OpenCable™ Specifications
ETV

Enhanced TV Application Messaging Protocol 1.0

OC-SP-ETV-AM1.0.1-120614

Issued

Notice

Document Status Sheet

Document Control Number:	OC-SP-ETV-AM1.0.1-1206xx
Document Title:	Enhanced TV Application Messaging Protocol 1.0
Revision History:	I01 - Issued April 18, 2005
	I02 - Issued July 27, 2005
	I03 - Issued July 14, 2006
	I04 - Issued September 21, 2007
	I05 - Issued November 25, 2009
	I06 - Issued January 28, 2011
	1.0.1 - Issued June 14, 2012
Date:	June 14, 2012

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

List of Figures

List of Tables

1   SCOPE

1.1   Introduction and Overview

Broadcasters and network operators around the world are deploying interactive applications by creating enhancements to a broadcast video stream. These Enhanced Television (ETV) applications rely on embedding various types of data in the video stream, including programs, images, and triggers.

This document specifies the synchronization and signaling mechanisms to be used by ETV applications, regardless of the target receiver or middleware environment. ETV mechanisms must be implementable by legacy set-top boxes as well as OpenCable (OCAP) host devices, and this implementation requirement implies that more than one option must exist for the physical transmission of the signaling and trigger data. This document addresses those various options and describes how a set-top box should interpret signals and triggers delivered via each of those methods.

1.2   Purpose of document

The purpose of this document is to specify ETV application signaling and synchronization mechanisms that meet all of the objectives/requirements of North American cable systems for delivering video-synchronous ETV applications, whether they are broadcast or delivered on-demand.

The intent is to propose a uniform method of inserting signals and triggers that is independent of application environments and software/technology vendors. That said, it is understood that accommodations must be made for the support of specific legacy set-top boxes such as the DCT-2000 and Explorer 2000, while also supplying a standard rich enough to work with advanced set-top boxes based on the OpenCable Host 2.0 Core Functional Requirements [HOST2.1].

In some cases, the need to support a range of devices may result in the need to have more than one signaling packet delivered through the network for the same application. As the number of legacy set-top boxes drops to zero, in any given division over the next several years, this requirement would be relaxed.

This document does not attempt to impose a selection of a particular vendor for implementation. The design of the system is largely based on open industry standards with an objective to leverage currently existing equipment and tools available for implementing such a system.

1.3   Organization of document

This document is divided into four parts:

- a description of the type of applications to be addressed by this specification,
- application signaling and life-cycle management,
- application synchronization and timeline management,
- platform-specific constraints imposed by legacy environments.

1.4 Requirements

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

"SHALL"   This word means that the item is an absolute requirement of this specification.

"SHALL NOT"   This phrase means that the item is an absolute prohibition of this specification.

"SHOULD"   This word means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

"SHOULD NOT"   This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

"MAY"   This word means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

2 REFERENCES

2.1 Normative References

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

[ETV-BIF]	OC-SP-ETV-BIF1.0.1-120614, Enhanced Television (ETV) Binary Interchange Format 1.0, June 14, 2012, Cable Television Laboratories, Inc.
[IEC 13818-1]	ISO/IEC 13818-1:2000, Information technology—Generic coding of moving pictures and associated audio information: Systems, 2000.
[IEC 13818-6]	ISO/IEC 13818-6:1998(E), Information technology—Generic coding of moving pictures and associated audio information: Extensions for DSM-CC, 1998.
[MHP]	ETSI ES 201 812 v1.1.2 (2006-08), DVB Multimedia Home Platform (MHP) 1.0.3.
[OCAP1.2]	OC-SP-OCAP1.2.3-120531, OpenCable Application Platform Specification, Profile 1.2, May 31, 2012, Cable Television Laboratories, Inc.
[RFC 4122]	IETF RFC 4122, A Universally Unique IDentifier (UUID) URN Namespace, July 2005.
[RFC 4648]	IETF RFC 4648, The Base16, Base32, and Base64 Data Encodings, October 2006.
[SMPTE 343M]	SMPTE 343M-2002, Declarative Data Essence – Local Identifier (lid:) URI Scheme.
[UTF-8]	IETF RFC 3629, UTF-8, A Transformation Format of ISO 10646.

2.2 Informative References

[HOST2.1]	OC-SP-HOST2.1-CFR-I16-120531, OpenCable Host 2.1 Core Functional Requirements, May 31, 2012, Cable Television Laboratories, Inc.

2.3 Reference Acquisition

CableLabs Specifications:

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027;
Phone 303-661-9100; Fax 303-661-9199; Internet: http://www.cablelabs.com

ETSI Specifications:

- European Telecommunications Standards Institute, http://www.etsi.org

ISO Specifications:

- International Organization for Standardization, http://www.iso.org

IETF RFC Specifications

- The Internet Engineering Task Force (IETF), http://www.ietf.org/

3   TERMS AND DEFINITIONS

This specification uses the following terms:

Application Signal	A broadcast message that provides information to a receiver necessary to acquire, launch, and terminate an ETV application.
Enhanced Television (ETV)	A general term that refers to interactive services and applications provided in conjunction with video programming.
Enhancement	A software application that executes in conjunction with video programming.
Trigger	A broadcast message that provides a synchronization mechanism to an enhancement. Triggers may be embedded in the associated video program, or delivered via another means such as OOB. Triggers may also be used for the delivery of unsolicited data to an enhancement. Triggers may include application signals and stream events.
Stream Event	A type of Trigger that conveys application defined messages to an enhancement. ETV stream events are normatively defined within this specification.
User Agent	An application running on a receiver that decodes and executes the enhancement.

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations:

AIT	Application Information Table
bslbf	bit-string left bit first
DCII	DigiCipher II
DSM-CC	Digital Storage Module - Command and Control
DTD	Document Type Definition
DVR	Digital Video Recorder
EBIF	ETV Binary Interchange Format
ETV	Enhanced Television
EISS	ETV Integrated Signaling Stream
IB	In-band
NPT	Normal Play Time
OOB	Out-of-Band
PMT	Program Map Table
PTS	Presentation Time Stamp
rpchof	remainder polynomial coefficients, highest order first
uimsbf	unsigned integer most significant bit first
URI	Uniform Resource Identifier
VBI	Vertical Blanking Interval

5   ENHANCED TELEVISION APPLICATION MODEL

5.1   ETV Applications

This specification is intended to support a wide variety of program synchronous applications, such as:

- Interactive Advertising
- Game Shows
- News
- Sports Events
- Voting applications
- Impulse upgrade promotions
- E-commerce applications

This specification addresses both live broadcasts and pre-recorded programs and supports "real-time" viewing as well as time-shifted (DVR) viewing and interaction.

5.2   ETV Authoring Process

There are several important factors in the creation and deployment of ETV applications:

- Production Team
- Cable Operator
- Network Broadcaster
- Application Server

Figure 1 provides a graphical illustration of the relationships between these functional groups.

5.2.1   Production Team

Application production teams generate the interactive enhancements in conjunction with the studios that produce the video. Applications are often built around templates for the triggers and data that are inserted by a production team.

For pre-recorded shows, the application signaling and triggers are mastered during the video post-production process before the show is broadcast. Throughout the production process, the production team uses a media timeline based on SMPTE time codes, which are used for synchronization of video tape.

For live shows, the application itself may be prepared in advance, but the actual data to feed the application is inserted dynamically—for example: Who just won the Oscar for Best Director?

5.2.2   Network Broadcaster

Enhancements cannot be bound to a program until after the video has been digitized for final broadcast. Furthermore, the video feeds from the major networks are often converted back to analog by a local network affiliate and then re-digitized for use by the local cable company, losing many enhancements along the way. Cable channels have the advantage of staying in the digital domain and so can insert all enhancements at the digital encoding stage.

Applications and triggers must be inserted into the video stream by the broadcaster using synchronization triggers and play lists that are appropriate for the type of equipment used in the broadcast environment. At the same time, broadcasters may have to rely on sending those enhancements across a broadband connection to cable companies and affiliates who do not preserve the full digital signal from beginning to end. Some enhancements can be sent ahead of time for automatic insertion on a given schedule. Others may be transmitted in real time and synchronized to live events—either by production teams directly, or via the broadcasters or the cable operators.

5.2.3   Cable Operators

Some cable operators generate their own programming and enhancements. Again, they are responsible for obtaining the enhancement from the Production Team and inserting the application, signaling and triggers into the video streams, adding to or replacing video segments received from the satellite. At this point, cable operators may be

working in either the analog or digital domain, because the video may be in either state before final transmission to the subscriber. Unfortunately, analog and digital video place different requirements on the data insertion equipment.

5.2.4   Application Servers

Although it probably does not signal a base enhancement directly, a polling or score server MAY be used to dynamically process subscriber votes or quiz answers and send responses back to particular client receivers. Messages sent from the polling server MAY require routing or insertion by the cable operator back into either an IB or OOB data stream to a client.
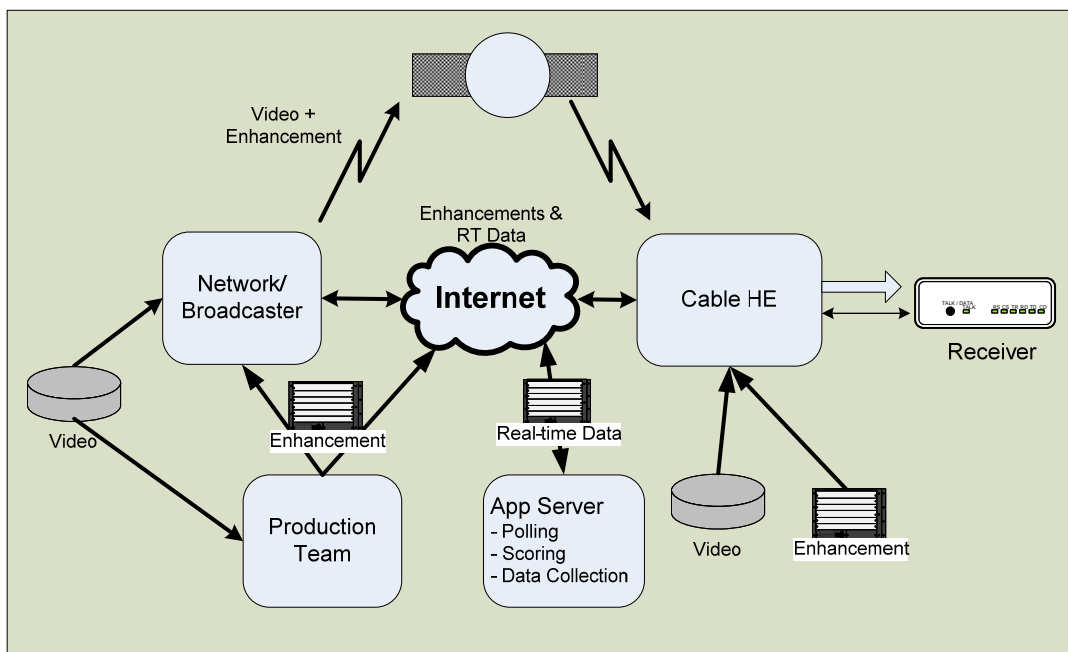
Figure 1 - Enhancement Distribution Process

5.3   Enhanced Television Components

An enhanced television application is made up of several components:

- Broadcast Audio/Visual program
- ETV Application Program and Resource Data
- Application Signaling
- Stream Events/Triggers
- Media Timeline

In addition, an ETV application MAY send data to an application server. Responses from the server MAY come in the form of new application resources or stream events.

These components are illustrated in Figure 2.
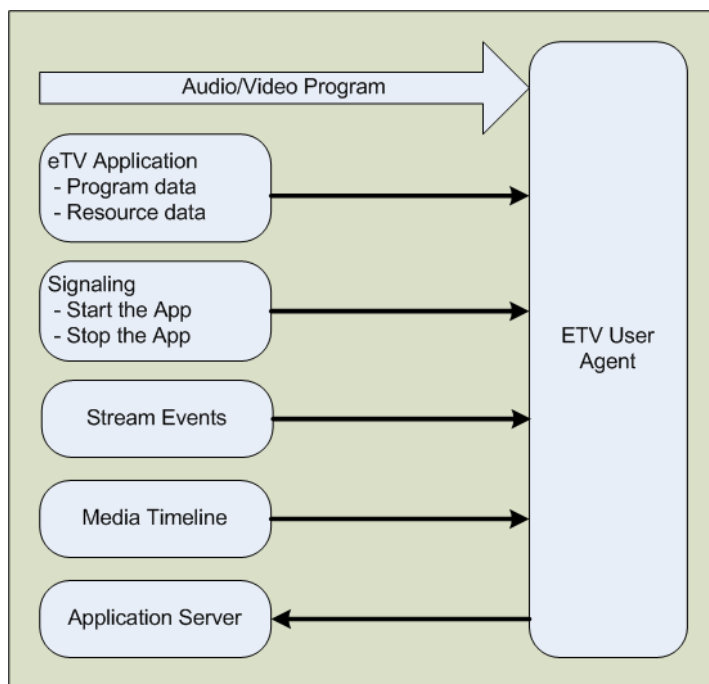
Figure 2 - **Enhanced Television components**

While all of these components together complete the ETV application, it helps to look at them separately in the context of discussing their delivery and formats:

- The **ETV application** is the subject of other specifications, such as OpenCable Enhanced Television (ETV) [ETV-BIF], although this specification will discuss the delivery and encapsulation of the ETV application data.
- **Signaling** commands defined in this specification tell the ETV User Agent in the client receiver to start and stop the application, and identify how to find and load the primary elements of the application.
- **Stream Events** deliver "unsolicited" data to an application, synchronized with the video stream.
- The **Media Timeline** provides the basic timing references to which stream events are synchronized.
- ETV **Application Servers** are used for collecting the results of an ETV application, such as requests for additional information or storing poll or voting responses.

One advantage of separating the ETV application format from this signaling specification is that this specification can be used to signal a number of different types of applications, including ones written to OpenCable Enhanced Television (ETV) [ETV-BIF], as well as to other specifications yet to be written. In general, the problems of signaling and media synchronization are universal, regardless of the type of application being signaled.

5.4   Media Timeline

A Media Timeline is a continuous timeline over the duration of an event. An event is defined in [IEC 13818-1] as a collection of elementary streams with a common time base, an associated start time, and an associated end time. A typical, but not normative example, is the video and audio for a single television show.

The Media Timeline refers to the real time of the event. For example, when an event is presented in reverse, the timeline counts down rather than up; and when an event is presented at 10 times the normal rate, the timeline progresses at 10 times the normal rate. In this way, the Media Timeline increases and decreases in a way similar to a counter on a video tape recorder and provides an absolute timeline to which references can be made for operations such as jumping to a particular point in the event.

Media timelines can be paused, for example, during a commercial break which occurs interstitially between two segments of the event. Media timelines can also be nested, for example, if the commercial itself contains an enhancement.

6   ENHANCED TELEVISION APPLICATION SIGNALING

6.1   Introduction

This section defines the data formats and transmission mechanisms that enable receivers to discover, download, launch, and terminate ETV applications.

6.2   Program Map Table Descriptors

The following PMT descriptors are defined by this specification:

- ETV registration descriptor
- ETV integrated signaling descriptor
- ETV-BIF platform descriptor

6.2.1   ETV Registration Descriptor

The `etv_registration_descriptor` is defined for use in the elementary stream information loop of the PMT. The ETV registration descriptor [IEC 13818-1], table 2-46 – Registration Descriptor, clause 2.6.8 is defined to identify unambiguously the programs that comply with this specification. The ETV registration descriptor SHALL be carried in the elementary stream information loop of the PMT for each program component that conveys an ETV integrated signaling stream or ETV application resource stream. The presence of the registration descriptor uniquely and unambiguously identifies the formats of the private data used within this specification, including private use fields and identifiers within *user private* ranges such as descriptor tags and MPEG private section table ids.

The `etv_registration_descriptor` is defined in Table 1.

Table 1 - ETV Registration Descriptor Syntax

Syntax	Bits	Mnemonic
etv registration descriptor() {		
descriptor tag	8	uimsbf
descriptor length	8	uimsbf
etv format identifier	32	bslbf
}		

descriptor_tag This 8-bit integer with value `0x05` identifies this descriptor.

descriptor_length This 8-bit integer indicates the number of bytes following the descriptor length field.

etv_format_identifier CableLabs has assigned a value of 0x45545631 (ASCII "ETV1") to this 4-byte field to identify the program component (within a multiplex) in which it is carried as complying with this specification.

Note: This value must be registered with the SMPTE Registration Authority, LLC and is subject to change. SMPTE is authorized by the International Organization for Standardization (ISO) to register MPEG-2 format identifiers per [IEC 13818-1]:2000 Annex O: Registration procedure [for "format_identifier"].

6.2.2   ETV Integrated Signaling Descriptor

The `etv_integrated_signaling_descriptor` is defined for use in the elementary stream information loop of the PMT. This descriptor indicates that the associated elementary stream contains an ETV integrated signaling stream, as defined in Section 7 of this specification. Only one elementary stream signaled by the PMT SHALL contain an `etv_integrated_signaling_descriptor`. In the event that more than one PMT entry contains an `etv_integrated_signaling_descriptor`, the behavior of the ETV receiver is undefined.

A PMT entry with an `etv_integrated_signaling_descriptor` SHALL be associated with a stream_type of 0xC0 or 0x05.

The `etv_integrated_signaling_descriptor` is defined in Table 2.

Table 2 - ETV Integrated Signaling Descriptor Syntax

Syntax	Bits	Mnemonic
etv integrated signaling descriptor() {		
descriptor tag	8	uimsbf
descriptor length	8	uimsbf
platform id length	8	uimsbf
for (i=0; i<m; i++){		
etv bif platform id()		
}		
for (i=0; i<n; i++) {		
private use	8	bslbf
}		
}		

descriptor_tag	This 8-bit integer with value `0xA2` identifies this descriptor.
descriptor_length	This 8-bit integer indicates the number of bytes following the descriptor length field.
platform_id_length	This 8-bit integer specifies the number of bytes of the `etv_bif_platform_id` fields immediately following this `platform_id_length` field. This value MAY be zero.
etv_bif_platform_id	This field contains zero or more `etv_bif_platform_id` structures, as defined in Table 4.
	If `platform_id_length` is zero (i.e., there are no `etv_bif_platform_id` structures listed by this descriptor), the associated elementary stream SHALL be decoded on ALL platforms. If `platform_id_length` is non-zero, this descriptor SHALL contain an `etv_bif_platform_id` structure for each supported platform.
private_use	This field may be used to carry private data to a receiver or user agent which interprets this signaling stream. Its use is not defined by this specification.

6.2.3 ETV-BIF Platform Descriptor

The `etv_bif_platform_descriptor` is defined for use in the elementary stream information loop of the PMT. This descriptor indicates to a receiver that the associated elementary stream carries ETV-BIF application resources for one or more hardware/software platforms (as identified by [ETV-BIF]) each of which SHALL be listed in this descriptor.

Resources that target baseline receivers SHALL be transmitted on a single elementary stream. Resources that target other receivers MAY be transmitted across multiple elementary streams. In this case, an `etv_bif_platform_descriptor`, which identifies the targeted platform, SHALL be included in the PMT entry that corresponds to each elementary stream carrying those resources.

An elementary stream MAY contain resources that target different collections of platforms. An elementary stream MAY also contain resources for multiple applications; (for example, resources for the primary broadcast program and resources for an enhanced advertisement). The PMT elementary stream information loop MAY contain multiple instances of an `etv_bif_platform_descriptor`.

The `etv_bif_platform_descriptor` is defined in Table 3.

Table 3 - ETV-BIF Platform Descriptor Syntax

Syntax	Bits	Mnemonic
etv bif platform descriptor() {		
descriptor tag	8	uimsbf
descriptor length	8	uimsbf
for (i=0; i < n; i++) {		
etv bif platform id()		
}		
}		

descriptor_tag   This 8-bit integer with value `0xA1` identifies this descriptor.

descriptor_length   This 8-bit integer indicates the number of bytes following this field.

etv_bif_platform_id   This field contains zero or more `etv_bif_platform_id` structures, as defined in Table 4.

                                      If descriptor_length is zero (i.e., there are no `etv_bif_platform_id` structures listed by this descriptor), the associated elementary stream SHALL carry resources that target ALL platforms.

Table 4 - ETV-BIF Platform ID Syntax

Syntax	Bits	Mnemonic
etv bif platform id() {		
pdtHWManufacturer	24	uimsbf
pdtHWModel	16	uimsbf
pdtHWVersionMajor	8	uimsbf
pdtHWVersionMinor	8	uimsbf
pdtSWManufacturer	24	uimsbf
pdtSWModel	16	uimsbf
pdtSWVersionMajor	8	uimsbf
pdtSWVersionMinor	8	uimsbf
pdtProfile	8	uimsbf
}		

All fields in this table are defined in [ETV-BIF], section 9.5.

6.3 Application Signaling for Analog Services

Application Signaling for analog services is out-of-scope for this specification.

7 ENHANCED TELEVISION SYNCHRONIZATION SIGNALING

Synchronization of an application to a video program requires the establishment of a reference media timeline. A media timeline allows a receiver to deliver stream events to an application at very specific points within the timeline.

OCAP devices are capable of using NPT descriptors and stream events embedded within a DSM-CC Object Carousel for this purpose as defined in sections 8.1 and 8.3 of DSM-CC [IEC 13818-6]. NPT descriptors establish the reference timeline. Stream events carry synchronous application data. Limited-capability devices, however, are not capable of properly interpreting and processing the DSM-CC Object Carousel and the descriptors carried therein.

This section defines a set of descriptors which all devices are capable of receiving, designed for maintaining a media timeline and delivering synchronous data to an application.

Because of the limitations of the very low-end legacy receivers, there are requirements to combine the media timeline messages and stream events into a single, unified elementary stream. This unified stream SHALL also carry additional descriptors as identified in this section.

This unified stream SHALL be known as an ETV integrated signaling stream (EISS). It is possible that an operator MAY deliver an EISS as well as a DSM-CC object carousel with embedded NPT & stream event descriptors. In this case, the receiver MAY use the media timeline that is best suited for its capabilities.

Only one EISS stream SHALL be conveyed for an MPEG-2 program. The EISS stream may carry EISS tables for multiple applications, each identified by an application identifier and application instance identifier. An individual EISS table for an application SHALL NOT be interleaved with EISS table sections of another application. Therefore, this specification recommends that individual EISS tables be kept small to avoid any latency in delivering the EISS tables for other applications.

7.1 EISS Table

The descriptors defined in this section are carried in an EISS Table. This table is contained in one or more MPEG-2 sections with syntax as specified in Table 5.

Table 5 - EISS Section Syntax

Syntax	Bits	Mnemonic
eiss section() {		
table id	8	uimsbf
section syntax indicator	1	bslbf
reserved1	3	bslbf
section length	12	uimsbf
reserved2	8	uimsbf
section number	8	uimsbf
last section number	8	uimsbf
protocol version major	8	uimsbf
protocol version minor	8	uimsbf
application type	16	uimsbf
application identifier()	48	
application instance identifier length	8	uimsbf
for (i=0; i<application instance identifier length; i++) {		
application_instance_identifier_data[]	8	uimsbf
}		

Syntax	Bits	Mnemonic
```platform id length```	8	uimsbf

```for (i=0; i<m; i++) {```
```   etv bif platform id()```
```}```
```for (i=0; i<N; i++) {```
```   eiss descriptor()```
```}```		

```CRC 32```	32	rpchof
```}```		

table_id	This 8-bit integer with value `0xE2` identifies this table.
section_syntax_indicator	The `section_syntax_indicator` is a 1-bit field that SHALL be set to `0`.
reserved1	This 3-bit field SHALL be set to `000`.
section_length	This is a 12-bit field that specifies the number of bytes of the section starting immediately following the `section_length` field, up to and including the `CRC_32` field. The value in this field SHALL not exceed 1021.
reserved2	This 8-bit field SHALL be set to `0x00`.
section_number	This 8-bit field gives the number of the section. The `section_number` of the first section in the table SHALL be `0x00`. The `section_number` SHALL be incremented by 1 with each additional section with the same `table_id`.
last_section_number	This 8-bit field specifies the number of the last section (that is, the section with the highest `section_number`) of the table of which this section is part.
protocol_version_major	This 8-bit field specifies a major version number for the ETV messaging protocol being conveyed in the ETV streams for the currently signaled application. The required value is expected to be incremented each time a backward-incompatible revision is published to the implied user agent semantics or to a fundamental syntactic structure of the ETV-AM protocol. The major version number corresponding to this specification is 6.
	If the specified major version for the ETV messaging protocol is not supported and the application has not yet been loaded, then the user agent SHALL ignore the application being signaled; otherwise, if the application has been loaded, then the user agent SHALL cause an `UnsupportedVersionError` condition to be signaled, and subsequently, a terminating transition SHALL be triggered.
protocol_version_minor	This 8-bit field specifies a minor version number for the ETV messaging protocol being conveyed in the ETV streams for the currently signaled application. The required value is expected to be incremented each time a backward-compatible revision is published to the implied user agent semantics or to a fundamental syntactic structure of the ETV-AM protocol. The minor version number corresponding to this specification is 0.
	If the specified minor version for the ETV messaging protocol is not supported, then the user agent SHALL cause an `UnsupportedVersionError` condition to be signaled once a loading transition has been effected.
application_type	This 16-bit integer identifies the type of application being signaled. The following application types are defined:

Table 6 - Application Types

application_type	Description
0x0000 – 0x0007	Reserved by CableLabs

application_type	Description
0x0008	ETV-Binary Interchange Format (ETV-BIF) application
0x0009	Switch Engine application
0x000A	Decision Engine application
0x000B – 0xBFFF	Reserved by CableLabs
0xC000 – 0xFFFD	Private use
0xFFFE – 0xFFFF	Reserved by CableLabs

application_identifier   This 48-bit integer identifies the application according to DVB-MHP section 10.5 [MHP].

application_instance_identifier_length   An unsigned integer that denotes the number of bytes in the application instance identifier data string.

application_instance_identifier_data[]   An array of bytes that comprise the UTF-8 form of the encoded string as defined by [UTF-8] which identifies the application instance. This array of bytes SHALL NOT contain a NUL termination byte (0x00).

Note: This format supports a Version 1 (MAC-based) UUID as defined in [RFC 4122] and encoded as a Base64URL string as defined in [RFC 4648]. The 16-byte binary UUID will encode to a 22-byte string by using a modified Base64 for URL variant, where no padding '=' will be used, and the '+' and '/' characters of standard Base64 are respectively replaced by '-' and '_'.

platform_id_length   This 8-bit integer specifies the number of bytes of the `etv_bif_platform_id` fields immediately following this `platform_id_length` field. This value MAY be zero.

etv_bif_platform_id   This field contains zero or more `etv_bif_platform_id` structures, as defined in Table 4. If `platform_id_length` is zero, this eiss_section applies to all platforms. If `platform_id_length` is non-zero, this field identifies the set of platforms for which this eiss_section applies. All of the eiss_sections for a single eiss table SHALL specify the same set of platforms.

eiss_descriptor   Zero or more descriptors as specified in Section 7.2.

CRC_32   This 32-bit field SHALL be set as defined in [IEC 13818-1], Annex B.

7.2   EISS Descriptors

The contents of this section are applicable when the value of the application_type field of the eiss_section equals 0x0008 (ETV_BIF). This section defines the following EISS descriptors:

- ETV Application Information Descriptor
- ETV Media Time Descriptor
- ETV Stream Event Descriptor
- ETV Application Metadata Descriptor

7.2.1   ETV Application Information Descriptor

Because limited-capability devices cannot process normal AITs, the relevant fields from the AIT SHALL be embedded in the EISS as an application information descriptor, described in Table 7.

Table 7 - ETV Application Information Descriptor Syntax

Syntax	Bits	Mnemonic
etv application information descriptor() {		
descriptor tag	8	uimsbf
descriptor length	8	uimsbf
application control code	8	uimsbf
application version()		
max_protocol_version_major	8	uimsbf
max_protocol_version_minor	8	uimsbf
application_flags()		
application priority	8	uimsbf
initial resource locator()		
for (i=0; i<n; i++) {		
private data	8	bslbf
}		
}		

descriptor_tag   This 8-bit integer with value `0xE0` identifies this descriptor.

descriptor_length   This 8-bit integer indicates the number of bytes following this field.

application_control_code   This 8-bit integer controls the state of the application. The semantics of this field are application type-dependent. This field is interpreted according to Table 8.

Table 8 - ETV-BIF Application Control Code Values

Code	Identifier	Semantics
0x00		reserved_for_future_use
0x01	AUTOSTART	The primary application resource SHALL be loaded and transport layer signaling SHALL indicate that the application MAY be resumed, in accordance with the [ETV-BIF] Lifecycle section requirements for *eligible to run*.
0x02	PRESENT	The primary application resource MAY be loaded, but transport layer signaling SHALL NOT indicate that the application MAY be resumed, in accordance with the [ETV-BIF] Lifecycle section requirements for *eligible to run*.
0x03	DESTROY	Transport layer signaling SHALL indicate that the application SHALL be terminated, in accordance with the [ETV-BIF] Terminating section requirements for the Terminating transition.
0x04-0x06		reserved_for_future_use
0x07	SUSPEND	Transport layer signaling SHALL indicate that the application SHALL be suspended, in accordance with the [ETV-BIF] Suspending section requirements for the Suspending transition.
0x08-0xff		reserved_for_future_use

A decoder SHALL implement the following behaviors when processing an `application_control_code`:

- A decoder SHALL register a loss of signaling event after 4 seconds of not receiving an AUTOSTART or PRESENT signal. A loss of signaling event SHALL be interpreted as semantically equivalent to a SUSPEND signal.

- After a maximum of 10 minutes of sustained loss of signaling for an application, a decoder SHALL terminate the application if not already terminated.

- A decoder SHALL terminate the application immediately on receiving a DESTROY signal.

- An application which is signaled with the SUSPEND control code SHALL NOT be eligible to run.

- An application which is signaled with the AUTOSTART control code SHALL be considered eligible to run from the transport layer signaling perspective, in accordance with the [ETV-BIF] Lifecycle section.

- If an application is signaled with the AUTOSTART or PRESENT control code but there is no page resource available, then a decoder SHALL ignore the descriptor. In this case, a decoder SHALL attempt to process subsequent descriptors.

application_version	This 16-bit field carries the application version. It is structured according to Table 9.

Table 9 - ETV-BIF Application Version

Syntax	Bits	Mnemonic
application version () {		
version major	8	uimsbf
version minor	8	uimsbf
}		

version_major	This 8 integer carries the major version of the application.
version_minor	This 8 integer carries the minor version of the application.
max_protocol_version_major	This 8-bit field, if non-zero, specifies the maximum major protocol version supported by user agents that should decode this application. An ETV user agent that supports EISS protocol versions greater than `max_protocol_version_major` SHALL ignore this descriptor. This enables the user agent to ignore prior app version signaling in cases where a later version of the app is being signaled in this or subsequent EISS tables.
max_protocol_version_minor	This 8-bit field, if non-zero, specifies the maximum minor protocol version supported by user agents that should decode this application. If the maximum major protocol version supported by the user agent is equal to `max_protocol_version_major`, then an ETV user agent that supports EISS protocol versions greater than `max_protocol_version_minor` SHALL ignore this descriptor. This enables the user agent to ignore prior app version signaling in cases where a later version of the app is being signaled in this or subsequent EISS tables.
application_flags	This 32-bit field carries application flags. It is structured according to Table 10.

Table 10 - ETV-BIF Application Flags

Syntax	Bits	Mnemonic
application flags() {		
test flag	8	bslbf
resource update flags	4	bslbf
reserved	20	bslbf
}		

test_flag	This 8-bit field conveys a set of test flags used to target this application to a specific population of set-top boxes. The usage of this field is governed by the cable operator and not further defined by this specification.
resource_update_flags	This 4-bit field is a sequence number that is incremented whenever either (i) the *downloadID* field of the associated application carousel *DownloadInfoIndication* messages is changed, or (ii) the CRC32 field changes

	in the case that an associated data carousel is signaled using Section 8.2 *Alternate Constrained Data Carousels* format. When incrementing the value starts at 0x1 and from 0xf the value wraps to 0x1. A value of 0x0 indicates this field is not used and SHALL NOT increment.
reserved	This 20-bit field is reserved by this specification and SHALL be set to 0x00000.
application_priority	This field identifies a relative priority between the applications signaled in this service.

- Where there is more than one application with the same Application identification, this priority SHALL be used to determine which application is started.

- Where there are insufficient resources to continue running a set of applications, this priority SHALL be used to determine which applications to terminate.

- The greater the numerical value, the higher the application priority.

initial_resource_locator	This field identifies the locator for the initial page resource to be loaded by the receiver to execute the application identified by `application_identifier`. The initial page resource shall be conveyed in an elementary stream that is signaled by the same PMT that signaled the stream containing this EISS table. The format of the locator SHALL comply with the `ebiLocator` structure as defined in [ETV-BIF] Section 11.13 Locator.

For the purposes of this specification, the `lsType` locator type field of the `ebiLocator` structure SHALL be restricted to the following locator type:

- type 4 – URI Locator, wherein the URI SHALL match the URI that is conveyed in the DII message of the initial page resource. This form of locator SHALL be used to reference an initial page resource that conveys a Local Identifier (lid) URI scheme in its associated DII message.

If `application_control_code` equals 0x03 (DESTROY), the user agent SHALL ignore the value conveyed by initial_resource_locator, and in this case the value of the `lsLength` field of `ebiLocator` MAY be zero (0).

private_data	This field is defined as private use and is dependent upon the Application Type being signaled. This field SHALL be used to carry an application argument string as identified by [ETV-BIF] Application Arguments Section.

7.2.2   ETV Media Time Descriptor

ETV media time descriptors enable a receiver to maintain a program-specific timeline that can be referenced by a stream event for synchronization of an application to a broadcast program. This descriptor contains a value that allows the receiver to establish a unique time for each point within the program, even when that program is interrupted for advertisements or is joined in progress.

The ETV media time descriptor is defined in Table 11.

Table 11 - ETV Media Time Descriptor Syntax

Syntax	Bits	Mnemonic
etv media time descriptor() {		
descriptor tag	8	uimsbf
descriptor length	8	uimsbf
time value	32	uimsbf
}		

descriptor_tag	This 8-bit integer with value `0xE1` identifies this descriptor.
descriptor_length	This 8-bit integer indicates the number of bytes following the `descriptor_length` field.
time_value	This field contains the time in milliseconds since the beginning of the current program.

It is not necessary for a program to actually start at time 0, as long as the Stream Event descriptors tied to these time codes accommodate for the actual time values delivered in this stream. Consecutive `time_values` SHALL NOT have negative time discontinuities after discounting the interval of time passed between two consecutive `etv_media_time_descriptors`. Consecutive `time_values` MAY have positive time discontinuities between two consecutive `etv_media_time_descriptors`.

The user agent SHALL provide forwards extrapolation of `time_value` for the application between receiving two consecutive `etv_media_time_descriptors`.

7.2.3   ETV Stream Event Descriptor

ETV stream event descriptors carry application data to be delivered to an application synchronously with the broadcast event. The ETV stream event descriptor is defined in Table 12.

Table 12 - ETV Stream Event Descriptor Syntax

Syntax	Bits	Mnemonic
`etv stream event descriptor() {`		
`    descriptor tag`	8	uimsbf
`    event counter`	4	bslbf
`    descriptor length`	12	uimsbf
`    time value`	32	uimsbf
`    header type`	3	uimsbf
`    payload type`	5	uimsbf
`    for (i=0; i<N; i++) {`		
`        payload byte`	8	bslbf
`    }`		
`}`		

descriptor_tag	This 8-bit integer with value `0xE2` identifies this descriptor.
event_counter	This 4-bit field provides an event counter and SHALL increment by 0x1 for each non-duplicated stream event with the same application identifier, application instance identifier and platform id. When event_counter reaches 0xf it is incremented to 0x0. Duplicate stream events MAY be signaled by conveying the same value of the event counter in consecutive stream events. The bits of `etv_stream_event_descriptor` for duplicate stream events SHALL be identical. One and only one of the duplicated stream events SHALL be delivered to the application. Duplicate stream events (i) SHALL be consecutive to their first occurrence (i.e. SHALL NOT be interleaved with other stream events); and (ii) SHALL be delivered in separate EISS tables. The numeric sequencing of the event_counter field is scoped to application identifier, application instance identifier and platform id fields in the EISS.
descriptor_length	This 12-bit integer indicates the number of bytes following the `descriptor_length` field.
time_value	This field indicates the time at which the event SHOULD be delivered to the application on the receiver. There will be some inevitable delay based on the processing power of the receiver. If `time_value` is equal to 0, the event SHALL be delivered immediately. The semantics related to this field are described in Section 7.2.2.

header_type	An enumeration value indicating the event header type. The values of this enumeration are defined by [ETV-BIF] Trigger Format section, Trigger Header Types Table.
payload_type	An enumeration value indicating the event payload type. The values of this enumeration are defined by [ETV-BIF] Trigger Format section, Trigger Payload Types Table.
payload_byte	This field contains application-dependent data. This field SHALL be interpreted according to the format described by [ETV-BIF] Serialized Trigger section. In this case, and in order to provide EBIF defined trigger payload, the trigger header type SHALL NOT be the value for none; see [ETV-BIF]. This version of the specification does not support serialized triggers with the header type of None.

7.2.4   ETV Application Metadata Descriptor

ETV application metadata descriptors carry application metadata information synchronously with the broadcast event. These metadata items override the same metadata items embedded in the ETV application. The metadata items could be assembled from more than one MPEG sections.

An ETV application metadata descriptor SHALL be delivered in the same table as an ETV application information descriptor.

The ETV application metadata descriptor is defined in Table 13.

Table 13 - ETV Application Metadata Descriptor Syntax

Syntax	Bits	Mnemonic
```etv application metadata descriptor() {```		
``` descriptor tag,```	8	uimsbf
``` reserved1,```	4	bslbf
``` descriptor length,```	12	uimsbf
``` count```	8	uimsbf
``` for (i=0; i<count; i++) {```		
``` metadata item id```	24	uimsbf
``` metadata item type```	4	bslbf
``` metadata item size in bytes```	12	uimsbf
``` for (j=0; j<metadata item size in bytes; j++) {```		
``` metadata item value byte```	8	bslbf
``` }```		
``` }```		
```}```		

descriptor_tag	This 8-bit integer with value `0xE5` identifies this descriptor.
reserved1	This 4-bit field SHALL be set to `0000`.
descriptor_length	This 12-bit integer indicates the number of bytes following the `descriptor_length` field.
count	This 8-bit integer indicates the number of metadata items contained in this descriptor.
metadata_item_id	This 24-bit integer represents the metadata item id as defined in the Metadata Items Annex in [ETV-BIF]. In this version of the specification, only metadata items within the private use range (0xFF0000 - 0xFFFFFE) SHALL be supported for use with this descriptor.
metadata_item_type	This 4-bit integer indicates the type of the metadata item value, and is interpreted according to Table 14 - Metadata Item Type Values.

Table 14 - Metadata Item Type Values

Value	Description
0x0	unsigned integer
0x1	Boolean (0 = FALSE, non-zero = TRUE)
0x2	String (UTF-8)
0xF	reserved_for_future_use

Note: Metadata item types and sizes SHALL conform to the types defined in the [ETV-BIF] Common Data Types Section.

metadata_item_size_in_bytes   This 12-bits integer indicates the number of bytes used by this metadata item value.

metadata_item_value_byte   This 8-bit field makes up the value of the metadata item.

7.3 Synchronization in Analog Services

Synchronization and signaling for analog services is outside the scope of this document.

8   CARRIAGE OF ETV APPLICATION RESOURCE DATA

This section describes the carriage of ETV application resource data within an MPEG-2 Transport Stream. A common portable format must be established that may be interpreted by all receivers, and which can be generated by all authoring and packaging tools.

8.1   DSM-CC Data Carousel

If the Elementary Stream that carries ETV Resources is signaled with a `stream_type` of `0x0B` (IEC 13818-6 Type B - DSM-CC Data Carousel [IEC 13818-6]), the application resource data is carried in a DSM-CC Data Carousel as defined in [IEC 13818-6]. No constraints are placed on the Data Carousel by this specification.

The contents of sections 7 and 9 of [IEC 13818-6] are hereby incorporated into this specification; and, for the purposes of the OpenCable Contribution Agreement, SHALL be considered a "Contribution" to this specification, subject to the IPR terms and conditions (including each signatory's opportunity to provide notice) of the OpenCable Contribution Agreement.

The following fields of the DownloadInfoIndication (DII) message are further specified by this specification:

moduleInfoByte   The moduleInfoByte fields of the DII message MAY specify the *abs_path* component of the Local Identifier (lid:) URI Scheme [SMPTE 343M] to be used to locate this module when used by a URI Locator as defined by [ETV-BIF]. If moduleInfoByte fields are specified, then the user agent SHALL provide a mapping to this module for lid: URI Locators that reference this module using *authority* and *abs_path*. The moduleInfoByte fields contain the *abs_path* structure as defined in Table 15.

Table 15 - abs_path

Syntax	Bits	Mnemonic
`abs_path() {`		
`    abs_path_length`	8	uimsbf
`    for (i=0; i<N; i++) {`		
`        abs_path_byte`	8	bslbf
`    }`		
`}`		

abs_path_length   This 8-bit integer indicates the number of bytes of the abs_path structure following the abs_path_length field.

abs_path_byte   This field contains the bytes of the *abs_path* component of the lid: URI scheme.

privateDataByte   The privateDataByte fields of the DII message MAY specify the *authority* component of the Local Identifier (lid:) URI Scheme [SMPTE 343M] to be used to locate the resources identified by this DII message when referenced by an lid: URI Locator as defined by [ETV-BIF]. The privateDataByte fields contain the *authority* structure as defined in Table 16.

<div align="center">***Table 16 - authority***</div>

Syntax	Bits	Mnemonic

authority() {
    authority_length
    for (i=0; i<N; i++) {
        authority_byte
    }
}
	8
	
	8
		uimsbf
		
		bslbf

authority_length   This 8-bit integer indicates the number of bytes of the *authority* structure following the authority_length field.

authority_byte   This field contains the bytes of the *authority* component of the lid: URI scheme. Resources for a single authority SHALL NOT be conveyed on more than one data carousel in a given elementary stream. Resources for different authorities SHALL be conveyed on different data carousels in a given elementary stream. In other words, a DII SHALL map all resources that are being conveyed for a given authority in the elementary stream at any given moment.

8.2 Alternate Constrained Data Carousels

If the Elementary Stream that carries ETV Resources is signaled with a stream_type of 0xC0 (DCII Text Message), ETV Resources SHALL be encapsulated within a constrained data carousel as defined in Table 17. This constrained data carousel carries a DSM-CC_Section as defined in [IEC 13818-6]. The primary constraint imposed by this format is the limitation that each dc2_data_carousel_section is limited in size to less than 1 KB.

<div align="center">***Table 17 - DCII Data Carousel Message Syntax***</div>

Syntax	Bits	Mnemonic

dc2 data carousel section() {
    table id
    section syntax indicator
    reserved1
    section length
    filter info
    reserved2
    for (i=0; i<N; i++) {
        dsmcc section()
    }
    CRC32
}
	8
	1
	3
	12
	16
	8
	
	32
		uimsbf
		bslbf
		bslbf
		uimsbf
		uimsbf
		bslbf
		
		rpchof

table_id   This 8-bit integer with value 0xE3 or 0xE4 identifies this descriptor. If the enclosed dsmcc_section carries a DII message, this field SHALL convey the value 0xE3. If the enclosed dsmcc_section carries a DownloadDataBlock (DDB) message, this field SHALL convey the value 0xE4.

section_syntax_indicator   The section_syntax_indicator is a 1-bit field which SHALL be set to 0.

reserved1   This 3-bit field is reserved by this specification and SHALL be set to '100'.

section_length   This 12-bit field specifies the number of bytes in the section starting immediately following the section_length field. The value in this field SHALL not exceed 1021, indicating that the encapsulated dsmcc_section SHALL have a maximum length of 1014 bytes.

filter_info	This 16-bit field is intended to accommodate hardware filtering of messages. If the enclosed `dsmcc_section` carries a DII message, this field SHALL convey the value 0xFBFB. If the enclosed `dsmcc_section` carries a DDB message, this field SHALL convey a copy of the moduleId field of the conveyed DSMCC DDB message. The set of values for the moduleId SHALL be limited to within the range 0x0001 and 0xFBEF. This enables a receiver to set hardware filters on all DSMCC DII control messages and specific download data modules in `dsmcc_section` sections.
reserved2	This 8 bit field SHALL be set to `0x00`.
dsmcc_section	This field carries a DSM-CC_Section as defined in [IEC 13818-6], table 9-2. When a DSM-CC section is encapsulated by a `dc2_data_carousel_section` the maximum length of that DSM-CC section is 1014 bytes.
CRC-32	This field SHALL be set as defined in [IEC 13818-1], Annex B.

8.3 Timing of EISS signal and DII message

If a new application is signaled by an EISS Application Information Descriptor with either a PRESENT or AUTOSTART control code, the first DII message corresponding to the signaled initial page resource and other resources SHALL be inserted before EISS control codes.

9 APPLICATION SIGNALING AND SYNCHRONIZATION FOR LIMITED CAPABILITY DEVICES

9.1 Introduction

ETV applications will be deployed on a wide range of receivers, including devices such as the Motorola DCT-2000. Some devices do not have the resources necessary to support advanced signaling techniques. Programmers and network operators may choose to simultaneously broadcast both messaging types in order to target the widest range of devices; however, all devices must be capable of reading the base signaling stream.

The critical resource constraint of limited-capability devices is the number of PID filters, which has required the combination of several elements into the Integrated Signaling Stream as described in Section 7 of this specification.

An overall review of the descriptor elements introduced in this specification is illustrated in Figure 3:
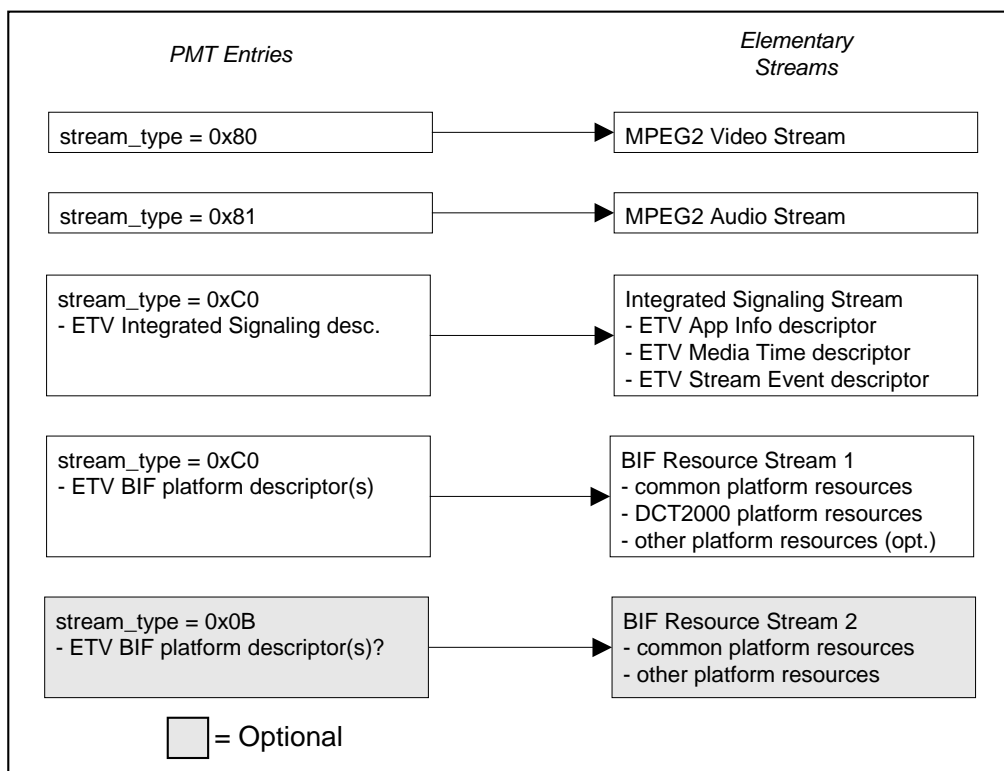

Figure 3 - PMT Signaling Motorola DCT-2000 Specific Behaviors

The User Agent running on a DCT-2000 SHALL read the first four PMT entries in the above table. The ETV Integrated Signaling Descriptor in the third PMT entry indicates that the associated elementary stream carries a signaling stream that includes an AIT-like Application Information descriptor, the Media Time descriptors, and any Stream Event descriptors.

The ETV-BIF Platform descriptor in the fourth entry indicates that this elementary stream carries ETV-BIF resources for the specified platforms.

If one or more elementary streams contains ETV-BIF resources targeted to a DCT-2000 receiver, those streams will have a `stream_type` of `0xC0` (DCII Text Message). Those resources that are not required to be processed by a DCT-2000 MAY be signaled with a `stream_type` of `0xC0` or `0x0B` (DSM-CC Data Carousel).

9.2   All other Set-Top Specific Behaviors

User Agents running on all other receivers SHALL read the first three PMT entries, just as the DCT-2000. The Application Information descriptor in the EISS will contain an `application_identifier`, which, in this case, MAY reflect the presence of resources in either the fourth or fifth stream (or both) above. The User Agent SHALL read each PMT entry to find the resources most suitable for the given hardware and/or software platform and load from those streams as required.

9.3   OpenCable Host Specific Behaviors

No specific signaling or behaviors have been identified for OpenCable hosts.

Appendix I   Revision History

The following ECNs were incorporated into OC-SP-ETV-AM-I02-050726:

Number	Description	Date
ETV-AM-N-05.0785-2	Support for single ETV data PID	7/13/05
ETV-AM-N-05.0786-2	Correction to filter_info values in App Resource stream	7/13/05
ETV-AM-N-05.0804-1	Improved specification for time_value field of etv_media_time_descriptor	7/20/05

The following ECNs were incorporated into OC-SP-ETV-AM1.0-I03-060714:

Number	Description	Date
ETV-AM-N-05.0828-3	Collected changes for ETV AM	1/16/06
ETV-AM1.0-N-06.0861-3	Improved support for enhanced Ads	4/3/06

The following ECN was incorporated into OC-SP-ETV-AM1.0-I04-070921:

Number	Description	Date
ETV-AM1.0-N-06.0963-2	Collected and editorial changes	4/10/07

The following ECNs were incorporated into OC-SP-ETV-AM1.0-I05-091125:

Number	Description	Date
ETV-AM1.0-N-08.1196-1	Clarify Hardware/Software Version field descriptions	4/1/08
ETV-AM1.0-N-09.1401-2	Application Priority Clarification	11/25/09
ETV-AM1.0-N-09.1429-2	Initial Page Resource location clarification	11/25/09

The following ECNs were incorporated into OC-SP-ETV-AM1.0-I06-110128:

Number	Description	Date
ETV-AM1.0-N-09.1474-3	Change to eiss_section() table_id	1/28/11
ETV-AM1.0-N-09.1476-10	Addition of SUSPEND Control Code, Resource Update and Stream Event Sequence Numbers	1/28/11
ETV-AM1.0-N-10.1494-1	Explicit definition of the timing requirement for DII and corresponding EISS signaling	1/28/11
ETV-AM1.0-N-10.1532-1	Define a new EISS descriptor for signaling and interchanging metadata items	1/28/11

The following ECNs were incorporated into OC-SP-ETV-AM1.0.1-120614:

Number	Description	Date
ETV-AM1.0-N-12.1778-1	Update References	6/14/12
ETV-AM1.0-N-12.1786-1	Make PMT etv_integrated_signaling_descriptor stream_type requirement a SHALL	6/14/12