OpenCable™ Specifications

Digital Receiver Interface Protocol Specification

OC-SP-DRI-I01-060109

ISSUED

Document Status Sheet

Document Control Number:	OC-SP-DRI-I01-060109			
Document Title:	Digital Receiver Interface Protocol Specification			
Revision History:	D01 – Released June 2, 2005 D02 – Released November 18, 2005 I01 – Released January 9, 2006			
Date:	January 9, 2006			
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.

Contents

Figures

Tables

This page left blank intentionally.

1 SCOPE

1.1 Introduction and Overview

This specification defines the characteristics and normative specifications for the interface between DRI Transceiver (DRIT) and the Home Media Server (HMS). The DRIT receives content from a cable system and delivers it over the DRI to a local HMS.

Cable operators may lease DRITs as well as support DRITs purchased at retail. The combination of a DRIT and a Home Media Server permits the secure storage and distribution of cable programming across multiple rendering devices within the home.

This specification leverages the Universal Plug and Play standard (UPnP v1.0) for device enumeration, control and streaming. This specification supports a variety of Digital Rights Management systems ("DRM systems") to enforce the handling of cable programming according to the rules defined by the cable operator. The interface defines a unique content protection layer ("DRI Security") that needs to be supported by all DRM systems.

The interface defines a mechanism for remotely selecting any analog, digital standard or high definition cable channel, and organizing the adequate streaming method across the interface. The transmission of an analog NTSC channel requires digital conversion and compression for bandwidth optimization.

The interface supports an extension of all Card interface messaging required for remote navigation and user interface, such as out-of-band service information, emergency alert messages, Man Machine Interface and diagnostics.

A DRIT may implement the DRI on any physical bus that supports IP streaming of 39Mbps HDTV content and for which the certification submitter provides a bridge to a CableLabs specified standard interface.

1.2 Purpose of this document

This document intends to define a new class of digital output that supports full remote control of the DRIT receiving resources and Card interface. It departs from the existing point to point protection mechanisms by defining a generic plug-in for any approved DRM system.

1.3 Historical Perspective

This specification has its origins in OC-SP-CC-IF, the OpenCable™ CableCARD™ Interface Specification, which was initially issued in October 1999. This specification defines the interface protocols to separate the cable network private and security features from the DRIT receiving device. At that time, the requirement was that the DRIT was either a set-top box or a television. By the end of 2003, the emergence of personal video recorder and new home networking technologies created a need for a new OpenCable interface able to project some of the Card signaling messages over IP to enable secure use of the audio video cable programming anywhere in the home network.

This specification employs the Universal Plug and Play specifications published by the UPnP Forum for device enumeration, control and eventing.

1.4 Organization of document

This specification is organized five main chapters:

Chapter 5 “System Overview” describes how the DRI connects a DRIT device to a Home Media Server and how the function of navigation, decoding and rendering can be translated to the latter.

Chapter 6 “Interface Services” defines the interface components and the underlying internet protocols

Chapter 7 “DRI Initialization” defines the steps for device enumeration, reservation and initialization

Chapter 8 “DRI Security” defines the DRM plug-in requirements.

Chapter 9 ”Interface Extensibility” defines the requirements for adding new services to the interface

1.5 Requirements (Conformance Notation)

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

“SHALL/MUST”	This word or the adjective “REQUIRED” means that the item is an absolute requirement of this specification.
“SHALL NOT / MUST NOT”	This phrase means that the item is an absolute prohibition of this specification.
“SHOULD”	This word or the adjective “RECOMMENDED” 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 or the adjective “OPTIONAL” 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.

[47CFR15] FCC 47 CFR Chapter 1 (10-1-98 Edition), Part 15 – Radio Frequency Devices, Class B

[47CFR76] FCC 47 CFR Chapter 1 (10-1-98 Edition), Part 76 – Cable Television Service

[A/53] ATSC A/53C: ATSC Digital Television Standard

[AES] FIPS-197: Advanced Encryption Standard (AES), 2001 November 26**.**

[CCIF] OC-SP-CCIF2.0-I03-051117: CableCARD Interface 2.0 Specification

[CCCP] OC-SP-CCCP2.0-IF-I02-050708: CableCARD Copy Protection 2.0 Specification

[CEA542] CEA-542-B: Cable Television Channel Identification Plan

[HTML] Hypertext Markup Language (http://www.w3.org/TR/REC-html32.html)

[HTTP] RFC 2616 “Hypertext Transfer Protocol” (http://www.ietf.org/rfc/rfc2616.txt)

[IP] RFC 791 “Internet Protocol” (http://www.ietf.org/rfc/rfc791.txt)

[MPEG-S] ISO/IEC 13818-1, 2000, Information technology—Generic coding of moving pictures and associated audio: Systems

[OCUR] OpenCable Unidirectional Receiver [OC-SP-OCUR-D02-051118]

[RTSP] RFC 2326 “Real Time Streaming Protocol” (http://www.ietf.org/rfc/rfc2326.txt)

[RTP] RFC 2250 "RTP Payload Format for MPEG1/MPEG2 Video" (http://www.ietf.org/rfc/rfc2250.txt)

[SCTE18] SCTE 18 2001 (formerly DVS 208): Emergency Alert Message for Cable

[SCTE43] ANSI/SCTE 43 2005 (formerly DVS/258), Digital Video Systems Characteristics Standard for Cable Television

[SCTE54] SCTE 54 2004 (formerly DVS 241): Digital Video Service Multiplex and Transport System Standard for Cable Television

[SCTE65] ANSI/SCTE 65 2002 (formerly DVS 234): Service Information Delivered Out-of-Band for Digital Cable Television

[SOAP] Simple Object Access Protocol (http://www.w3.org/TR/2003/REC-soap12-part1-0030624/)

[SSDP] Simple Service Discovery Protocol (http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt)

[TCP] RFC 793 “Transmission Control Protocol” (http://www.ietf.org/rfc/rfc0793.txt)

[UDP] RFC 768 “User datagram protocol” (http://www.ietf.org/rfc/rfc768.txt)

[UPNPAV] AVTransport:1 Service Template Version 1.01 (http://www.upnp.org/standardizeddcps/documents/AVTransport1.0.pdf)

[UPNPCM] ConnectionManager:1 Service Template Version 1.01 (http://www.upnp.org/standardizeddcps/documents/ConnectionManager1.0.pdf)

[UPNPDA] UPnP™ Device Architecture 1.0, Version 1.0.1, 2 December 2003 (http://www.upnp.org/download/UPnPDA10_20000613.htm)

[XML] Extensible Markup Language. W3C recommendation (http://www.w3.org/XML/)

2.2 Informative References

[A/65] ATSC A/65B: Program and System Information Protocol for Terrestrial Broadcast and Cable

[MPEG-V] ISO/IEC 13818-2, 2000, Information technology—Generic coding of moving pictures and associated audio: Video

[MPEG-A] ISO/IEC 11172-3:1993, Information technology—Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s—Part 3: Audio

[SCTE20] ANSI/SCTE 20 2004 (formerly DVS 157): Standard Methods for Carriage of Closed Captions and Non-Real Time Sampled Video. Note: Non-Real Time Sampled Video support is "optional" for Host Devices.

[SCTE21] ANSI/SCTE 21 2001 (formerly DVS 053): Standard for Carriage of NTSC VBI Data in Cable Digital Transport Streams

[CEA608] CEA-608-C: Recommended Practice for Line 21 Data Service

[CEA708] CEA-708-B: Digital Television (DTV) Closed Captioning

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; http://www.cablelabs.com .

SCTE/DVS Standards:
SCTE - Society of Cable Telecommunications Engineers Inc., 140 Philips Road, Exton, PA 19341
Phone: 610-363-6888 / 800-542-5040; Fax: 610-363-5898; http://www.scte.org .

ISO/IEC Standards:
ISO Central Secretariat: International Organization for Standardization (ISO), 1, rue de Varembé, Case postale 56, CH-1211 Geneva 20, Switzerland; Internet: http://www.iso.ch .

UPnP Specifications:
UPnP Forum, Email: upnpfeedback@forum.upnp.org; Internet:
http://www.upnp.org/standardizeddcps/default.asp

3 TERMS AND DEFINITIONS

This specification uses the following terms:

DRIT	DRI Transceiver - A cable receiving device licensed by CableLabs to host a Card and transmit content over the DRI. The OpenCable Unidirectional Receiver (OCUR) is one example of DRIT.
FDC	Forward Data Channel: The SCTE 55 compliant data transmission from headend to the DRIT. The FDC carries configuration and control messages from the Network Controller to DRITs on a carrier separate form the main video services. This typically includes the following types of messages: • Conditional Access (CA) messages including entitlements • System Information (SI) messages • Electronic Program Guide (EPG) messages • Emergency Alert System (EAS) messages • Other generic messages
Card	The PCMCIA form factor CableCARD™ security card provided by the cable operator to enable reception of CA protected digital services. Also referred to as "Point of Deployment" (POD) module. The interface between the Card and the receiver is specified by the OpenCable platform. Card functionality includes copy protection and signal demodulation.
Home Media Server	A Home Media Server is the sink device for content sourced by the DRIT on the DRI. The HMS provides control, and user interface to the DRIT when the DRI Protocol is active. The HMS may securely store, distribute or display cable audio-visual programming services.
Digital Receiver Interface	An implementation of the DRI on a physical bus.
Digital Rights Management	A technology that manages access to digital content or services to enable access and use as designated by the provider and to prevent unauthorized access and use. DRM may prevent the sharing or copying of digital content or tie the use or viewing of content to specific individuals, operating systems, or hardware.
Tuner	The main video service receiver. The Tuner selects, demodulates and processes a desired main service signal from the broadband cable network and outputs either an MPEG-TS or a baseband analog TV signal.

Data Types

b	Binary value suffix symbol. Indicates the preceding value is a binary integer,
Base64	Base64-encoded arbitrary binary data.
Boolean	0 or 1, where 0 is defined as "false" and 1 as "true"
i4	Binary integer, four bytes (32-bits), optional sign, in the range: -2,147,483,648 to 2,147,483,647

string	Sequence of characters, bytes, of unspecified length
ui1	Unsigned binary integer, one byte (8-bits), in the range: 0 to 255
ui2	Unsigned binary integer, two bytes (16-bits), in the range 0 to 65,535
ui4	Unsigned binary integer, four bytes (32-bits), in the range 0 to 4,294,967,295
uimsbf	Unsigned integer, most significant bit first

4 ABBREVIATIONS, ACRONYMS, SYMBOLS

This specification uses the following abbreviations:

ACCI	Aggregated Content Control Information. Includes CCI and all content protection parameters required by the relevant DRIT specification, e.g., [OCUR].
AES	Advanced Encryption Standard, see [AES]
APS	Analog Protection System – CCI bits to control analog anti-copy coding
CA	Conditional Access – the cable operator's means of protecting content and of conveying content use permissions to DRITs.
CCI	Copy Control Information – as defined in [CCCP].
CGMS-A	Copy Generation Management System – Analog
CHI	Card-Host Interface, see [CCIF]
DRI	Digital Receiver Interface
DRIT	DRI Transceiver, see Section 3
DRI-TI	DRI Transport Interface, the conduit for MPEG program content on the DRI.
DRM	Digital Rights Management
DVS	Digital Video Subcommittee of the SCTE
EAS	Emergency Alert System
FAT	Forward Application Transport Channel
FDC	Forward Data Channel, see Section 3
HMS	Home Media Server
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
MMI	Man Machine Interface
MPEG	Moving Picture Experts Group
NTSC	National Television Standards Committee, refers to analog TV signals
PES	Program Elementary Stream, see [MPEG-S]
PID	Packet ID, see [MPEG-S]
RTP	Real-time Transport Protocol
RTSP	Real-Time Streaming Protocol
SAP	Second Audio Program, delivered with BTSC format audio in NTSC signals
SCTE	Society of Cable Telecommunications Engineers
SI	System Information
SOAP	Simple Object Access Protocol
SSDP	Simple Service Discovery Protocol

TID	Table ID
TS	Transport Stream, defined in [MPEG-S]
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UUID	Universal Unique IDentifier.
XML	Extensible Markup Language
|	Binary concatenation operator symbol, the "bar character", means combine the bit strings of two binary parameters, e.g., for A = 111 and B = 000, A|B = 111000
b	Number suffix indicating that the preceding number is an integer expressed in binary format. Space characters are inserted each 4 characters to improve readability, e.g. 1001 1001b is a binary integer with value equal to 153 decimal.
0x	Number prefix indicating that the following number is expressed in hexadecimal format. Space characters are inserted each 4 characters as radix to improve readability, e.g., 0x99 is binary integer with value equal to 153 decimal.

5 SYSTEM OVERVIEW

5.1 Definition

The DRI is a digital interface that allows a DRIT to provide tuning and security services to a Home Media Server.

DRITs may only be used in conjunction with a Home Media Server. In this scenario, some functionality can be eliminated to create much simpler receivers with no user interface. The DRI Protocol is designed in such a way that DRITs can be connected with a single Home Media Server, which will project cable programming in the heart of a true multi-room, multi-user experience.

REQ1522 Although the Home Media Server may manage multiple DRITs, the HMS and DRIT SHALL implement all of the interface protocols defined in this specification as one-to-one interactions independent of any other activity on the IP link between them.

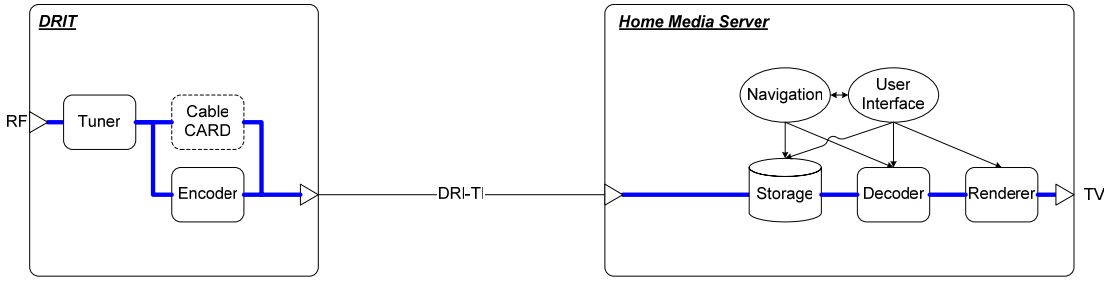

Figure 5.1-1 - DRI Connection

6 INTERFACE SERVICES

6.1 Introduction

The DRI Protocol leverages the internet protocols shown in Figure 6.1-1. Positioned over IP, the interface is fully independent of the physical bus it is ported to.

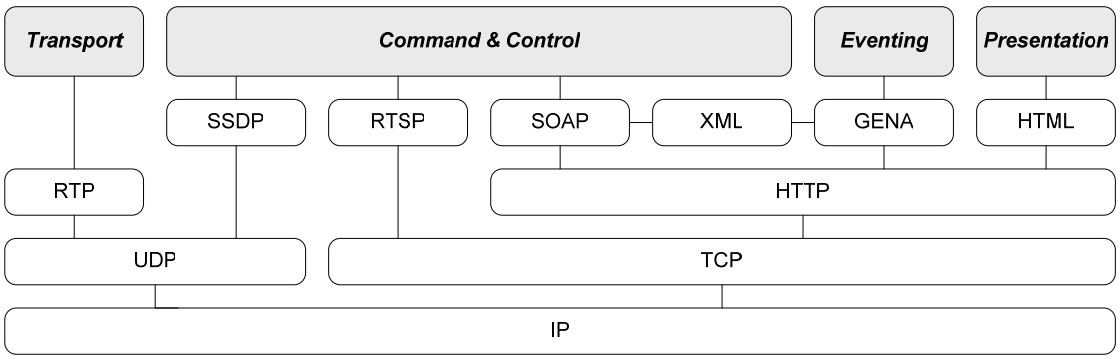

Figure 6.1-1 - Internet protocol

REQ116 As a UPnP™ device, the DRIT SHALL implement the internet protocols required in [UPNPDA].

6.2 Command and Control

The DRI Command and Control layer is architected around the device topology. Functions that are involved in the programming datapath are grouped per service, as shown in Figure 6.2-1.

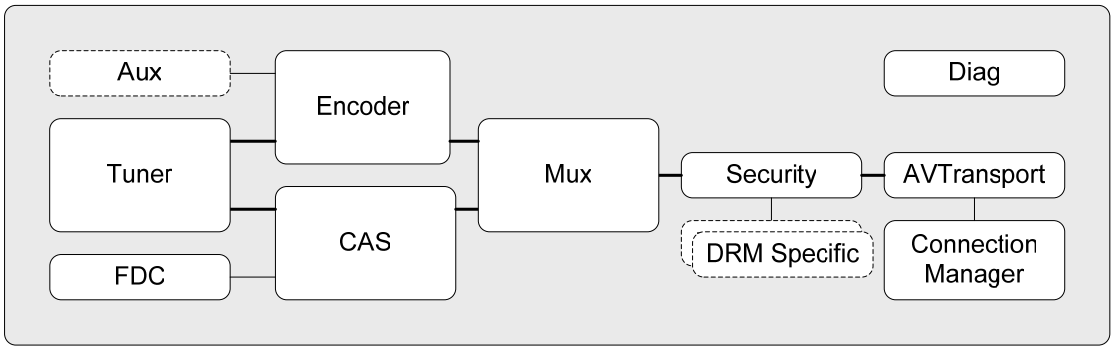

Figure 6.2-1 - DRI Command and Control Services

- **Tuner** – This service is used to control the tuner and the demodulator for both digital and analog channels.

- **FDC** – This service is used to expose the FDC tuner diagnostics and transmit [SCTE18] and [SCTE65] table sections.

- **Aux (optional) –** This service is used to enumerate and configure baseband audio/video inputs, if any.
- **Encoder** – This service is used to enumerate and configure audio/video encoding capabilities.
- **CAS** – This service is used to extend the Card messages for setup, status, user interface and descrambling.
- **Mux** – This service is used to define the TS Packets that need to be carried over the DRI Transport Interface.
- **Security** – This service is used to enumerate and select an approved DRM system
- **DRM Specific (private**) – This service is nested behind the generic DRM service and provides support to a specific approved DRM system for device registration and license generation.
- **Diag** – This service is used to provide diagnostic information about the DRIT.
- **AV Transport** – This service is a generic UPnP service defined in [UPNPAV]
- **Connection Manager** – This service is a generic UPnP service defined in [UPNPCM].

  REQ117 Card device operations, such as initialization, time, OOB tuner control, copy protection and firmware upgrade, SHALL be handled by the DRIT independently of the DRI.

  REQ242 If the DRIT supports multiple DRM systems, there SHALL be one DRM Specific service per system.

The following sections describe the state variables, actions and arguments attached to each service. For a more accurate description, refer to Annex A for the XML description.

6.2.1 Tuner Service

6.2.1.1 Service Type

The following service type identifies a service that is compliant with this template:

urn:schemas-opencable-com:service:Tuner:1

6.2.1.2 State Variables

The Tuner service is architected around the state variables defined in Table 6.2–1.

REQ243 The DRIT SHALL support the state variables defined in Table 6.2–1 for the Tuner Service.

REQ118 The DRIT SHALL support the Tuner service state variables within the range specified in the Allowed Value column of Table 6.2–1.

REQ244 All state variables in the Tuner Service designated as "Evented" SHALL be reported to subscribers in event messages.

Table 6.2–1 - State Variables for Tuner Service

Variable Name	Data Type	Allowed Value	Default Value	Evented
A_ARG_TYPE_Increment	ui4			No
A_ARG_TYPE_SeekModulation	String			No
A_ARG_TYPE_SeekUp	Boolean		True	No
A_ARG_TYPE_StopFrequency	ui4			No
A_ARG_TYPE_TimeToBlock	ui2			No
CarrierLock	Boolean			No
Frequency	ui4			No
Modulation	String	"UNKNOWN", "QAM64", "QAM-64", "QAM256", "QAM-256" "NTSC", "NTSC-M", "8VSB", "8-VSB"		No
ModulationList	String			No
PCRLock	Boolean			Yes
Seeking	Boolean			Yes
SignalLevel	i4			No
SNR	ui4			No

- **A_ARG_TYPE_Increment:** This variable is used as parameter of the SeekSignal {} action. It defines in kHz how the Frequency state variable is incremented to calculate the scanned frequencies.

- **A_ARG_TYPE_SeekModulation:** This variable is used as parameter of the SeekSignal {} action. It defines a subset of modulation types that will be looked for during the action. It is a list of modulation values, separated by comma. "ALL" means all possible modulations.

- **A_ARG_TYPE_SeekUp**: This variable is set by the SeekSignal action to define if the search is performed by increasing the frequency (TRUE) or by decreasing the frequency (FALSE).

- **A_ARG_TYPE_StopFrequency:** This variable is used as parameter of the SeekSignal {} action. It defines the frequency value, once reached by the Frequency state variable that will terminate this action.

- **A_ARG_TYPE_TimeToBlock**: This variable is set by the SeekSignal action to define a time out value in second.

- **CarrierLock**: This variable reflects the current CarrierLock status of the tuner.

- **Frequency:** This variable reflects the current frequency of the tuner and is expressed in kHz.
- **Modulation**: This variable reflects the current modulation detected by the demodulator.
- **ModulationList**: This variable is used to enumerate the modulation types supported by the DRIT. It is a list of modulation value, separated by comma. An empty list assumes that only the mandatory modulation types are supported.
- **PCRLock:** This variable reflects the current PCRLock status of the FAT tuner when tuned to a digital channel and the PCRLock status of the A/V encoded signal when tuned to an analog channel.
- **Seeking:** This variable reflects the seeking state of the FAT tuner. It is set to TRUE after a SeekSignal action, and reset to FALSE when either the tuner has found a valid signal, has reached the outbound frequency or as timed out.
- **SignalLevel:** This variable reflects the current signal level as measured in dBmV by the demodulator.
- **SNR:** This variable reflects the current signal noise ratio level as measured in dB by the demodulator.

REQ119 If the tuner is locked to a carrier with a non-supported modulation, then Modulation state variable SHALL be set to "UNKNOWN".

REQ120 If the tuner has been able to resolve the UNKNOWN modulation setting, then that argument SHALL be set to the detected modulation.

REQ121 The DRIT SHALL support the Tuner Service state variables within the range specified in the Allowed Value column of Table 6.2–1.

REQ122 All state variables in the Tuner Service designated as "Evented" SHALL be reported to subscribers in event messages.

6.2.1.3 Actions

REQ123 The Tuner Service SHALL support the actions as defined in Table 6.2–2

Table 6.2–2 - Tuner Service Actions

Actions	Requirements
REQ123.1 SetTunerParameters {}	Mandatory
REQ123.2 GetTunerParameters{}	Mandatory
REQ123.3 SeekSignal{}	Optional
REQ123.4 SeekCancel{}	Optional

6.2.1.3.1 SetTunerParameters

Arguments for SetTunerParameters are defined in Table 6.2–3.

REQ124 The SetTunerParameters action SHALL return tuning status in less than 1s.

Table 6.2–3 - Arguments for SetTunerParameter

Argument	Direction	Related State Variable
NewFrequency	In	Frequency
NewModulation	In	ModulationList
CurrentFrequency	Out	Frequency
CurrentModulation	Out	Modulation
PCRLockStatus	Out	PCRLock

- **NewFrequency (Input):** This argument set the Frequency state variable to the selected frequency.
- **NewModulation (Input):** This argument sets the Modulation state variable to the first in a list of selected modulation types.

  REQ1523 If the modulation list is empty, the DRIT SHALL set Modulation to "UNKNOWN".

  REQ1524 If the modulation list has more than one entry, the DRIT SHALL try each listed modulation type until it achieves demodulation or completes the list.

  REQ268 If the modulation is unknown, then the value SHALL be set to "UNKNOWN".

- **CurrentFrequency (Output):** This argument provides the value in Frequency state variable when the action response is created.
- **CurrentModulation (Output):** This argument provides the value of the Modulation state variable when the action response is created.
- **PCRLockStatus (Output):** This argument provides the value of the PCRLock state variable when the action response is created..

6.2.1.3.2 GetTunerParameters

Arguments for GetTunerParameters are defined in Table 6.2–4.

REQ126 The GetTunerParameters action SHALL return tuning status in less than 1s.

Table 6.2–4 - Arguments for GetTunerParameter

Argument	Direction	Related State Variable
CurrentCarrierLock	Out	CarrierLock
CurrentFrequency	Out	Frequency
CurrentModulation	Out	Modulation
CurrentPCRLock	Out	PCRLock
CurrentSignaLevel	Out	SignalLevel
CurrentSNR	Out	SNR

- **CurrentCarrierLock (Output):** This argument provides the value of the CarrierLock state variable when the action response is created.

- **CurrentFrequency (Output):** This argument provides the value of the Frequency state variable when the action response is created.
- **CurrentModulation (Output):** This argument provides the value of the Modulation state variable when the action response is created.
- **CurrentPCRLock (Output):** This argument provides the value of the PCRLock state variable when the action response is created.
- **CurrentSignalLevel (Output):** This argument provides the value of the SignalLevel state variable when the action response is created.
- **CurrentSNR (Output):** This argument provides the value of the SNR state variable when the action response is created.

6.2.1.3.3 SeekSignal

Arguments for SeekSignal are defined in Table 6.2–5.

REQ127 Upon receipt of the SeekSignal action, the DRIT SHALL perform a signal search according to the input arguments within the limit of the timeout period.

Table 6.2–5 - Arguments for SeekSignal

Argument	Direction	Related State Variable
StartFrequency	In	Frequency
StopFrequency	In	A_ARG_TYPE_StopFrequency
ModulationList	In	A_ARG_TYPE_SeekModulation
Increment	In	A_ARG_TYPE_Increment
SeekUp	In	A_ARG_TYPE_SeekUp
TimeToBlock	In	A_ARG_TYPE_TimeToBlock

- **StartFrequency (Input):** This argument sets the Frequency state variable.
- **StopFrequency (Input):** This argument sets the A_ARG_TYPE_StopFrequency state variable.
- **ModulationList (Input):** This argument sets the A_ARG_TYPE_SeekModulation state variable
- **SeekUp (Input):** This argument sets the A_ARG_TYPE_SeekUp state variable.
- **Increment (Input):** This argument sets the A_ARG_TYPE_Increment state variable.
- **TimeToBlock (Input):** This argument sets the A_ARGTYPE_TimeToBlock state variable.

6.2.1.3.4 SeekCancel

This function doesn't have any argument.

REQ128 Upon receipt of the SeekCancel action, the DRIT SHALL cancel any SeekSignal action in less than 500ms.

6.2.2 FDC Service

6.2.2.1 Service Type

The following service type identifies a service that is compliant with this template:

urn:schemas-opencable-com:service:FDC:1

6.2.2.2 State Variables

The FDC service is architected around the states variable defined in Table 6.2–6.

REQ245 The DRIT SHALL support the state variable defined in Table 6.2–6 for the FDC Service.

REQ129 The DRIT SHALL support the FDC service state variables within the range specified in the Allowed Value column of Table 6.2–6.

REQ246 All state variables in the FDC Service designated as "Evented" SHALL be reported to subscribers in event messages.

REQ1525 All state variables of data type "string" SHALL represent hex values with numerals 0-9 and upper case characters A – F only (no 0x prefix).

Table 6.2–6 - State Variables for FDC Service

Variable Name	Data Type	Allowed Value	Default Value	Evented
A_ARG_TYPE_PidListChanges	string			No
A_ARG_TYPE_TID	string			No
Bitrate	ui4			No
CarrierLock	Boolean			No
CCFreePidFilter	ui1			No
Frequency	ui4			No
PidList	string			No
SpectrumInversion	Boolean			No
TableSection	Base64			Yes

- **A_ARG_TYPE_PidListChanges**: This variable is used as an argument of the AddPid {} and RemovePid {} actions to modify the PidList state variable. It is defined as a string of TS Packet Identifier (PID) values in hex separated by comma.

- **A_ARG_TYPE_TID**: This variable is a string of MPEG Table ID (TID) values in hex separated by comma, set by the RequestTables action to filter which cached MPEG table sections shall be re-evented through the TableSection state variable. "ALL" means no filtering.

- **Bitrate**: This variable is set by the Card with a OOB_RX_tune_req() APDU and defines the expected bit rate in kbps of the FDC stream.

- **CarrierLock**: This variable reflects the current CarrierLock status of the FDC tuner.

- **CCFreePidFilter**: This variable reflects the number of remaining MPEG flows on the Card. If no Card is inserted, the state variable value is 0.

- **Frequency:** This variable reflects the current frequency of the FDC tuner and is expressed in kHz.

- **PidList**: This variable is a string of TS Packet ID (PID) values in hex separated by comma, which represents the MPEG Flows opened with the Card across the Extended Channel as defined by [CCIF].

  REQ240 The DRIT SHALL open the first MPEG flow with PID = 0x1FFC without any support from the DRI.

- **SpectrumInversion**: This variable is set by the Card with a OOB_RX_tune_req() APDU and defines the expected spectrum inversion status of the FDC stream.

- **TableSection:** This variable is a base64 object defined in Table 6.2–7. It is a complex object that includes the latest MPEG table section received from the Card and its corresponding PID.

Table 6.2–7 - TableSection Type Definition

	No. of bits	Mnemonic
Reserved	3	uimsbf
PID	13	uimsbf
MPEG Table Section{}		

REQ132 The DRIT SHALL update the TableSection state variable every time it receives a new table section from the Card.

REQ133 The DRIT SHALL cache all table sections, as defined in Table 6.2–8, in non-volatile memory so that they can be independently retrieved with a RequestTables{} action.

Table 6.2–8 -Card Tables

Table	PID	TID	Cache
NIT	1FFC	C2	Yes
NTT	1FFC	C3	Yes
SVCT	1FFC	C4	Yes
STT	1FFC	C5	No
MGT	1FFC	C7	No
RRT	1FFC	CA	Yes
LVCT	1FFC	C9	Yes
AEIT-0	Per MGT	D6	No
AETT-0	Per MGT	D7	No
EAS	1FFC	D8	No

6.2.2.3 Actions

REQ134 The FDC Service SHALL support the actions as defined in Table 6.2–9.

Table 6.2–9 - FDC Service Actions

Actions	Requirements
REQ134.1 GetFDCStatus {}	Mandatory
REQ134.2 RequestTables {}	Mandatory
REQ134.3 AddPid {}	Mandatory
REQ134.4 RemovePid {}	Mandatory

6.2.2.3.1 GetFDCStatus

Arguments for GetFDCStatus are defined in Table 6.2–10. The FDC tuner is only controlled by the Card.

REQ135 Upon receipt of the GetFDCStatus action, the DRIT SHALL return tuning status in less than 500ms.

Table 6.2–10 - Arguments for GetFDCStatus

Argument	Direction	Related State Variable
CurrentBitrate	Out	Bitrate
CurrentCarrierLock	Out	CarrierLock
CurrentFrequency	Out	Frequency
CurrentSpectrumInversion	Out	SpectrumInversion
CurrentPidList	Out	PidList

- **CurrentBitrate (Output):** This argument provides the value of the Bitrate state variable when the action response is created.
- **CurrentCarrierLock (Output):** This argument provides the value of the CarrierLock state variable when the action response is created.
- **CurrentFrequency (Output):** This argument provides the value of the Frequency state variable when the action response is created.
- **CurrentSpectrumInversion (Output):** This argument provides the value of the SpectrumInversion state variable when the action response is created
- **CurrentPidList (Output):** This argument provides the value of the PidList state variable when the action response is created.

6.2.2.3.2 RequestTables

Arguments for RequestTables are defined in Table 6.2–11.

REQ136 Upon receipt of the RequestTables action, the DRIT SHALL event each cached table section filtered by the A_ARG_TYPE_TID state variables using the TableSection state variable in less than 5s.

Table 6.2–11 - Arguments for RequestTables

Argument	Direction	Related State Variable
TID	In	A_ARG_TYPE_TID

- **TID (Input):** This argument sets the A_ARG_TYPE_TID state variable to the selected TID values.

6.2.2.3.3 AddPid

Arguments for AddPid are defined in Table 6.2–12.

REQ137 Upon receipt of the AddPid action, the DRIT SHALL send a new_flow_request() APDU with an MPEG flow type to the Card for each additional PID value in less than 500ms.

Table 6.2–12 - Arguments for AddPid

Argument	Direction	Related State Variable
AddPidList	In	A_ARG_TYPE_PidListChanges
RemainingPidFilter	Out	CCFreePidFilter

- **AddPidList (Input):** This argument provides a list of PID value that need to be added to the PidList state variable.

- **RemainingPidFilter (Output):** This argument provides the value of the CCFreePidFilter when the action response is created.

6.2.2.3.4 RemovePid

Arguments for RemovePid are defined in Table 6.2–13.

REQ138 Upon receipt of the RemovePid action, the DRIT SHALL send a delete_flow_request() APDU to the Card for each PID value in less than 500ms.

Table 6.2–13 - Arguments for RemovePid

Argument	Direction	Related State Variable
RemovePidList	In	A_ARG_TYPE_PidListChanges

- **RemovePidList (Input):** This argument provides a list of PID value that need to be removed from the PidList state variable.

6.2.3 Aux Service

REQ139 The DRIT SHALL support the Aux service only if it includes one or more baseband audio and video inputs.

6.2.3.1 Service Type

The following service type identifies a service that is compliant with this template:

urn:schemas-opencable-com:service:Aux:1

6.2.3.2 State Variables

The Aux service is architected around the state variables defined in Table 6.2–14.

REQ247 The DRIT SHALL support the state variables defined in Table 6.2–14 for the Aux Service.

REQ140 The DRIT SHALL support the Aux service state variables within the range specified in the Allowed Value column of Table 6.2–14.

REQ248 All state variables in the Aux Service designated as "Evented" SHALL be reported to subscribers in event messages.

Table 6.2–14 - State Variables for Aux Service

Variable Name	Data Type	Allowed Value	Default Value	Evented
GenLock	Boolean			Yes
Format	String	"UNKNOWN", "NTSC-M"		No
FormatList	String	"NTSC-M"		No
InputNumber	ui1			No
InputType	String	"S-VIDEO", "VIDEO"		No
SignalLevel	i4			No
SNR	ui4			No
SVideoInputs	ui1			No
VideoInputs	ui1			No

- **GenLock**: This variable is true if the DRIT received a gen-lock signal on the selected input. This variable is evented.
- **Format**: This variable defines the video format of the gen-lock signal on the selected input.
- **FormatList**: This variable is a string of video formats separated by comma supported by the DRIT.
- **InputNumber**: This variable defines which input among the selected input type group is active. The first input is numbered 1.
- **InputType:** This variable defines which input type group applies for input selection.
- **SignalLevel:** This variable reflects the current signal level in dBmV of the selected input.
- **SNR:** This variable reflects the current signal noise ratio value in dB of the selected input.
- **SVideoInputs:** This variable defines the number of s-video (Y/C) inputs supported by the DRIT.

- **VideoInputs**: This variable defines the number of composite (CVBS) inputs supported by the DRIT.

6.2.3.3 Actions

REQ141 The Aux Service SHALL support the actions as defined in Table 6.2–15.

Table 6.2–15 - Aux Service Actions

Actions	Requirements
REQ141.1 GetAuxCapabilities {}	Mandatory
REQ141.2 SetAuxParameters {}	Mandatory

6.2.3.3.1 GetAuxCapabilities

Arguments for GetAuxCapabilities are defined in Table 6.2–16.

REQ142 Upon receipt of the GetAuxCapabilities action, the DRIT SHALL provide a detailed description of its baseband video inputs in less than 500ms.

Table 6.2–16 - Arguments for GetAuxCapabilities

Argument	Direction	Related State Variable
SupportedFormat	Out	FormatList
SVideoNbr	Out	SVideoInputs
VideoNbr	Out	VideoInputs

- **SupportedFormat (Output):** This argument provides the value of the FormatList state variable when the action response is created.

- **SVideoNbr (Output):** This argument provides the value of the SVideoInputs state variable when the action response is created.

- **VideoNbr (Output):** This argument provides the value of the VideoInputs state variable when the action response is created.

6.2.3.3.2 SetAuxParameters

Arguments for SetAuxParameters are defined in Table 6.2–17.

REQ143 Upon receipt of the SetAuxParameters action, the DRIT SHALL select the required inputs, detect signal presence, and format in less than 5 seconds.

Table 6.2–17 - Arguments for SetAuxParameters

Argument	Direction	Related State Variable
SelectType	In	InputType
SelectInput	In	InputNumber
SelectFormat	In	Format
ActualFormat	Out	Format

Argument	Direction	Related State Variable
CurrentGenLock	Out	GenLock

- **SelectType (Input):** This argument sets the InputType state variable.
- **SelectInput (Input):** This argument sets the InputNumber state variable.
- **SelectFormat (Input):** This argument sets the Format state variable.

  REQ249 If the SelectFormat of the SetAuxParameter is unknown, then the value SHALL be set to "UNKNOWN".

- **ActualFormat (Output):** This argument reflects the value of the Format state variable.

  REQ144 If the DRIT has been able to resolve the UNKNOWN format setting in the SelectFormat of the SetAuxParameter, then the ActualFormat argument SHALL be set to the detected format.

- **CurrentGenLock (Output):** This argument provides the value of the GenLock state variable when the action response is created.

6.2.4 Encoder Service

6.2.4.1 Service Type

The following service type identifies a service that is compliant with this template:

urn:schemas-opencable-com:service:Encoder:1

6.2.4.2 State Variables

The Encoder service is architected around the state variables defined in Table 6.2–18.

REQ250 The DRIT SHALL support the state variables defined in Table 6.2–18 for the Encoder Service.

REQ251 The DRIT SHALL support the Encoder Service state variables within the range specified in the Allowed Value column of Table 6.2–18.

REQ252 All state variables in the Encoder Service designated as "Evented" SHALL be reported to subscribers in event messages.

Table 6.2–18 - State Variables for Encoder Service

Variable Name	Data Type	Allowed Value	Default Value	Evented
AudioBitrateMax	ui4			No
AudioBitrateMin	ui4			No
AudioBitrateMode	string	"CBR", "AVR", "VBR"	"CBR"	No
AudioBitrateStepping	ui4			No

Variable Name	Data Type	Allowed Value	Default Value	Evented
AudioBitrateTarget	ui4			No
AudioEncoderMethodList	Base64			No
AudioEncoderMethodNumber	ui1			No
AudioMute	Boolean			No
FieldOrder	String	"Lower", "Higher"		No
InputSelection	String	"Tuner", "Aux"		No
NoiseReduction	Boolean			No
PulldownDetection	Boolean			Yes
PulldownSelection	Boolean			No
SAPDetection	Boolean			Yes
SAPSelection	Boolean			No
VideoBitrateMax	ui4			No
VideoBitrateMin	ui4			No
VideoBitrateMode	string	"CBR", "AVR", "VBR"	"AVR"	No
VideoBitrateStepping	ui4			No
VideoBitrateTarget	ui4			No
VideoEncoderMethodList	Base64			No
VideoEncoderMethodNumber	ui1			No

- **AudioBitrateMax:** This variable defines the maximum bit rate value of the DRIT audio encoder method defined by the AudioEncoderMethodNumber state variable, and operating under the mode defined by the AudioBitrateMode state variable.

- **AudioBitrateMin:** This variable defines the minimum bit rate value of the DRIT audio encoder method defined by the AudioEncoderMethodNumber state variable, and operating under the mode defined by the AudioBitrateMode state variable.

- **AudioBitrateMode:** This variable defines the compression mode of the DRIT audio encoder. Three values are possible: 1- Constant Bit Rate (CBR), 2- Average Bit Rate (AVR), 3- Variable Bit Rate (VBR).

  REQ145 The DRIT Audio encoder SHALL support at least one of the following modes: 1- Constant Bit Rate (CBR), 2- Average Bit Rate (AVR), 3- Variable Bit Rate (VBR).

- **AudioBitrateStepping:** This variable defines the minimum increment bit rate value of the DRIT audio encoder method defined by the AudioEncoderMethodNumber state variable, and operating under the mode defined by the AudioBitrateMode state variable.

- **AudioBitrateTarget:** This variable defines the target bit rate of the DRIT audio encoder method defined by the AudioEncoderMethodNumber state variable, and operating under the mode defined

by the AudioBitrateMode state variable. If no value has been set by the SetEncoderParameters {} action, then the encoder default value is reported.

- **AudioEncoderMethodList:** This variable is a base64 object defined in Table 6.2–19. It is a complex object that can include one or more audio compression format combinations.

Table 6.2–19 - AudioEncoderMethodList Type Definition

	No. of bits	Mnemonic
number_audio_compression_format	8	uimsbf
for (i=0; i< number_audio_compression_format; i++) {		
Audio_algorithm_code	8	uimsbf
Sampling_rate	32	uimsbf
Bit_depth	8	uimsbf
Number_channel	8	uimsbf
}		

- **Algorithm_code:** This parameter is defined in Table 6.2–20.

  REQ146 The DRIT SHALL support at least one of the approved values defined in Table 6.2–20.

Table 6.2–20 - Audio_algorithm_code values

Audio_algorithm_code	Value
MPEG1 Layer II	0x00
Dolby AC3	0x01
Reserved	0x02-0xFF

- **Sampling_rate:** This parameter defines the rate in Hz at which the audio is sampled.
- **Bit_depth:** This parameter defines the number of bits per audio sample.
- **Number_Channel:** This parameter defines the number of audio channels supported by the DRIT audio encoder.

  REQ147 The DRIT SHALL support Audio sampling rate of at least 48kHz.

  REQ148 The DRIT SHALL support audio bit depth of at least 16bit per sample.

  REQ149 The DRIT SHALL support stereo audio.

- **AudioEncoderMethodNumber:** This variable defines the active audio encoder method.
- **AudioMute:** This variable is set to TRUE when the DRIT audio encoder provides silent audio.
- **FieldOrder:** This variable is set to "LOWER" if the lower field is presented first, and "HIGHER" if the higher field is presented first.

- **InputSelection:** This variable is set to "TUNER" when the DRIT encoder is processing the signal from the Tuner, and "AUX" when it is processing the signal from the Aux input.

- **NoiseReduction:** This variable is set to TRUE when the DRIT encoder is applying noise reduction processing on the video signal.

- **PulldownDetection**: This variable is TRUE when the DRIT encoder detect a video signal that supports a 3:2 pulldown.

- **PulldownSelection**: This variable is TRUE to enable the DRIT encoder to perform a 32 pulldown operation when the PulldownDetection state variable is set to TRUE.

- **SAPDetection**: This variable is TRUE when the DRIT encoder detects a second audio program (SAP).

- **SAPSelection**: This variable is TRUE to enable the DRIT encoder to select the second audio program when the SAPDetection state variable is set to TRUE.

  REQ1526 If the DRIT can simultaneously encode both the primary audio and the SAP, the DRIT SHALL send both audio streams.

- **VideoBitrateMax:** This variable defines the maximum bit rate value of the DRIT video encoder method defined by the VideoEncoderMethodNumber state variable, and operating under the mode defined by the VideoBitrateMode state variable.

- **VideoBitrateMin:** This variable defines the minimum bit rate value of the DRIT video encoder method defined by the VideoEncoderMethodNumber state variable, and operating under the mode defined by the VideoBitrateMode state variable.

- **VideoBitrateMode:** This variable defines the compression mode of the DRIT video encoder. Three values are possible: 1- Constant Bit Rate (CBR), 2- Average Bit Rate (AVR), 3- Variable Bit Rate (VBR).

  REQ150 The DRIT Video encoder SHALL support at least one of the following modes: 1- Constant Bit Rate (CBR), 2- Average Bit Rate (AVR), 3- Variable Bit Rate (VBR).

- **VideoBitrateStepping:** This variable defines the minimum increment bit rate value of the DRIT video encoder method defined by the VideoEncoderMethodNumber state variable, and operating under the mode defined by the VideoBitrateMode state variable.

- **VideoBitrateTarget:** This variable defines the target bit rate of the selected DRIT video encoder method defined by the VideoEncoderMethodNumber state variable, and operating under the mode defined by the VideoBitrateMode state variable. If no value has been set by the SetEncoderParameters {} action, then the encoder default value is reported

- **VideoEncoderMethodList:** This variable is a base64 object defined in Table 6.2–21. It is a complex object that can include one or more video compression format combinations.

Table 6.2–21 - VideoEncoderMethodList Type Definition

	No. of bits	Mnemonic
number_video_compression_format	8	uimsbf
for (i=0; i< number_video_compression_format; i++) {		
Vertical_size	16	uimsbf
Horizontal_size	16	uimsbf
Aspect_ratio_information	8	uimsbf
Frame_rate_code	8	uimsbf
Progressive_sequence	8	uimsbf
}		

All parameters are defined by [SCTE43].

REQ238 The DRIT video encoder MAY support any subset of [SCTE43] combinations.

- **VideoEncoderMethodNumber:** This variable defines the selected video encoder method.

6.2.4.3 Actions

REQ151 The Encoder Service SHALL support the actions as defined in Table 6.2–22.

Table 6.2–22 - Encoder Service Actions

Actions	Requirements
REQ151.1 GetEncoderCapabilities {}	Mandatory
REQ151.2 SetEncoderParameters {}	Mandatory
REQ151.3 GetEncoderParameters {}	Mandatory

6.2.4.3.1 GetEncoderCapabilities

Arguments for GetEncoderCapabilities are defined in Table 6.2–23.

REQ152 Upon receipt of the GetEncoderCapabilities action, the DRIT SHALL provide a detailed description of its audio and video encoder characteristics in less than 500ms.

Table 6.2–23 - Arguments for GetEncoderCapabilities

Argument	Direction	Related State Variable
AudioProfile	Out	AudioEncoderMethodList
VideoProfile	Out	VideoEncoderMethodList

- **AudioProfile (Output):** This argument provides the value of the AudioEncoderMethodList state variable when the action response is created.

- **VideoProfile (Output):** This argument provides the value of the VideoEncoderMethodList state variable when the action response is created.

6.2.4.3.2 SetEncoderParameters

Arguments for SetEncoderParameters are defined in Table 6.2–24.

REQ153 Upon receipt of the SetEncoderParameters action, the DRIT SHALL configure the audio and video encoder based on the input parameters in less than 2s.

Table 6.2–24 - Arguments for SetEncoderParameters

Argument	Direction	Related State Variable
AudioMode	In	AudioBitrateMode
AudioBitrate	In	AudioBitrateTarget
AudioMethod	In	AudioEncoderMethodNumber
Mute	In	AudioMute
FieldToggle	In	FieldOrdering
SignalSource	In	InputSelection
NoiseFilter	In	NoiseReduction
Pulldown	In	PulldownSelection
SAP	In	SAPSelection
VideoMode	In	VideoBitrateMode
VideoBitrate	In	VideoBitrateTarget
VideoMethod	In	VideoEncoderMethodNumber

- **AudioMode (Input):** This argument sets the AudioBitrateMode state variable.
- **AudioBitrate (Input):** This argument defines the AudioBitrateTarget state variable.
- **AudioMethod (Input):** This argument defines the AudioEncoderMethodNumber state variable.
- **Mute (Input):** This argument sets the AudioMute state variable.
- **FieldToggle (Input):** This argument sets the FieldOrdering state variable.
- **SignalSource (Input):** This argument sets the InputSelection state variable.
- **NoiseFilter (Input):** This argument sets the NoiseReduction state variable.
- **Pulldown (Input):** This argument sets the PulldownSelection state variable.
- **SAP (Input):** This argument set the SAPSelection state variable.
- **VideoMode (Input):** This argument sets the VideoBitrateMode state variable.
- **VideoBitrate (Input):** This argument defines the VideoBitrateTarget state variable.
- **VideoMethod (Input):** This argument defines the VideoEncoderMethodNumber state variable.

6.2.4.3.3 GetEncoderParameters

Arguments for GetEncoderParameters are defined in Table 6.2–25.

REQ154 Upon receipt of the GetEncoderParameters action, the DRIT SHALL report the current configuration of the audio and video encoder in less than 500ms.

Table 6.2–25 - Arguments for GetEncoderParameters

Argument	Direction	Related State Variable
CurrentAudioMax	Out	AudioBitrateMax
CurrentAudioMin	Out	AudioBitrateMin
CurrentAudioMode	Out	AudioBitrateMode
CurrentAudioStepping	Out	AudioBitrateStepping
CurrentAudioBitrate	Out	AudioBitrateTarget
CurrentAudioMethod	Out	AudioEncoderMethodNumber
CurrentMuteStatus	Out	AudioMute
CurrentFieldOrder	Out	FieldOrdering
CurrentSignalSource	Out	InputSelection
CurrentNoiseFilter	Out	NoiseReduction
CurrentPulldownStatus	Out	PulldownDetection
CurrentPulldownSetting	Out	PulldownSelection
CurrentSAPStatus	Out	SAPDetection
CurrentSAPSetting	Out	SAPSelection
CurrentVideoMax	Out	VideoBitrateMax
CurrentVideoMin	Out	VideoBitrateMin
CurrentVideoMode	Out	VideoBitrateMode
CurrentVideoBitrate	Out	VideoBitrateTarget
CurrentVideoStepping	Out	VideoBitrateStepping
CurrentVideoMethod	Out	VideoEncoderMethodNumber

- **CurrentAudioMax (Output):** This argument provides the value of the AudioBitrateMax state variable when the action response is created.
- **CurrentAudioMin (Output):** This argument provides the value of the AudioBitrateMin state variable when the action response is created.
- **CurrentAudioMode (Output):** This argument provides the value of the AudioBitrateMode state variable when the action response is created.
- **CurrentAudioStepping (Output):** This argument provides the value of the AudioBitrateStepping state variable when the action response is created.
- **CurrentAudioBitrate (Output):** This argument provides the value of the AudioBitrateTarget state variable when the action response is created.

- **CurrentAudioMethod (Output):** This argument provides the value of the AudioEncoderMethodNumber state variable when the action response is created.

- **CurrentMuteStatus (Output):** This argument provides the value of the AudioMute state variable when the action response is created.

- **CurrentFieldOrder (Output):** This argument provides the value of the FieldOrdering state variable when the action response is created.

- **CurrentSignalSource (Output):** This argument provides the value of the InputSelection state variable when the action response is created.

- **CurrentNoiseFilter (Output):** This argument provides the value of the NoiseReduction state variable when the action response is created.

- **CurrentPulldownStatus (Output):** This argument provides the value of the PulldowDetection state variable when the action response is created.

- **CurrentPulldownSetting (Output):** This argument provides the value of the PulldownSelection state variable when the action response is created.

- **CurrentSAPStatus (Output):** This argument provides the value of the SAPDetection state variable when the action response is created.

- **CurrentSAPSetting (Output):** This argument provides the value of the SAPSelection state variable when the action response is created

- **CurrentVideoMax (Output):** This argument provides the value of the VideoBitrateMax state variable when the action response is created.

- **CurrentVideoMin (Output):** This argument provides the value of the VideoBitrateMin state variable when the action response is created.

- **CurrentVideoMode (Output):** This argument provides the value of the VideoBitrateMode state variable when the action response is created.

- **CurrentVideoStepping (Output):** This argument provides the value of the VideoBitrateStepping state variable when the action response is created.

- **CurrentVideoBitrate (Output):** This argument provides the value of the VideoBitrateTarget state variable when the action response is created.

- **CurrentVideoMethod (Output):** This argument provides the value of the VideoEncoderMethodNumber state variable when the action response is created.

6.2.5 CAS Service

The CAS Service supports conditional access related communications over the DRI. The DRIT will typically interact with the Card to implement this service.

6.2.5.1 Service Type

The following service type identifies a service that is compliant with this template:

urn:schemas-opencable-com:service:CAS:1

6.2.5.2 State Variables

The CAS service is architected around the states variable defined in Table 6.2–26.

REQ253 The DRIT SHALL support the state variables defined in Table 6.2–26 for the CAS Service.

REQ155 The DRIT SHALL support the CAS Service state variables within the range specified in the Allowed Value column of Table 6.2–26.

REQ254 All state variables in the CAS Service designated as "Evented" SHALL be reported to subscribers in event messages.

Table 6.2–26 - State Variables for CAS Service

Variable Name	Data Type	Allowed Value	Default Value	Evented
A_ARG_TYPE_MMIDialogNumber	ui1			No
ApplicationList	Base64			No
CardMessage	string			Yes
CardManufacturer	string			No
CardVersion	string			No
CardStatus	enumerated string	Per Table 6.2–28		Yes
DaylightSaving	boolean			No
DescramblingMessage	string			No
DescramblingStatus	enumerated string	Per Table 6.2–29		Yes
EALocationCode	ui4			No
Language	string			No
Lock	boolean			Yes
MMIMessage	Base64			Yes
RatingRegion	ui1			No
SourceId	ui4			No
TimeZone	i4			No
VirtualChannelNumber	ui4			No

- **A_ARG_TYPE_MMIDialogNumber**: This variable is set by the NotifyMMIClose action to report the dialog number that has been closed by the HMS.

  REQ157 The DRIT SHALL include in the ApplicationList variable all information received from the Card application_info_cnf() APDU.

  REQ156 The DRIT SHALL report the MMI dialog number information from the A_ARG_TYPE_MMIDialogNumber back to the Card using a close_mmi_cnf() APDU.

- **ApplicationList**: This variable is a base64 object defined in Table 6.2–27. It is a complex object that can include one or more Card application descriptions.

Table 6.2–27 - ApplicationList Type Definition

	No. of bits	Mnemonic
application_count	8	uimsbf
for (i=0 ; i < application_count; i++) {		
Application_type	8	uimsbf
Application_version	16	uimsbf
Application_name_length	8	uimsbf
For (j=0 ; j<application_name_length; j++) {		
application_name_byte	8	uimsbf
}		
application_url_length	16	uimsbf
For (j=0 ; j< application_url_length; j++) {		
application_url_byte	8	uimsbf
}		
}		

- Every time the HMS will want to retrieve Card HTML content, it will send an HTTP request with a translated URL, as defined below

http://<DRIT IP ADDRESS>/get_cc_url?<**application_url**>

REQ158 Upon receipt of an HTTP request with a translated Card application URL, the DRIT SHALL send a server_query() APDU with the original content URL to the Card, which responds with the HTML page in the server_reply() APDU.

- **CardMessage:** This variable defines a warning message.

  REQ159 When the CardStatus state variable is set to "Error", the DRIT SHALL set the CardMessage variable to the error message defined in Appendix E of [CCIF].

  REQ160 When the CardStatus state variable is set to "FirmwareUpgrade", the DRIT SHALL set the CardMessage variable to the user_notification_text message received in the firmware_upgrade() APDU.

- **CardManufacturer:** This variable defines the Card manufacturer.

  REQ161 The DRIT SHALL set the CardManufacturer variable to the manufacturer name defined in the Card Information Structure (CIS).

- **CardVersion:** This variable defines the Card version number.

  REQ162 The DRIT SHALL set the CardVersion variable to the pod_version_number parameter of the received application_info_cnf() APDU.

- **CardStatus:** This variable defines the Card's status.

REQ163 The DRIT SHALL set the CardStatus variable based on the condition defined in Table 6.2–28.

Table 6.2–28 - CardStatus Conditions

Value	Condition
"Inserted"	If a Card is inserted with no error and no firmware upgrade condition
"Removed"	If there is no Card inserted
"Error"	If a Card is inserted and there is an error detected as defined in Appendix E of [CCIF]
"Firmware Upgrade"	If a Card is inserted with no error, but there is a pending firmware_upgrade() APDU

- **DaylightSaving:** This variable defines the DaylightSaving status of the DRIT.

  REQ165 The DRIT SHALL set the DaylightSaving variable to the value of the daylight_saving() parameter of the received feature_parameters() APDU.

  REQ166 If the daylight_saving() parameter is not available, then the DaylightSaving variable SHALL be set to null.

- **DescramblingMessage:** This variable complements the DescramblingStatus state variable to reports the full interpreted text message provided by the Card in the ca_pmt_reply message.

- **DescramblingStatus:** This variable codifies the descrambling status of the channel identified by the VirtualChannelNumber state variable.

  REQ167 The DRIT SHALL set the DescramblingStatus variable based on the ca_pmt_reply() APDU from the Card, as defined in Table 6.2–29.

Table 6.2–29 - DescramblingStatus Conditions

DescramblingStatus Variable	Condition
"Unknown"	NoCard response or ca_pmt_reply() with ca_enable = 0x74 to 0xFF
"Possible"	ca_pmt_reply() with ca_enable = 0x01
"Possible (purchase dialogue)"	CA_pmt_reply() with ca_enable = 0x02
"Possible (technical dialogue)"	ca_pmt_reply() with CA_enable = 0x03
Reserved	0x04 – 0x70
"Not possible (no entitlement)"	ca_pmt_reply() with ca_enable = 0x71
"Not possible (technical reason)"	ca_pmt_reply() with ca_enable = 0x73

- **EALocationCode:** This variable defines the Emergency Alert location code of the DRIT.

  REQ168 The DRIT SHALL set the EALocationCode variable to the value of the EA_location_code() parameter of the received feature_parameters() APDU.

  REQ169 If the EA_location_code() parameter is not available, then the EALocationCode variable SHALL be set to null.

- **Language:** This variable is set by the SetPreferredLanguage action in standard ISO639 format.

- **Lock:** The DRIT sets this value True whenever tune-by-VCN has succeeded and PCRLock is True.

- **MMIMessage:** This variable is a base64 object defined in Table 6.2–30. It is a complex object that allows the Card to request the HMS to open/close a dialog.

  REQ170 The DRIT SHALL include in the MMIMessage variable all information received from the Card open_MMI() and close_MMI() APDUs.

Table 6.2–30 - MMIMessage Type Definition

	No. of bits	Mnemonic
dialog_number	8	uimsbf
display_type	8	uimsbf
Action	8	uimsbf
If (Action == "Open") {		
Content_url_length	16	uimsbf
for (j=0 ; j< Content_url_length; j++) {		
Content_url_byte	8	uimsbf
}		
}		

Every time the HMS will want to retrieve Card HTML content, it will send an HTTP request with a translated URL, as defined below

http://<HOST IP ADDRESS>/get_cc_url?<**content_url**>

REQ171 Upon receipt of an HTTP request with a translated Card content URL, the DRIT SHALL send a server_query() APDU with the original content URL to the Card, which responds with the HTML page of the server_reply() APDU.

- Allowed values for action:

action	Value
Close	0x00
Open	0x01

- Allowed values for display_type:

display_type	Value
Full screen	0x00
Overlay	0x01
New window	0x02
Reserved	0x03-0xFF

- **RatingRegion:** This variable defines the rating region code of the DRIT.

  REQ172 The DRIT SHALL set the RatingRegion variable to the value of the rating_region() parameter of the received feature_parameters() APDU.

  REQ173 If the rating_region() parameter is not available, then the RatingRegion variable SHALL be set to null.

- **SourceId**: This variable is set by the SetChannel and GetEntitlement actions.

  REQ174 The DRIT SHALL only use the SourceId variable when a valid Card is inserted.

- **TimeZone:** This variable defines the time zone of the DRIT.

  REQ175 The DRIT SHALL set the TimeZone variable to the value of the time_zone() parameter of the received feature_parameters() APDU.

  REQ176 If the time_zone() parameter is not available, then the TimeZone variable SHALL be set to null.

- **VirtualChannelNumber:** This variable is set by the SetChannel and GetEntitlement actions.

  REQ177 The DRIT SHALL only use the VirtualChannelNumber variable when a valid Card is inserted.

6.2.5.3 Actions

REQ178 The CAS service SHALL support the actions as defined in Table 6.2–31.

Table 6.2–31 - CAS Service Actions

Actions	Requirements
REQ178.1 GetCardStatus {}	Mandatory
REQ178.2 GetEntitlement {}	Optional
REQ178.3 NotifyMmiClose {}	Mandatory
REQ178.4 SetChannel {}	Mandatory
REQ178.5 SetPreferedLanguage {}	Optional

6.2.5.3.1 GetCardStatus

Arguments for GetCardStatus are defined in Table 6.2–32.

REQ179 Upon receipt of the GetCardStatus action, the DRIT SHALL retrieve from the

Card all the necessary information to respond with updated information in less than 500ms.

Table 6.2–32 - Arguments for GetCardStatus

Argument	Direction	Related State Variable
CurrentCardStatus	out	CardStatus
CurrentCardManufacturer	out	CardManufacturer
CurrentCardVersion	out	CardVersion
CurrentDaylightSaving	out	DaylightSaving
CurrentEALocationCode	out	EALocationCode
CurrentRatingRegion	out	RatingRegion
CurrentTimeZone	out	TimeZone

- **CurrentCardStatus (Output):** This argument provides the value of the CardStatus state variable when the action response is created.
- **CurrentCardManafacturer (Output):** This argument provides the value of the CardManufacturer state variable when the action response is created.
- **CurrentCardVersion (Output):** This argument provides the value of the CardVersion state variable when the action response is created.
- **CurrentDaylightSaving (Output):** This argument provides the value of the DaylightSaving state variable when the action response is created.
- **CurrentEALocation (Output):** This argument provides the value of the EALocationCode state variable when the action response is created.
- **CurrentRatingRegion (Output):** This argument provides the value of the RatingRegion state variable when the action response is created.
- **CurrentTimeZone (Output):** This argument provides the value of the TimeZone state variable when the action response is created.

6.2.5.3.2 GetEntitlement

Arguments for GetEntitlement are defined in Table 6.2–33.

REQ180 Upon receipt of the GetEntitlement action, the DRIT SHALL perform in sequence the following actions in less than 1s.

1. REQ180.1 Clear all the previous program PID from the PidList (Mux service) state variable.
2. REQ180.2 Configure the tuner based on the service information tables received from the Card and either of the NewChannelNumber or the NewSourceId arguments. If both are defined and don't point to the same physical channel, then NewChannelNumber shall prevail.
3. REQ180.3 Set the ProgramNumber (Mux service) state variable.
4. REQ180.4 Reset the ACCI status message (see section 8.3) to 0.

5. REQ180.5 Set the DescramblingStatus state variable to “Unknown”.

6. REQ180.6 Set the DescramblingMessage state variable to Null.

7. REQ180.7 Send a ca_pmt(Query) APDU to the Card.

Table 6.2–33 - Arguments for GetEntitlement

Argument	Direction	Related State Variable
NewChannelNumber	In	VirtualChannelNumber
NewSourceId	In	SourceId
CurrentEntitlement	Out	DescramblingStatus
EntitlementMessage	Out	DescramblingMessage

Whereas:

- **NewChannelNumber (Input):** This argument sets the VirtualChannelNumber state variable.
- **NewSourceId (Input):** This argument sets the SourceId state variable
- **CurrentEntitlement (Output):** This argument provides the value of the DescramblingStatus state variable when the action response is created.
- **EntitlementMessage (Output):** This argument provides the value of the DescramblingMessage state variable when the action response is created.

6.2.5.3.3 NotifyMmiClose

Arguments for NotifyMmiClose are defined in Table 6.2–34.

REQ182 Upon receipt of the NotifyMmiClose action, the DRIT SHALL send a close_mmi_cnf() APDU to the Card with the MMI dialog number in less than 500ms.

Table 6.2–34 - Arguments for NotifyMmiClose

Argument	Direction	Related State Variable
MMIDialogNumber	In	A_ARG_TYPE_MMIDialogNumber

- **MMIDialogNumber (Input):** This argument sets the A_ARG_TYPE_MMIDialogNumber state variable.

6.2.5.3.4 SetChannel

Arguments for SetChannel are defined in Table 6.2–35.

REQ183 Upon receipt of the SetChannel action, the DRIT SHALL perform in sequence the following actions in less than 1s.

1. REQ183.1 Clear all the previous program PID from the PidList (Mux service) state variable.

2. REQ183.2 Configure the tuner based on the service information tables received from the Card and either of the NewChannelNumber or the NewSourceId arguments. If both are defined and don't point to the same physical channel, then NewChannelNumber prevails.

3. REQ183.3 Set the ProgramNumber (Mux service) state variable.

4. REQ183.4 Reset the ACCI status message (see section 8.3) to 0.

5. REQ183.5 Set the DescramblingStatus (CAS service) state variable to "Unknown".

6. REQ183.6 Set the DescramblingMessage (CAS service) state variable to Null.

7. REQ183.7 Send a ca_pmt(Ok_descrambling) APDU to the Card.

8. REQ183.8 Add all PID referenced in the new Program Map Table (PMT) and the PMT PID into the PidList (Mux service) state variable.

Table 6.2–35 - Arguments for SetChannel

Argument	Direction	Related State Variable
NewChannelNumber	In	VirtualChannelNumber
NewSourceId	In	SourceId
LockAchieved	Out	Lock

- **NewChannelNumber (Input):** This argument sets the VirtualChannelNumber state variable.

- **NewSourceId (Input):** This argument sets the SourceId state variable

6.2.5.3.5 SetPreferredLanguage

Arguments for SetPreferredLanguage are defined in Table 6.2–36.

REQ184 Upon receipt of the SetPreferredLanguage action, the DRIT SHALL send a feature_parameters() APDU to the Card with the language() parameter set to NewLIanguage.

Table 6.2–36 - Arguments for SetPreferredLanguage

Argument	Direction	Related State Variable
NewLanguage	In	Language

- **NewLanguage (Input):** This argument sets the Language state variable.

6.2.6 Mux

6.2.6.1 Service Type

The following service type identifies a service that is compliant with this template:

urn:schemas-opencable-com:service:Mux:1

6.2.6.2 State Variables

The Mux service is architected around the states variable defined in Table 6.2–37.

REQ255 The DRIT SHALL support the state variable defined in Table 6.2–37 for the Mux Service.

REQ185 The DRIT SHALL support the Mux Service state variables within the range specified in the Allowed Value column of Table 6.2–37.

REQ256 All state variables in the Mux Service designated as "Evented" SHALL be reported to subscribers in event messages.

Table 6.2–37 - State Variables for Mux Service

Variable Name	Data Type	Allowed Value	Default Value	Evented
A_ARG_TYPE_PidListChanges	String			No
PIDList	string			No
ProgramNumber	ui2			No

- A_ARG_TYPE_PidListChanges: This variable is used as an argument of the AddPid {} and RemovePid {} actions to modify the PidList state variable. It is defined as a string of TS Packet Identifier (PID) values in hex separated by comma.

- **PidList**: This variable is a string of TS Packet Identifier (PID) values in hex separated by comma, which represents all PID that have been enabled to go across the DRI Transport Interface. This variable includes:

    - PID 0x0000 for the Program Association Table (PAT)

    - The PID that have been directly set by the AddPid action. For security reasons, some active PID may be blocked by the DRI Security layer.

    - All PID referenced in the Program Map Table (PMT) and the PMT PID of the program directly selected by a SetProgram action

    - All PID referenced in the Program Map Table (PMT) and the PMT PID of the program indirectly selected by a SetChannel (CAS service) action.

  For security reasons, the DRI security layer may block some selected PIDs (See chapter 8)

- **ProgramNumber:** This variable defines the current MPEG program number as defined in the Program Association Table (PAT).

6.2.6.3 Actions

REQ186 The Mux service SHALL support the actions as defined in Table 6.2–38.

Table 6.2–38 - Mux Service Actions

Actions	Requirements
REQ186.1 SetProgram {}	Mandatory

Actions	Requirements
REQ186.2 AddPid {}	Mandatory
REQ186.3 RemovePid {}	Mandatory

6.2.6.3.1 SetProgram

Arguments for SetProgram are defined in Table 6.2–39.

REQ187 Upon receipt of the SetProgram action, the DRIT SHALL perform in sequence the following actions in less than 1s, only if a Card is not inserted.

1. REQ187.1 Clear all the previous program PID from the PidList state variable.
2. REQ187.2 Send a ca_pmt() APDU to the Card.
3. REQ187.3 Add all PID referenced in the new Program Map Table (PMT) and the PMT PID into the PidList state variable.

Table 6.2–39 - Arguments for SetProgram

Argument	Direction	Related State Variable
NewProgram	in	ProgramNumber

- **NewProgram (Input):** This argument sets the ProgramlNumber state variable.

6.2.6.3.2 AddPid

Arguments for AddPid are defined in Table 6.2–40.

REQ188 Upon receipt of the AddPid action, the DRIT SHALL add all new PID values to the state variable in less than 500ms.

Table 6.2–40 - Arguments for AddPid

Argument	Direction	Related State Variable
AddPidList	in	A_ARG_TYPE_PidListChanges

- **AddPidList (Input):** This argument provides a list of PID values that need to be added to the PidList state variable.

6.2.6.3.3 RemovePid

Arguments for RemovePid are defined in Table 6.2–41.

REQ189 Upon receipt of the RemovePid action, the DRIT SHALL remove all old PID values from the state variable in less than 500ms.

Table 6.2–41 - Arguments for RemovePid

Argument	Direction	Related State Variable
RemovePidList	in	A_ARG_TYPE_PidListChanges

- **RemovePidList (Input):** This argument provides a list of PID values that need to be removed from the PidList state variable.

6.2.7 Security Service

6.2.7.1 Service Type

The following service type identifies a service that is compliant with this template:

urn:schemas-opencable-com:service:Security:1

6.2.7.2 State Variables

The Security service is architected around the state variables defined in Table 6.2–42.

REQ257 The DRIT SHALL support the state variables defined in Table 6.2–42 for the Security Service .

REQ190 The DRIT SHALL support the Security Service state variables within the range specified in the Allowed Value column of Table 6.2–42.

REQ258 All state variables in the Security Service designated as "Evented" SHALL be reported to subscribers in event messages.

Table 6.2–42 - State Variables for Security Service

Variable Name	Data Type	Allowed Value	Default Value	Evented
DrmUUIDList	String			No
DrmUUID	String			No
DrmPairingStatus	String	“Green”, “Orange”, “Red”		Yes
HmsAssociationList	Base64			No

- **DrmUuidList**: This variable is a string of Universal Unique Identifier (UUID) values in hex separated by comma, which identify the CableLabs approved DRM systems supported by the DRIT and currently not listed in Drm_revocation_list.

- **DrmUuid**: This variable defines which DRM system is currently active.

- **DrmPairingStatus:** This variable defines the current DRM pairing status of the DRIT. The values are defined in Table 6.2–43.

Table 6.2–43 - DrmPairingStatus Values

Value	Condition
“Green”	The DRIT is paired with a HMS, controlled content is released under DRM protection
“Orange”	The DRIT is paired with a HMS, but this pairing is going

Value	Condition
	to time out. The HMS is expected to refresh its pairing in background. Controlled content is released under DRM protection
"Red"	The DRIT is not paired. No controlled content is released.

- **HmsAssociationList:** This variable is an authenticated message that lists all the HMS Association records as defined by [OCUR]. The syntax is defined in Table 6.2–44.

Table 6.2–44 - HMS Association Message

Syntax	# of bits	Mnemonic
hms_association_message () {		
hms_association_message_tag	24	uimsbf
length_field ()		
HOST_ID	40	uimsbf
CARD_ID	64	uimsbf
current_hms_association_index	8	uimsbf
number_of_records	8	uimsbf
for (i = 0; i < number_of_records; i ++) {		
hms_association_record	352	uimsbf
}		
rsa_signature ()	1024	uimsbf
}		

Whereas:

- **hms_association_message_tag** - This variable is set to 0x9F9072
- **HOST_ID** – The DRIT's unique Host ID defined in the CableLabs-issued Host device certificate embedded in the DRIT.
- **CARD_ID** – The Card's unique ID defined in the CableLabs-issued Card device certificate embedded in the Card
- **Current_hms_association_index –** The index in the array of records that corresponds to the currently DRM paired HMS.
- **Number_of_record –** Number of HMS devices currently registered by the DRIT.
- **hms_association_record** – As defined in [OCUR]
- **rsa_signature** – RSA signature of the full message with the DRIT private key.

6.2.7.3 Actions

REQ192 The Security service SHALL support the actions as defined in Table 6.2–45.

Table 6.2–45 - Security Service Actions

Actions	Requirements
REQ192.1 SetDRM {}	Mandatory

6.2.7.3.1 SetDRM

Arguments for SetDRM are defined in Table 6.2–46.

REQ193 Upon receipt of the SetDRM action, the DRIT SHALL set the DrmPairingStatus state variable to "Red" and switch to the designated DRM systems in less than 5s.

Table 6.2–46 - Arguments for SetDRM

Argument	Direction	Related State Variable
NewDrm	in	DrmUUID

- **NewDrm (Input):** This argument sets the DrmUUID state variable.

  REQ194 If the NewDrm argument identifies a DRM system that is not supported by the DRIT, the DRIT SHALL set the DrmPairingStatus to "Red".

6.2.8 Diag Service

6.2.8.1 Service Type

The following service type identifies a service that is compliant with this template:

urn:schemas-opencable-com:service:Diag:1

6.2.8.2 State Variables

The Diag service is architected around the states variable defined in Table 6.2–47.

REQ259 The DRIT SHALL support the state variables defined in Table 6.2–47 for the Diag Service.

REQ195 The DRIT SHALL support the Diag Service state variables within the range specified in the Allowed Value column of Table 6.2–47.

REQ260 All state variables in the Diag Service designated as "Evented" SHALL be reported to subscribers in event messages.

Table 6.2–47 - State Variables for Diag Service

Variable Name	Data Type	Allowed Value	Default Value	Evented
A_ARG_TYPE_Parameter	String			No
A_ARG_TYPE_Value	String			No
A_ARG_TYPE_Volatile	Boolean			No
ParameterList	String			No

- **A_ARG_TYPE_Parameter**: This variable is used as argument of the GetParameter action to provide the name of the requested parameter.

- **A_ARG_TYPE_Value:** This variable is used as argument of the GetParameter action to return the value of the requested parameter.

- **A_ARG_TYPE_Volatile:** This variable is a Boolean that is set to TRUE when the requested parameter is dynamic and FALSE when it is static for the DRIT.

- **ParameterList:** This variable is a string of parameter names separated by comma, which enumerates the diagnostic parameters supported by the DRIT.

REQ196 The DRIT SHALL support the diagnostic parameters defined in Table 6.2–48.

Table 6.2–48 - Diagnostic Parameter List

Parameter Name	Description
"Host Manufacturer"	Name of the DRIT manufacturer
"Host Serial Number"	Serial Number of the DRIT
"Host ID"	Unique ID of the DRIT used for Card/Host binding
"Host Power Status"	Explicit description of the current power status
"Host Boot Status"	Explicit description of the current boot status
"Host Memory Report"	Explicit description of the current memory allocation"
"Host Application"	Explicit description of the DRM application supported by the devices, including name, version number and date.
"Host Firmware"	Explicit description of the DRIT firmware including name, version number and date.

REQ197 If other parameters are added, they SHOULD be formatted such that each parameter value is a string that includes the value of the parameter and its complete description:

The following is an example of this:

```
A_ARG_TYPE_Parameter:   "Host Firmware"
A_ARG_TYPE_Value:       "xxxx - Version: 1.02 - Date: xx/xx/xx"
```

6.2.8.3 Actions

REQ198 The Diag service SHALL support the action as defined in Table 6.2–49.

Table 6.2–49 - Diag Service Actions

Actions	Requirements
REQ198.1 GetParameter {}	Mandatory

6.2.8.3.1 GetParameter

Arguments for GetParameter are defined in Table 6.2–50.

REQ199 Upon receipt of the GetParameter action, the DRIT SHALL return the value and

the type of the parameter in less than 500ms.

Table 6.2–50 - Arguments for GetParameter

Argument	Direction	Related State Variable
Parameter	in	A_ARG_TYPE_Parameter
Value	out	A_ARG_TYPE_Value
Volatile	out	A_ARG_TYPE_Volatile

- **Parameter (Input):** This argument sets the A_ARG_TYPE_Parameter state variable.
- **Value (Output):** This argument provides the value of the A_ARG_TYPE_Value state variable when the action response is created.
- **Volatile (Output):** This argument provides the value of the A_ARG_TYPE_Volatile state variable when the action response is created**.**

6.2.9 AVTransport Service

REQ200 The DRIT SHALL provide an implementation of the UPnP AVTransport Service as defined in [UPNPAV].

6.2.10 ConnectionManager Service

REQ201 The DRIT SHALL provide an implementation of the UPnP ConnectionManager as defined in [UPNPCM].

REQ202 The ConnectionManager Service of the Media Server Device SHALL implement the following Actions along with the required Actions defined in [UPNPCM]:

REQ202.1 PrepareForConnection

REQ202.2 ConnectionComplete

REQ203 The SourceProtocolInfo State Variable of the ConnectionManager SHALL support the following transport protocol / content format definitions:

REQ203.1 rtsp-rtp-udp::dri-mp2t:**

6.3 Eventing

REQ204 The DRI specification defines nine events, listed in Table 6.3–1, all of which SHALL be made available to any connected device that subscribed to its services according to [UPNPAV].

REQ204.1 The nine events listed in Table 6.3–1 SHALL be sent within one second of the triggering system state transition.

Table 6.3–1 - DRI Events

Event Name	Service
PCRLock	Tuner
Seeking	Tuner

Event Name	Service
TableSection	FDC
GenLock	Aux
CardStatus	CAS
CardMessage	CAS
MMIMessage	CAS
DescramblingStatus	CAS
DrmPairingStatus	Security
PulldownDetection	Encoder
SAPDetection	Encoder

6.4 DRI Transport

6.4.1 Transport Characteristics

The DRI Transport Interface (DRI-TI) implements a controlled-jitter transmission compliant with the following requirements.

REQ1527 The DRIT SHALL output the selected program content as a single program MPEG-TS in RTP packets according to [RTSP] and [RTP].

REQ261 The DRIT SHALL release all MPEG PES packets of the selected program unaltered and in the order received.

REQ262 The DRIT SHALL NOT remove any TS Packets of the selected program in which the transport_error indicator is set.

REQ263 The maximum jitter of TS Packets on the DRI-TI SHALL NOT exceed 50ms.

REQ264 The DRIT SHALL NOT send a RTP/UDP packet more often than every 5ms.

REQ265 The MPEG transport stream output by the DRIT on the DRI-TI SHALL otherwise comply with [MPEG-S].

These requirements create the notion of a periodic burst transmission: At the end of a Transport Interval, the DRIT releases one UDP packet with all aggregated MPEG packets during the interval.

REQ1528 The DRIT SHALL use a Transport Interval between 5ms and 50 ms.

Figure 6.4-1 and Figure 6.4-2 depict examples of how a DRIT may transmit programs of widely differing rates on the DRI-TI. Both depict a DRI-TI with 60Mbps available for transmission by the DRIT such as a 100Mbps Ethernet connection with limited other traffic.

Figure 6.4-1 depicts an HDTV program with a variable rate averaging 12Mbps and peaking at 38 Mbps, near the full rate of a 256-QAM cable channel. The HDTV program data is accumulated in a buffer and then transmitted every 20ms at the maximum rate available on the DRI until the buffer is empty.

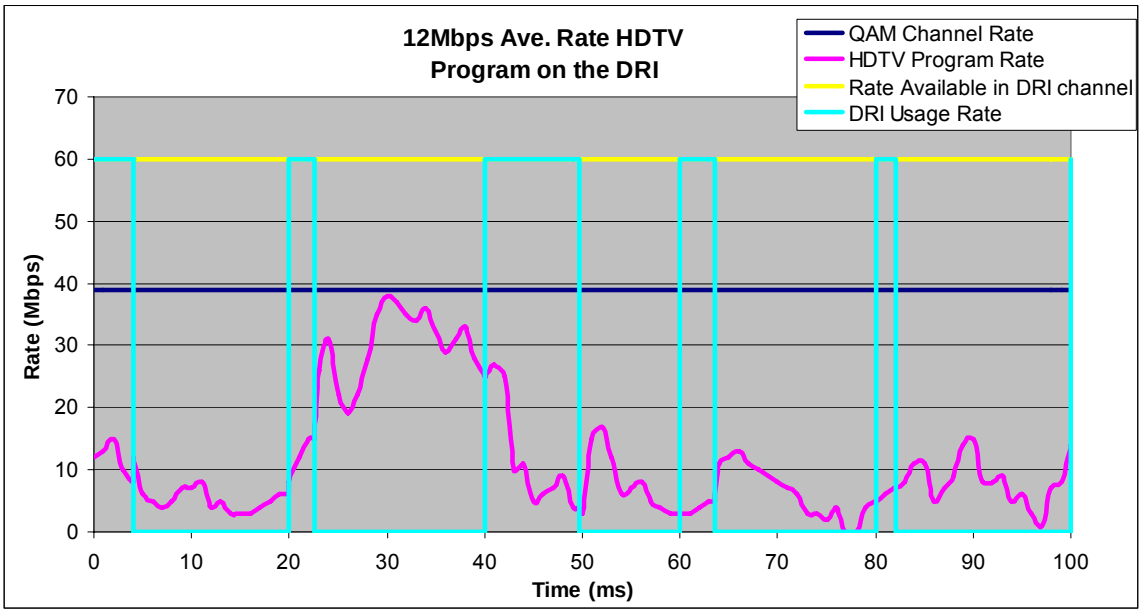

Figure 6.4-1 – High Bit-rate Program on DRI (Informative)

Figure 6.4-2 depicts an audio-only program with a fixed rate of 256kbps received from a 64-QAM cable channel. The audio program data is accumulated in a buffer and then transmitted every 50ms at the maximum rate available on the DRI until the buffer is empty.

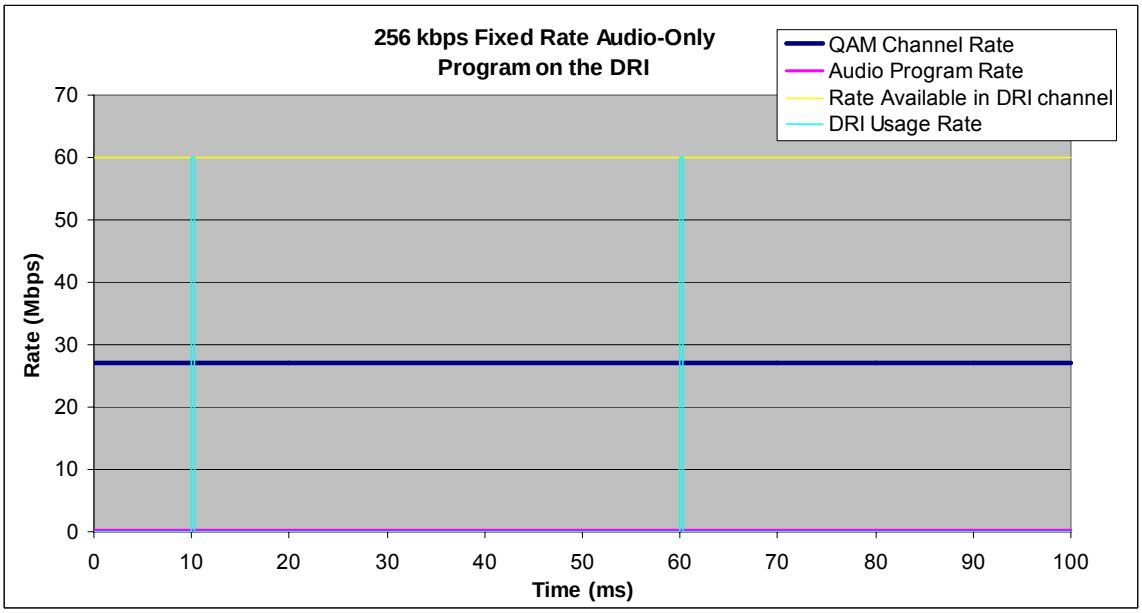

Figure 6.4-2 - Low-bit rate example of a Controlled-jitter Transmission (Informative)

6.4.2 Transport Quality of Service

The intent of this specification is to ensure uninterrupted flow of content. However transmission errors may occur. The requirement to not delete TS Packets with the transport error flag set is to provide the HMS with the means to discriminate between errors induced by the RF transmission and errors induced in delivery across the DRI.

7 DRI INITIALIZATION

On power-up, the DRIT shall follow the initialization of the DRI output as defined in this section.

7.1 UPnP Device Discovery

REQ211 The DRIT SHALL provide an implementation of a UPnP Device as defined in [UPNPDA] for being discovered and controlled by standard UPnP Control Point on the LAN.

7.2 DRM Registration

The HMS asks the DRIT for its list of supported DRM systems. The HMS selects one from this list and proprietary DRM pairing follows. If the DRIT confirms that it has established an exclusive cryptographically-secure relationship (DRM Pairing) with the HMS, it indicates to the HMS and enables delivery of protected content under that DRM.

The following requirements apply:

REQ212 When a Card is inserted, the DRIT SHALL NOT release any content if the DRM Registration hasn't successfully occurred.

REQ213 When there is no Card, the DRIT SHALL NOT release any copy-protected content (including Macrovision, APS, and CGMS-A) if the DRM Registration hasn't successfully occurred.

REQ214 The DRIT SHALL only be paired with one HMS at a given time.

REQ215 The DRIT SHALL support pairing to only one DRM system at a given time.

The DRM registration process represented in Figure 7.2-1 calls for a Proprietary DRM Pairing operation that is beyond the scope of this document.

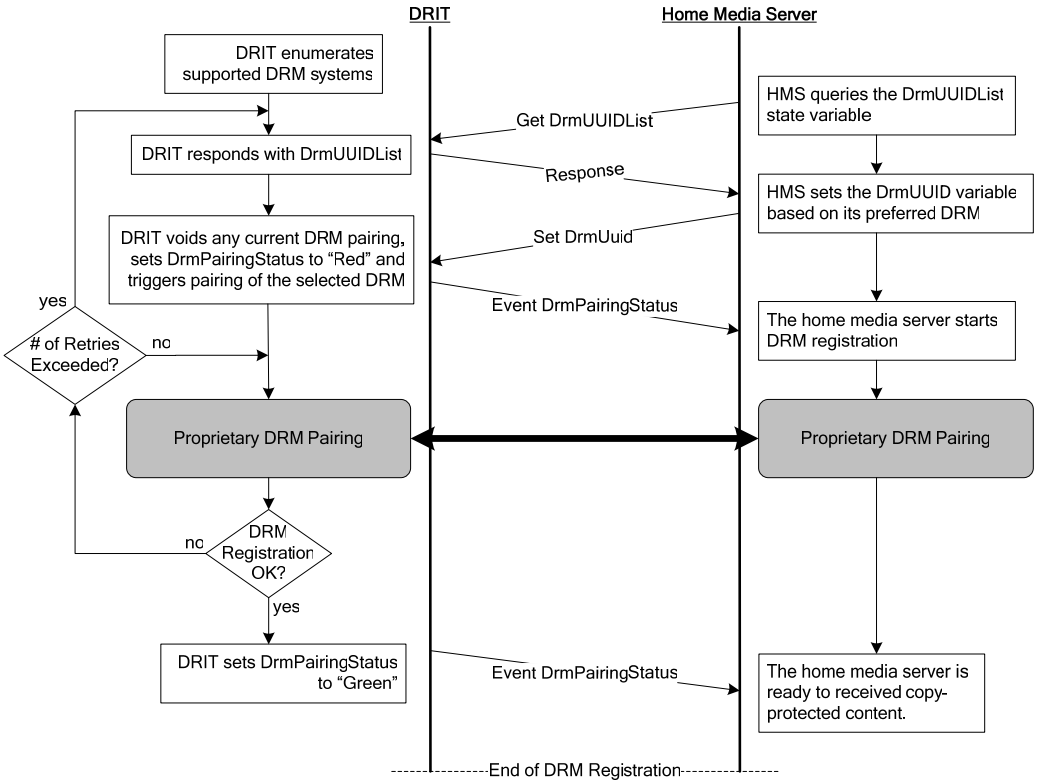

Figure 7.2-1 - DRM Registration(Informative)

8 DRI SECURITY

8.1 Introduction

The DRI protocol defines a common security framework for a DRM plug-in. By standardizing some aspects of the content protection layer, the DRI protocol enables one DRM system to share recorded content with another DRM system without re-encryption, as long as the content licenses are bridged properly and the decryption keys are passed from the originating DRM to the receiving DRM.

The following sub-sections define the DRI security requirements in terms of:

- DRM Pairing Control – This section defines how DRM pairing is dependent on Card/DRIT binding.
- License Generation – This section defines the conditions for generating and transmitting a DRM license after a channel change.
- Content Protection – This section defines the scrambling methods and algorithms that the DRIT applies to protected content.
- Content Identification – This section defines the indexing information that the DRIT adds in-band for the purpose of retrieving the matching DRM license, when the content is consumed.

8.2 DRM Pairing Control

The DRM pairing operation is under control of the DRM system.

REQ217 Immediately following successful Card-Host authentication, per [CCCP], the DRIT SHALL void any current DRM pairing, set its DRM pairing status to "Red", and send DrmPairingStatus to any connected HMS.

8.3 License Generation Sequence

REQ219 The DRIT SHALL generate and send a new DRM license immediately following Channel Change without a Card.

REQ1529 The DRIT SHALL generate and send a new DRM license immediately following Channel Change with a Card.

REQ1530 The DRIT SHALL generate and send a new DRM license immediately following any change in the value of ACCI.

8.3.1 Scenario 1: After Channel Change Without a Card

Figure 8.3-1 shows the Aggregate Content Control Information (ACCI, as described in [OCUR]) transitions after a channel change when there is no Card.

(A) – The channel change is an asynchronous event triggered by the user. At instant (A), $ACCI_A$ stops applying and ACCI is reset by the DRIT.

(B) – The DRIT terminates the encryption of the content in less than 100ms, after event (A), shown as ΔTe.

(C) – The ACCI status message change is an asynchronous event triggered by the cable head-end. At instant (C) the DRM license generation process starts.

(D) – The DRM license generation process completes in less than one (1) second, or, after event (C). It creates a DRM license with policy based on the ACCI status message and an associated AES key. Both become valid after the next PES payload.

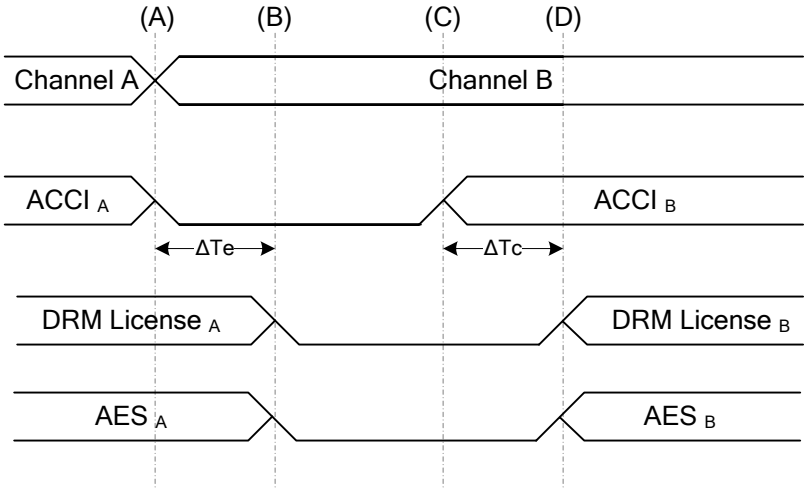

Figure 8.3-1 - DRM License Generation after a Channel Change without a Card

8.3.2 Scenario 2: After a Channel Change With a Card

Figure 8.3-2 shows the ACCI transitions after a program change when there is a Card inserted and powered. The DRM system is unaware of the CCCP key transitions.

(A) – The channel change is an asynchronous event triggered by the user. At instant (A), $ACCI_A$ stops applying, ACCI is reset by the DRIT and the DRM license generation process starts.

(B) – The DRM license generation process completes in less than one (1) second, or ΔT, after event (A). It creates a DRM license with policy based on the reset ACCI status message and an associated AES key. Both become valid after the next PES payload.

(C) – The ACCI status message change is an asynchronous event triggered by the cable head-end. At instant (C) the DRM license generation process starts.

(D) – The DRM license generation process completes in less than one (1) second, or ΔT, after event (C). It creates a DRM license with policy based on the ACCI status message and an associated AES key. Both become valid after the next PES payload.

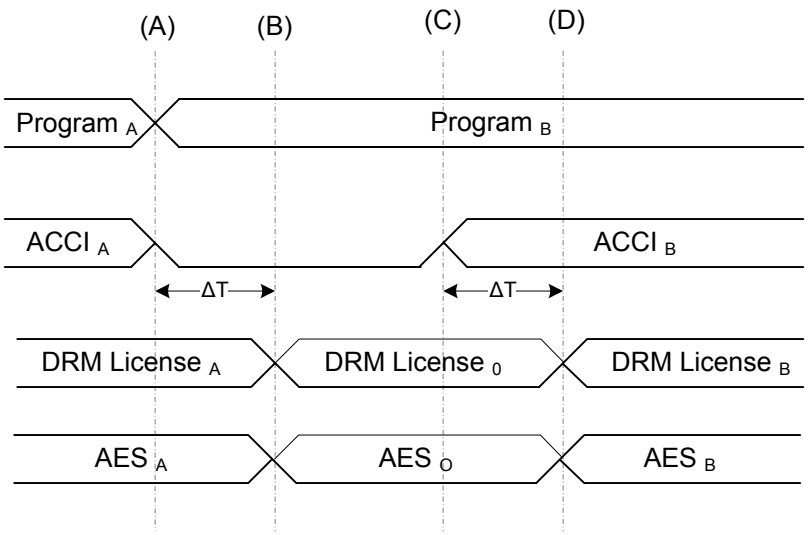

Figure 8.3-2 - DRM License Generation after a Channel Change with a Card

Note (Informative): Based on the refreshing rate of the components of the ACCI status message, as set by the cable headend, event (C) can happen before (B).

8.3.3 Scenario 3: After an ACCI Update

ACCI may change at any time due to a change in any of the multiple sources of ACCI data, including CCI delivery by the Card and changes in SCTE 21 or CGMS-A data. Figure 8.3-3 shows the ACCI transition on the DRI immediately following any change in ACCI status of a currently tuned channel. The DRM system is unaware of the CPkey transitions on the CHI.

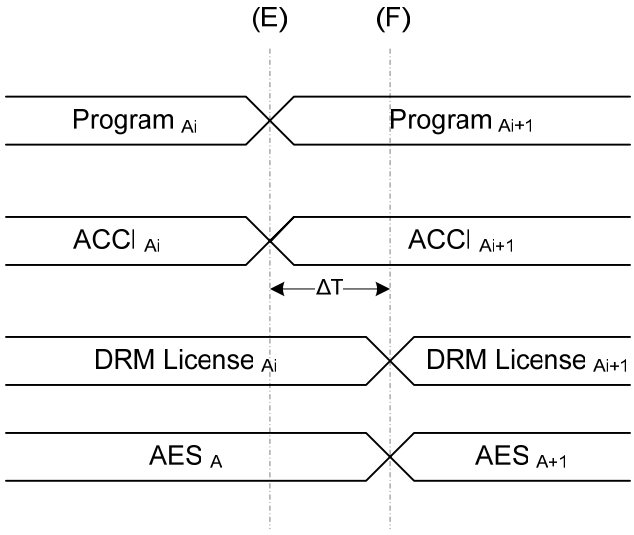

Figure 8.3-3 - DRM License Generation after a ACCI Update

(E) – The ACCI update is an asynchronous event triggered by the cable head-end. It differs from transition (C) because the active license is not based on the reset ACCI status message. At instant (E), the DRM license generation process starts.

(F) – The DRM license generation process completes in less than one (1) second, or ΔT, after event (E). It creates a DRM license with policy defined by the updated ACCI state, and an associated AES key. Both become valid after the next PES payload.

Note (Informative): Based on the ACCI update policy as set by the cable headend, the DRM system can issue more than one license per program.

8.4 Content Protection

The DRI security layer consists of an encryption layer (AES) and a content identification mechanism (TAG), as shown in Figure 8.4-1.

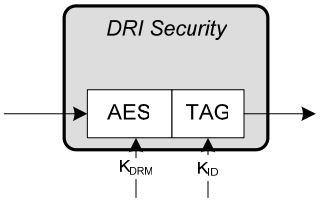

Figure 8.4-1 - DRI Security

8.4.1 Content Encryption

Content Encryption is consists of AES 128-bit key in counter mode, where all the following requirements apply:

REQ1531 The DRIT SHALL apply 128-bit AES encryption to all programs delivered over the DRI.

REQ1532 The DRIT SHALL employ counter (CTR) mode of 128-bit AES per [AES].

REQ1533 The DRIT SHALL AES128 encrypt 98% or more of the payload portion of each recognized format audio or video PES over every 100-second period of that PES.

REQ1534 The DRIT SHALL AES128 encrypt 100% of the payload portion of each TS Packet carrying data types other than recognized formats of audio or video PES.

REQ1535 The DRM system SHALL provide a secure means to establish and change the shared secret 128-bit AES encryption keys in the DRIT and HMS.

REQ1536 The DRM system SHALL protect ACCI values against unauthorized modification, substitution, and loss of temporal association such that the CCI delivered for specific content will control the specified parameters for output of that content.

The means for DRM pairing, generation, and synchronization of encryption keys is proprietary to the DRM system and beyond the scope of this specification.

8.4.2 Content Identification

The content identification based on insertion of data into the content stream in the AV Transport layer of the DRI is provided to facilitate synchronization of ACCI with its associated content. The means of such association and use of the TAG mechanism are beyond the scope of this specification.

REQ1537 The DRIT SHALL maintain and make available on the DRI the temporal association of ACCI and content of one second (1s) or less; see [CCCP].

9 INTERFACE EXTENSIBILITY

When a DRIT supports additional functionality, it may do so by exposing additional embedded UPnP devices, additional UPnP services, or by extending the defined UPnP services with vendor extensions as described in the UPnP documentation. This extension should be done is such a way as to not compromise the functionality, interoperability, user experience, or robustness of the DRIT or its security and content protection features.

9.1 Presentation Page

REQ267 This Presentation Page specification requires that a minimum set of information SHALL be projected by the DRIT through the means of a UPnP presentation page, as shown in Figure 9.1-1.

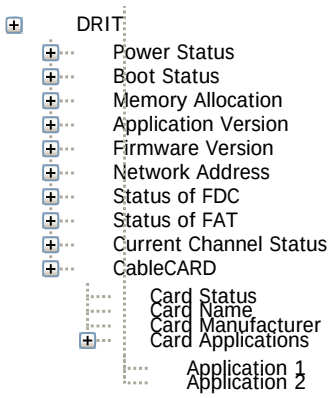

Figure 9.1-1 - DRI Presentation Page

Most of these parameters are defined by [OCUR]. The following requirements also apply:

REQ235 The DRI Device - Application Version Page SHALL include the DRM application details.

REQ1538 The presentation page SHALL be structured so that it can be rendered with a regular browser (i.e., Internet Explorer, Netscape…).

Annex A DRI XML Schema (Normative)

A.1 Aux

```
<?xml version="1.0" encoding="utf-8" ?>
- <scpd xmlns="urn:schemas-upnp-org:service-1-0">
- <specVersion>
  <major>1</major>
  <minor>0</minor>
  </specVersion>
- <actionList>
- <action>
  <name>GetAuxCapabilities</name>
- <argumentList>
- <argument>
  <name>SupportedFormat</name>
  <direction>out</direction>
  <relatedStateVariable>FormatList</relatedStateVariable>
  </argument>
- <argument>
  <name>SVideoNbr</name>
  <direction>out</direction>
  <relatedStateVariable>SVideoInputs</relatedStateVariable>
  </argument>
- <argument>
  <name>VideoNbr</name>
  <direction>out</direction>
  <relatedStateVariable>VideoInputs</relatedStateVariable>
  </argument>
  </argumentList>
  </action>
- <action>
  <name>SetAuxParameters</name>
- <argumentList>
- <argument>
  <name>SelectType</name>
  <direction>in</direction>
  <relatedStateVariable>InputType</relatedStateVariable>
  </argument>
- <argument>
  <name>SelectInput</name>
  <direction>in</direction>
  <relatedStateVariable>InputNumber</relatedStateVariable>
  </argument>
- <argument>
  <name>SelectFormat</name>
  <direction>in</direction>
  <relatedStateVariable>Format</relatedStateVariable>
```

```
    </argument>
- <argument>
  <name>ActualFormat</name>
  <direction>out</direction>
  <relatedStateVariable>Format</relatedStateVariable>
    </argument>
- <argument>
  <name>CurrentGenLock</name>
  <direction>out</direction>
  <relatedStateVariable>GenLock</relatedStateVariable>
    </argument>
    </argumentList>
    </action>
    </actionList>
- <serviceStateTable>
- <stateVariable sendEvents="no">
  <name>Format</name>
  <dataType>string</dataType>
- <allowedValueList>
  <allowedValue>UNKNOWN</allowedValue>
  <allowedValue>NTSC-M</allowedValue>
    </allowedValueList>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>InputType</name>
  <dataType>string</dataType>
- <allowedValueList>
  <allowedValue>S-VIDEO</allowedValue>
  <allowedValue>VIDEO</allowedValue>
    </allowedValueList>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>InputNumber</name>
  <dataType>ui1</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>SignalLevel</name>
  <dataType>i4</dataType>
    </stateVariable>
- <stateVariable sendEvents="yes">
  <name>GenLock</name>
  <dataType>boolean</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>VideoInputs</name>
  <dataType>ui1</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>SVideoInputs</name>
```

```
<dataType>ui1</dataType>
</stateVariable>
- <stateVariable sendEvents="no">
<name>SNR</name>
<dataType>ui4</dataType>
</stateVariable>
- <stateVariable sendEvents="no">
<name>FormatList</name>
<dataType>string</dataType>
- <allowedValueList>
<allowedValue>NTSC-M</allowedValue>
</allowedValueList>
</stateVariable>
</serviceStateTable>
</scpd>
```

A.2 CAS

```
<?xml version="1.0" encoding="utf-8" ?>
- <scpd xmlns="urn:schemas-upnp-org:service-1-0">
- <specVersion>
<major>1</major>
<minor>0</minor>
</specVersion>
- <actionList>
- <action>
<name>GetCardStatus</name>
- <argumentList>
- <argument>
<name>CurrentCardStatus</name>
<direction>out</direction>
<relatedStateVariable>CardStatus</relatedStateVariable>
</argument>
- <argument>
<name>CurrentCardManufacturer</name>
<direction>out</direction>
<relatedStateVariable>CardManufacturer</relatedStateVariable>
</argument>
- <argument>
<name>CurrentCardVersion</name>
<direction>out</direction>
<relatedStateVariable>CardVersion</relatedStateVariable>
</argument>
- <argument>
<name>CurrentDaylightSaving</name>
<direction>out</direction>
<relatedStateVariable>DaylightSaving</relatedStateVariable>
</argument>
- <argument>
```

```
<name>CurrentEALocationCode</name>
<direction>out</direction>
<relatedStateVariable>EALocationCode</relatedStateVariable>
</argument>
- <argument>
<name>CurrentRatingRegion</name>
<direction>out</direction>
<relatedStateVariable>RatingRegion</relatedStateVariable>
</argument>
- <argument>
<name>CurrentTimeZone</name>
<direction>out</direction>
<relatedStateVariable>TimeZone</relatedStateVariable>
</argument>
</argumentList>
</action>
- <action>
<name>GetEntitlement</name>
- <argumentList>
- <argument>
<name>NewChannelNumber</name>
<direction>in</direction>
<relatedStateVariable>VirtualChannelNumber</relatedStateVariable>
</argument>
- <argument>
<name>NewSourceId</name>
<direction>in</direction>
<relatedStateVariable>SourceId</relatedStateVariable>
</argument>
- <argument>
<name>CurrentEntitlement</name>
<direction>out</direction>
<relatedStateVariable>DescramblingStatus</relatedStateVariable>
</argument>
- <argument>
<name>EntitlementMessage</name>
<direction>out</direction>
<relatedStateVariable>DescramblingMessage</relatedStateVariable>
</argument>
</argumentList>
</action>
- <action>
<name>NotifyMmiClose</name>
- <argumentList>
- <argument>
<name>MMIDialogNumber</name>
<direction>in</direction>
relatedStateVariable>A_ARG_TYPE_MMIDialogNumber</relatedStateVariable>
</argument>
```

```
    </argumentList>
    </action>
- <action>
  <name>SetChannel</name>
- <argumentList>
- <argument>
  <name>NewChannelNumber</name>
  <direction>in</direction>
  <relatedStateVariable>VirtualChannelNumber</relatedStateVariable>
    </argument>
- <argument>
  <name>NewSourceId</name>
  <direction>in</direction>
  <relatedStateVariable>SourceId</relatedStateVariable>
    </argument>
    </argumentList>
    </action>
- <action>
  <name>SetPreferredLanguage</name>
- <argumentList>
- <argument>
  <name>NewLanguage</name>
  <direction>in</direction>
  <relatedStateVariable>Language</relatedStateVariable>
    </argument>
    </argumentList>
    </action>
    </actionList>
- <serviceStateTable>
- <stateVariable sendEvents="no">
  <name>CardManufacturer</name>
  <dataType>string</dataType>
    </stateVariable>
- <stateVariable sendEvents="yes">
  <name>CardStatus</name>
  <dataType>string</dataType>
- <allowedValueList>
  <allowedValue>Inserted</allowedValue>
  <allowedValue>Removed</allowedValue>
  <allowedValue>Error</allowedValue>
  <allowedValue>Firmware Upgrade</allowedValue>
    </allowedValueList>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>ApplicationList</name>
  <dataType>bin.base64</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>SourceId</name>
```

```
  <dataType>ui4</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>VirtualChannelNumber</name>
  <dataType>ui4</dataType>
    </stateVariable>
- <stateVariable sendEvents="yes">
  <name>DescramblingStatus</name>
  <dataType>string</dataType>
- <allowedValueList>
  <allowedValue>Unknown</allowedValue>
  <allowedValue>Possible</allowedValue>
  <allowedValue>Possible (purchase dialogue)</allowedValue>
  <allowedValue>Possible (technical dialogue)</allowedValue>
  <allowedValue>Not possible (no entitlement)</allowedValue>
  <allowedValue>Not possible (technical reason)</allowedValue>
    </allowedValueList>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>DescramblingMessage</name>
  <dataType>string</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>DaylightSaving</name>
  <dataType>boolean</dataType>
    </stateVariable>
- <stateVariable sendEvents="yes">
  <name>CardMessage</name>
  <dataType>string</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>CardVersion</name>
  <dataType>string</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>TimeZone</name>
  <dataType>i4</dataType>
    </stateVariable>
- <stateVariable sendEvents="yes">
  <name>MMIMessage</name>
  <dataType>bin.base64</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>EALocationCode</name>
  <dataType>ui4</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>RatingRegion</name>
  <dataType>ui1</dataType>
```

```
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>Language</name>
  <dataType>string</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>A_ARG_TYPE_MMIDialogNumber</name>
  <dataType>ui1</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>X_CopyProtectionStatus</name>
  <dataType>ui4</dataType>
    </stateVariable>
    </serviceStateTable>
    </scpd>
```

A.3 Diag

```
    <?xml version="1.0" encoding="utf-8" ?>
- <scpd xmlns="urn:schemas-upnp-org:service-1-0">
- <specVersion>
  <major>1</major>
  <minor>0</minor>
    </specVersion>
- <actionList>
- <action>
  <name>GetParameter</name>
- <argumentList>
- <argument>
  <name>Parameter</name>
  <direction>in</direction>
  <relatedStateVariable>A_ARG_TYPE_Parameter</relatedStateVariable>
    </argument>
- <argument>
  <name>Value</name>
  <direction>out</direction>
  <relatedStateVariable>A_ARG_TYPE_Value</relatedStateVariable>
    </argument>
- <argument>
  <name>Volatile</name>
  <direction>out</direction>
  <relatedStateVariable>A_ARG_TYPE_Volatile</relatedStateVariable>
    </argument>
    </argumentList>
    </action>
    </actionList>
- <serviceStateTable>
- <stateVariable sendEvents="no">
  <name>A_ARG_TYPE_Parameter</name>
```

```
  <dataType>string</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>A_ARG_TYPE_Value</name>
  <dataType>string</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>A_ARG_TYPE_Volatile</name>
  <dataType>boolean</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>ParameterList</name>
  <dataType>string</dataType>
    </stateVariable>
    </serviceStateTable>
    </scpd>
```

A.4 Encoder

```
    <?xml version="1.0" encoding="utf-8" ?>
- <scpd xmlns="urn:schemas-upnp-org:service-1-0">
- <specVersion>
  <major>1</major>
  <minor>0</minor>
    </specVersion>
- <actionList>
- <action>
  <name>GetEncoderCapabilities</name>
- <argumentList>
- <argument>
  <name>AudioProfile</name>
  <direction>out</direction>
  <relatedStateVariable>AudioEncoderMethodList</relatedStateVariable>
    </argument>
- <argument>
  <name>VideoProfile</name>
  <direction>out</direction>
  <relatedStateVariable>VideoEncoderMethodList</relatedStateVariable>
    </argument>
    </argumentList>
    </action>
- <action>
  <name>SetEncoderParameters</name>
- <argumentList>
- <argument>
  <name>AudioMode</name>
  <direction>in</direction>
  <relatedStateVariable>AudioBitrateMode</relatedStateVariable>
    </argument>
```

```
- <argument>
  <name>AudioBitrate</name>
  <direction>in</direction>
  <relatedStateVariable>AudioBitrateTarget</relatedStateVariable>
    </argument>
- <argument>
  <name>AudioMethod</name>
  <direction>in</direction>
  <relatedStateVariable>AudioEncoderMethodNumber</relatedStateVariable>
    </argument>
- <argument>
  <name>Mute</name>
  <direction>in</direction>
  <relatedStateVariable>AudioMute</relatedStateVariable>
    </argument>
- <argument>
  <name>FieldToggle</name>
  <direction>in</direction>
  <relatedStateVariable>FieldOrdering</relatedStateVariable>
    </argument>
- <argument>
  <name>SignalSource</name>
  <direction>in</direction>
  <relatedStateVariable>InputSelection</relatedStateVariable>
    </argument>
- <argument>
  <name>NoiseFilter</name>
  <direction>in</direction>
  <relatedStateVariable>NoiseReduction</relatedStateVariable>
    </argument>
- <argument>
  <name>Pulldown</name>
  <direction>in</direction>
  <relatedStateVariable>PulldownSelection</relatedStateVariable>
    </argument>
- <argument>
  <name>SAP</name>
  <direction>in</direction>
  <relatedStateVariable>SAPSelection</relatedStateVariable>
    </argument>
- <argument>
  <name>VideoMode</name>
  <direction>in</direction>
  <relatedStateVariable>VideoBitrateMode</relatedStateVariable>
    </argument>
- <argument>
  <name>VideoBitrate</name>
  <direction>in</direction>
  <relatedStateVariable>VideoBitrateTarget</relatedStateVariable>
```

```
    </argument>
- <argument>
  <name>VideoMethod</name>
  <direction>in</direction>
  <relatedStateVariable>VideoEncoderMethodNumber</relatedStateVariable>
    </argument>
    </argumentList>
    </action>
- <action>
  <name>GetEncoderParameters</name>
- <argumentList>
- <argument>
  <name>CurrentAudioMax</name>
  <direction>out</direction>
  <relatedStateVariable>AudioBitrateMax</relatedStateVariable>
    </argument>
- <argument>
  <name>CurrentAudioMin</name>
  <direction>out</direction>
  <relatedStateVariable>AudioBitrateMin</relatedStateVariable>
    </argument>
- <argument>
  <name>CurrentAudioMode</name>
  <direction>out</direction>
  <relatedStateVariable>AudioBitrateMode</relatedStateVariable>
    </argument>
- <argument>
  <name>CurrentAudioStepping</name>
  <direction>out</direction>
  <relatedStateVariable>AudioBitrateStepping</relatedStateVariable>
    </argument>
- <argument>
  <name>CurrentAudioBitrate</name>
  <direction>out</direction>
  <relatedStateVariable>AudioBitrateTarget</relatedStateVariable>
    </argument>
- <argument>
  <name>CurrentAudioMethod</name>
  <direction>out</direction>
  <relatedStateVariable>AudioEncoderMethodNumber</relatedStateVariable>
    </argument>
- <argument>
  <name>CurrentMuteStatus</name>
  <direction>out</direction>
  <relatedStateVariable>AudioMute</relatedStateVariable>
    </argument>
- <argument>
  <name>CurrentFieldOrder</name>
  <direction>out</direction>
```

```
<relatedStateVariable>FieldOrdering</relatedStateVariable>
</argument>
- <argument>
<name>CurrentSignalSource</name>
<direction>out</direction>
<relatedStateVariable>InputSelection</relatedStateVariable>
</argument>
- <argument>
<name>CurrentNoiseFilter</name>
<direction>out</direction>
<relatedStateVariable>NoiseReduction</relatedStateVariable>
</argument>
- <argument>
<name>CurrentPulldownStatus</name>
<direction>out</direction>
<relatedStateVariable>PulldownDetection</relatedStateVariable>
</argument>
- <argument>
<name>CurrentPulldownSetting</name>
<direction>out</direction>
<relatedStateVariable>PulldownSelection</relatedStateVariable>
</argument>
- <argument>
<name>CurrentSAPStatus</name>
<direction>out</direction>
<relatedStateVariable>SAPDetection</relatedStateVariable>
</argument>
- <argument>
<name>CurrentSAPSetting</name>
<direction>out</direction>
<relatedStateVariable>SAPSelection</relatedStateVariable>
</argument>
- <argument>
<name>CurrentVideoMax</name>
<direction>out</direction>
<relatedStateVariable>VideoBitrateMax</relatedStateVariable>
</argument>
- <argument>
<name>CurrentVideoMin</name>
<direction>out</direction>
<relatedStateVariable>VideoBitrateMin</relatedStateVariable>
</argument>
- <argument>
<name>CurrentVideoMode</name>
<direction>out</direction>
<relatedStateVariable>VideoBitrateMode</relatedStateVariable>
</argument>
- <argument>
<name>CurrentVideoBitrate</name>
```

```
  <direction>out</direction>
  <relatedStateVariable>VideoBitrateTarget</relatedStateVariable>
    </argument>
- <argument>
  <name>CurrentVideoStepping</name>
  <direction>out</direction>
  <relatedStateVariable>VideoBitrateStepping</relatedStateVariable>
    </argument>
- <argument>
  <name>CurrentVideoMethod</name>
  <direction>out</direction>
  <relatedStateVariable>VideoEncoderMethodNumber</relatedStateVariable>
    </argument>
    </argumentList>
    </action>
    </actionList>
- <serviceStateTable>
- <stateVariable sendEvents="no">
  <name>AudioBitrateMax</name>
  <dataType>ui4</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>AudioBitrateMin</name>
  <dataType>ui4</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>AudioBitrateMode</name>
  <dataType>string</dataType>
- <allowedValueList>
  <allowedValue>CBR</allowedValue>
  <allowedValue>AVR</allowedValue>
  <allowedValue>VBR</allowedValue>
    </allowedValueList>
  <defaultValue>CBR</defaultValue>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>AudioBitrateStepping</name>
  <dataType>ui4</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>AudioBitrateTarget</name>
  <dataType>ui4</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>AudioEncoderMethodList</name>
  <dataType>bin.base64</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>AudioEncoderMethodNumber</name>
```

```
  <dataType>ui1</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>AudioMute</name>
  <dataType>boolean</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>FieldOrdering</name>
  <dataType>string</dataType>
- <allowedValueList>
  <allowedValue>Lower</allowedValue>
  <allowedValue>Higher</allowedValue>
    </allowedValueList>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>InputSelection</name>
  <dataType>string</dataType>
- <allowedValueList>
  <allowedValue>Tuner</allowedValue>
  <allowedValue>Aux</allowedValue>
    </allowedValueList>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>NoiseReduction</name>
  <dataType>boolean</dataType>
    </stateVariable>
- <stateVariable sendEvents="yes">
  <name>PulldownDetection</name>
  <dataType>boolean</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>PulldownSelection</name>
  <dataType>boolean</dataType>
    </stateVariable>
- <stateVariable sendEvents="yes">
  <name>SAPDetection</name>
  <dataType>boolean</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>SAPSelection</name>
  <dataType>boolean</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>VideoBitrateMax</name>
  <dataType>ui4</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>VideoBitrateMin</name>
  <dataType>ui4</dataType>
```

```
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>VideoBitrateMode</name>
  <dataType>string</dataType>
- <allowedValueList>
  <allowedValue>CBR</allowedValue>
  <allowedValue>AVR</allowedValue>
  <allowedValue>VBR</allowedValue>
    </allowedValueList>
  <defaultValue>AVR</defaultValue>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>VideoBitrateStepping</name>
  <dataType>ui4</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>VideoBitrateTarget</name>
  <dataType>ui4</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>VideoEncoderMethodList</name>
  <dataType>bin.base64</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>VideoEncoderMethodNumber</name>
  <dataType>ui1</dataType>
    </stateVariable>
    </serviceStateTable>
    </scpd>
```

A.5 FDC

```
    <?xml version="1.0" encoding="utf-8" ?>
- <scpd xmlns="urn:schemas-upnp-org:service-1-0">
- <specVersion>
  <major>1</major>
  <minor>0</minor>
    </specVersion>
- <actionList>
- <action>
  <name>AddPid</name>
- <argumentList>
- <argument>
  <name>AddPidList</name>
  <direction>in</direction>
  <relatedStateVariable>A_ARG_TYPE_PidListChanges</relatedStateVariable>
    </argument>
- <argument>
  <name>RemainingPidFilter</name>
```

```
<direction>out</direction>
<relatedStateVariable>CCFreePidFilter</relatedStateVariable>
</argument>
</argumentList>
</action>
- <action>
<name>GetFDCStatus</name>
- <argumentList>
- <argument>
<name>CurrentBitrate</name>
<direction>out</direction>
<relatedStateVariable>Bitrate</relatedStateVariable>
</argument>
- <argument>
<name>CurrentCarrierLock</name>
<direction>out</direction>
<relatedStateVariable>CarrierLock</relatedStateVariable>
</argument>
- <argument>
<name>CurrentFrequency</name>
<direction>out</direction>
<relatedStateVariable>Frequency</relatedStateVariable>
</argument>
- <argument>
<name>CurrentSpectrumInversion</name>
<direction>out</direction>
<relatedStateVariable>SpectrumInversion</relatedStateVariable>
</argument>
- <argument>
<name>CurrentPidList</name>
<direction>out</direction>
<relatedStateVariable>PidList</relatedStateVariable>
</argument>
</argumentList>
</action>
- <action>
<name>RemovePid</name>
- <argumentList>
- <argument>
<name>RemovePidList</name>
<direction>in</direction>
<relatedStateVariable>A_ARG_TYPE_PidListChanges</relatedStateVariable>
</argument>
</argumentList>
</action>
- <action>
<name>RequestTables</name>
- <argumentList>
- <argument>
```

```
  <name>TID</name>
  <direction>in</direction>
  <relatedStateVariable>A_ARG_TYPE_TID</relatedStateVariable>
    </argument>
    </argumentList>
    </action>
    </actionList>
- <serviceStateTable>
- <stateVariable sendEvents="no">
  <name>A_ARG_TYPE_PidListChanges</name>
  <dataType>string</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>A_ARG_TYPE_TID</name>
  <dataType>string</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>PidList</name>
  <dataType>string</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>Frequency</name>
  <dataType>ui4</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>Bitrate</name>
  <dataType>ui4</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>SpectrumInversion</name>
  <dataType>boolean</dataType>
    </stateVariable>
- <stateVariable sendEvents="yes">
  <name>TableSection</name>
  <dataType>bin.base64</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>CarrierLock</name>
  <dataType>boolean</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>CCFreePidFilter</name>
  <dataType>ui1</dataType>
    </stateVariable>
    </serviceStateTable>
    </scpd>
```

A.6 Mux

```
<?xml version="1.0" encoding="utf-8" ?>
- <scpd xmlns="urn:schemas-upnp-org:service-1-0">
- <specVersion>
  <major>1</major>
  <minor>0</minor>
  </specVersion>
- <actionList>
- <action>
  <name>SetProgram</name>
- <argumentList>
- <argument>
  <name>NewProgram</name>
  <direction>in</direction>
  <relatedStateVariable>ProgramNumber</relatedStateVariable>
  </argument>
  </argumentList>
  </action>
- <action>
  <name>AddPid</name>
- <argumentList>
- <argument>
  <name>AddPidList</name>
  <direction>in</direction>
  <relatedStateVariable>A_ARG_TYPE_PidListChanges</relatedStateVariable>
  </argument>
  </argumentList>
  </action>
- <action>
  <name>RemovePid</name>
- <argumentList>
- <argument>
  <name>RemovePidList</name>
  <direction>in</direction>
  <relatedStateVariable>A_ARG_TYPE_PidListChanges</relatedStateVariable>
  </argument>
  </argumentList>
  </action>
  </actionList>
- <serviceStateTable>
- <stateVariable sendEvents="no">
  <name>ProgramNumber</name>
  <dataType>ui2</dataType>
  </stateVariable>
- <stateVariable sendEvents="no">
  <name>PIDList</name>
  <dataType>string</dataType>
  </stateVariable>
- <stateVariable sendEvents="no">
```

```
<name>A_ARG_TYPE_PidListChanges</name>
<dataType>string</dataType>
</stateVariable>
</serviceStateTable>
</scpd>
```

A.7 Security

```
<?xml version="1.0" encoding="utf-8" ?>
- <scpd xmlns="urn:schemas-upnp-org:service-1-0">
- <specVersion>
<major>1</major>
<minor>0</minor>
</specVersion>
- <actionList>
- <action>
<name>SetDRM</name>
- <argumentList>
- <argument>
<name>NewDrm</name>
<direction>in</direction>
<relatedStateVariable>DrmUUID</relatedStateVariable>
</argument>
</argumentList>
</action>
</actionList>
- <serviceStateTable>
- <stateVariable sendEvents="yes">
<name>DrmPairingStatus</name>
<dataType>string</dataType>
- <allowedValueList>
<allowedValue>Green</allowedValue>
<allowedValue>Orange</allowedValue>
<allowedValue>Red</allowedValue>
</allowedValueList>
</stateVariable>
- <stateVariable sendEvents="no">
<name>DrmUUID</name>
<dataType>string</dataType>
</stateVariable>
- <stateVariable sendEvents="no">
<name>DrmUUIDList</name>
<dataType>string</dataType>
</stateVariable>
- <stateVariable sendEvents="no">
<name>HmsAssociationList</name>
<dataType>bin.base64</dataType>
</stateVariable>
</serviceStateTable>
```

```
</scpd>
```

A.8 Tuner

```
<?xml version="1.0" encoding="utf-8" ?>
- <scpd xmlns="urn:schemas-upnp-org:service-1-0">
- <specVersion>
  <major>1</major>
  <minor>0</minor>
</specVersion>
- <actionList>
- <action>
  <name>GetTunerParameters</name>
- <argumentList>
- <argument>
  <name>CurrentCarrierLock</name>
  <direction>out</direction>
  <relatedStateVariable>CarrierLock</relatedStateVariable>
</argument>
- <argument>
  <name>CurrentFrequency</name>
  <direction>out</direction>
  <relatedStateVariable>Frequency</relatedStateVariable>
</argument>
- <argument>
  <name>CurrentModulation</name>
  <direction>out</direction>
  <relatedStateVariable>Modulation</relatedStateVariable>
</argument>
- <argument>
  <name>CurrentPCRLock</name>
  <direction>out</direction>
  <relatedStateVariable>PCRLock</relatedStateVariable>
</argument>
- <argument>
  <name>CurrentSignalLevel</name>
  <direction>out</direction>
  <relatedStateVariable>SignalLevel</relatedStateVariable>
</argument>
- <argument>
  <name>CurrentSNR</name>
  <direction>out</direction>
  <relatedStateVariable>SNR</relatedStateVariable>
</argument>
</argumentList>
</action>
- <action>
  <name>SeekCancel</name>
</action>
```

```
- <action>
  <name>SeekSignal</name>
- <argumentList>
- <argument>
  <name>StartFrequency</name>
  <direction>in</direction>
  <relatedStateVariable>Frequency</relatedStateVariable>
    </argument>
- <argument>
  <name>StopFrequency</name>
  <direction>in</direction>
  <relatedStateVariable>A_ARG_TYPE_StopFrequency</relatedStateVariable>
    </argument>
- <argument>
  <name>ModulationList</name>
  <direction>in</direction>
  <relatedStateVariable>A_ARG_TYPE_SeekModulation</relatedStateVariable>
    </argument>
- <argument>
  <name>Increment</name>
  <direction>in</direction>
  <relatedStateVariable>A_ARG_TYPE_Increment</relatedStateVariable>
    </argument>
- <argument>
  <name>SeekUp</name>
  <direction>in</direction>
  <relatedStateVariable>A_ARG_TYPE_SeekUp</relatedStateVariable>
    </argument>
- <argument>
  <name>TimeToBlock</name>
  <direction>in</direction>
  <relatedStateVariable>A_ARG_TYPE_TimeToBlock</relatedStateVariable>
    </argument>
    </argumentList>
    </action>
- <action>
  <name>SetTunerParameters</name>
- <argumentList>
- <argument>
  <name>NewFrequency</name>
  <direction>in</direction>
  <relatedStateVariable>Frequency</relatedStateVariable>
    </argument>
- <argument>
  <name>NewModulation</name>
  <direction>in</direction>
  <relatedStateVariable>Modulation</relatedStateVariable>
    </argument>
- <argument>
```

```
<name>CurrentFrequency</name>
<direction>out</direction>
<relatedStateVariable>Frequency</relatedStateVariable>
</argument>
- <argument>
<name>CurrentModulation</name>
<direction>out</direction>
<relatedStateVariable>Modulation</relatedStateVariable>
</argument>
- <argument>
<name>PCRLockStatus</name>
<direction>out</direction>
<relatedStateVariable>PCRLock</relatedStateVariable>
</argument>
</argumentList>
</action>
</actionList>
- <serviceStateTable>
- <stateVariable sendEvents="no">
<name>A_ARG_TYPE_Increment</name>
<dataType>ui4</dataType>
</stateVariable>
- <stateVariable sendEvents="no">
<name>A_ARG_TYPE_SeekModulation</name>
<dataType>string</dataType>
</stateVariable>
- <stateVariable sendEvents="no">
<name>A_ARG_TYPE_StopFrequency</name>
<dataType>ui4</dataType>
</stateVariable>
- <stateVariable sendEvents="yes">
<name>Seeking</name>
<dataType>boolean</dataType>
</stateVariable>
- <stateVariable sendEvents="yes">
<name>PCRLock</name>
<dataType>boolean</dataType>
</stateVariable>
- <stateVariable sendEvents="no">
<name>A_ARG_TYPE_SeekUp</name>
<dataType>boolean</dataType>
</stateVariable>
- <stateVariable sendEvents="no">
<name>SignalLevel</name>
<dataType>i4</dataType>
</stateVariable>
- <stateVariable sendEvents="no">
<name>SNR</name>
<dataType>ui4</dataType>
```

```
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>Frequency</name>
  <dataType>ui4</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>ModulationList</name>
  <dataType>string</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>Modulation</name>
  <dataType>string</dataType>
- <allowedValueList>
  <allowedValue>QAM64</allowedValue>
  <allowedValue>QAM-64</allowedValue>
  <allowedValue>QAM256</allowedValue>
  <allowedValue>QAM-256</allowedValue>
  <allowedValue>UNKNOWN</allowedValue>
  <allowedValue>8VSB</allowedValue>
  <allowedValue>8-VSB</allowedValue>
  <allowedValue>NTSC</allowedValue>
  <allowedValue>NTSC-M</allowedValue>
    </allowedValueList>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>A_ARG_TYPE_TimeToBlock</name>
  <dataType>ui2</dataType>
    </stateVariable>
- <stateVariable sendEvents="no">
  <name>CarrierLock</name>
  <dataType>boolean</dataType>
    </stateVariable>
    </serviceStateTable>
    </scpd>
```

Appendix I Content Protection Implementation (Informative)

The following is an informative example of one means to provide some aspects of a content protection method for the DRI.

I.1 Encryption Control

Figure I–2 below depicts a state diagram for control of content scrambling on the DRI.

- The following terms are used to navigate the state diagram.
  - **scr** – This boolean variable is set to "yes" if the current TS Packet needs to be scrambled, "no" otherwise.
  - **key_sync** – This boolean variable is set to "yes" if the DRIT is renewing the AES key, "no" otherwise.
  - **base_counter** – This sixty four (64) bit field is uniquely defined by the DRIT across its lifetime, as shown in Figure I–1.

    Where:

    PID (13-bit) – The PID value of the selected elementary stream.

    Section_counter (51-bit) - A cyclic counter incremented for each no-to-yes transition of the **scr** state variable.

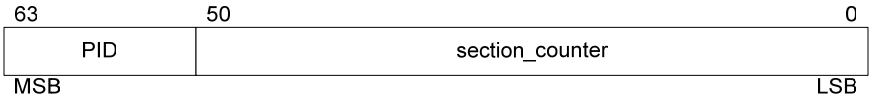

Figure I–1 - AES base_counter

  - **minor_counter** – A sixty four (64) bit counter incremented for each block of 16 scrambled bytes.
  - **i** – A four (4) bit counter incremented for each scrambled byte.
  - **scramble16** - = AES_{KEY} [base_counter | minor_counter], where AES_K[D] - Denotes that data, D, is encrypted with key, K, using the AES algorithm. Note that the AES mode of operation is imposed by the DRM.

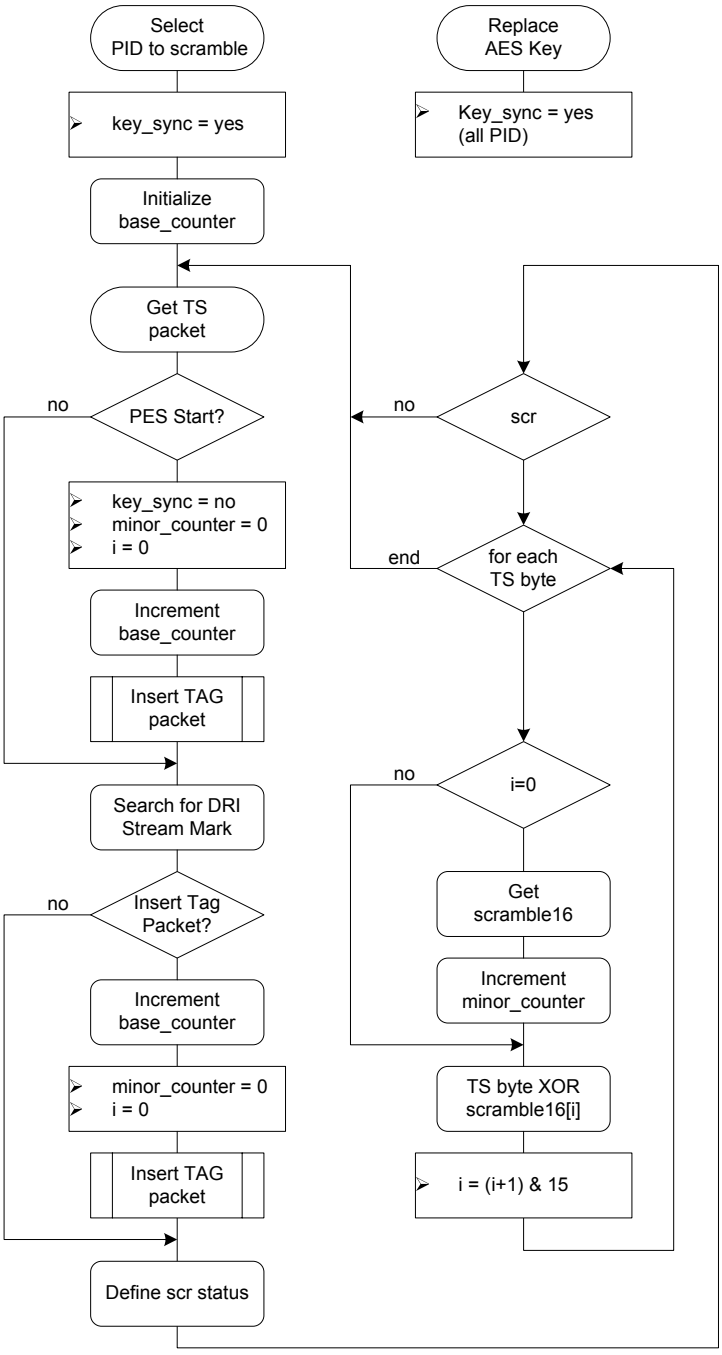

Figure I–2 - AES State Diagram

Following comments apply:

- After the ***Replace AES Key*** event occurs, the DRIT stops immediately scrambling all PID until it resynchronizes with each PES component. This transition guarantees that all PID from the same program are scrambled with the same key.
- During the action ***Define scr status***, the DRIT sets, for each received TS Packet, the **scr** state variable to "no" if any of the following conditions apply:

  Key_sync = yes

  The TS Packet includes whole or part of a PES header

The TS Packet includes whole or part containing other than the first 3 bytes, of one or more of the DRI Stream Marks[1] listed in Table I–1. A DRI Stream Mark is composed of an MPEG Start code and its following data payload.

Table I–1 - DRI Stream Mark

Stream Mark	Start Code	Byte Sequence	Maximum Data Payload Length (Byte)
Sequence header	B3	00 00 01 B3	12
GOP header0	B8	00 00 01 B8	8
Picture header	00	00 00 01 00	6
Private data	B2	00 00 01 B2	107

Note: The content protection layer of the DRI Security is a Transport Stream scrambling protocol, which means that each TS Packet can only be scrambled or in-the-clear, but not partially scrambled. Additional requirements are placed to the DRIT to leave few packets in the clear (usually 2-3 packets per video frame) to facilitate the indexing of the video stream by the connected device without the need for descrambling.

I.2 Content Identification

The content identification is achieved by periodically inserting into each scrambled elementary stream, a single TS Packet with some private DRM data field (TAG packet), which will further allow the DRM system to retrieve the matching DRM license when the content is consumed.

The TAG packet is an adaptation-field only packet compliant with the [MPEG-S] standard. As such, it doesn't require the DRIT to re-adjust the continuity counter of the following TS Packets.

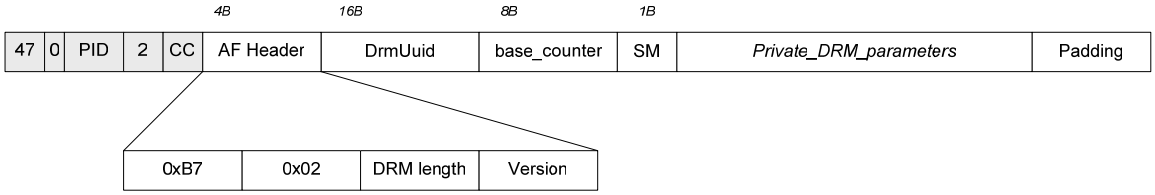

Figure I–3 - TAG Packet

The *adaptation_field_control* bits are set to 10b (Adaptation field only, no payload), so there is no requirement to increment the continuity counter.

AF Header consists of four bytes to be compliant with MPEG specification.

1st Byte = Adaptation Field length

2nd Byte = Adaptation Field presence flag (Private data = 0x02)

3rd Byte = Private data length

4th Byte = Version number (currently 0x00)

DrmUuid identifies the DRM system that relates to the TAG packet.

Base_counter resynchronizes the AES counter for the following encrypted packets.

[1] If a stream mark is spanned across two packets but less than three (3) bytes are in the first packet, it is not possible at the reception of the first packet to know if it SHALL be left in the clear or not. As a result this first packet SHALL always be left encrypted.

SM byte (Stream Mark) indicates that the following packet includes the beginning of a DRI Protocol Stream Mark, from which the first few bytes may be missing.

- SM = 0 – Next packet carries the beginning of a PES header or an entire PES header
- SM = 1 – Next packet includes the beginning of a stream mark
- SM = 2 – Next packet includes the beginning of a stream mark, from which the first byte 00 is missing
- SM = 3 – Next packet includes the beginning of a stream mark, from which the first two bytes 00 00 are missing
- SM = 4 – Next packet includes the beginning of a stream mark, from which the first three bytes 00 00 01 are missing
- SM = other – reserved

Private_DRM_parameter – Privately defined parameter by the DRM system.

The packet is padded with 0xFF.

- **Case A**: A TAG packet is inserted in front of a PES header
- **Case B:** A TAG packet is inserted in front of a packet containing the totality of DRI Protocol Stream Mark
- **Case C:** A TAG packet is inserted in front of two packets containing a DRI Protocol spanned stream mark, where the first packet contains at least 4 bytes.
- **Case D:** A TAG packet is inserted between two packets containing a spanned DRI Protocol Stream Mark, where the first packet contains less than 4 bytes.

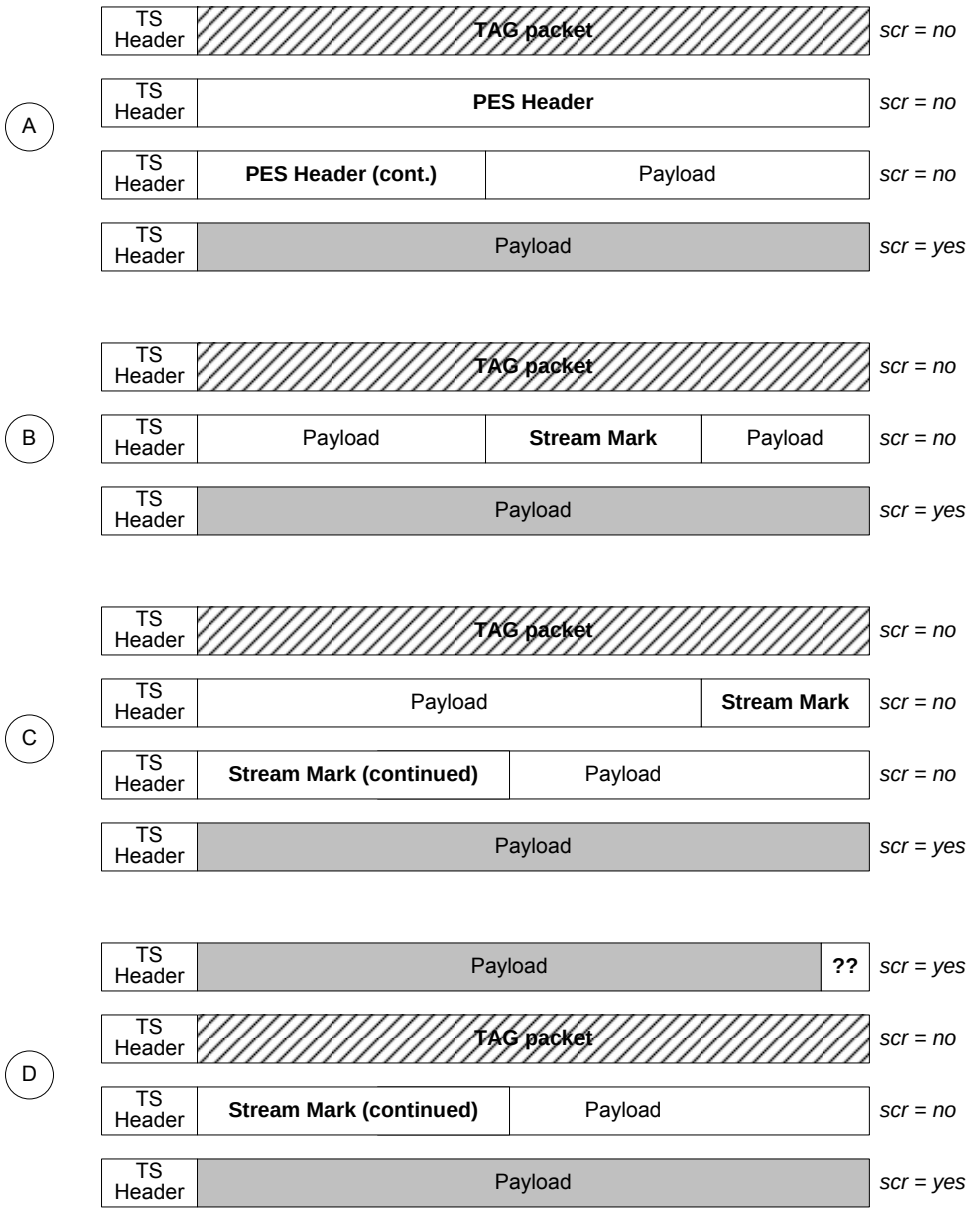

Figure I–4 - Partial Scrambling

Note: This content identification algorithm generates at least one re-synchronization point of the AES-based counter per video frame. As such, it provides a stream that is tolerant to packet losses across the DRI connection.