Framework Specification

## CL-SP-SM-I01-080708

**ISSUED**

# Document Status Sheet

| | | | | |
|---|---|---|---|---|
| **Document Control Number:** | CL-SP-SM-I01-080708 | | | |
| **Document Title:** | Software Management Framework Specification | | | |
| **Revision History:** | I01 – Released 07/08/2008 | | | |
| **Date:** | July 8, 2008 | | | |
| **Status:** | ~~Work in Progress~~ | ~~Draft~~ | **Issued** | ~~Closed~~ |
| **Distribution Restrictions:** | ~~Author Only~~ | ~~CL/Member~~ | ~~CL/ 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

# Contents

# Figures



# Tables

This page left blank intentionally.

# 1 PURPOSE AND SCOPE

The purpose of this document is to specify a framework to address the management of cable client software for operator managed clients. The framework specified in this document is termed the Software Management Framework. It presents interfaces, data model requirements, and protocol options. The framework does not present any specific solutions. CableLabs specifications efforts to specify new, or to revise existing, client software management solutions are recommended to adopt a solution based on the framework specified in this document.

The framework does not supersede or replace any cable software management solutions such as DOCSIS Secure Software Download. Further, the scope is limited to operational software modules, and does not address management of applications that are independent of operational software.

## 1.1 Requirements

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

| | |
|---|---|
| "MUST" | This word means that the item is an absolute requirement of this specification. |
| "MUST 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.

[ID-FLUTE] IETF Internet-Draft, FLUTE - File Delivery over Unidirectional Transport, draft-ietf-rmt-flute-revised-05, October 26, 2007.

[RFC 1350] IETF RFC 1350, The TFTP Protocol (Revision 2), July 1992.

[RFC 2616] IETF RFC 2616, "Hypertext Transfer Protocol -- HTTP/1.1", June 1999.

[RFC 2617] IETF RFC 2617, "HTTP Authentication: Basic and Digest Access Authentication", June 1999.

[RFC 3280] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, April 2002.

[RFC 3877] IETF RFC 3877, Alarm Management Information Base, Section 5 ITU Alarm, September 2004.

[RFC 4346] IETF RFC 4346, The Transport Layer Security (TLS) Protocol Version 1.1, April 2006.

[X.733] ITU Recommendation X.733 Information Technology – Open Systems Interconnection – Systems Management: Alarm Reporting Function.

## 2.2 Informative References

This specification uses the following informative references.

[RFC 3411] IETF RFC 3411/STD0062, D. Harrington, et al., An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks, December 2002.

[RFC 3450] IETF RFC 3450, Asynchronous Layered Coding (ALC) Protocol Instantiation, December 2002.

[RFC 3451] IETF RFC 3451, Layered Coding Transport (LCT) Building Block, December 2002.

[RFC 3452] IETF RFC 3452, Forward Error Correction (FEC) Building Block, December 2002.

## 2.3 Reference Acquisition

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027; Phone +1-303-661-9100; Fax +1-303-661-9199; http://www.cablelabs.com
- Internet Engineering Task Force (IETF) Secretariat, 46000 Center Oak Plaza, Sterling, VA 20166, Phone +1-571-434-3500, Fax +1-571-434-3535, http://www.ietf.org

- Internet Engineering Task Force (IETF), Internet: http://www.ietf.org/
  Note: Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time.

  The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.
  Internet-Drafts may also be accessed at http://tools.ietf.org/html/

# 3 TERMS AND DEFINITIONS

This specification uses the following terms:

| | |
|---|---|
| **Cable Client** | An edge device attached to a Cable Operator's network (e.g., Cable Modem). |
| **Cable Modem (CM)** | A modulator-demodulator at subscriber locations intended for use in conveying data communications on a cable television system. |
| **Configuration Server** | The Operator network element responsible for providing the configuration to a Cable Client; sometimes referred to as the Provisioning Server. |
| **Embedded Cable Modem (eCM)** | An embedded Cable Modem that has been enhanced with the features of the CableLabs eDOCSIS specification. |
| **eDOCSIS** | The embedded DOCSIS specification that defines the interface between the eCM and an eSAFE. |
| **Embedded Digital Multimedia Adaptor (eDVA)** | A PacketCable compliant eSAFE that provides real-time voice services. |
| **Embedded Multimedia Terminal Adapter (eMTA)** | A PacketCable compliant eSAFE that provides real-time voice services. |
| **Embedded Portal Service Element (ePS)** | A CableHome-compliant eSAFE that provides management and network address translation functions between the DOCSIS network and the home network. |
| **Embedded Service/Application Functional Entity (eSAFE)** | An embedded version of CableLabs-specified application, such as a PacketCable Multimedia Terminal Adapter (MTA), that provides a service using the DOCSIS IP platform, or a function or set of functions, such as the CableHome Portal Services logical element, that supports the delivery of one or more services over an IP platform. |
| **Embedded Set-top Box (eSTB)** | Embedded Set-Top Box: An eSAFE that is compliant with OpenCable DSG specifications and provides video, audio and data services. |
| **Management server** | The Operator network element responsible for monitoring and managing a Cable Client. |
| **Monolithic Firmware Image** | A single firmware image containing one or more code images for the entire cable client device. As an example for an eDOCSIS device containing an eSTB, the monolithic firmware image contains the eCM code image as well as the eSTB code image (which may also be composed from multiple eSTB code images). |
| **Network management server (NMS)** | The hardware and software components used by the Network Provider to manage its networks as a whole. The Network Management System provides an end-to-end network view of the entire network enabling management of the network elements contained in the network. |
| **Simple Network Management Protocol (SNMP)** | A network management protocol of the IETF. |
| **Software Download Server** | The hardware and software components containing cable client firmware images and which support the required transport protocols for file transfers. |
| **Trivial File Transfer Protocol (TFTP)** | An Internet protocol for transferring files without the requirement for user names and passwords that is typically used for automatic downloads of data and software. |

| | |
|---|---|
| **Type/Length/Value (TLV)** | An encoding of three fields, in which the first field indicates the type of element, the second the length of the element, and the third field the value of the element. |

# 4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations and acronyms:

| | |
|---|---|
| **ALC** | Asynchronous Layered Coding |
| **ASM** | Any Source Multicast |
| **CM** | A DOCSIS Cable Modem |
| **DOCSIS** | Data-Over-Cable Service Interface Specifications. |
| **eCM** | Embedded Cable Modem |
| **eDOCSIS** | Embedded DOCSIS specification |
| **eDVA** | Embedded Digital Multimedia Adaptor |
| **eMTA** | Embedded Multimedia Terminal Adaptor |
| **ePS** | Embedded Portal Service Element |
| **eSAFE** | Embedded Service/Application Functional Entity |
| **eSTB** | Embedded Set-Top Box |
| **FDT** | File Delivery Table |
| **FEC** | Forward Error Correction |
| **FLUTE** | File Delivery over Unidirectional Transport |
| **HTTP** | Hypertext Transport Protocol |
| **IETF** | Internet Engineering Task Force |
| **IP** | Internet Protocol |
| **LCT** | Layered Coding Transport |
| **Mfg CSA** | Manufacturer Code Signing Agent |
| **Mfg CVS** | Manufacturer Code Verification Signature |
| **MIME** | Multipurpose Internet Mail Extensions |
| **NMS** | Network Management System |
| **RFC** | Request for Comments |
| **SNMP** | Simple Network Management Protocol |
| **SSD** | Secure Software Download |
| **SSM** | Source Specific Multicast |
| **STB** | OpenCable Set-Top Box |
| **TCP** | Transport Control Protocol |
| **TFTP** | Trivial File Transport Protocol |
| **TLV** | Type/Length/Value |
| **TOI** | Transport Object Identifier |
| **TSI** | Transport Session Identifier |
| **UDP** | User Datagram Protocol |

| | |
|---|---|
| **URI** | Uniform Resource Identifier |
| **XML** | Extensible Markup Language |

# 5 OVERVIEW

CableLabs specifications often specify clients that are used by cable subscribers to connect to their cable operator's network and obtain services. Examples of such clients include DOCSIS Cable Modems (CMs), PacketCable 1.5 Embedded Multimedia Terminal Adaptors (E-MTAs) and OpenCable Set-Top Boxes (STBs). Cable clients can support one or more applications. For example, a CM provides broadband access where as an E-MTA can be used for broadband access and telephony. Cable clients can be hardware- or software-based. For instance, a CM is a hardware client, where as a PacketCable 2.0 User Equipment (UE) can manifest as a software client residing on a STB.

Cable clients contain one or more software modules that allow for it to operate in compliance with CableLabs specifications. These are termed operational software. Such software modules are different from middleware platforms such as the OpenCable Application Platform (OCAP) and any cable applications. The operational software modules on a client are collectively termed client software, or software, within this document. Operators have a need to manage such software to ensure continued and consistent client operations. For example, an Operator may want to ensure that all deployed E-MTAs from a certain manufacturer contain a specific revision of software. This requires the ability to identify the software revision and to provide a different version if required. Additionally, the operator may want to make updates at a later date due to reasons like software bug fixes and feature enhancements.

CableLabs specifications, outside of this framework, have presented solutions to perform such updates. For example, DOCSIS and OpenCable specifications have specified mechanisms to manage client software for DOCSIS CMs and OpenCable STBs, respectively. The procedures are identified as Secure Software Download (SSD) and OpenCable Common Download, respectively. They specify the following: data elements to identify the client software, mechanisms for the operator to direct a client to obtain specific software, software download procedures, and requirements for the client to verify the authenticity of downloaded software. These requirements are integral to the framework described in this document. However, there are enhancements. Some examples include:

- Support for multiple software modules, instead of a monolithic image;
- Ability to delay activation of downloaded software;
- Protocol choices for multicast;
- Support for a single CableLabs Root CA for Software Verification.

Software management needs to address the various operational considerations and identify the functional areas.

## 5.1 Operational Considerations

To effectively manage software on cable clients requires a better understanding of the client types, access networks, and applications.

### 5.1.1 Client Types

Cable clients can be hardware- or software-based. They can be standalone, or embedded with cable modems or other cable clients. For example, a triple play device can combine a CM, eMTA, and ePS that comply with the DOCSIS, PacketCable, and CableHome specifications, respectively. Similarly, a software-based PacketCable 2.0 UE can be embedded in a STB to provide value-added services to customers.

Some cable clients are required to support only Monolithic Software Images, such as DOCSIS 1.0 Cable Modems. Other cable clients may be allowed to support multiple software modules.

### 5.1.2 Access Networks

Cable clients can connect via a DOCSIS network, or by using a different access network. For example, PacketCable 2.0 clients (termed User Equipment or UEs) can connect via Wi-Fi. Such clients can also connect via NAT and Firewall devices.

### 5.1.3 Applications

Clients can support one or more applications. For example, PacketCable 2.0 allows for a UE to support multiple applications. Some of these applications may provide real-time services, such as voice. Such applications may be embedded with operational software. For example, PacketCable 2.0 E-DVAs are required to support a monolithic software image that complies with the PacketCable 2.0 and PacketCable RST specifications. Alternatively, such applications can be contained in individual software modules, e.g., the OCAP middleware supports multiple applications that may not be part of the operational software. Applications that are not contained within the operational or middleware components are not within the scope of this document.

## 5.2 Functional Areas

Five functional areas have been identified to effectively address software management, as highlighted in this section.

### 5.2.1 Client and Software Identification

To manage a client, an Operator needs the ability to uniquely identify a client within the network. The Operator also needs to be able to verify the software modules supported by a client. This allows an operator to ensure the presence of the required software modules. The functional area that deals with such identification requirements is termed Client and Software Identification.

### 5.2.2 Software Management Triggers

If the client software needs to be modified for any reason, the Operator needs to have the ability to send across instructions to the client. The instructions provide information such as the software modules to download and how to obtain them, among other things. These instructions are termed Software Management Triggers.

### 5.2.3 Software Download

When a client receives a software management trigger, it needs to obtain the instructed software modules. The processes involved in obtaining the software images containing the modules and their verification is termed Software Download.

### 5.2.4 Interruption Impact Reporting

The cable client may support applications that provide critical or real-time services that should not be interrupted due to management activities such as software management. In such cases, the client needs to be able to identify the impact of any interruptions. This is termed Interruption Impact Reporting. This can be used by management servers to minimize or eliminate any service interruption.

### 5.2.5 Software Management Reporting

Cable clients need to notify the Management Server of the status and any conditions resulting from software management operations. For example, a cable client needs to report the success or failure of a software download process and, in the case of a failure, the error conditions. The capability of the cable client to report this information is termed Software Management Reporting.

# 6 SOFTWARE MANAGEMENT FRAMEWORK

The Software Management Framework is specified in Figure 1.

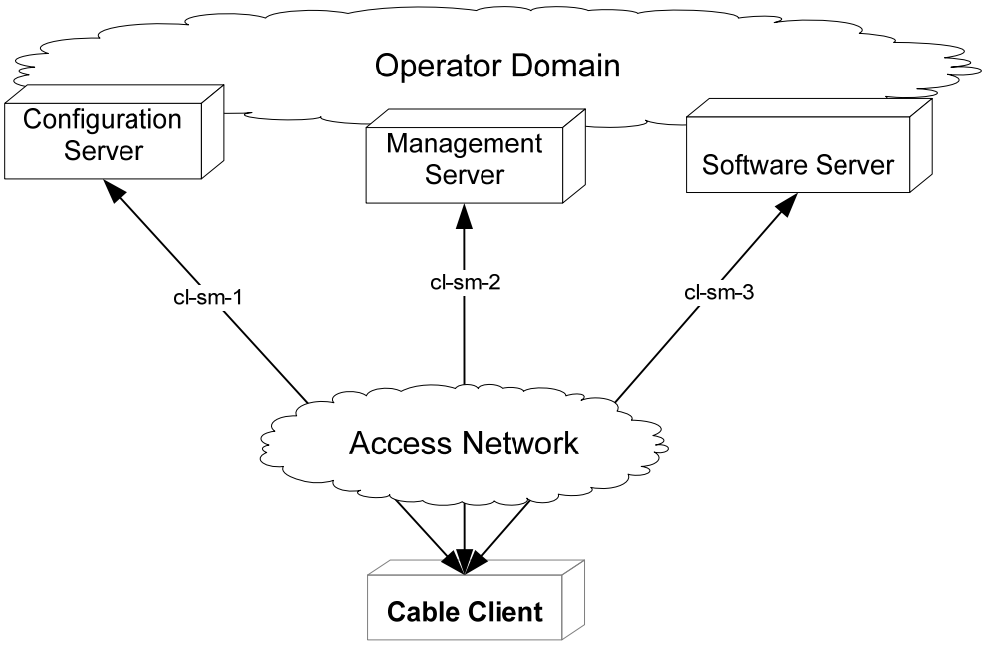



***Figure 1 – Software Management Framework***

Figure 1 identifies the functional components and the interfaces required for the Software Management Framework specified in this document. The interfaces are used to address the different functional areas supported by the framework. The following sub-sections provide an overview of the functional components, the interfaces, and the addressed functional areas.

## 6.1 Functional Components

The Software Management Framework identifies four functional components: cable client, configuration server, management server, and a software server.

### 6.1.1 Cable Client

A Cable Client is an Operator managed entity that interfaces with the Operator's network to provide user services, e.g., Cable Modem. Cable clients may need one or more software modules. Each software module can be categorized into three categories: operational, middleware, and application. A cable client may include all the three categories within a single image. This framework does not specify independent management of middleware and applications. This framework allows for the support of cable clients that are required to support Monolithic Software Images such as DOCSIS 1.0 CMs, and clients that may support multiple software modules with embedded applications. Figure 2 illustrates the separation of software contained within a client.

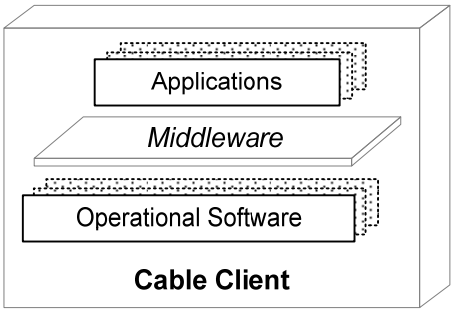



***Figure 2 - Conceptual Cable Client Software Module Architecture***

### 6.1.2 Configuration Server

The Configuration Server within the Operator domain is responsible for providing the required operator configuration to the client. The configuration is provided either due to a client request (e.g., bootup configuration) or due to changes in configuration initiated by the Operator's backoffice (e.g., run-time changes). The interface to the client is labeled SM-1. The interface is used by the Software Management framework to provide triggers.

### 6.1.3 Management Server

The Management Server within the Operator domain is responsible for monitoring and management of the cable client. This includes receiving events from the client, e.g., failure reports and providing the necessary information for continuous client operation. The interface to the client is labeled SM-2. The interface is used by the Software Management framework to provide triggers, and to identify success, errors, and warnings that the client may report during or after a software download process.

### 6.1.4 Software Server

The Software Server within the Operator domain is responsible for storing any software that is required by the client. The interface to the client is labeled SM-3. The Software Server information is provided via the triggers and the client communicates using SM-3 to obtain one or more software images.

## 6.2 Interfaces

The Software Management Framework requires the use of three interfaces: cl-sm-1, cl-sm-2, and cl-sm-3.

### 6.2.1 cl-sm-1

Cable clients need to be configured to obtain operational and application-specific data such as operational state and network element information. This is accomplished via an interface between the Configuration Server and the cable client. This interface is termed cl-sm-1. The software management framework does not specify any protocols, but allows project-specific protocols to be used. The interface is used to transmit Software Management Triggers.

### 6.2.2 cl-sm-2

Cable clients are managed to maintain service integrity. CableLabs projects specify a management protocol such as SNMP for this purpose. Such a management interface is referred to as cl-sm-2. The interface is used to transmit Software Management Triggers. As with configuration, the software management framework does not specify any protocols and relies on project-specific choices.

### 6.2.3 cl-sm-3

To download software modules, the cable client needs to contact a software server. The software management framework refers to the interface between the cable client and a software server as cl-sm-3. Protocol options for this interface may be specified by CableLabs projects utilizing the software management framework. In cases where they are not specified, the framework specified in this document presents options in Annex B.

## 6.3 Functional Areas

The Software Management Framework addresses various functional areas. The functional areas and the associated requirements are specified in this section.

### 6.3.1 Client and Software Identification

To manage the software on a cable client, the Operator needs to identify the cable client to be managed and information pertaining to the software modules that it contains. This is termed client and software identification.

Specifically, a solution that complies with this framework needs to report the following:

- A unique identifier that identifies the device, e.g., mac address or a unique URI;
- A list of the contained software modules, and for each software module:
  - The module version; and,
  - If supported, the activation status.

In some instances the client identifier can be a locally unique value within the Operator's network, such as an IP address. For clients that do not support run-time activation and deactivation of modules, software modules are always assumed to be active.

The following sub-sections provide options for the Operator to obtain this information.

#### *6.3.1.1 Configuration*

For cable clients that are deployed directly by the Operator, i.e., provided after being associated with a subscription, the required client and software identification information may already be known during pre-configuration.

Operators who support the dynamic configuration model, i.e., dynamic association of cable clients with a subscription may obtain client and software information prior to dynamic configuration. For example, the DHCP process or the initial configuration request can be used.

#### *6.3.1.2 Management*

Independent of the indicated configuration mechanisms, the cable client MUST make the required information available via the management interface. The data representation and the management protocol for obtaining this information are project-specific and out of scope for this document. Management provides a definitive means for the Operator to obtain client and software identification parameters in real-time.

### 6.3.2 Software Management Triggers

When the required software modules are not present on a client, the Operator directs the cable client to download the required modules. This directive is presented in messages termed Software Management Triggers. These triggers are transmitted via either configuration or management. The framework specified in this document does not recommend any specific protocols, but relies on project-specific choices for configuration and management.

Configuration and Management protocol options need to provide for security requirements such as authentication, data integrity and privacy.

A trigger consists of the following:

- list of software modules that need to be modified;
- software server to obtain the alternate software modules;
- software download protocol, if required (e.g., when the client supports more than one protocol);
- software image identifier that contains the software module(s) such as the software image filename;
- activation status for each of the software modules, if activation is supported by the client;
- security information for software verification, such as a CVC.

The normative list of data elements is specified in Annex A. Two models are defined for triggers: the push and the pull models. In the push model, the management server sends a trigger to the cable client (e.g., via a configuration file). In the pull mechanism, the client obtains the trigger either via a scheduled polling request or an ad-hoc query (e.g., user-initiated request).

After obtaining triggers, the client may be instructed to either obtain the software from a location, or to obtain it from a communications channel established by a software server. An example of the latter is a multicast or broadcast session to which the client connects.

### 6.3.3 Software Download

This section deals with the mechanisms for the cable client to download the software modules, based on the trigger. The process involves three steps: transport, verification, and activation.

#### *6.3.3.1 Software Transport*

Cable clients complying with the requirements in this document SHOULD support one or more of the following transport protocols:

- HTTP as specified in Annex B.2
- FLUTE as specified in Annex B.3
- TFTP, for legacy clients

If TFTP is supported, it MUST comply with the requirements in Annex B.1. Cable clients MAY also specify project-specific download mechanisms. Any such project-specific scenarios SHOULD specify adequate security requirements. An example of an exception to security requirements is the reuse of a transport protocol that is previously authenticated and offers integrity-protection between the cable client and the Software Server.

Refer to Annex B for security considerations associated with the indicated protocols.

#### *6.3.3.2 Software Verification*

Any software modules that are downloaded by a cable client need to be verified for applicability. This involves authentication and integrity checks, both of which are vital to the overall operation and security of cable networks.

Refer to Annex C for software verification requirements. A cable client that is compliant with this document MUST support the process as specified in Annex C.

#### *6.3.3.3 Software Activation*

After successful software verification, the downloaded software modules need to be activated for operation. This can be done by specifying default behavior (e.g., activate by default), or based on Operator direction (e.g., via trigger information). Cable clients that support real-time applications SHOULD provide activation controls that allow an Operator to activate the software modules based on criteria that limit the disruption to real-time applications. Cable clients that do not support real-time applications MAY still provide such controls.

If client support activation, the following requirements apply:

- For any software module that is successfully downloaded, the cable client MUST act on any activation controls provided in the triggers;
- If the download is unsuccessful, the cable client MUST NOT act on the activation controls.

Activation controls need to accommodate for the inability to activate a given module within the specified criteria, and be associated with management events to address error scenarios. Activation controls can be classified into the multiple categories, based on the intent. A non-exhaustive set is indicated in the following sub-sections. The choice and implementation of activation controls is left to solutions based on the framework specified in this document.

##### *6.3.3.3.1 Time-based*

Time-based activation controls specify a relative or absolute time for download. An example of relative time is the period to wait after the software download process, before the software module is activated. This is useful in certain operational scenarios. For example, if software download is performed across a large client base and is expected to result in mass registrations, this process can be used to minimize the peaks by providing different values across the clients (or subsets of clients).

Relative times may not always be directly mapped to the absolute time due. For example, the different cable clients from the same manufacturer may not be able to complete the software download process at the same time due to network and operational considerations.

Absolute times can be useful when cable clients need to download and prepare for a specific date and time before a new module is made operational. This assumes that the software download and software verification processes are successfully completed by the indicated time.

Time-based activation controls MUST specify the desired time components, such as hours, minutes and seconds. Activation controls specifying absolute times need more details than those specifying relative times. For instance, absolute time requires the specification of time zones.

##### *6.3.3.3.2 Status-based*

Status-based activation controls identify the criteria for activation, such as status information that can be reported by the client. For example, "activate when no real-time application is operational". Such controls can only be specified when the cable client is capable of determining such states.

##### *6.3.3.3.3 Conditional Logic*

Conditional Logic activation controls can be based on conditional logic such as "If a real-time application has not been active for more than sixty minutes or it has been seventy-two hours since the module was downloaded, then activate the module." This is most useful for intelligent clients that support conditional logic and the Operator can delegate the responsibility to the client. This category combines activation control behavior from the other categories like status-based (real-time application status) or time-based (72 hours) controls.

#### *6.3.3.3.4 Management-based*

The activation controls that require a separate management trigger for activation fall under the category of management-based activation controls. This is useful in cases where the cable client is unable to take actions based on status. For example, a cable client can be directed to perform a download, but to wait for a management trigger (e.g., SNMP SET) before it activates the module. This may be useful with cable clients that can report status information (e.g., a specific real-time application is operational) but may not have the ability to perform actions based on the status (e.g., do not activate until a specific real-time application is inactive).

### 6.3.4 Interruption Impact Reporting

If a cable client obtains Software Management Triggers while it is operational, the activation of a downloaded software module can affect operations, especially those that rely on real-time cable client availability such as voice or real-time multimedia services. To address such scenarios, the cable client MAY support interruption impact reporting and handling. If supported, the cable client MUST provide the following information whenever requested via a management interface:

- the impact status for each of the real-time applications supported by the application, e.g., PacketCable Voice,
- detailed information on the reason for the impact, e.g., ongoing voice call on line #n.

### 6.3.5 Software Management Reporting

Cable clients need to report events related to Software Management. This document does not specify a specific reporting mechanism. However, it specifies the high level requirements for reporting in Annex D. A cable client compliant with the framework specified in this document MUST support the event definitions and alarm formats specified in Annex D.

# Annex A Information Model for Triggers

The information model for the triggers is contained in this section. Annex A.1 defines the Trigger message. Annex A.2 provides the details regarding its constituents. Annex A.3 provides the details of the SwModuleControlData element.

## A.1 Trigger Message

The format of the Trigger message is depicted in Table 1.

***Table 1 - Trigger Message***

| Message | Definition | Explanation |
|---|---|---|
| Trigger | Sequence of three elements:<br>• TriggerDownloadType<br>• TriggerControlDataVersion<br>• SwModuleControlData | This message includes all the necessary information required to download one or more modules.<br>The triplet of elements may be repeated multiple times within one Trigger message. |

## A.2 Trigger Message Elements

The elements of the Trigger message are indicated in Table 2. An explanation of the elements is presented in the sub-sections following Table 2.

***Table 2 - Trigger Message Elements***

| Element Name | Type | Type Constraints |
|---|---|---|
| TriggerDownloadType | pull \| push | |
| TriggerControlDataVersion | String | |
| SwModuleControlData | Sequence of data defining the control information (complex type) | |

### A.2.1 TriggerDownloadType

TriggerDownloadType specifies the method used for retrieving the control data for downloading the software. If the element indicates "pull", the cable client should retrieve the control data associated with the control data version attribute from the server.

### A.2.2 TriggerControlDataVersion

TriggerControlDataVersion specifies the version of the control data that should be used by the cable client in downloading the software modules.

### A.2.3 SwModuleControlData

SwModuleControlData is a sequence of elements defining relevant control information associated with one or more software modules that are to be downloaded. The list of elements may occur one or more times. When the trigger

download type (TriggerDownloadType) is "pull", this element is not present. This element is composed of other elements as described in Annex A.3.

## A.3 SwModuleControlData Elements

The elements of the SwModuleControlData element are indicated in Table 3. An explanation of the elements is presented in the sub-sections following Table 3.

***Table 3 - SwModuleControlData Elements***

| Element Name | Type | Type Constraints |
|---|---|---|
| SwModuleControlData | Sequence of various elements defining the control information for the software modules to be downloaded (complex type)<br>• TargetDevice<br>• DownloadTime<br>• ActivationTime<br>• ActivateAfterModule<br>• ModuleFileURI<br>• ModuleFileSize<br>• HashType<br>• ModuleFileHash<br>• ValidationCVC<br>• AllowModuleVersion<br>• Transport | |
| TargetDevice | Client specific | |
| DownloadTime | DateTime | A non-zero value may only be specified if the software is downloaded using the pull model. |
| ActivationTime | immediately \|<br>dateAndTimeIndicated \|<br>uponCriticalServices | When the specified date and time is set to zero, it is equivalent to 'immediately' (e.g., the software is activated immediately upon download). |
| ActivateAfterModules | <client specific> | Any constraints is outside the scope of this document. It is vendor specific. |
| ModuleFileURI | URI | |
| ModuleFileSize | Integer | |
| HashType | Enumerated list (SHA-1, SHA-256, MD5, MMH, <project specific> | This list may be augmented with project specific hash types. |
| ModuleFileHash | HexBinary | |
| ValidationCVC | HexBinary | |

| Element Name | Type | Type Constraints |
|---|---|---|
| AllowModuleVersion | Enumeration (older, same, new) | |
| Transport | Complex Type with protocol specific information. The following choices are included:<br>Protocol specific information is included.<br>Choice {<br>pull URL<br><br>push enumeration {dsm-cc, dsg, mpeg, ipmulticast)<br>} | In the case of pull, URL can be used providing the link for the location to download the file using http, tftp, or ftp. In the case of push model, the choice is application specific. For example some identified choices are dsm-cc, dsg, mpeg, and multicast. |
| TransportSpecificControlInfo | <Any – project/application specific> | |

### A.3.1 TriggerControlData

This is a complex type with a list of elements identified in the Table 3. Each element is described below.

#### *A.3.1.1 TargetDevice*

This element specifies the identity of the target device where the software is to be downloaded. The data type for the identifier is cable client specific. For example, it may be an OUI, Host ID, etc.

#### *A.3.1.2 DownloadTime*

This element specifies the date and time for the download. A value of zero corresponds to download immediately. A non-zero value indicates deferring until the specified time. The option to defer to a specific time is valid only if the transport element is selected to use the pull model.

#### *A.3.1.3 ActivationTime*

This element specifies when the software module should be activated after the download. If 'uponCriticalServices' is selected, the syntax is project specific. It may contain a list of real-time applications and the activation time would be after the real-time applications in progress are terminated. When the specified date and time is set to zero, it is equivalent to 'immediately' (e.g., the software is activated immediately upon download).

#### *A.3.1.4 ActivateAfterModules*

This element specifies the dependent module or modules which need to be activated first to ensure successful activation. The format of this object is vendor-specific.

#### *A.3.1.5 ModuleFileURI*

This element specifies a URI. It may include a URL (for pull mode) which indicates the location where the file to be downloaded resides or a URI that is a pointer to a multicast stream or to an MPEG stream etc.

#### *A.3.1.6 ModuleFileSize*

This element specifies the size of the file in bytes.

#### *A.3.1.7 HashType*

This element specifies the type of the hash algorithm used for calculating the hash included in the control data. The hash algorithm may be project specific or SHA-1, SHA-256, MD5, MMH, etc.

#### *A.3.1.8 ModuleFileHash*

This element specifies the value of the hash of the module file calculated using the hash type (HashType). The value of hash is calculate on both the header information and the image.

#### *A.3.1.9 ValidationCVC*

This element specifies the CVC for validation as specified in Annex C.

#### *A.3.1.10 AllowModuleVersion*

This element specifies whether the device is allowed to download an older version of the module or overwrite the current version or a new version.

#### *A.3.1.11 Transport*

This element specifies the details for downloading the image. If the pull model is used, then the URL the cable client MUST use to download the image is specified. In the case of push model different application specific choices are identified. Based on the selected choice, different parameters and formats will be required. These definitions are project specific. The details of the parameters are described below.

##### *A.3.1.11.1 TransportSpecificControlInfo*

This attribute contains parameters specific to the transport attribute for the push model. The parameters are specific to the application.

# Annex B Software Retrieval Options

## B.1 TFTP

Cable clients and Software Servers complying with this specification and supporting TFTP as a transport protocol MUST be compliant with the requirements specified in [RFC 1350]. It is to be noted that TFTP in itself does not provide authentication, integrity protection, or privacy. This presents various security threats such as impersonation, Man-in-the-Middle, and Denial of Service (DoS) attacks. The software verification requirements presented in this document prevent any active threats due to erroneous modules. However, it does not eliminate threats such as DoS that require authentication and integrity protection. Thus, the use of TFTP is not recommended in scenarios where security attacks can be easily mounted by unwanted elements.

## B.2 HTTP

Cable clients and Software Servers complying with this specification and supporting HTTP as a transport protocol MUST be compliant with the client requirements specified in [RFC 2616].

To support authentication and integrity protection, a cable client supporting HTTP MUST also support HTTP over TLS as specified in [RFC 4346]. Additionally, mutual authentication SHOULD be supported either via certificates, or by using a combination of Server Side Authentication and client Digest Authentication. Refer to [RFC 2617] for more information on Digest Authentication.

## B.3 FLUTE

Cable clients and Software Servers complying with this specification and supporting FLUTE as a transport protocol MUST follow the requirements specified in this sub-section. The Software Server MUST support the "sender" requirements defined in [ID-FLUTE]. The terms "Software Server" and "sender" are used interchangeably with respect to the FLUTE protocol. The cable client MUST support the "receiver" requirements defined in [ID-FLUTE]. The terms "cable client" and "receiver" are used interchangeably with respect to the FLUTE protocol.

The following sections define an overview of the FLUTE protocol. Any requirements stated in this specification supersede those stated in [ID-FLUTE].

FLUTE (File Delivery over Unidirectional Transport) [ID-FLUTE] is built upon the Asynchronous Layered Coding (ALC) [RFC 3450] protocol, for the unidirectional delivery of files over any network. The ALC protocol is the transport for FLUTE and provides for massive scalability over IP multicast networks for the delivery of binary objects (i.e., files). In the context of this document, FLUTE is specified for simultaneously delivering small or large files (e.g., software updates) to many clients from a single server.

In addition to providing scalability, FLUTE provides congestion control using ALC [RFC 3450], reliability using FEC building block [RFC 3452], and session management using LCT building block [RFC 3451]. FLUTE can be used with IPv4 and IPv6, unicast and multicast (ASM and SSM).

### B.3.1 File Delivery Session

The file delivery portion of the protocol defines how the attributes (e.g., filename, location, MIME type, size, content encoding and security properties) are carried in-band with the delivered file.

FLUTE uses ALC/LCT sessions consisting of a set of ALC/LCT channels. The sender of the file(s) defines the channel and is associated with an address. A receiver joins a channel to receive files sent by the sender to that channel and leaves a channel to stop receiving such files. The ALC/LCT header (32-bits wide) includes a TSI field (Transport Session Identifier) where a source (sender's) IP Address and TSI pair identify a session to a receiver. The

ALC/LCT header also includes a TOI field (Transport Object Identifier) to identify a source file. Multiple files may be carried in a session. A TOI value of zero is reserved for delivery of File Delivery Table (FDT) Instances. A session will have a FDT local to it. Files and FDT Instances may be multiplexed in any order. Refer to the next section for details on FDTs. Refer to [RFC 3451] for details on the LCT Default Header containing the TOI and TSI fields.

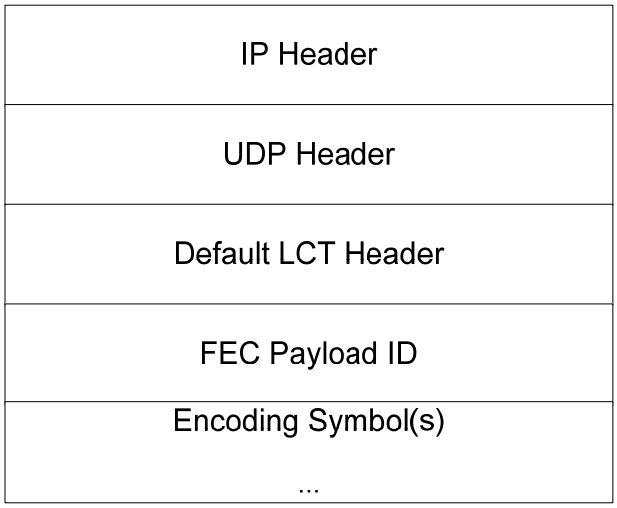



***Figure 3 - ALC Packet with Default LCT header***

### B.3.2 File Delivery Table

A File Delivery Table (FDT) provides a means to describe attributes associated with files to be delivered in the file delivery session. More specifically, the FDT is a set of file description entries for files to be delivered in the session. Within the session, the FDT is delivered as FDT Instances which contains one or more file description entries of the FDT. Each receiver of an FDT maintains an internal FDT database.

***Table 4 - Example FDT Attributes***

| Delivery Attributes | File Attributes |
|---|---|
| TOI for file* | URI* |
| FEC Object Transmission Information | MIME Media Type |
| Size of Transport Object | File Size |
| Aggregate Rate | Encoding and Security |
| *Required attributes | |

### B.3.3 FDT Instances

For every file delivered in a FLUTE ALC/LCT session, there must be a file description entry within at least one FDT Instance sent in the session. At a minimum, the file description entry maps the TOI and URI (see required fields in Table 1).

A TOI value (present in the LCT header) is reserved for delivery of FDT Instances.

The FDT Instance is identified by a fixed length 32-bit LCT Header Extension (EXT_FDT). An FDT Instance is uniquely identified within a file delivery session by a FDT Instance Identifier contained in the EXT_FDT.

An FDT Instance is valid until it reaches its expiration time specified in the payload. FDT Instances are higher priority packets than the files they describe.

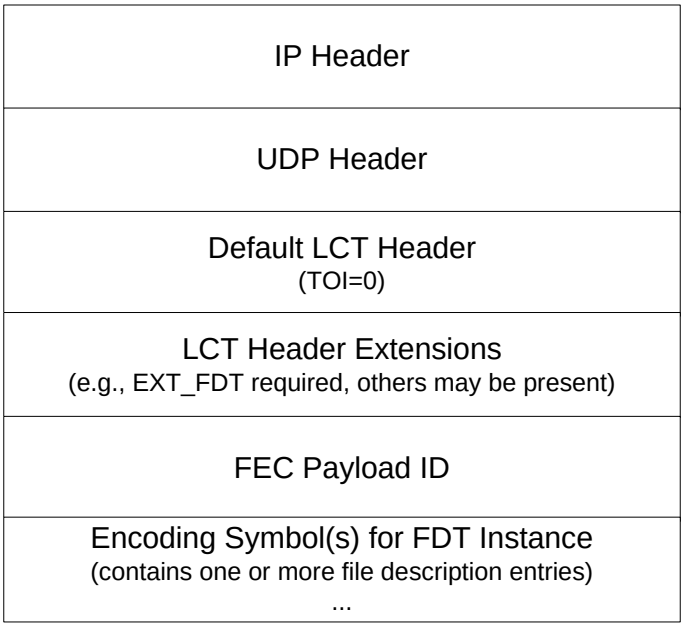



***Figure 4 - FDT Instances sent in ALC/LCT packets***

The EXT_FDT header contains three fields: an 8-bit Header Extension Type (HET) field, a 4-bit Version (V) field and a 20-bit FDT Instance Id field. For each file delivery session, the FDT Instance ID starts at zero and increments.

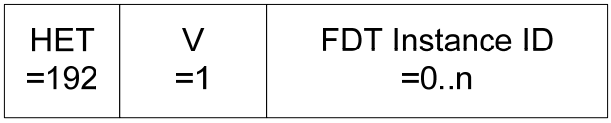



***Figure 5 - EXT_FDT Header***

The FDT Instance syntax is XML Schema, with a root element of "FDT-Instance". This provides extensibility for private attributes beyond the FDT Instance XML Schema defined in [ID-FLUTE].

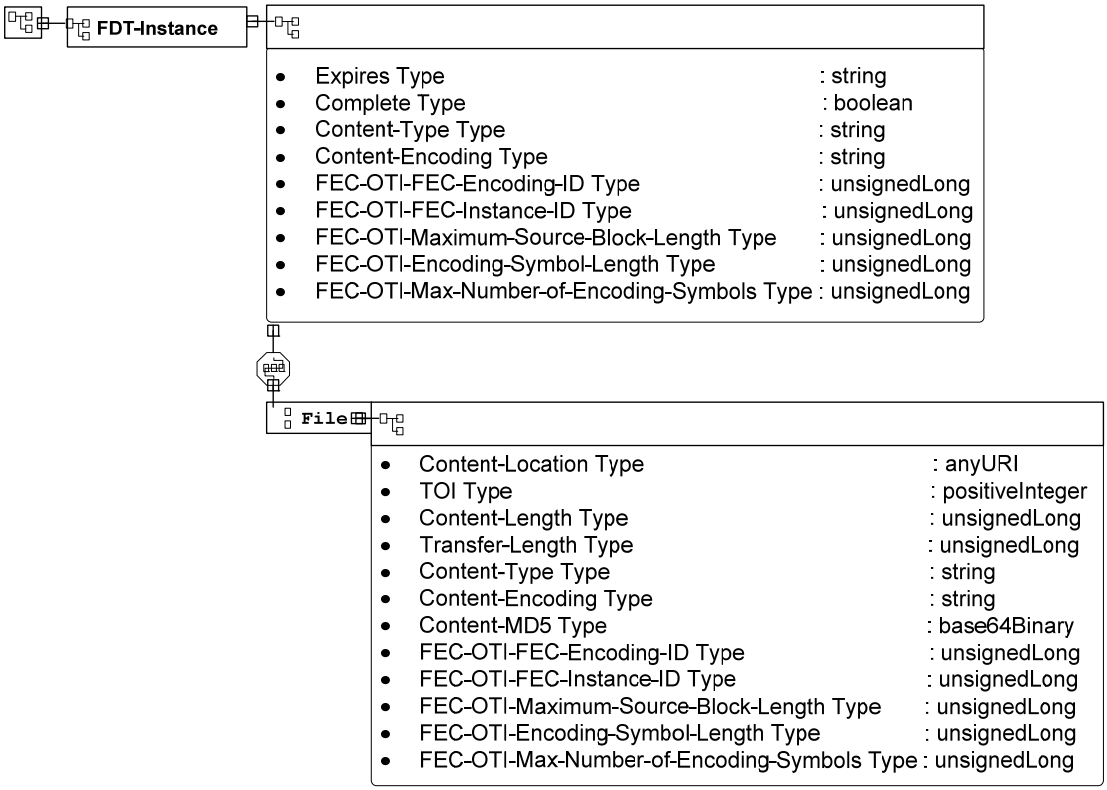



***Figure 6 - FDT Instance XML Schema***

An FDT Instance may be content encoded (e.g., compressed with GZIP). If content encoding is used, a fixed length 32-bit FDT Instance Content Encoding Header (EXT_CENC) is required as an LCT Header Extension.

The EXT_CENC header contains three fields: an 8-bit Header Extension Type (HET) field, an 8-bit Content Encoding Algorithm (CENC) field and a 16-bit Reserved field. The CENC field specifies the content encoding algorithm for the FDT Instance payload (e.g., for GZIP N=3).

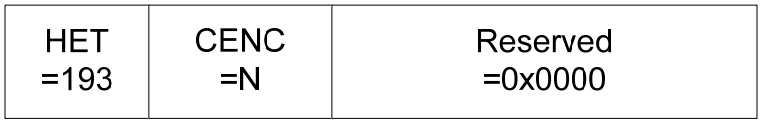



***Figure 7 - EXT_CENC Header***

There are two methods for delivering FEC Object Transmission Information in-band within the session: 1) EXT_FTI header and 2) FDT. Both methods will deliver exactly the same information.

The 128-bit fixed length EXT_FTI header is higher priority and is required as an LCT Header Extension. It will be included in at least one packet for each FDT Instance. The FEC Encoding ID (8-bits) is carried in the Codepoint field of the ALC/LCT header [RFC 3450]. The EXT_FTI header contains five fields: an 8-bit Header Extension Type (HET) field, an 8-bit Header Extension Length (HEL) field (where L equals the length of the whole Header Extension field), a 48-bit Transfer Length field (which indicates the length in bytes of the transport object that carries the file, or the file length if the file is not content encoded), a 16-bit FEC Instance ID field and a FEC Encoding ID Specified Format field (depends on FEC Encoding ID as specified in [ID-FLUTE].

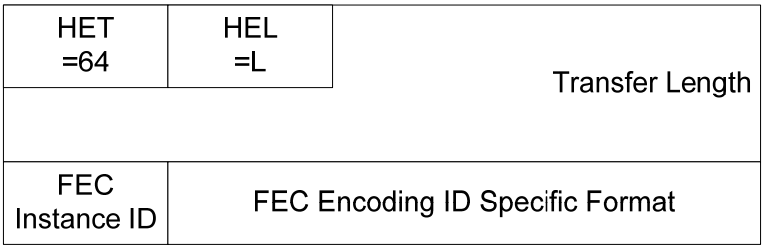



***Figure 8 - EXT_FTI Header***

The FDT method is required when TOI values other than zero are used. This method defines appropriate attributes for each file within the "FDT-Instance" or "File" element of the FDT structure as specified in the FDT Instance Schema shown in Figure 6. These are optional fields with the mappings shown in the table below.

***Table 5 - FEC Object Attribute Mappings***

| FEC Object Schema Attributes | Mapping |
|---|---|
| Transfer-Length | Transfer Length |
| FEC-OTI-FEC-Encoding-ID | FEC Encoding ID |
| FEC-OTI-FEC-Instance-ID | FEC Instance ID |
| FEC-OTI-Maximum-Source-Block-Length<br>FEC-OTI-Encoding-Symbol-Length<br>FEC-OTI-Max-Number-of-Encoding-Symbols | FEC Encoding ID Specified Format* |
| *Refer to [ID-FLUTE] | |

### *B.3.3.1 Congestion Control and Channels*

There are four scenarios with FLUTE congestion control:

- Single channel, single-rate congestion control;
- Single channel, no congestion control (assumes flow and congestion control are performed by other mechanisms);
- Multiple channels, multiple-rate congestion control;
- Multiple channels, no congestion control (assumes flow and congestion control are performed by other mechanisms).

Those scenarios with multiple channels may have timing constraints for files sent on different channels arriving out of order.

Refer to [ID-FLUTE] for additional details on congestion control.

### *B.3.3.2 FLUTE Session Configuration*

Prior to starting a FLUTE session, a receiver MUST be configured with the associated transport parameters. It is out of scope of this specification, as well as [ID-FLUTE], to specify how the transport configuration parameters are communicated to and configured within the client devices. Some examples which might be employed are:

- SDP or XML session description file sent over a suitable transport protocol (e.g., SIP);

- TLV entries in a Configuration File sent over a suitable transport protocol (e.g., TFTP);
- SNMP MIB Sets.

The following table lists the required parameters according to [ID-FLUTE].

***Table 6 - Required Transport Configuration Parameters***

| Parameter | Description |
|---|---|
| FluteIndicator | Indicates the session is a FLUTE session |
| SrcIpAddress | IP Address of the Sender transferring the file(s) |
| NumChannels | Number of channels in the file delivery session(n) |
| DestIpAddress(1..n) | Destination IP Address for each channel |
| DestPortNum(1..n) | Destination Port Number for each channel |
| SessTsi | Transport Session Identifier (TSI) of the file delivery session |

The following table lists the optional parameters according to [ID-FLUTE].

***Table 7 - Optional Transport Configuration Parameters***

| Parameter | Description |
|---|---|
| SessStartTime | Start time of the session as advertised by the Sender |
| SessEndTime | End time of the session as advertised by the Sender |
| FecEncodingId | FEC Encoding ID when the default FEC Encoding ID 0 is not used for the delivery of FDT |
| FecInstanceId | FEC Instance ID when the default FEC Encoding ID 0 is not used for the delivery of FDT |
| ContentEncodingFormat | Format of content encoding if used (e.g., compression) |
| SessOfInterest | Indication to the receiver that a session contains files that are of interest |

#### *B.3.3.3 Multicast Approach*

Source Specific Multicast (SSM) is recommended over Any Source Multicast (ASM) due to security concerns.

#### *B.3.3.4 Security Considerations*

FLUTE security considerations include, but are not limited to:

- Vulnerable to Denial-of-Service attacks
  - Forged packets
- Spoofed FDT Instance packets
  - Could alter Content-Location field to point to Trojan Horse or other virus (if executed by client)
  - Generating a bad Content-MD5 sum
  - Modify Content-Encoding

- Application should perform integrity check on entire reconstructed file, especially for FDT Instances
- Packet source authentication and integrity check
- Clients should be authorized to join a session
- Should secure the process of the client receiving the Session Description information

As described in clause 7 of [ID-FLUTE], there are several security issues with the use of FLUTE as a transport protocol. Because the FLUTE protocol uses FEC, it is vulnerable to denial-of-service attacks where attackers attempt to send forged packets to the session to prevent successful reconstruction or cause inaccurate construction of the FDT of file by clients. Because FLUTE is a multicast transport protocol, these types of attacks are a more significant issue as many clients may receive the same spoofed packet. Also, attackers may spoof FDT Instance packets with incorrect FDT-Instance fields. For example, an attacker could send an FDT-Instance with an incorrect TOI Content-Location field pointing to a malicious system file or user configuration file. Another example is an incorrect Content-MD5 sum causing the receiver to reject the associated file as corrupted. Attackers can also attempt to inject forged packets with incorrect congestion control information into the multicast stream potentially adversely affecting network elements and clients downstream of the attack.

In order to protect against reconstruction of invalid objects, clients MUST follow the software verification procedures described in Section 6.3.3.2.

Another vulnerability is the potential for receivers to receive incorrect session description information, leading to legitimate receivers unable to receive the session content, or attempting to receive at a much higher rate than they are capable of.

# Annex C Software Verification

The requirements defined in this section address these primary security goals for the code download and verification process:

- The Client should have a means to authenticate that the originator of any download code is a known and trusted source;
- The Client should have a means to verify that the downloaded code has not been altered from the original form in which it was provided by the trusted source;
- The process should strive to simplify the Operator's code file-handling requirements and provide mechanisms for the Operator to upgrade or downgrade the code version of Clients on their network;
- The process allows Operators to dictate and control their policies with respect to: (a) which code files will be accepted by Clients within their network; and (b) security controls that define the security of the process on their network;
- Clients are able to move freely among systems controlled by different Operators;
- Support updating Root CA Certificates in the Client;
- Support updating the Manufacturer CA Certificate in the Client;
- Support updating the CableLabs CVC Root CA Certificate in the Client;
- Support updating the CableLabs CVC CA Certificate in the Client.

This document limits its scope to these primary system security requirements, but acknowledges that in some cases additional security may be desired. The concerns of individual Operators or Client manufacturers may result in additional security related to the distribution or installation of code into a Client or other CableLabs network element. This specification does not restrict the use of further protections, as long as they do not conflict with the requirements of this specification.

Multiple levels of protection are required to protect and verify the code download:

- The manufacturer of the Client code always applies a digital signature to the code file. The signature is verified with a certificate chain that extends up to the DOCSIS Root CA or CableLabs CVC Root CA. The manufacturer's signature affirms the source and integrity of the code file to the Client;
- Though the manufacturer always signs its code file, an Operator may later apply its code signature in addition to the manufacturer's signature. If a second signature is present, the Client verifies both signatures with a certificate chain that extends up to the DOCSIS Root CA or CableLabs CVC Root CA before accepting a code file;
- OSS mechanisms for the provisioning and control of the Client are important to the proper execution of this process. The code-upgrade capability of a Client is enabled during the provisioning and registration process. Code downloads are initiated during the provisioning and registration process, or can be initiated in normal operation using management triggers.

The code file is built using a PKCS#7-compliant structure that is defined below. Included in this structure are:

- the code image; i.e., the upgrade code image;
- the Code Verification Signature (CVS); i.e., the digital signature over the code image and any other authenticated attributes as defined in the structure;
- the Code Verification Certificate (CVC); i.e., an PKCS#7-compliant certificate that is used to deliver and validate the public code verification key that will verify the signature over the code image. The DOCSIS Certification Authority or CableLabs CVC Certification Authority, a trusted party whose certificate is already stored in the Client, signs this CVC.

Figure 9 shows the basic steps required for the signing of a code image when the code file is signed only by the Client manufacturer, and when the code file is signed by the Client manufacturer and co-signed by an Operator.

The code manufacturer builds the code file by signing the code image using a CableLabs PKCS#7 digital signature structure with a CVC. The code file is then sent to the Operator. The Operator verifies that the code file is from a trusted manufacturer and has not been modified. At this point, the Operator has the option of loading the code file on the download server as-is, or of adding its signature and Operator CVC to the code file. During the code upgrade process, the CM will retrieve the code file from the downloadserver and verify the new code image before installing it.

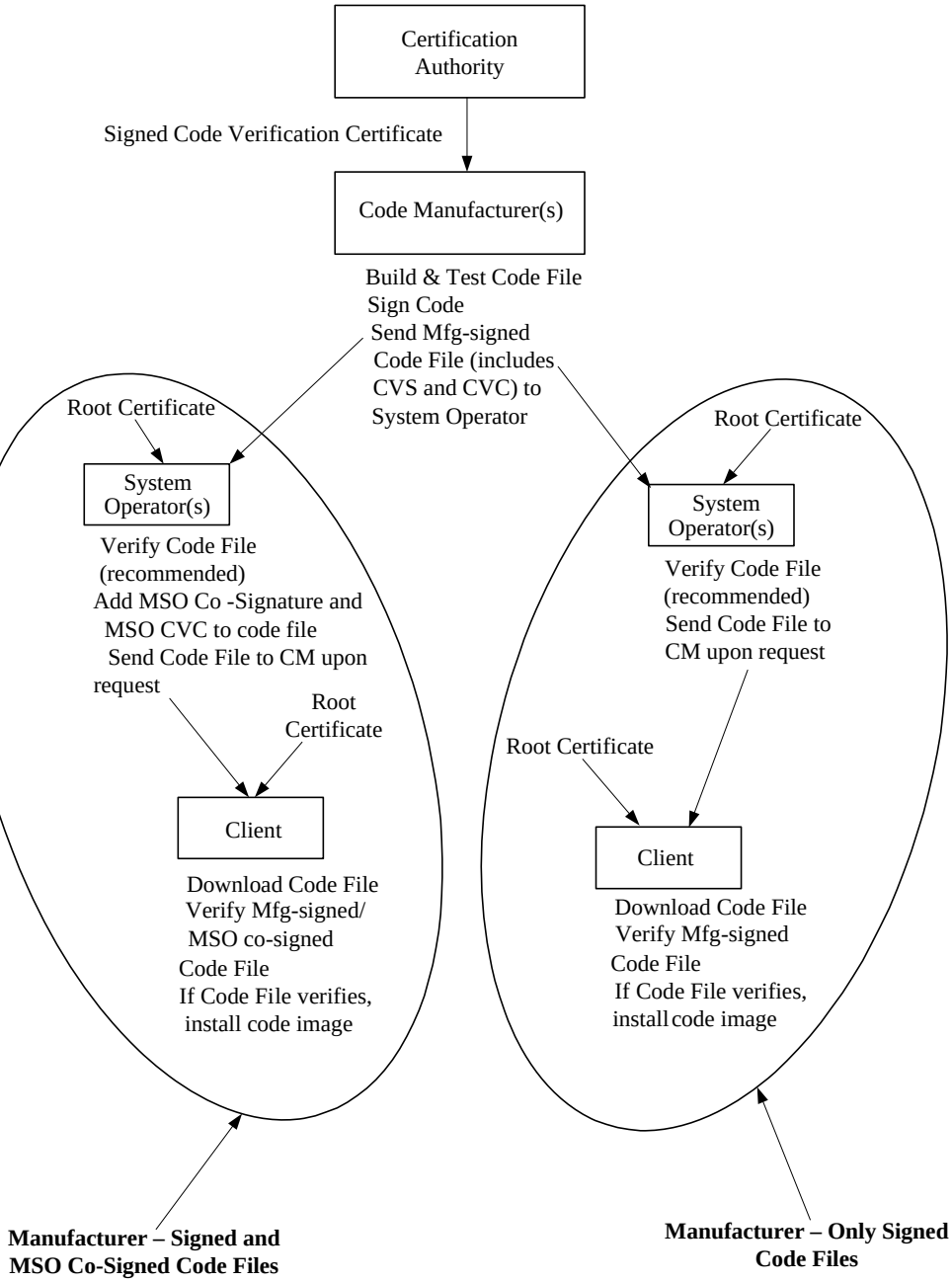



***Figure 9 - Typical Code Validation Hierarchy***

## C.1 Software Code Upgrade Requirements

The following sections define requirements of the Client software code upgrade verification process. All Clients MUST verify code upgrades according to this specification.

### C.1.1 Code File Format

A single file is used to encapsulate the code for the Client. The code file is a CableLabs PKCS#7 signed data message that includes:

- The Manufacturer's Code Verification Signature (CVS);
- The Manufacturer's Code Verification Certificate (CVC);
- The code image (compatible with the destination Client) as signed content;
- Module information of the code image.
- Optionally, when the MSO co-signs the code file:
  - The MSOs CVS;
  - The MSOs CVC.
- Optional DOCSIS Root CA Certificate for CVC verification;
- Optional Manufacturer CA Certificate for Client authentication;
- Optional CableLabs CVC Root CA Certificate for CVC verification;
- Optional CableLabs CVC CA Certificate for CVC verification.

The code file complies with PKCS#7 and is DER encoded. The code file matches the structure shown below.

| Code File Structure | Description |
|---|---|
| PKCS#7 Digital Signature{ | |
| ContentInfo | |
| contentType | SignedData |
| signedData() | EXPLICIT signed-data content value; includes CVS and PKCS#7 CVC |
| } | |
| SignedContent{ | |
| DownloadParameters { | Mandatory TLV format (Type 28) defined in the subclause PKCS#7 |
| ModuleInfo() | Mandatory TLV for providing information about the software module(s) in the code image. |
| DOCSISRootCACert() | Optional TLV for one DER-encoded certificate. |
| MfgCACert() | Optional TLV for one DER-encoded Manufacturer CA Certificate. |
| ClabCVCRootCACert() | Optional TLV for one DER-encoded certificate. |
| ClabCVCCACertificate() | Optional TLV for one DER-encoded certificate. |
| } | |
| CodeImage() | Upgrade code image |
| } | |

#### *C.1.1.1 CableLabs PKCS#7 Signed Data*

The signedData field of the CableLabs PKCS#7 Digital Signature matches the DER encoded structure defined in the following table:.

***Table 8 - CableLabs 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 { | Code Verification Certificates (CVCs) |
| mfgCVC() | Required for all code files |
| msoCVC() | Optional; required for cable operator co-signatures |
| } end certificates | |
| SignerInfo{ | |
| MfgSignerInfo { | Required for all code files |
| Version | version = 1 |
| issuerAndSerialNumber | from the signer's certificate |
| issuerName | distinguished name of the CVC issuer |
| CountryName | country name of the CVC issuer |
| organizationName | organization name of the CVC issuer |
| organizationalUnitName | organizational unit name of the CVC issuer |
| commonName | common name of the CVC issuer |
| certificateSerialNumber | from CVC; Integer, size (1..20) octets |
| digestAlgorithm | SHA-1 |
| authenticatedAttributes | |
| contentType | data; contentType of signedContent |
| signingTime | UTCTime (GMT), YYMMDDhhmmssZ |
| messageDigest | digest of the content as defined in PKCS#7 |
| digestEncryptionAlgorithm | rsaEncryption |
| encryptedDigest | |
| } end mfg signer info | |
| MsoSignerInfo { | OPTIONAL; required for cable operator co-signatures |
| Version | version =1 |

| | |
|---|---|
| issuerAndSerialNumber | from the signer's certificate |
| issuerName | distinguished name of the CVC issuer |
| CountryName | country name of the CVC issuer |
| organizationName | organization name of the CVC issuer |
| organizationalUnitName | organizational unit name of the CVC issuer |
| commonName | common name of the CVC issuer |
| certificateSerialNumber | from CVC; Integer, size (1..20) octets |
| digestAlgorithm | SHA-1 |
| authenticatedAttributes | |
| contentType | data; contentType of signedContent |
| signingTime | UTCTime (GMT), YYMMDDhhmmssZ |
| messageDigest | digest of the content as defined in PKCS#7 |
| digestEncryptionAlgorithm | rsaEncryption |
| encryptedDigest | |
| } end mso signer info | |
| } end signer info | |
| } end signed data | |

#### *C.1.1.1.1 Code Signing Keys*

The digital signature uses the RSA Encryption Algorithm PKCS#7, with SHA-1 PKCS#7. The RSA key modulus is either 1024 bits, 1536 bits, or 2048 bits in length.

#### *C.1.1.1.2 Code Verification Certificate Format*

The format used for the CVC is a DER encoded structure that is PKCS#7 compliant. However, the certificate format may vary due to differences in certificate policy for each CableLabs project.

#### *C.1.1.1.3 Code Verification Certificate Revocation*

The Client is not required to support CVC revocation.

However, there is a method for revoking CVCs based on the validity start date of the certificate. This method requires that an updated CVC be delivered to the Client with an updated validity start time. Once the CVC is successfully validated, the validity start time will update the Client's current value of cvcAccessStart.

To expedite the delivery of an updated CVC without requiring the cable modem to process a code upgrade, the CVC may be delivered in either the Client's configuration file or via management. The format of a CVC is the same whether it is in a code file, configuration file, or sent via management.

### *C.1.1.2 Signed Content*

The SignedContent field of the code file contains the CodeImage and the DownloadParameters fields, which may contain up to four items: a DOCSIS Root CA certificate, a Manufacturer CA certificate, a CableLabs CVC Root CA certificate, and a CableLabs CVC CA certificate.

The final code image is in a binary format compatible with the destination Client. A code image may contain one or more modules used to update different parts of the Client. In support of the PKCS#7 signature requirements, the code content is encoded as an OCTET STRING.

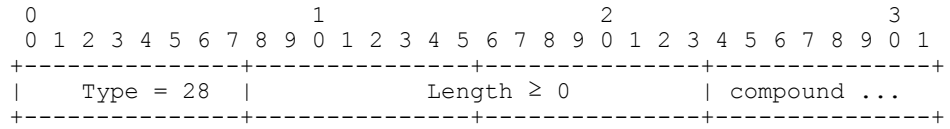
Each manufacturer should build their code with additional mechanisms to verify that an upgrade code image is compatible with the destination Client.

#### *C.1.1.2.1 DownloadParameters Attribute*

This attribute is used in the Client Code File. This is a compound attribute, consisting of an ordered collection of sub-attributes.

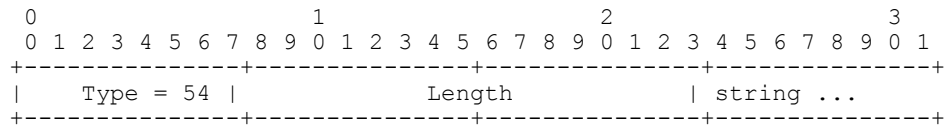
The sub-attributes include zero or more of the following attribute(s) in this order:

- One instance of Module-Info;
- Zero or one instance of DOCSIS-Root-CA-Certificate;
- Zero or one instance of CA-Certificate;
- Zero or one instance of CableLabs-CVC-Root-CA-Certificate;
- Zero or one instance of CableLabs-CVC-CA-Certificate (see Section C.1.1.2.6).




#### *C.1.1.2.2 Module-Info*

This attribute contains a textual message that provides information about the software module(s) in the code image. Information should include name, version, etc., for each module.

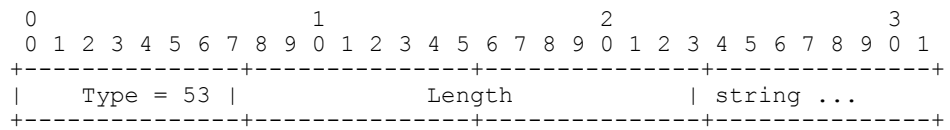



`Length` Variable

`string` A string of octets.

#### *C.1.1.2.3 DOCSIS-Root-CA-Certificate*

This attribute contains a DER-encoded DOCSIS Root CA certificate.

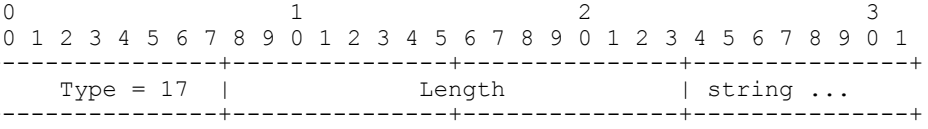


`Length` Variable

`string` DER-encoded DOCSIS Root CA Certificate

#### *C.1.1.2.4 CA-Certificate*

This attribute contains a DER-encoded Manufacturer CA or CableLabs Mfg CA certificate.

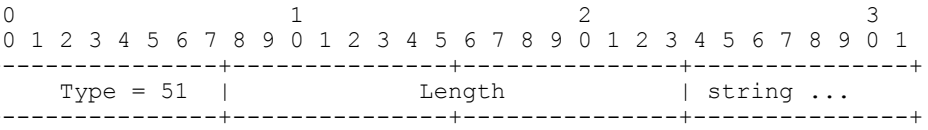



`Length` Variable

`string` DER-encoded Manufacturer CA or CableLabs Mfg CA certificate

#### *C.1.1.2.5 CableLabs CVC-Root-CA-Certificate*

This attribute contains a DER-encoded CableLabs CVC Root CA certificate.

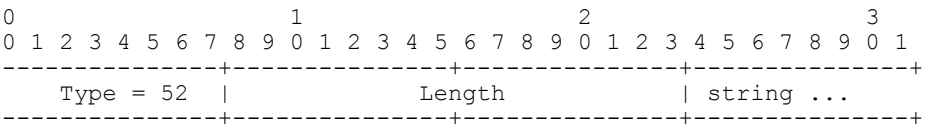



`Length` Variable

`string` DER-encoded CVC Root CA Certificate

#### *C.1.1.2.6 CableLabs CVC-CA-Certificate*

This attribute contains a DER-encoded CableLabs CVC CA certificate.




`Length` Variable

`string` DER-encoded CVC CA Certificate

### C.1.2 Code File Processing Requirements

The Client MUST reject the CableLabs PKCS#7 code file if the signedData field does not match the DER encoded structure in Table 8.

The Client MUST be able to verify CableLabs code file signatures that are signed using key modulus lengths of 1024, 1536 and 2048 bits. The public exponent is $F_4$ (65537 decimal).

The Client MUST NOT install the upgraded code image unless the code image has been verified as being compatible with the Client.

If the code download and installation is successful, then the Client MUST replace its currently stored DOCSIS Root CA certificate with the DOCSIS Root CA certificate in the SignedContent field, if one was present.

If the code download and installation is successful, then the Client MUST replace its currently stored Manufacturer CA certificate with the Manufacturer CA certificate received in the SignedContent field, if any were present.

If the code download and installation is successful, then the Client MUST replace its currently stored CableLabs CVC Root CA certificate and/or CableLabs CVC CA certificate with the CableLabs CVC Root CA certificate and/or CableLabs CVC CA certificate received in the SignedContent field, if any were present.

### C.1.3 Code File Access Controls

In addition to the cryptographic controls provided by the digital signature and the certificate, special control values are included in the code file for the Client to check before it accepts a code image as valid. The conditions placed on the values of these control parameters MUST be satisfied before the Client attempts to validate the CVC and the CVS.

#### *C.1.3.1 Subject Organization Names*

The Client MUST recognize up to two names that it considers a trusted code-signing agent in the subject field of a code file CVC. These are:

- **The cable modem manufacturer:** The Client MUST verify that the manufacturer name in the manufacturer's CVC subject field exactly matches the manufacturer name stored in the Client's non-volatile memory by the manufacturer. A manufacturer CVC is always included in the code file.
- **A co-signing agent:** As described above, CableLabs permits another trusted organization to co-sign code files destined for the Client. In most cases this organization is the Operator. The organization name of the co-signing agent is communicated to the Client via a co-signer's CVC in the configuration file when initializing the Client's code verification process. When validating the code file, the Client MUST verify that the co-signer's organization name in the co-signer's CVC subject field exactly matches the co-signer's organization name previously received in the co-signer's initialization CVC, and stored by the CM.

#### *C.1.3.2 Time Varying Controls*

In support of the code upgrade process, the Client MUST keep two UTC time values associated with each code-signing agent. These values are known as codeAccessStart and cvcAccessStart. The Client MUST store and maintain one pair of time values for the Client manufacturer signing agent. If the CM is assigned a code co-signing agent, the Client MUST maintain a pair of time values for the code co-signing agent.

These values are used to control code file access to the Client by individually controlling the validity of the CVS and the CVC. Stored and maintained time values in the Client MUST have a precision of one second. Stored and maintained time values in the Client MUST be capable of representing all times (with one second precision) between midnight, Jan 1 1950 and midnight Jan 1 2050.

The Client MUST NOT allow the values of codeAccessStart and cvcAccessStart corresponding to the cable modem's manufacturer signing agent to decrease. The Client MUST NOT allow the value of codeAccessStart and cvcAccessStart corresponding to the co-signing agent to decrease as long as the co-signing agent does not change and the CM maintains that co-signer's time-varying control values.

### C.1.4 Cable Modem Code Upgrade Initialization

Before the Client can upgrade code, it should be properly initialized. Its manufacturer first initializes the Client. Every time a Client registers on a CableLabs network, it MUST check its current initialization state with respect to the operational needs of the particular network. It may be necessary for the Client to reinitialize at registration, particularly if the Client has moved from one network to another.

#### *C.1.4.1 Manufacturer Initialization*

It is the responsibility of the manufacturer to install the initial code version in the Client.

In support of code upgrade verification, values for the following parameters MUST be loaded into the Client's non-volatile memory:

- Client manufacturer's organizationName;
- codeAccessStart initialization value;
- cvcAccessStart initialization value.

The manufacturer MUST initialize the values of codeAccessStart and cvcAccessStart to a UTCTime equal to the validity start time of the manufacturer's latest CVC. These values will be updated periodically under normal operation via manufacturer's CVCs that are received and verified by the Client.

In support of the CVC validation process, the manufacturer MUST install the following certificates in the Client's non volatile memory:

- CableLabs CVC Root CA Certificate;
- CableLabs CVC CA Certificate;
- DOCSIS Root CA Certificate.

#### *C.1.4.2 Network Initialization*

The Client receives settings relevant to code upgrade verification in its configuration file. The Client MUST NOT use these settings until after the CMTS has successfully registered the Client.

The configuration file normally includes the most up-to-date CVC applicable for the destination Client. When the configuration file is used to initiate a code upgrade, it will include a CVC to initialize the Client for accepting code files according to this specification. Regardless of whether a code upgrade is required, a CVC in the configuration file MUST be processed by the Client.

A configuration file may contain:

- No CVC;
- A Manufacturer's CVC only;
- A Co-Signer's CVC only;
- Both a Manufacturer's CVC and a Co-Signer's CVC.

Before the Client will enable its ability to upgrade code files from the network, it MUST receive a valid CVC in a configuration file and successfully register. If the Client's configuration file does not contain a valid CVC and its ability to upgrade code files has been disabled, then the Client MUST reject any information in a CVC delivered via management.

When the Client's configuration file does not contain a co-signer's CVC, the Client MUST NOT accept code files that have been co-signed.

If the Client is configured to accept code co-signed by a code-signing agent, the following parameters MUST be stored in the Client's memory when the co-signer CVC is processed:

- co-signing agent's organizationName;
- co-signer cvcAccessStart;
- co-signer codeAccessStart.

Unlike the manufacturer's organizationName and time varying control values, the co-signer organizationName and time varying control values are not required to be stored in non-volatile memory.

#### *C.1.4.2.1 Processing the Configuration File CVC*

When a CVC is included in the configuration file, the Client MUST verify the CVC before accepting any of the code upgrade settings it contains. Upon receipt of the CVC, the Client MUST perform the following validation and procedural steps. If any of the following verification checks fail, the Client MUST immediately halt the CVC verification process. If the Client configuration file does not include a valid CVC, the Client MUST NOT download upgrade code files, whether triggered by the Client configuration file or via management. If the Client configuration file does not include a valid CVC, the Client SHOULD NOT process CVCs subsequently delivered via an management. If the Client configuration file does not include a valid CVC, the Client MUST NOT accept information from a CVC subsequently delivered via management.

Following receipt of a CVC in a configuration file, the Client MUST:

1. If the CVC is a Manufacturer's CVC (Type 32), verify that the manufacturer's CVC validity start time is greater than or equal to the manufacturer's cvcAccessStart value currently held in the Client and the subject organizationName is identical to the Client's manufacturer name. If either of these conditions are not true, reject this CVC and log an error.
2. If the CVC is a Co-signer's CVC (Type 33) and the subject organizationName is identical to the CM's current code co-signing agent, verify that the validity start time is greater than or equal to the co-signer's cvcAccessStart value currently held in the Client. If this condition is not true, reject this CVC and log an error.
3. If the CVC is a Co-signer's CVC (Type 33) and the subject organizationName is not identical to the current code co-signing agent name, make this subject organization name become the Client's new code co-signing agent.
4. Verify the CVC chains to a root CA certificate held by the Client using the method defined by the Basic Path Validation section in [RFC 3280].
5. Update the Client's current value of cvcAccessStart corresponding to the CVC's subject organizationName (i.e., manufacturer or code co-signing agent) with the validity start time value from the validated CVC. If the validity start time value is greater than the Client's current value of codeAccessStart, update the Client's codeAccessStart value with the validity start time value.

#### *C.1.4.2.2 Processing CVC obtained via management*

The Client MUST process CVCs received via management when it is enabled to upgrade code files. When the Client is not enabled to upgrade code files, it MUST reject all CVCs received via SNMP. When validating a CVC received via management, the Client MUST perform the following validation and procedural steps. If any of the following verification checks fail, the Client MUST immediately halt the CVC verification process, log the error if applicable, and remove all remnants of the process up to that step.

When a Client receives a CVC via management, it MUST:

1. Verify that the manufacturer's CVC validity start time is greater than the manufacturer's cvcAccessStart value currently held in the CM, if the CVC subject organizationName is identical to the CM's manufacturer name .
2. Verify that the validity start time is greater than the co-signer's cvcAccessStart value currently held in the CM, if the CVC subject organizationName is identical to the cable modem's current code co-signing agent.
3. Reject this CVC if the CVC subject organizationName is not identical to CM's manufacturer or current code co-signing agent name.
4. Verify the CVC chains to a root CA certificate held by the Client using the method defined by the Basic Path Validation section in [RFC 3280].
5. Update the current value of the subject's cvcAccessStart values with the validated CVC's validity start time value. If the validity start time value is greater than the Client's current value of codeAccessStart, the Client MUST replace its codeAccessStart value with the validity start value.

### C.1.5 Code Signing Guidelines

Client vendors and Operators can control the Secure Software Download process based on their policies by updating the Manufacturer/Co-Signer CVC or by changing the signingTime in the Manufacturer/Co-Signer CVS (Code Verification Signature). At this time, this specification does not specify the policy related to the Client Code File signing process. However, an example of a policy is provided in this section.

#### *C.1.5.1 Manufacturer Client Code File Signing Policy*

In order to sign code files, the manufacturer obtains a valid CVC from CableLabs.

When signing a code file, a manufacturer may choose not to update the PKCS#7 signingTime value in the manufacturer's signing information. The signingTime value should be equal to or greater than the CVC's validity start time. The PKCS#7 signingTime value is not less than the CVC's validity start time.

The Client vendor and its Manufacturer Code Signing Agent (Mfg CSA), which securely stores the RSA private key corresponding to the RSA public key in the Manufacturer CVC and generates the CVS for the Client Code File, might employ the following policy for the Client Code File signing process.

The Mfg CSA continues to put exactly the same date and time value (T1) in the signingTime field in the Mfg CVS of the Client Code File as long as the vendor does not have any Client Code File to revoke.

Once the vendor realizes there are certain issues in one or more Client Code File(s) and wants to revoke them, the vendor chooses the current date and time value (T2) and starts using T2 as the signingTime value in the Mfg CVS for all the newly created Client Code File(s) from that point. In addition, any Client Code files signed with T1 that are still good are re-signed using T2.

Under this policy, because the multiple Client Code Files make a group of the Client Code Files with the exact same signingTime value in the Mfg CVS, the Operator can download any Client Code File in the group in any order. That is, among the Client Code Files in the same group, the Client’s software can be downgraded if necessary.

#### *C.1.5.2 Operator CM Code File Signing Policy*

Operators should verify that the code image as received from a vendor is exactly as built by the trusted manufacturer. The Operator may choose to co-sign code images destined for use on its network. All code images downloaded to a Client across the network are signed in accordance with this specification.

### C.1.6 Code Verification Requirements

The Client MUST NOT install upgrade code unless the code has been verified.

#### *C.1.6.1 Client Code Verification Steps*

When downloading code, the Client MUST perform the verification checks presented in this sub clause. If any of the verification checks fail, or if any of the code file is rejected due to invalid formatting, the Client MUST 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. The verification checks can be made in any order.

1. For the manufacturer code file signature, the Client MUST verify that:
   - the value of signingTime is equal to or greater than the manufacturer's codeAccessStart value currently held in the Client;
   - the value of signingTime is equal to or greater than the manufacturer's CVC validity start time;
   - the value of signingTime is less than or equal to the manufacturer's CVC validity end time.
2. For the manufacturer CVC, the Client MUST verify that:
   - the CVC subject organizationName is identical to the manufacturer name currently stored in the Client's memory;
   - the CVC validity start time is equal to or greater than the manufacturer's cvcAccessStart value currently held in the Client;
3. The Client MUST verify the manufacturer CVC chains to a root CA certificate held by the Client using the method defined by the Basic Path Validation section in [RFC 3280].
4. The Client MUST verify the manufacturer's code file signature. If the signature does not verify, the Client MUST reject all components of the code file (including the code image), and any values derived from the verification process should be immediately discarded.
5. If the manufacturer signature verifies and a co-signing agent signature is required:
   a. For the co-signer code file signature, the Client MUST verify that:
      1. the co-signer's signature information is included in the code file;
      2. the value of signingTime is equal to or greater than the corresponding codeAccessStart value currently held in the Client;
      3. the value of signingTime is equal to or greater than the corresponding CVC validity start time;
      4. the value of signingTime is less than or equal to the corresponding CVC validity end time.
   b. For the co-signer CVC, the Client MUST verify that:
      1. the CVC subject organizationName is identical to the co-signer's organization name currently stored in the Client's memory;
      2. the CVC validity start time is equal to or greater than the cvcAccessStart value currently held in the Client for the corresponding subject organizationName.
   c. The Client MUST verify the co-signer CVC chains to a root CA certificate held by the Client using the method defined by the Basic Path Validation section in [RFC 3280].
   d. The Client MUST verify the co-signer's code file signature. If the signature does not verify, the Client MUST reject all components of the code file (including the code image), and any values derived from the verification process should be immediately discarded.
6. Once the manufacturer's, and optionally the co-signer'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.

7. The Client upgrades its software by installing the code file.
8. If the code installation is unsuccessful, the Client MUST discard the PKCS#7 signingTime values and CVC validity start values it just received in the code file.
9. Once the code installation is successful, the Client MUST:
   a. update the current value of manufacturer codeAccessStart with the PKCS#7 signingTime value;
   b. update the current value of manufacturer cvcAccessStart with the CVC validity start value.
10. If the code installation is successful, and if the code file was co-signed, the Client MUST:
    a. update the current value of the co-signer's codeAccessStart with the PKCS#7 signingTime value;
    b. update the current value of the co-signer's cvcAccessStart with the CVC validity start value.

### C.1.7 Error Codes

The Client MUST log the following nine error events when they occur during the code verification process:

1. Improper code file controls

   Conditions:

   a. CVC subject organizationName for manufacturer does not match the Client's manufacturer name.
   b. CVC subject organizationName for code co-signing agent does not match the Client's current code co-signing agent.
   c. The manufacturer's PKCS#7signingTime value is less-than the codeAccessStart value currently held in the Client.
   d. The manufacturer's PKCS#7 validity start time value is less-than the cvcAccessStart value currently held in the Client.
   e. The manufacturer's CVC validity start time is less-than the cvcAccessStart value currently held in the Client.
   f. The manufacturer's PKCS#7 signingTime value is less-than the CVC validity start time.
   g. The co-signer's PKCS#7 signingTime value is less-than the codeAccessStart value currently held in the Client.
   h. The co-signer's PKCS#7 validity start time value is less-than the cvcAccessStart value currently held in the Client.
   i. The co-signer's CVC validity start time is less-than the cvcAccessStart value currently held in the Client.
   j. The co-signer's PKCS#7 signingTime value is less-than the CVC validity start time.
2. Code file manufacturer CVC validation failure
3. Code file manufacturer CVS validation failure
4. Code file co-signer CVC validation failure
5. Code file co-signer CVS validation failure
6. Improper Configuration File CVC format

   Conditions:

   a. CVC not compliant with X.509 standard.

7. Configuration File CVC validation failure
8. Improper Management CVC format

   Conditions:

   a. CVC subject organizationName for manufacturer does not match the Client's manufacturer name.

   b. CVC subject organizationName for code co-signing agent does not match the Client's current code co-signing agent.

   c. the CVC validity start time is less-than or equal-to the corresponding subject's cvcAccessStart value currently held in the Client.
9. Management CVC validation failure

## C.2 Security Considerations

The protection afforded private keys is a critical factor in maintaining security. Users authorized to sign code, i.e., manufacturers and Operators who have been issued code-signing verification certificates (CVCs) by CableLabs, should protect their private keys. An attacker with access to the private key of an authorized code-signing user can create, at will, code files that are potentially acceptable to a large number of Clients.

The defense against such an attack is for the Operator to revoke the certificate whose associated code-signing private key has been learned by the attacker. To revoke a certificate, the Operator delivers to each affected Client, an updated CVC with a validity start time that is newer than that of the certificate(s) being revoked. The new CVC can be delivered via any of the supported mechanisms: configuration file, code file, or SNMP. The new CVC implicitly revokes all certificates whose validity start time is earlier than that of the new CVC.

To reduce the vulnerability to this attack, Operators should regularly update the CVC in each CM, at a frequency comparable to how often the Operator would update a CRL if one were available. Regular updates help manage the time interval during which a compromised code-signing key is useful to an attacker. CVCs should also be updated if it is suspected that a code-signing key has been compromised. To update the CVC, the user needs a CVC whose validity start time is newer than the CVC in the CM. This implies that the Certification Authority regularly issues new CVCs to all authorized code-signing manufacturers and Operators, to make the CVCs available for update.

When a Client is attempting to register on the network for the first time or after being off-line for an extended period, it should receive a trusted CVC as soon as possible. This provides the Client with the opportunity to receive the most up-to-date CVC available and deny access to CVCs that need to be revoked since the Client last initialization. The first opportunity for the Client to receive a trusted CVC is in its configuration file. If the configuration file does not include a valid CVC, the Client will not request or have the ability to remotely upgrade code files. In addition, the Client will not accept CVCs subsequently delivered via SNMP.

To mitigate the possibility of a Client receiving a previous code file via a replay attack, the code files include a signing-time value in the PKCS#7 structure that can be used to indicate the time the code image was signed. When the Client receives a code file signing-time that is later than the signing-time it last received, it will update its internal memory with this value. The Client will not accept code files with an earlier signing-time than this internally stored value. To upgrade a Client with a new code file without denying access to past code files, the signer may choose not to update the signing-time. In this manner, multiple code files with the same code signing-time allow an Operator to freely downgrade a Client’s code image to a past version (that is, until the CVC is updated). This has a number of advantages for the Operator, but these advantages should be weighed against the possibilities of a code file replay attack.

Without a reliable mechanism to revert back to a known good version of code, any code-update scheme, including the one in this specification, has the weakness that a single, successful forced update of an invalid code image by a Client may render the Client useless, or may cause the Client to behave in a manner harmful to the network. Such a

Client may not be repairable via a remote code update, since the invalid code image may not support the update scheme.

# Annex D Software Management Reporting

ITU Recommendation [X.733] defines a standard alarm reporting function. This section defines the specific events and fields within the events which align with the [X.733] Recommendation.

## D.1 Event Fields

### D.1.1 Event Category

See the Textual Convention IANAItuEventType from the IANA-ITU-ALARM-TC-MIB of [RFC 3877]. IANAItuEventType defines additional values beyond X.733.

| X.733 Event Type |
|---|
| Communications Alarm |
| QoS Alarm |
| Processing Error Alarm |
| Equipment Alarm |
| Environmental Alarm |

### D.1.2 Event Cause

See the Textual Convention IANAItuProbableCause from the IANA-ITU-ALARM-TC-MIB of [RFC 3877]. IANAItuProbableCause defines additional values beyond X.733.

### D.1.3 Event Severity

See the Textual Convention ItuPerceivedSeverity from the ITU-ALARM-TC-MIB of [RFC 3877].

| X.733 Perceived Severity |
|---|
| Cleared |
| Indeterminate |
| Critical |
| Major |
| Minor |
| Warning |

### D.1.4 Event Text

See the Textual Convention SnmpAdminString from the SNMP-FRAMEWORK-MIB of [RFC 3411].

## D.2 Event Definitions

***Table 9 - Event Definitions***

| Cause | Category | Severity | Text | Notes |
|---|---|---|---|---|
| SwDownloadFailure | Processing Error | Warning | Software Download Failed: <reason description> | Possible reasons include: transport protocol max retries exceeded, file server not found, file not found on server, power disruption, link broken, insufficient resources |
| SwFileVerificationFailure | Processing Error | Warning | Software Downloaded Failed Verification: <reason description> | Possible reasons include: incompatible SW file, file corrupted |
| SwAuthenticationFailure | Processing Error | Warning | Software Downloaded Failed Authentication: <reason description> | Possible reasons include: improper code file controls, code file manufacturer CVC validation, code file manufacturer CVS validation, code file co-signer CVC validation, code file co-signer CVS validation, improper configuration file CVC format, configuration file CVC validation, improper SNMP CVC format, SNMP CVC validation |

# Appendix I Use Cases and Examples

## I.1 FLUTE Use Case

This section describes a general FLUTE use case for a file delivery and associated FDT instance delivery.

1. Cable client receives Session Description information (see subsections that follow).
2. Cable client joins the channels in order to receive packets associated with the Session.
3. Cable client receives packets and checks against the TSI; discarding those that don't match.
4. Cable client demultiplexes and stores packets based on TOI for later reconstruction.
5. Cable client recovers a file based on the amount of packets received.
   a. If the file is an FDT Instance, the cable client updates its database.
   b. If the file is not an FDT Instance but a delivered file, the cable client looks up the properties for the file from its database and assigns to the file (also performs some other checks against the file and what's stored in the DB).
   c. If a file is recovered before the associated file description entry is available, the cable client may wait for the FDT Instance.
6. The process ends if the file delivery session end time has been reached; otherwise, processing goes back to step 3.

## I.2 Example SSD FDT Instance Doc

The following represents an FDT Instance describing a CM monolithic firmware image file "cm_image1.bin" without any compression encoding and an extension field "Digital-Signature" defined specifically for CableLabs.

```
<?xml version="1.0" encoding="UTF-8"?>
<FDT-Instance xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ietf:params:xml:ns:fdt
                                ietf-flute-fdt.xsd"
Expires="2890842807">
        <File
           Content-Location=
               "http://www.cablelabs.com/DOCSIS/cm_image1.bin"
           TOI="1"
           Content-Length="6126300"
           Content-Type= "text"
           Digital-Signature="SHA-1 with RSA"/>
</FDT-Instance>
```

## I.3 DOCSIS FLUTE Session Configuration

There are two DOCSIS use cases for FLUTE session configuration. Refer to Table 4 and Table 5 for details on the required and optional parameters that are configured with the following two use cases.

- CM Configuration File TLVs

DOCSIS defines TLVs (Type/Length/Values) which are encoded into a binary CM configuration file which is downloaded a boot time. The file performs configuration of various parameters within the CM. SNMP MIB objects can also be included as TLV-11 entries, to provide configuration of SNMP objects that support write access.

- SNMP MIB Objects

DOCSIS defines several standard and CableLabs enterprise specific MIBs for configuring the CM. Such MIB objects can be set at run time by a Network Management System, or at boot time by including the MIB objects as TLV-11 entries in the CM configuration file.

## I.4 SDP FLUTE Session Configuration

This section illustrates a use case for a FLUTE session configuration via SDP.

The FLUTE/UDP protocol identifier is used to denote the session as FLUTE over UDP. It's not necessary to identify a media type (MIME types). There are two different behavior descriptions:

- Restricted Behavior: SDP defines one session

  The attributes Src Addr, TSI, Timing, Channel/FEC are defined per Session (1 instance).

- Composite Session: SDP defines multiple sessions

  The attributes Src Addr, TSI, Channel/FEC are defined per Primary Media.

  The first media line declared for a Composite Session group is the Primary Media.

  Multiple session descriptions can be defined.

An SDP for FLUTE Session with One Channel is shown below.

```
v=0
o=user123 2890844526 2890842807 IN IP6 2201:056D::112E:144A:1E24
s=File delivery session example
i=More information
t=2873397496 2873404696
a=source-filter: incl IN IP6 * 2001:210:1:2:240:96FF:FE25:8EC9
a=flute-tsi:2
a=flute-ch:1
m=application 12345 FLUTE/UDP *
c=IN IP6 FF1E:03AD::7F2E:172A:1E24
a=FEC-declaration:0 encoding-id=128; instance-id=0
```

# Appendix II Acknowledgements

CableLabs would like to thank the following individuals for their contributions to the development of this specification.

Balu Ambady

Brian Hedstrom

Deepak Kharbanda

Joel Michon

Lakshmi Raman

Stuart Hoggan

Sumanth Channabasappa

Brian Hedstrom is thanked for his efforts related to the transport protocols, Lakshmi Raman for the Information Model and trigger requirements, Stuart Hoggan for the requirements around Software Verification, Deepak Kharbanda for managing reviews and incorporating feedback, and Sumanth for coordinating the efforts. We would also like to thank Balu Ambady for feedback related to security and early contributions, and Joel Michon for the OpenCable related clarifications.