Advanced Access Content System (AACS)

HD DVD and DVD Pre-recorded Book

Intel Corporation
International Business Machines Corporation
Matsushita Electric Industrial Co., Ltd.
Microsoft Corporation
Sony Corporation
Toshiba Corporation
The Walt Disney Company
Warner Bros.

Revision 0.912
August 15, 2006

This page is intentionally left blank.

Preface

Notice

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. IBM, Intel, Matsushita Electric Industrial Co., Ltd., Microsoft Corporation, Sony Corporation, Toshiba Corporation, The Walt Disney Company and Warner Bros. disclaim all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

This document is subject to change under applicable license provisions.

Copyright © 2005-2006 by Intel Corporation, International Business Machines Corporation, Matsushita Electric Industrial Co., Ltd., Microsoft Corporation, Sony Corporation, Toshiba Corporation, The Walt Disney Company, and Warner Bros. Third-party brands and names are the property of their respective owners.

Intellectual Property

Implementation of this specification requires a license from AACS LA.

Contact Information

Please address inquiries, feedback, and licensing requests to AACS LA:

- Licensing inquiries and requests should be addressed to licensing@aacsla.com.
- Feedback on this specification should be addressed to comment@aacsla.com.

The URL for the AACS LA web site is http://www.aacsla.com.

This page is intentionally left blank.

Table of Contents

PREFACE ..... **III**

Notice ..... **iii**

Intellectual Property ..... **iii**

Contact Information ..... **iii**

1 INTRODUCTION **11**

1.1 Purpose and Scope **11**

1.2 Overview **12**
1.2.1 AACS Mode 12
1.2.2 Overview 12

1.3 Organization of this Document **21**

1.4 References **21**

1.5 Notation **22**

1.6 Terminology **22**

1.7 Abbreviations and Acronyms **23**

2 AACS COMPONENTS IN LEAD-IN AREA AND BURST CUTTING AREA 25

2.1 Introduction **25**

2.2 Key Conversion Data **27**

2.3 AACS Components on HD DVD-ROM **27**
2.3.1 Control Data 28
2.3.2 Media Key Block 29
2.3.3 Volume Identifier 31
2.3.4 Pre-recorded Media Serial Number 32

2.4 AACS Components on DVD-ROM **33**
2.4.1 Control Data 33
2.4.2 Media Key Block 37
2.4.3 Volume Identifier 38
2.4.4 Pre-recorded Media Serial Number 39

3 AACS COMPONENTS IN DATA AREA **41**

3.1 Introduction **41**

3.2 Media Key Block 41

3.3 Sequence Key Block and Segment Key Files 42

3.4 Title Key File 49

3.5 Title Usage File 55

3.6 Content Certificate 69

3.7 Content Hash 71
3.7.1 Content Hash Table #1 71
3.7.2 Content Hash Table #2 72

3.8 Content Revocation List 75

3.9 Boot Sequence for Disc Application 75

3.10 Backups 76

4 PROTECTION OF AN HD DVD-VIDEO CONTENT ON A MEDIUM 79

4.1 Introduction 79

4.2 Stored data values in CPI field 81

4.3 Protection format for EVOB 86
4.3.1 Pack Types 87
4.3.2 Pack Encryption 88
4.3.3 Content Hash Check for EVOBs 90
4.3.4 Pack Decryption 91

4.4 Protection Formats for Advanced Resources 92
4.4.1 Protection Types for ARFs 92
4.4.2 Five Encapsulation Formats 94
4.4.3 Verification of Hash and/or Decryption 101

5 DOWNLOADING, STREAMING AND ONLINE-ENABLING 103

5.1 Introduction 103

5.2 Binding 104
5.2.1 Binding and Permissions 104
5.2.2 APIs used to Obtain the Binding Information 108

5.3 Managed Copy 109

5.4 APIs 114
5.4.1 API for Streaming 114
5.4.2 APIs for Online-Enabling 114
5.4.3 Examples of Usage Scenarios of Online-Enabling 115

5.5 AACS Object 118

6 PROTECTION OF HD DVD-VIDEO CONTENTS IN PERSISTENT STORAGES 123

6.1 Introduction 123

6.2 HD DVD-Video Contents in Persistent Storages 123
6.2.1 Contents in Persistent Storages Used by a Disc Application 124
6.2.2 Playlist Update 125
6.2.2.1 Data in a Persistent Storage 125
6.2.2.2 Boot Sequence 128
6.2.2.3 Content Hash Check 130
6.2.3 APIs for Loading and Saving 131

6.3 Protection of Directory Name for a Content Provider in Persistent Storages 133

7 SEQUENCE KEY 137

7.1 Introduction 137

7.2 Sequence Key for Standard VTS 138
7.2.1 Entering into a Sequence Key Section 139
7.2.2 Transitions among Interleaved Units in a Sequence Key Section 140
7.2.3 Getting Out of a Sequence Key Section 142

7.3 Sequence Key for Advanced VTS 142
7.3.1 Entering into a Sequence Key Section 144
7.3.2 Transition among Interleaved Units in a Sequence Key Section 144
7.3.3 Getting Out of a Sequence Key Section 145

A SCHEMA FOR MANAGED COPY MANIFEST FILE 147

B MANAGED COPY PERMISSION XML SCHEMA FOR HD DVD AND DVD155

C MANAGED COPY WEB SERVICE DESCRIPTION FOR HD DVD AND DVD 157

D ADDITIONAL REQUIREMENT FOR CARRIAGE OF SRM 163

D.1 Introduction 163

D.2 SRM (System Renewability Message) 163
D.2.1 SRM for DTCP 163
D.2.2 SRM for HDCP 163

E MCMAACS OBJECT FOR MANAGED COPY MACHINE 165

List of Figures

Figure 1-1: An Example of an HD DVD-Video Player System Model 17
Figure 1-2: A Rough Sketch of Relationship among the Components on an AACS Disc 19
Figure 1-3: A Rough Sketch of Relationship among the Components in Persistent Storages 20
Figure 2-1: Physical Layout of AACS Components on an (HD) DVD-ROM 26
Figure 2-2: Structure of BCA and Lead-in Area of an HD DVD ROM Medium 28
Figure 2-3: Structure of a Control Data Zone 28
Figure 2-4: Structure of a Data Segment in a Control Data Zone 29
Figure 2-5: Example of storing MKB on Lead-in Area of an HD DVD ROM medium 30
Figure 2-6: Structure of BCA and Lead-in Area of a DVD ROM Medium 33
Figure 2-7: Structure of the Control Data Zone 34
Figure 2-8: Structure of an ECC block in Control Data Zone 34
Figure 2-9: Data Frame Configuration for a DVD ROM medium 35
Figure 2-10: Example of storing MKB on Lead-in Area of a DVD-ROM medium 38
Figure 3-1: An Example of Coverage of Usage Rules 56
Figure 3-2: The Relationship between a BIFO and Binding MAC field and a Title Key 60
Figure 4-1: Protection Targets of an HD DVD-Video Content on an HD DVD-Video ROM Medium 80
Figure 4-2: An Example of Title Key assignment to EVOBs 86
Figure 5-1: Key Retrieval Flows for Medium Binding and Content Binding 105
Figure 5-2: Key Retrieval Flows for Medium & Device Binding and Content & Device Binding 106
Figure 5-3: Key Retrieval Flows for Content & Temporary Nonce Binding 107
Figure 5-4: Image of an Online-Enabling Scenario 116
Figure 5-5: Image of an Online-Enabling Scenario 116
Figure 6-1: Directory Structure for Persistent Storage 136
Figure 7-1: An Image of Sequence Key Section 139

List of Tables

Table 2-1: Copyright Protection Information in an HD DVD-Video ROM medium 29
Table 2-2: Volume Identifier Format for an AACS-Protected HD DVD-Video ROM medium 31
Table 2-3: Format of BCA Record Containing a Volume Identifier 31
Table 2-4: Format of BCA Record Containing a Pre-recorded Media Serial Number 32
Table 2-5: Copyright Protection Information in a DVD-Video ROM medium 34
Table 2-6: CPR_MAI Format in Content Provider Information Sectors 36
Table 2-7: CPR_MAI in Data Area 36
Table 2-8: Volume Identifier Format for an AACS-Protected DVD-Video ROM medium 38
Table 3-1: Format of SKBF 43
Table 3-2: Format of Segment Key File 45
Table 3-3: Format of SKG Field 46
Table 3-4: Format of Segment Key Unit Field 48
Table 3-5: Format of Title Key File 51
Table 3-6: Format of Binding Information 54
Table 3-7: Format for Title Usage File 57
Table 3-8: Format of BURS 59
Table 3-9: Format of Usage Rule Set 61
Table 3-10: Format of the Usage Rule of CCI for Update 64
Table 3-11: Usage Rule of Time-Based Conditions 65
Table 3-12: Format of REL Usage Rule 67
Table 3-13: Format of Output Control Bits 68
Table 3-14: Format of Content Certificate on an AACS Disc for HD DVD-Video 69
Table 3-15: Format of Content Hash Table #1 71
Table 3-16: Format of Content Hash Table #2 on an AACS Disc 73
Table 4-1: Stored data values in Content Protection Information (CPI) 81
Table 4-2: Stored data value in Key Management Information (KMI) 82
Table 4-3: Stored data value in Content Hash Management Information (CHMI) 83
Table 4-4: Stored data value in Usage Rule Management Information (URMI) 83
Table 4-5: Stored data value in CCI_SS 84
Table 4-6: Stored data value in CCI 85
Table 4-7: Encrypted Pack Format 88
Table 4-8: Encrypted Pack Format for HL_PCK 90
Table 4-9: Encapsulation Format for Encryption 94
Table 4-10: Encapsulation Format for Encryption and Hash 94

Table 4-11: Encapsulation Format for MAC 97
Table 4-12: Encapsulation Format for Hash 98
Table 4-13: Encapsulation Format for Non-Protected Advanced Element 100
Table 5-1: Required Information for Each Binding 109
Table 6-1: Format of Content Certificate File in a Persistent Storage 126
Table 6-2: Format of Directory Key File 134
Table 7-1: Bit Assignment for the Reserved Field of C_PBI 139
Table 7-2: SML_SEQI 141
Table 7-3: SML_SEQ_Cn_DSTA 141
Table 7-4: TMAP_GI for AACS-Compliant Disc 143
Table 7-5: TMAP_TY for AACS-Compliant Disc 143

Chapter 1
Introduction

1 Introduction

1.1 Purpose and Scope

The Advanced Access Content System (AACS) specification defines an advanced, robust and renewable method for protecting entertainment content, including high-definition audiovisual content. The specification is organized into several “books”. The *Introduction and Common Cryptographic Elements* book defines cryptographic procedures that are common among the various defined uses of the protection system. The *Pre-recorded Video Book* specifies additional details for using the system to protect audiovisual content distributed on pre-recorded (read-only) storage media.

This Book (the *AACS HD DVD and DVD Pre-recorded Book*) defines detailed adaptation of the AACS technical elements to the *HD DVD-Video Specifications* defined in the DVD Forum. Therefore, this book defines copyright protection of the HD DVD-Video contents on the HD DVD and DVD Pre-recorded media. An HD DVD-Video Player has capability of connecting to the Internet, and it has capability of managing Persistent Storages as non-volatile storages external to the optical disc. The Player behavior can be controlled by XML Documents and ECMAScript Codes provided by the content provider. These new capabilities of an HD DVD-Video Player, together with capability of playing back high-definition audiovisual and audio-only contents, could bring a brand-new user experience. This book provides a content protection scheme for Persistent Storages as well as an HD DVD-ROM or DVD-ROM medium itself.

Use of this specification and access to the intellectual property and the cryptographic materials required to implement it will be subjects of a license. A license authority referred to as AACS LA LLC (hereafter referred to as AACS LA) is responsible for establishing and administering the content protection system based, in part, on this specification.

1.2 Overview

1.2.1 AACS Mode

An AACS-Compliant Player has two modes: AACS Mode and Non-AACS Mode. The two modes are distinguished by the method of assignment of provider directory and by the boot sequence. The details are described in Section 6.3. In AACS Mode, behavior of some APIs is different from that defined in the *HD DVD-Video Specifications*. The description is seen in 6.2.3. And, in AACS Mode, some AACS APIs are introduced. They are described in 5.5. Above all, in AACS Mode, it is assumed that the content data may have different formats. For instance, an EVOB may be encrypted and an XML file shall be encapsulated.

A Player shall enter into AACS Mode before executing its boot sequence if and only if it decides that a Disc to be played back is an AACS Disc. Once a Player enters into Non-AACS Mode, the Player is not allowed to enter into AACS Mode before one of the following conditions hold:

- The Disc is ejected.
- The Player loses power.
- The boot sequence for an AACS Disc starts.
  - The boot sequence is described in 6.2.2.2.

A Player shall decide that a Disc to be played back is an AACS Disc if the AACS-Compliant drive for the Player is able to read the PMSN or if the drive is able to read the Volume ID. The following description in this book is applicable in AACS mode unless otherwise stated.

1.2.2 Overview

The following is the general principles of data protection for the content and the AACS components concerning an AACS-Compliant (HD) DVD-Video:

On an AACS Disc:

a. Title Usage Files *shall* be protected by means of Content Hashing and MAC.

b. Title Key Files *shall* be protected by means of encryption and MAC

c. All the P/S-EVOBs *shall* be protected by means of Content Hashing. They *may* also be protected by packet-based encryption.

d. Advanced Elements such as images, animations, fonts, etc. *may* be protected by encapsulation in Encapsulation Format for MAC or in Encapsulation Format for Encryption.

- Effect audios (WAV files) *must not* be encapsulated in Encapsulation Format for MAC.
- Every Advanced Element, which is to be reproduced as a part of contents, shall be encapsulated. See 4.4 for the details. Even a Non-Protected Advanced Element *shall* be encapsulated in Encapsulation Format for Non-Protected Advanced Element. On the other hand, for instance, a JPG file for a PC wallpaper or a JPG file used by a Player for showing an explanation of a Provider Directory should not be encapsulated.

e. All the XML Document files *shall* be protected by encapsulation in Encapsulation Format for Hash.
   - Exception: An XML Document for Advanced Subtitles *may* be protected by encapsulation in Encapsulation Format for Encryption and Hash if the content author wants it to be encrypted.
   - Exception: Managed Copy Manifest named "MNGCPY_MANIFEST.XML" *must not* be encapsulated though it *shall* be protected by means of Content Hashing.

f. All the ECMAScript files *shall* be protected by encapsulation in Encapsulation Format for Hash.
   - Exception: An ECMAScript file *may* be protected by encapsulation in Encapsulation Format for Encryption and Hash if the content author wants it to be encrypted.

In Persistent Storages:

a. Title Usage Files *shall* be protected by means of Content Hashing and MAC.
   - A Title Usage File may override Usage Rules for a P/S-EVOB on a Disc.

b. Title Key Files *shall* be protected by means of encryption and MAC.

c. S-EVOBs *may* be protected by packet-based encryption.
   - No content hash is required for an S-EVOB in a Persistent Storage.

d. Advanced Elements such as images, animations, effect audios and fonts *may* be protected by encapsulation in Encapsulation Format for MAC or in Encapsulation Format for Encryption.
   - A captured image *must not* be encrypted. It *may*, however, be protected by encapsulation in Encapsulation Format for MAC.
   - Effect audios (WAV files) *must not* be encapsulated in Encapsulation Format for MAC.
   - Every Advanced Element, which is to be reproduced as a part of contents, shall be encapsulated. See 4.4 for the details. Non-Protected Advanced Element *shall* be encapsulated in Encapsulation Format for Non-Protected Advanced Element. On the other hand, for

instance, a JPG file for a PC wallpaper or a JPG file used by a Player for showing an explanation of a Provider Directory should not be encapsulated.

e. All the XML Documents *shall* be protected by encapsulation in Encapsulation Format for MAC.
   - An application-generated XML Document shall also be encapsulated in Encapsulation Format for MAC.
     i) It is automatically encapsulated by the APIs for saving XML Documents.
   - Exception: An XML Document for Advanced Subtitles *may* be protected by encapsulation in Encapsulation Format for Encryption if the content author wants it to be encrypted.

f. All the ECMAScript codes *shall* be encapsulated in Encapsulation Format for MAC.
   - Exception: An ECMAScript file *may* be protected by encapsulation in Encapsulation Format for Encryption if the content author wants it to be encrypted.

The protection methods are different between contents on a Disc and those in Persistent Storages. On an AACS Disc, XML Document files and ECMAScript files, which govern navigation of contents playback and define behavior of a Player, shall be strictly protected by means of Content Hashing. Those hashes are compiled into a Content Hash Table and the hash of the Content Hash Table is contained in the Content Certificate which is signed by the AACS LA. And the content hashes of P/S-EVOBs on a Disc are also compiled into another Content Hash Table whose hash is contained in the signed Content Certificate.

On the other hand, in a Persistent Storage, XML Document files and ECMAScript files shall be protected by means of a MAC or encryption. There are no Content Hash Table and no Content Certificate for contents in Persistent Storages though there is a Content Certificate file for the Title Usage File. Data in a Persistent Storage which are associated with a Disc in an AACS-Compliant Player are stored in a directory whose name is known only by the Player which plays the concerned Disc. No other Disc than the concerned Disc is able to alter or manipulate those associated data in the Persistent Storage. A content provider's directory in a Persistent Storage is strictly protected from illegitimate access from other AACS-Compliant/Non-AACS-Compliant Discs played back on an AACS-Compliant Player:

g. Configuration File "DISCID.DAT" and Directory Key File *shall* be protected by means of Content Hashing.

The Configuration File has the data field called "Provider ID". Provider ID is cryptographically modified by the Directory Key in the Directory Key File in order to yield the name of the Provider Directory. These data are strictly protected by means of Content Hashing so that segmentation among content providers will never be violated.

Data to be stored in a Provider's directory in a Persistent Storage shall be created as described above. They shall have no content hash except the Title Usage Files. A content to be protected shall be encapsulated. There are three sources for the data: the first is an AACS Disc, the second is a Network Server and the third is application-generated data. Data from the former two sources shall be, in the first place, encapsulated in the appropriate formats and stored on the Disc or prepared in the Server. Note that All the Advanced Elements to be displayed as a content shall be encapsulated in one of the *five* encapsulation formats.

The *HD DVD-Video Specifications* defines the terms "Advanced Element" and "Advanced Navigation". Advanced Navigation contains XML Documents and ECMAScript Codes which define the architecture of content playback such as playback sequences, resource declarations, descriptions of menu, actions for menu buttons, etc. Advanced Navigation also contains XML Documents for Advanced Subtitle. The term "Advanced Element" denotes data used by Advanced Navigation. Advanced Elements contain still images, animations, fonts, etc. In this Adaptation Book, the term "Advanced Resources" is sometimes used to mean the sum of the data set "Advanced Navigation" and the data set "Advanced Element".

There are *five* kinds of encapsulation formats for Advanced Resource Files. One is the *Encapsulation Format for Hash*. Integrity of data encapsulated in this format is guaranteed by means of Content Hashing and Content Certificate. The second encapsulation format is *Encapsulation Format for Encryption and Hash*. This format is used for data which the content author wants to be encrypted when the data shall be protected by means of Content Hashing. These two formats serve for integrity check of XML Documents and ECMAScript Codes on an AACS Disc. The AACS signature is involved in the protection of them, which guarantees higher integrity.

The third format is *Encapsulation Format for MAC*. Integrity of data encapsulated in this format is guaranteed by means of MAC which is calculated using one of the Title Keys. The fourth format is *Encapsulation Format for Encryption*, which is for data to be protected by encryption. These two formats serve for integrity check of XML Documents and ECMAScript Codes in Persistent Storages. The fifth format is *Encapsulation Format for Non-Protected Advanced Element*. It is used to prevent alternation of a protected file with a non-protected file. In the format, only the filename of an Advanced Element is encrypted and protected.

An application may save a DOM Document in XML Parser into File Cache, into a Persistent Storage or into a Network Server, in the form of an XML Document. In AACS Mode, an XML Document shall be encapsulated in Encapsulation Format for MAC. Therefore, the API, XMLParser.write(), for saving a DOM Document shall calculate the MAC value of the XML Document to save. The MAC value is calculated using a Title Key which is indicated by the API, AACS.setMACKey().

There is an HD DVD-Video API, FileIO.openTextFile(), for reading an XML Document or ECMAScript Codes from File Cache or a Persistent Storage into a buffer in Script Engine. This API does not perform any AACS operation such as MAC verification. That is, an XML Document, for instance, read by openTextFile() is not MAC-verified and is not de-capsulated.

An AACS-Compliant Player provides two APIs for capturing image with MAC. One is for the Main Video Plane and the other is for the Graphics Plane. A captured image is saved in Encapsulation Format for MAC. See 6.2.3 for the APIs for loading/saving.

An HD DVD-Video disc may contain program streams, called P/S-EVOBs, of audiovisual data or audio-only data. As stated above, those EVOB files may be protected by encryption. Encryption shall be performed on a packet basis. Some packets may be encrypted and others not. The AACS protection format for the EVOB file is defined in Chapter 4 of this Book.

Figure 1-1 depicts an example of a system model for an HD DVD-Video Player with the AACS modules. The pink portions of the figure denote the AACS functions such as decryption or data integrity check. A P-EVOB in a Primary Video Set may contain ADV_PCK which is a container for Advanced Resources. Advanced Resources are packetized and multiplexed into a P-EVOB. A Secondary Video Set shall not contain a P-EVOB, but it may contain an S-EVOB, where an S-EVOB has no ADV_PCK.

In Figure 1-1, packetized Advanced Resources go from DEMUX to File Cache Manger. Each Advanced Resource File in File Cache is no longer packetized but is in the composed form. No cryptographic operation is performed in the DEMUX and the file transfer. Therefore, Advanced Resource Files shall be protected by encapsulation before they are multiplexed as ADV_PCKs into a P-EVOB. This is an important point at the authoring stage or at the AACS encryption stage of an HD DVD-Video Disc creation.

From a Disc, Persistent Storages or Network Servers, Advanced Resources come into File Cache. These Advanced Resource Files shall be encapsulated in the first place. The data transfer to File Cache from the Disc or from a Persistent Storage is just bit-by-bit copy which involves no cryptographic operation. This means that those Advanced Resource Files shall reside, in the first place, in the encapsulated format on the Disc or in a Persistent Storage. The data transfer to File Cache from a Network Server may involve a cryptographic operation such as HTTPS for protection of the data transfer. It is, however, considered as bit-by-bit copy of data from the server to File Cache in the aspect of the AACS content protection. This means that a Network Server shall prepare encapsulated Advanced Resource Files.

Streaming of audiovisual data or audio-only data comes from a Network Server or a Persistent Storage. They are S-EVOBs and may be encrypted on a packet basis. Streamed data are decrypted in Presentation Engine before decoding. Note that all the protected data are decrypted or MAC-checked just

before they are decoded, parsed or interpreted. Note that a Persistent Storage or a Network Server is not a source of a P-EVOB.

Following the above-mentioned policy for content protection, all the inputs from an AACS Disc to XML Parser and Script Engine are strictly protected from data manipulation. This means that the inputs are kept strictly the same as they were originally created by the content participant to the AACS LA. A Video Disc may leave some contents, XML Documents or ECMAScript Codes in Persistent Storages. If a content participant follows the AACS content protection scheme defined in this book, all the resources in Persistent Storages will never be affected on an AACS-Compliant Player by an HD DVD-Video Disc distributed by another content provider, regardless of whether the HD DVD-Video Disc is AACS-Compliant or not.

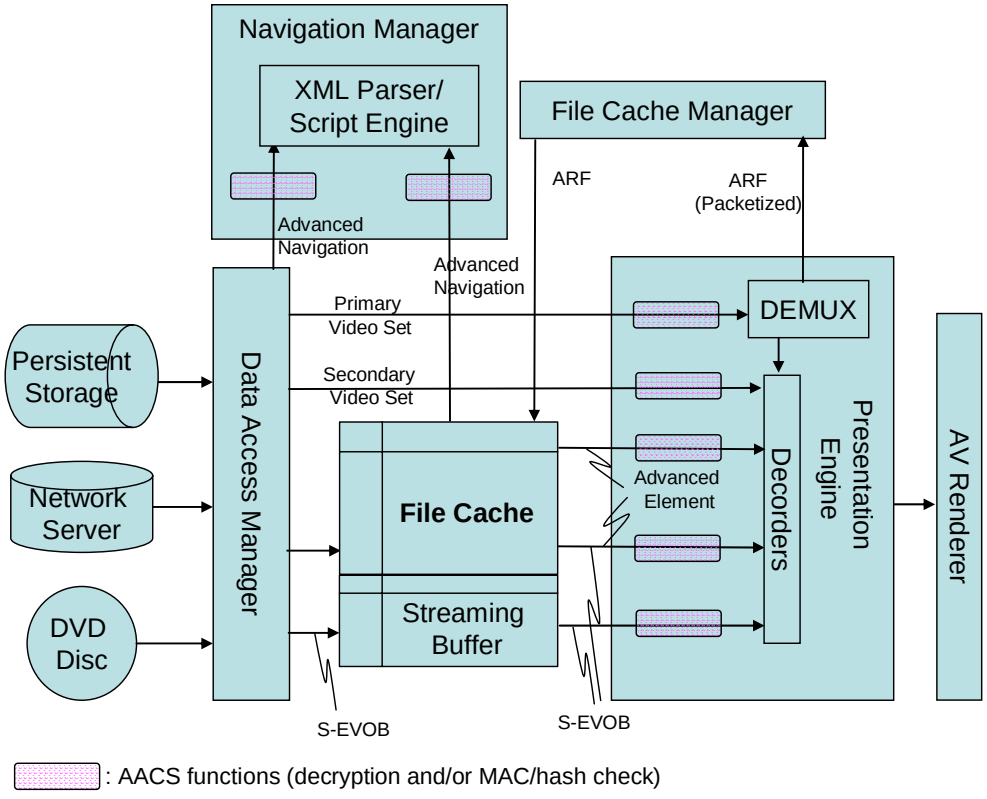

Figure 1-1: An Example of an HD DVD-Video Player System Model

For an Advanced Contents, there are two kinds of applications: One is a Disc Application and the other is a P-Storage Application. A Disc Application is an application which is invoked from an HD

DVD-Video disc, and a P-Storage Application is an application which is invoked from a Persistent Storage. An AACS Disc contains two Content Hash Tables: Content Hash Table #1 and #2. The Content Hash Table #1 contains the hashes of all the EVOBU/TUs in the P/S-EVOBs on the Disc. The Content Hash Table #2 contains the hashes of the following data:

i) Title Usage Files on the Disc
    a. These files are not encapsulated.
ii) All the XML Document files
    a. The files are encapsulated in Encapsulation Format for Hash or in Encapsulation Format for Encryption and Hash.
iii) All the ECMAScript files
    a. The files are encapsulated in Encapsulation Format for Hash or in Encapsulation Format for Encryption and Hash.

For a Disc Application, the AACS module verifies the Content Certificate, checks the hashes of the Content Hash Tables #1 and #2, and the hash of the TUF. Before or while a P/S-EVOB is played back, Content Hash Checking is performed on the P/S-EVOB using the Content Hash Table #1. Before an XML Document file or an ECMAScript file is processed by XML Parser/Script Engine, the AACS module shall check the hash of the concerned file referring to the Content Hash Table #2.

Figure 1-2 shows a rough sketch of relationship among the components on an AACS Disc. A “hash” arrow shows (a portion of) the source is hashed up into the sink. A “refer” arrow means that the source contains the field which refers to an entry in the sink data.

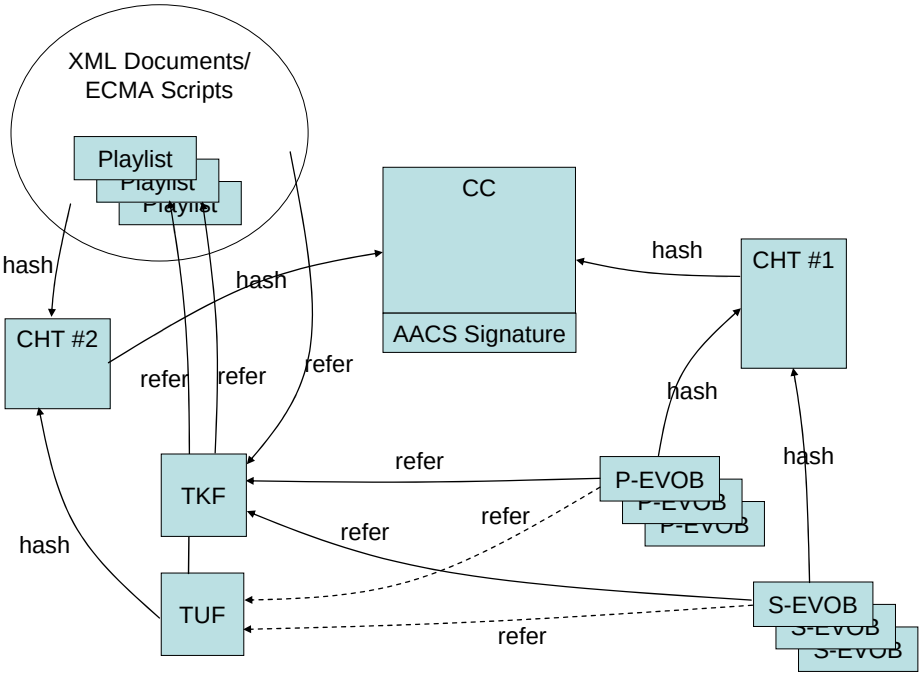

Figure 1-2: A Rough Sketch of Relationship among the Components on an AACS Disc

For a P-Storage Application, there is a Playlist in a Persistent Storage, and there are an associated Title Usage File (TUF) and an associated Title Key File (TKF) in the same directory. No Content Hash Table resides in a Persistent Storage. There is, however, a Content Certificate file associated with the Playlist in the same directory. The hash of a TUF is contained in the Content Certificate file. For a P-Storage Application, an AACS module shall verify the signature of the Content Certificate and the hash of the TUF. Figure 1-3 shows a rough sketch of relationship among the components on an AACS Disc.

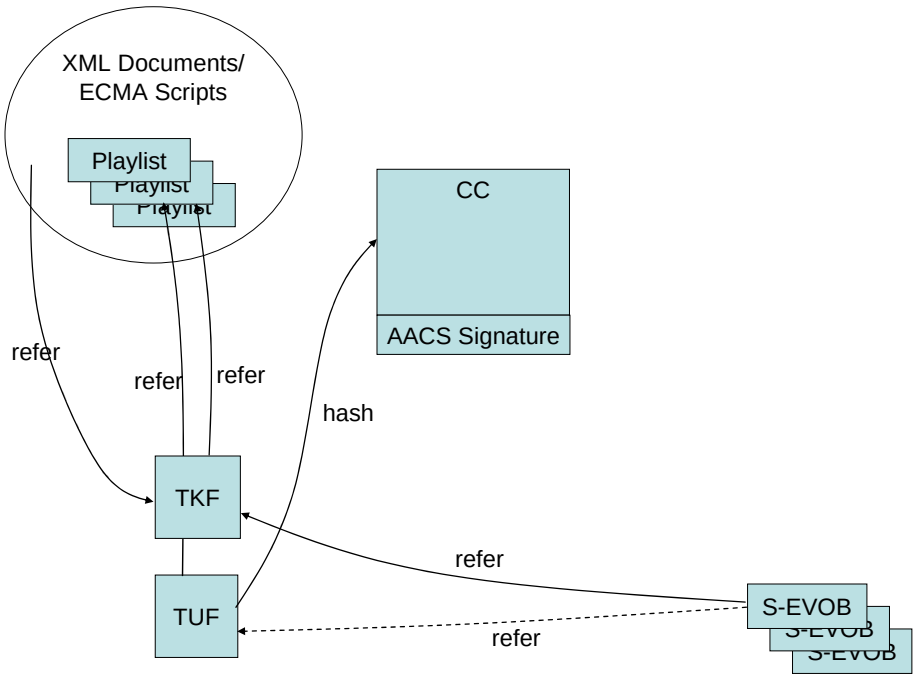

Figure 1-3: A Rough Sketch of Relationship among the Components in Persistent Storages

Note that this Book assumes a Playlist has one of the following names: “VPLST.XPL”, “VPLST%%%.XPL” or “APLST%%%.XPL”, where %%% runs 000 to 999. Although an arbitrary name is allowed for a Playlist in the *HD DVD-Video Specifications*, this Book places a restriction on them because of name convention used for a Playlist and the AACS components.

There are some AACS APIs exposed to the ECMAScript layer. The AACS APIs are enabled and usable when and only when the system is in AACS mode, i.e. when the AACS-Compliant Player plays back an AACS Disc. There is an AACS object, whose function properties are used mainly for network applications such as Online-Enabling. An Online-Enabling API exchanges a TKF in order to enable playback of EVOBs. AACS also has an impact on the HD DVD-Video APIs which load/save XML Documents or ECMASCript Codes. In AACS mode, the load/save APIs behave in a slightly different way from those described in the *HD DVD-Video Specifications*. The load APIs load and use contents in encapsulated files with hash or MAC as long as verification of them succeeds. And the save APIs save contents into encapsulated files with MAC. See 6.2.3 for the details.

1.3 Organization of this Document

This document is organized as follows:

- Chapter 1 provides an introduction and overview.
- Chapter 2 describes AACS components which reside in the Lead-in Area and the Burst Cutting Area of an HD DVD-ROM medium or a DVD-ROM medium.
- Chapter 3 provides a detailed description of the AACS components which reside in the Data Area of an HD DVD-ROM medium or a DVD-ROM medium.
- Chapter 4 provides a detailed description of the protection scheme for the data which specifically belong to the HD DVD-Video, especially to the Advanced Contents.
- Chapter 5 describes functions and protocols which are required for online applications such as downloading, streaming and Online-Enabling.
- Chapter 6 provides a detailed description of the protection scheme for contents in Persistent Storages.
- Chapter 7 describes the implementation and realization of the Sequence Key.

1.4 References

- *DVD Specifications for High Density Read-Only Disc, Part 1: Physical Specifications,* Version 1.0
- *DVD Specifications for Read-Only Disc, Part 1: Physical Specifications,* Version 1.04
- *DVD Specifications for High Definition Video,* Version 1.0
- *AACS Introduction and Common Cryptographic Elements*.
- *AACS Pre-recorded Video Book*.
- *Extensible Markup Language (XML) 1.0 (Third Edition),* W3C Recommendation 04 February 2004.
- *ECMAScript Language Specification 3rd edition,* Standard ECMA 262.

This specification shall be used in conjunction with the preceding publications. When the publications are superseded by an approved revision, the revision shall apply.

1.5 Notation

Except where specifically noted otherwise, this document uses the same notations and conventions for numerical values, operations, and bit/byte ordering as described in the *Introduction and Common Cryptographic Elements* book of this specification.

1.6 Terminology

- An HD DVD-Video ROM medium means an HD DVD ROM medium which contains an HD DVD-Video Content or a DVD-Video Content.
- In this specification, a DVD-Video ROM medium means a 3X-Speed DVD ROM medium which contains an HD DVD-Video Content or a DVD-Video Content.
- A Video Disc means an HD DVD-Video ROM medium or a DVD-Video ROM medium.
  - Each side of a double sided (HD) DVD-Video ROM medium is considered as one Disc.
- An HD DVD-Video Content is called AACS-Protected if it is protected by the AACS content protection scheme.
- A DVD-Video Content is called AACS-Protected if it is protected by the AACS content protection scheme.
- An AACS-Protected HD DVD-Video ROM medium means an HD DVD-Video ROM medium which contains an AACS-Protected HD DVD-Video content or an AACS-Protected DVD-Video content.
- An AACS-Protected DVD-Video ROM medium means a DVD-Video ROM medium which contains an AACS-Protected HD DVD-Video Content or an AACS-Protected DVD-Video Content.
- An AACS Disc means an AACS-Protected HD DVD-Video ROM medium or an AACS-Protected DVD-Video ROM medium.
- Advanced Resources mean the sum of two data sets, Advanced Navigation and Advanced Element, both of which are defined in the *HD DVD-Video Specifications*.
- An Advance Resource File means a data file of Advanced Resources.
- A Disc Application means an HD DVD-Video application which is booted from a Disc.
- A P-Storage Application means an application which is booted from a Persistent Storage.
- A Sequence Key Section means, for a Standard VTS, a combination of a Cell Block and a corresponding Interleaved Block in a P-EVOB which serves Sequence Key playback. For an Advanced VTS, a Sequence

Key Section means a combination of a TMAP and a corresponding Interleaved Block in a P-EVOB which serves Sequence Key playback.

1.7 Abbreviations and Acronyms

- GUID: Global Unique Identifier.
- (P/S-)EVOBU: EVOBU for P/S-EVOB respectively. An EVOBU is sometimes called just a VOBU.
- ARF: Advanced Resource File.
- ANF: Advanced Navigation File.
- *HD DVD-Video Specifications*: *DVD Specification for High Definition Video*, Version 1.0.

This page is intentionally left blank.

Chapter 2
AACS Components in Lead-in Area and Burst Cutting Area

2 AACS Components in Lead-in Area and Burst Cutting Area

2.1 Introduction

This chapter describes details of locations and/or formats of the AACS components defined in the AACS *Pre-recorded Video Book* which are stored in the Lead-in Area and the Burst Cutting Area of an AACS-Protected HD DVD-Video ROM or of an AACS-Protected DVD-Video ROM. The HD DVD-ROM format and the DVD-ROM format are licensed by the DVD Forum. The DVD Forum publishes the concerned specifications:

- *DVD Specifications for High Density Read-Only Disc, Part 1: Physical Specifications*
- *DVD Specifications for Read-Only Disc, Part 1: Physical Specifications*

A reader of this chapter is assumed to be familiar with the HD DVD-ROM format and the DVD-ROM format. This chapter focuses on the format which is relevant to the AACS content protection scheme. An overview of locations of the AACS components on an AACS-Protected (HD) DVD-Video ROM is shown in Figure 2-1.

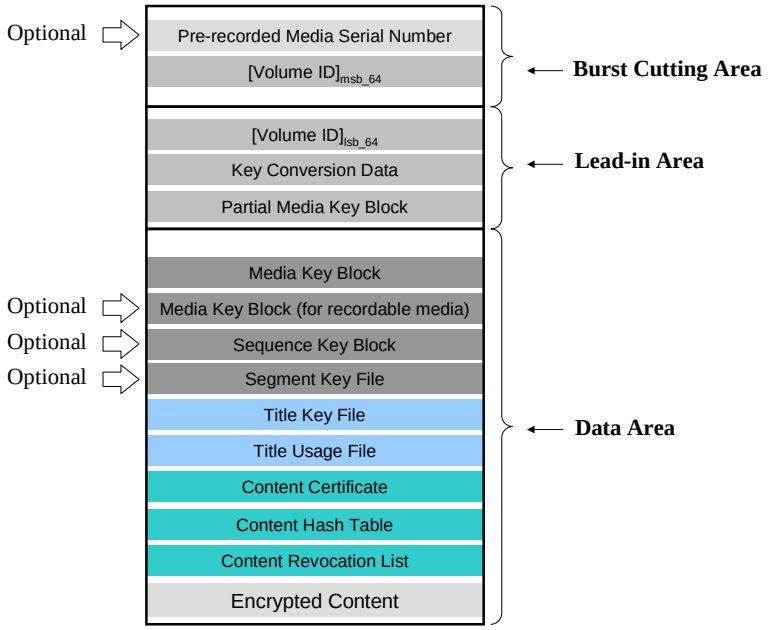

Figure 2-1: Physical Layout of AACS Components on an (HD) DVD-ROM

- The Volume ID (ID_{volume}) is separated into two parts which are stored in the Burst Cutting Area (BCA) and the Lead-in Area.
- The Pre-recorded Media Serial Number (PMSN) is stored in the BCA. It is optional.
- The Key Conversion Data (KCD) is stored in the Lead-in Area.
- The Media Key Block (MKB) for contents associated with the ROM medium is recorded in the Data Area while some records of the MKB are also stored in the Lead-in Area.

Some formats of the AACS components recorded in the Lead-in Area or the Burst Cutting Area (BCA) are different between an HD DVD-Video ROM and a DVD-Video ROM. The following subsections describe the details.

An AACS-Compliant HD DVD-ROM/DVD-ROM drive may support commands for format layer-change, which enable to change the format for a hybrid disc, i.e. an HD DVD-ROM/DVD-ROM Twin Format Disc, without disc ejection or power-off. Note that, however, from the viewpoint of the AACS content protection scheme, format layer-change from a layer which is protected by AACS to a layer which is not protected by AACS shall be regarded as disc ejection, even if no disc ejection is in fact occurred at the format layer-change. See the *AACS Introduction and Common Cryptographic Elements* for the drive behavior at disc ejection.

2.2 Key Conversion Data

An AACS Disc may contain a Key Conversion Data (KCD) of 128 bits. The KCD is stored in the Copyright Data Section in the manner described in the AACS *HD DVD and DVD Pre-recorded Book, Confidential Part*. Certain classes of devices, which are defined in the AACS *Compliance Rules*, need not perform the additional procedures for Drive-Host configuration defined in Chapter 4 of the AACS *Introduction and Common Cryptographic Elements*, provided that they are implemented with appropriate robustness defined in the AACS *Robustness Rules*. An AACS-Compliant (HD) DVD drive must not output the KCD if the connected device is not in the classes. Those classes of devices shall perform, however, key conversion in order to obtain a Media Key using the KCD if the KCD exists. The usage of KCD is described in the AACS *Introduction and Common Cryptographic Elements*.

2.3 AACS Components on HD DVD-ROM

Locations and formats of the AACS components stored in the BCA or a System Lead-in Area of an AACS-Protected HD DVD ROM medium are described in this section. Figure 2-2 depicts the structure of the BCA and the Lead-in Area of an HD DVD ROM medium. The details are provided in the following subsections.

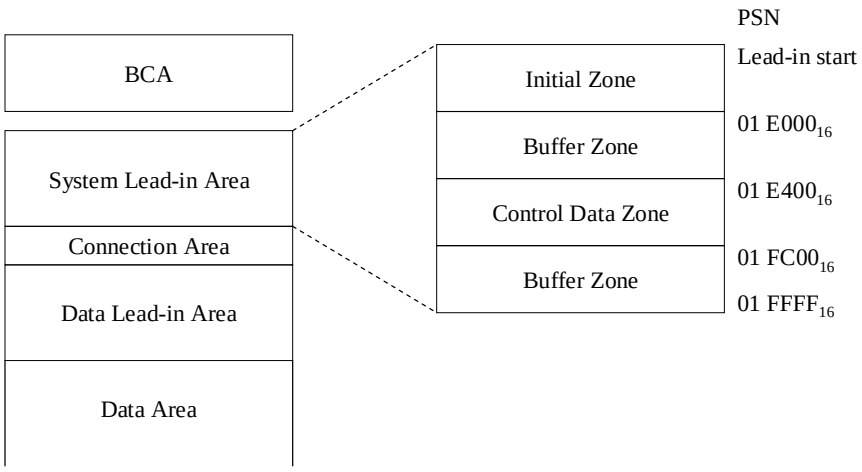

Figure 2-2: Structure of BCA and Lead-in Area of an HD DVD ROM Medium

2.3.1 Control Data

Figure 2-3 depicts the structure of the Control Data Zone. The Control Data Zone has 2 Control Data Sections and 2 Copyright Data Sections. A Control Data Section is comprised of 16 Data Segments all of which are equal one another. And a Copyright Data Section comprises 16 copies of the first Data Segment in the section. Figure 2-4 depicts structures of three kinds of Data Segments in the Control Data Zone. Note that each Data Segment consists of 32 Physical Sectors each of which has a length of 2048 bytes. Physical Sectors reserved in a Data Segment of a Control Data Section shall be filled with 00h. A Data Segment in a Copyright Data Section may contain copyright information. Otherwise, the Data Segment shall be filled with 00h.

The third Physical Sector in a Data Segment of a Control Data Section contains Copyright Protection Information. Table 2-1 shows the format of the Copyright Protection Information. The Copyright Protection System Type of 1 byte shall be equal to 01h if the AACS content protection is adopted for the medium. The following reserved field of 31 bytes shall be filled with 00h. See the AACS *HD DVD and DVD Pre-recorded Book, Confidential Part* for usage of the field of Reserved for Copyright Protection System Use in the Copyright Protection Information.

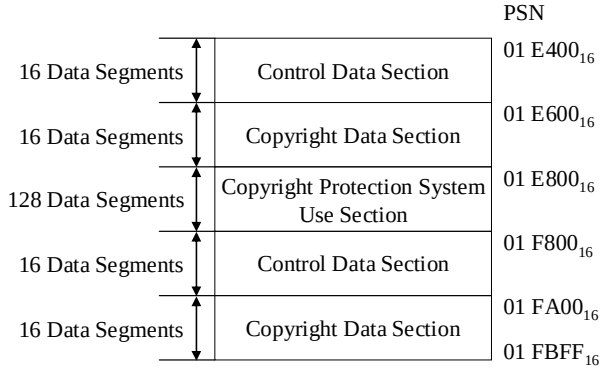

Figure 2-3: Structure of a Control Data Zone

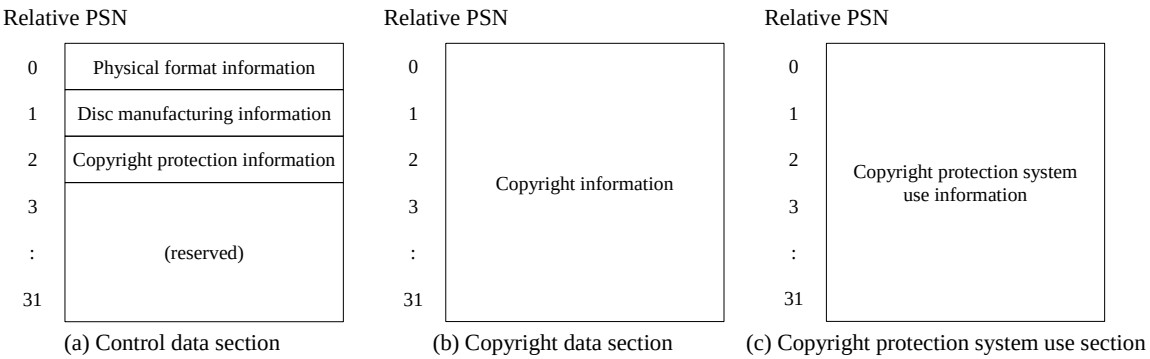

Figure 2-4: Structure of a Data Segment in a Control Data Zone

Table 2-1: Copyright Protection Information in an HD DVD-Video ROM medium

Bit / Byte	7	6	5	4	3	2	1	0
0	Copyright protection system type: 01_{16}							
1	Reserved							
:								
31								
32	MKB Packs							
33	Reserved for Copyright Protection System Use							
:								
2047								

2.3.2 Media Key Block

Each side of an AACS-Protected HD DVD-Video ROM medium shall contain one and only one Media Key Block (MKB) for decryption of contents on the medium. The whole MKB shall be stored in the Data Area while some records of the MKB shall also be stored in the Lead-in Area. The set of the records of the MKB is called a Partial MKB (P-MKB). A P-MKB consists of the Type and Version Record and the Host Revocation List Record. Drive Authentication shall use this P-MKB. See Chapter 4 of the *AACS Introduction and Common Cryptographic Elements* for the details of Drive Authentication Algorithm.

A P-MKB shall be recorded 8 times by a media manufacturer in the Copyright Protection System Use Section of the Control Data Zone. See Figure 2-4. The Copyright Protection System Use Section is divided into 8 portions. Each portion consists of contiguous 16 Data Segments. Every portion shall contain the same P-MKB. A P-MKB is recorded in the portion as shown in Figure 2-5. The maximum size of a P-MKB is 1 MB = 1048576 bytes. If the size of a P-MKB is less than 1 MB, the last MKB Pack may leave some residual bytes. The residual shall be filled with zero.

The size of a P-MKB shall be stored in the 32nd byte of the Copyright Protection Information as shown in Table 2-1. The MKB Packs field denotes the number of MKB Packs, which is equal to the smallest integer which is greater than or equal to the quotient of the P-MKB size divided by 32768.

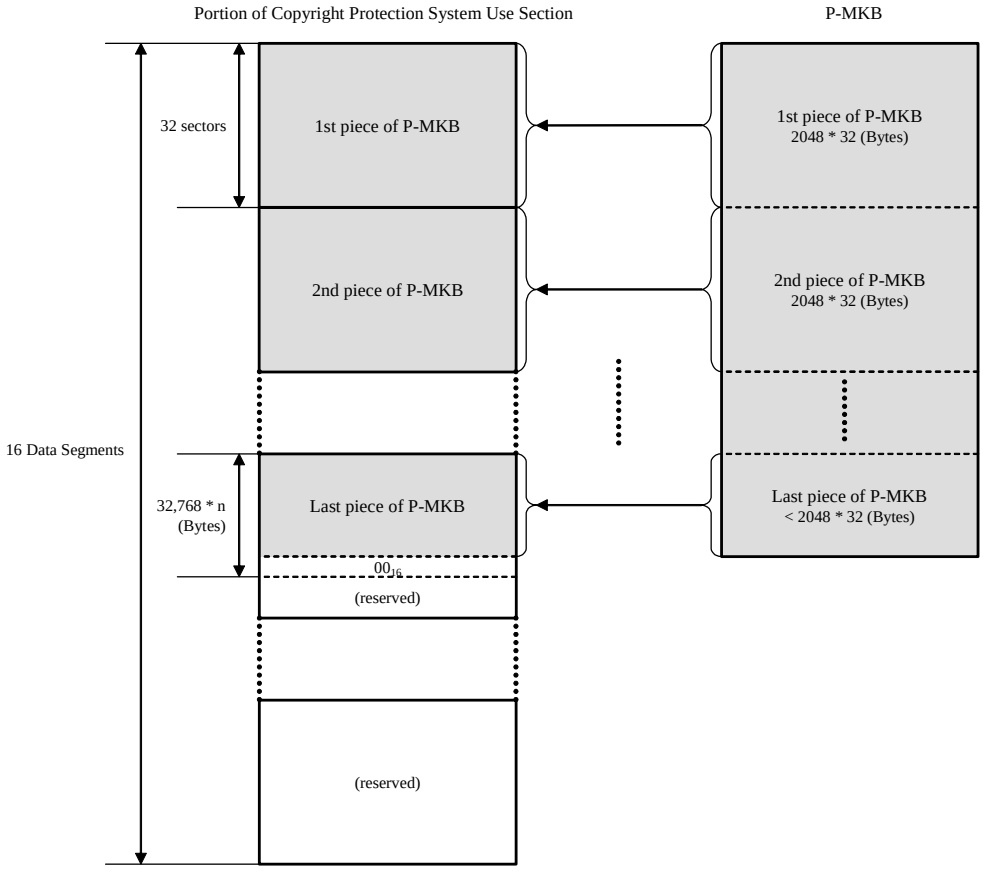

Figure 2-5: Example of storing MKB on Lead-in Area of an HD DVD ROM medium

2.3.3 Volume Identifier

Each side of an AACS-Protected HD DVD-Video ROM medium shall contain one and only one Volume Identifier (ID_{volume}) of 128 bits. The format of the Volume Identifier is shown on Table 2-2.

Table 2-2: Volume Identifier Format for an AACS-Protected HD DVD-Video ROM medium

Bit / Byte	7	6	5	4	3	2	1	0
0	Media Type: 40_{16}							
1	Reserved							
2 : 13	Unique Number							
14 15	Reserved							

A Volume Identifier contains the following values:

- Media Type of 1 byte. This value shall be 40h for an HD DVD-ROM medium.
- Unique Number of 12 bytes. This value shall be assigned by a content participant in order to identify a volume of an HD DVD-Video content.

The reserved fields shall be filled with 00h.

In an AACS-Protected HD DVD-Video ROM, the most significant 64 bits of the Volume Identifier shall be stored in the Burst Cutting Area (BCA) as shown in Table 2-3. The least significant 64 bits of the Volume Identifier shall be stored in a Copyright Data Section of the Control Data Zone (See 2.3.1) in a manner described in the AACS *HD DVD and DVD Pre-recorded Book, Confidential Part*.

Table 2-3: Format of BCA Record Containing a Volume Identifier

Bit / Byte	7	6	5	4	3	2	1	0

0	BCA Record ID: 1002_{16}
1	
2	Version: 10_{16}
3	Data Length: 08_{16}
4	Record Data: $[\text{Volume Identifier}]_{msb_64}$
:	
11	

The BCA may contain multiple contiguous data blocks called BCA Records. Each BCA Record begins with an Application ID of 2 bytes which indicates use of the Record. A 1-byte value of Version and a 1-byte value of Data Length follow the Application ID. The latter shows the length in byte of the remaining data in the Record. A Player may use the Application ID and the Data Length of a BCA Record so that it may skip Records and find the targeted one.

2.3.4 Pre-recorded Media Serial Number

A media manufacturer may write one and only one Pre-recorded Media Serial Number (PMSN) of 128 bits in a BCA Record of an AACS-Protected HD DVD-Video ROM. If a medium has a PMSN, it shall be contained in a BCA Record of the format shown in Table 2-4.

Table 2-4: Format of BCA Record Containing a Pre-recorded Media Serial Number

Bit / Byte	7	6	5	4	3	2	1	0
0	BCA Record ID: 1003_{16}							
1								
2	Version: 10_{16}							
3	Data Length: 10_{16}							
4	Record Data: Pre-recorded Media Serial Number							
:								
19								

2.4 AACS Components on DVD-ROM

Locations and formats of the AACS components that are stored in the BCA or the System Lead-in Area of an AACS-Protected DVD-Video ROM medium are described in this section. Figure 2-6 depicts the structure of the BCA and the Lead-in Area of a DVD ROM medium. The details are provided in the following subsections.

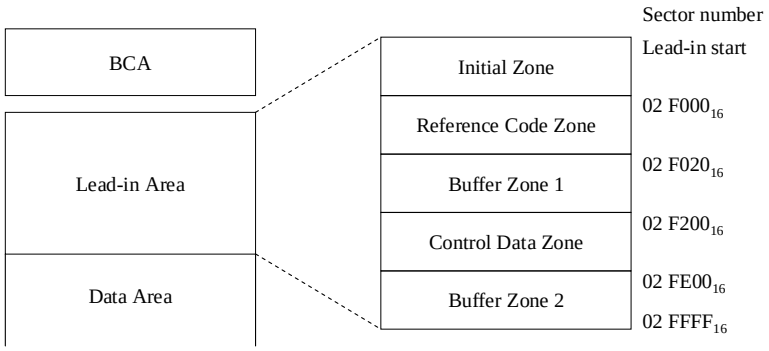

Figure 2-6: Structure of BCA and Lead-in Area of a DVD ROM Medium

2.4.1 Control Data

Figure 2-7 depicts structure of the Control Data Zone. The Control Data Zone has 2 Control Data Sections and 2 Copyright Data Sections. A Control Data Section is comprised of 16 ECC blocks all of which are equal to one another. And a Copyright Data Section comprises 16 copies of the first ECC Block in the section. Figure 2-8 depicts structure of three kinds of ECC blocks in the Control Data Zone. Note that each ECC Block consists of 16 Physical Sectors each of which has a length of 2048 bytes. The last 13 Physical Sectors reserved in an ECC Block of a Control Data Section shall be filled with 00h. The last 14 sectors in an ECC Block of a Copyright Data Section may contain copyright information. Otherwise, each of the sectors shall be filled with 00h.

The third Physical Sector in an ECC Block of a Control Data Section contains Copyright Protection Information. Table 2-5 shows the format of the Copyright Protection Information. The Copyright Protection System Type of 1 byte shall be 01h if the AACS content protection scheme is applied to the medium. The

following reserved field of 31 bytes shall be filled with 00h. See the AACS *HD DVD and DVD Pre-recorded Book, Confidential Part* for usage of the field of Reserved for Copyright Protection System Use in the Copyright Protection Information.

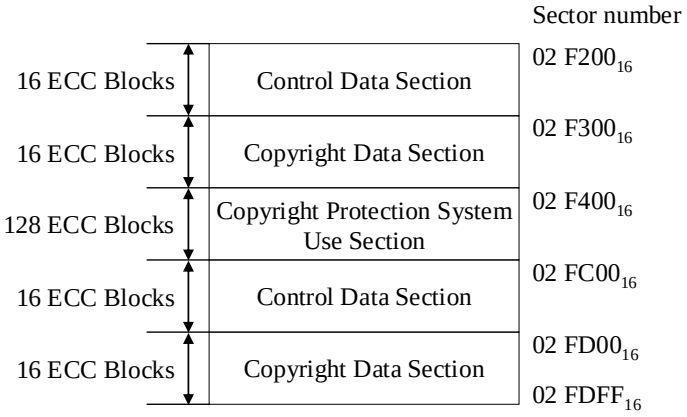

Figure 2-7: Structure of the Control Data Zone

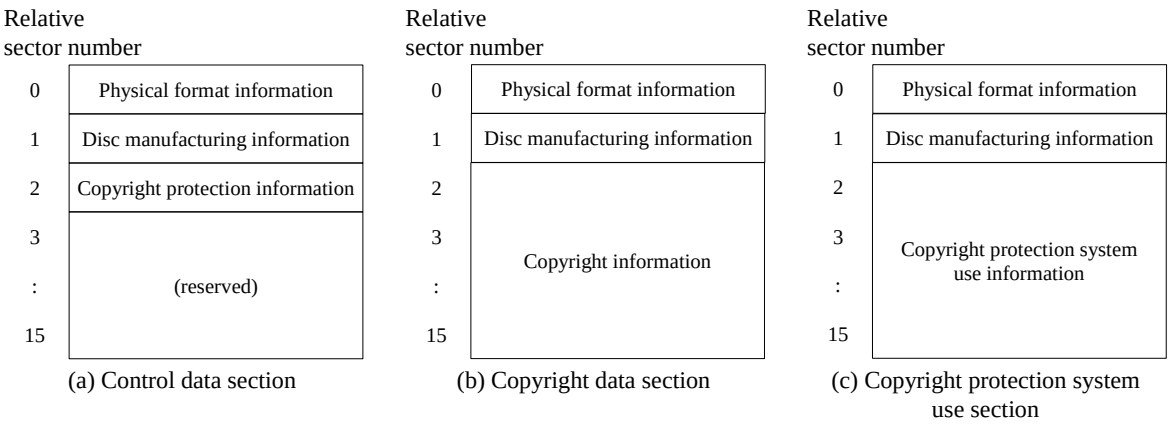

Figure 2-8: Structure of an ECC block in Control Data Zone

Table 2-5: Copyright Protection Information in a DVD-Video ROM medium

Byte \ Bit	7	6	5	4	3	2	1	0
0	Copyright Protection System Type: 01_{16}							
1	Reserved							
:								
31								
32	MKB Packs							
33	Reserved for Copyright Protection System Use							
:								
2047								

The last 14 sectors in an ECC block of the Control Data Zone, including the Copyright Protection Information in the Control Data Section, are called Content Provider Information Sectors. Figure 2-9 depicts the configuration of Data Frame, which is the logical structure of one physical sector. Table 2-6 shows the format of CPR_MAI in Content Provider Information Sectors. And Table 2-7 shows the CPR_MAI of a Data Frame in the Data area of an AACS-Protected DVD-Video ROM medium.

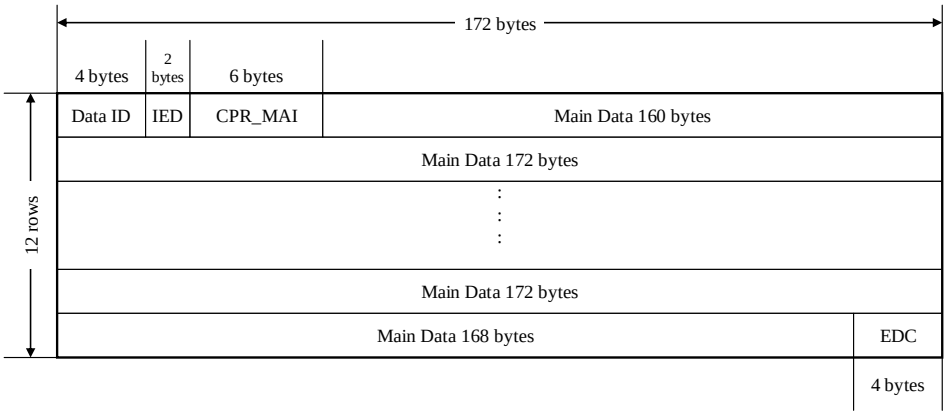

Figure 2-9: Data Frame Configuration for a DVD ROM medium

Table 2-6: CPR_MAI Format in Content Provider Information Sectors

Bit / Byte	7	6	5	4	3	2	1	0
0	CPS_TY: 03_{16}							
1	PM_TY: 00_{16}							
2	Reserved							
3								
4								
5								

CPR_MAI in Content Provider Information Sectors shall consist of the following fields:

- CPS_TY of 1 byte. This value indicates a copyright protection system applied to this medium. It shall be 03 for an AACS-Protected DVD-Video ROM medium.
- PM_TY of 1 byte. This value indicates the pre-recorded media type. It shall be 00h for a DVD ROM medium.
- A reserved field of 4 bytes which shall be filled with 00h.

Table 2-7: CPR_MAI in Data Area

Bit / Byte	7	6	5	4	3	2	1	0
0	CPM: 0_2	CP_SEC: 0_2	CGMS: 00_2		ADP_TY: 00_2		CP_MOD: 00_2	
1	Reserved							
2								
3								
4								
5								

A CPR_MAI in Data area shall consist of the following fields:

- CPM of 1-bit. This value shall be 0b for an AACS-Protected DVD-Video ROM medium.
- CP_SEC of 1 bit. This value shall be 0b if CPM is 0b.
- CGMS shall be 00b.
- ADP_TY of 2 bits which shall be equal to 00b
- A CP_MOD of 2 bits. This value shall be 00b if CPM is 0b.
- A reserved field of 4 bytes which shall be filled with 00h.

2.4.2 Media Key Block

Each side of an AACS-Protected DVD-Video ROM medium shall contain one and only one Media Key Block (MKB) for decryption of contents on the medium. The whole MKB shall be stored in the Data Area while some records of the MKB shall also be stored in the Lead-in Area. The set of the records of the MKB is called a Partial MKB (P-MKB). A P-MKB consists of the Type and Version Record and the Host Revocation List Record.

A P-MKB shall be recorded 4 times by a media manufacturer in the Copyright Protection System Use Section of the Control Data Zone. See Figure 2-8. The Copyright Protection System Use Section is divided into 4 portions. Each portion consists of contiguous 32 ECC blocks. Every portion shall contain the same P-MKB. A P-MKB is recorded in the portion as shown in Figure 2-10. The maximum size of the P-MKB is 1 MB - 128kB = 917504 bytes. If the size of the P-MKB is less than 1 MB-128kB, the last MKB Pack may leave some residual bytes. The residual shall be filled with zero.

The size of the P-MKB shall be stored in the 32nd byte of the Copyright Protection Information as shown in Table 2-5. The MKB Packs field denotes the number of MKB Packs, which is equal to the smallest integer which is greater than or equal to the quotient of the P-MKB size divided by 32768.

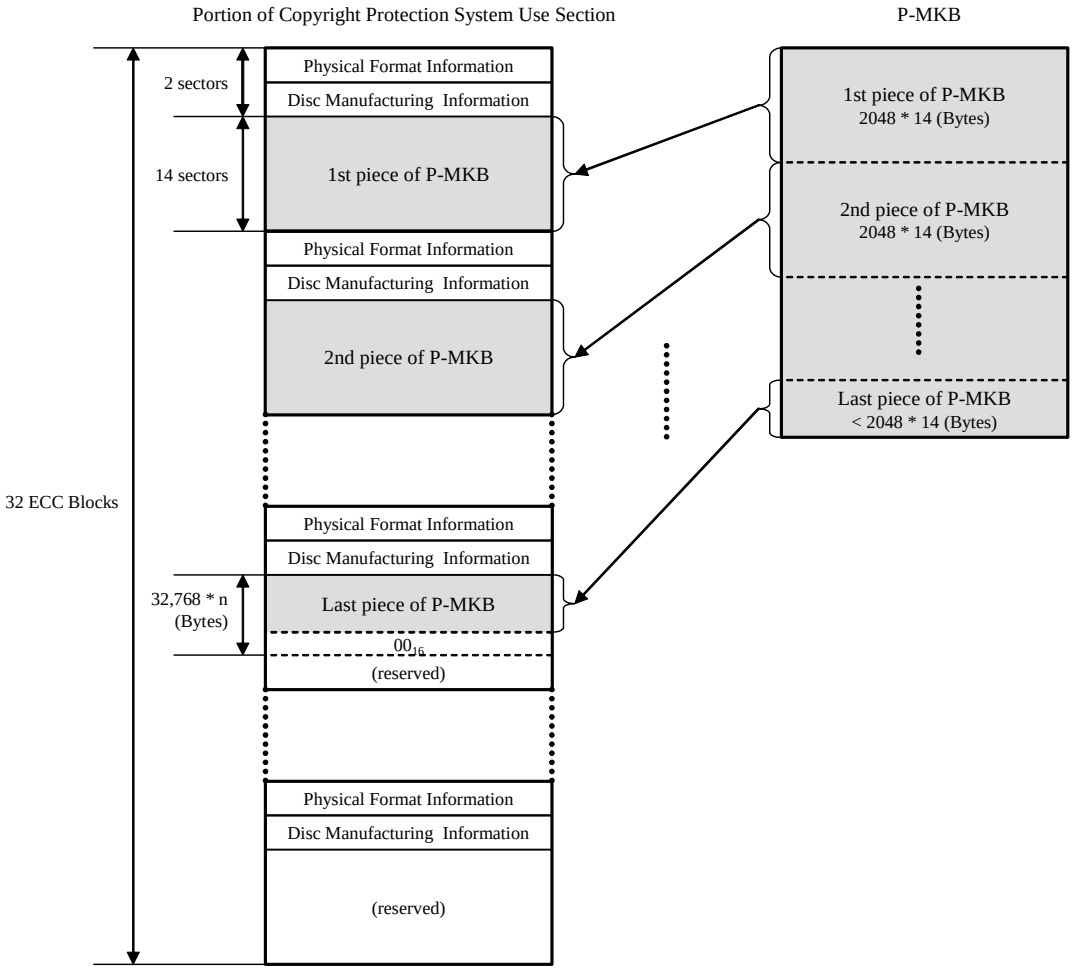

Figure 2-10: Example of storing MKB on Lead-in Area of a DVD-ROM medium

2.4.3 Volume Identifier

Each side of an AACS-Protected DVD-Video ROM shall contain one and only one Volume Identifier (ID_{volume}) of 128 bits. The format of the Volume Identifier is shown in Table 2-8.

Table 2-8: Volume Identifier Format for an AACS-Protected DVD-Video ROM medium

Bit / Byte	7	6	5	4	3	2	1	0
0	Media Type: 01_{16}							

1	Reserved
2 : 13	Unique Number
14 15	Reserved

A Volume Identifier contains the following fields:

- Media Type of 1 byte. A DVD-ROM medium shall have 01h for this value.
- The Unique Number field of 12 bytes. This value shall be assigned by a content participant in order to identify a volume of an HD DVD-Video content.

Note that the reserved fields shall be filled with 00h.

In an AACS-Protected DVD-Video ROM, the most significant 64 bits of Volume Identifier shall be stored in the BCA as shown in Table 2-3, and the least significant 64 bits of Volume Identifier shall be stored in the Copyright Data Section in a manner described in the AACS *HD DVD and DVD Pre-recorded Book, Confidential Part*.

2.4.4 Pre-recorded Media Serial Number

An AACS-Protected DVD-Video ROM may have a Pre-recorded Media Serial Number (PMSN) of 128 bits, which is placed in the Burst Cutting Area (BCA) by a media manufacturer. The format of PMSN is shown in Table 2-4.

This page is intentionally left blank.

Chapter 3

AACS Components in Data Area

3 AACS Components in Data Area

3.1 Introduction

This chapter describes details of locations and/or formats of the AACS components defined in the AACS *Pre-recorded Video Book* which are stored in the Data Area of an AACS-Protected HD DVD-Video ROM or of an AACS-Protected DVD-Video ROM. The HD DVD-ROM format and the DVD-ROM format are licensed by the DVD Forum. The DVD Forum publishes the concerned specifications:

- *DVD Specifications for High Density Read-Only Disc, Part 2: File System Specifications*
- *DVD Specifications for Read-Only Disc, Part 2: File System Specifications*

A reader of this chapter is assumed to be familiar with the HD DVD-ROM format and the DVD-ROM format. This chapter focuses on the format which is relevant to the AACS content protection scheme.

An overview of locations of the AACS components on an AACS Disc is shown in Figure 2-1. The following data will be stored in the Data Area: a Media Key Block (MKB), a Sequence Key Block File (SKBF), Segment Key Files (SKFs), Title Key Files (TKFs), Title Usage Files (TUFs), a Content Certificate, Content Hash Tables (CHTs), a Content Revocation List (CRL), encrypted contents and contents with MAC/hash. The encapsulation formats for Contents and other HD DVD-Video specific data for copyright protection are mentioned in the following chapters. An AACS Disc shall have, in the root directory, a directory for AACS components. The name “AACS” is reserved for the directory.

There are some reserved fields in data formats defined hereafter. Note that a reserved field shall be filled with 0b unless otherwise stated.

3.2 Media Key Block

An AACS Disc shall have one and only one Media Key Block (MKB) in the “AACS” directory for decryption of encrypted contents on the medium. The MKB is stored as a file in the Data Area of the medium, and the MKB file has the name “MKBROM.AACS”. Additionally, an AACS Disc may have another MKB, a Read/Write MKB for Update in the “AACS” directory. A Read/Write MKB is to be stored in a recording

device, which is used to update the MKB on a recordable medium. The usage of the Read/Write MKB is described in the *AACS HD DVD and DVD Recordable Book*. The name "MKBRECORDABLE.AACS" is reserved for the Read/Write MKB for Update on an AACS Disc. When an AACS Disc is inserted into an (HD) DVD recording device, the recording device may authenticate the Read/Write MKB for Update on the medium and check the version of the MKB for update. (See the *AACS Recordable Video Book*).

3.3 Sequence Key Block and Segment Key Files

An AACS-Protected HD DVD-Video ROM medium may have one and only one Sequence Key Block File (SKBF) in the "AACS" directory. The name "SKB.AACS" is reserved for the SKBF. The SKBF contains six SKBs in it. The format of SKBF is shown in Table 3-1. Details of the SKB format and the SKB process are described in the *AACS Pre-recorded Video Book*. The SKB process for one SKB yields one Volume Variant Unique Key (K_{vvu}) and Variant Data. The lower 10 bits of the Variant Data are, in this book, called Segment Key Unit Number (SKUN). Thus, the six times of the SKB processes for the SKBF yields in order six Volume Variant Unique Keys and six SKUNs: K_{vvu}[1], K_{vvu}[2], …, K_{vvu}[6], SKUN[1], SKUN[2], …, SKUN[6]. Note that 0 <= SKUN[j] <= 1023 for an integer j such that 1 <= j <= 6.

The SKBF accompanies a Segment Key File (SKF) in the "AACS" directory: "SKF.AACS". An SKF contains six Segment Key Group (SKG) fields. Each SKG field contains 1024 Segment Key Unit (SKU) fields. An SKU field contains 32 encrypted pairs of a Segment Key Number (SEG_NO) and a Segment Key of 16 bytes. Let j be an integer such that 1 <= j <= 6. From the 1024 SKU fields in the j-th SKG field, SKG #j, one SKU field is chosen according to the number SKUN[j]. The chosen SKU field, SKU #SKUN[j], can be decrypted by the Volume Variant Unique Key K_{vvu}[j].

One SKG corresponds to a Segment Key Range (SKR) which is a set of P-EVOBs on the AACS Disc. As described above, an AACS-Compliant Player has 32 decrypted pairs of an SEG_NO and a Segment Key for an SKR, which form a table called a Segment Key Table (SKT). Sequence Key Sections in a P-EVOB in an SKR share the same SKT. See Chapter 7 for Sequence Key Section.

Table 3-1: Format of SKBF

Byte \ Bit	7	6	5	4	3	2	1	0
0 : 11	SKBF_ID							
12 13	VERN							
14 : 17	SKB_SIZE #1							
18 : 33	SKB_SIZE #2 - #5							
34 : 37	SKB_SIZE #6							
38 : 47	Reserved							
48 : 47 + L#1	SKB #1							
48 + L#1 : 47 + L#1 + L#2	SKB #2							

48 + L#1 + L#2 **:** **47 + $\Sigma_{n=1}^{5}$L#n**	⋮
48 + $\Sigma_{n=1}^{5}$L#n **:** **47 + $\Sigma_{n=1}^{6}$L#n**	SKB #6

The SKBF contains the following fields:

- SKBF_ID of 12 bytes which is an identifier of a Segment Key File. The value shall be "DVD_HD_VSKBF" with character set code of ISO/IEC 646:1983 (a-characters).
- VERN of 2 bytes which is the version number of the Sequence Key Block File. This field shall be 0 for the current version of SKBF.
- The fields of SKB_SIZE #1 - #6. Each of the fields has length of 4 bytes. SKB_SIZE #n stores L#n, where L#n indicates the size of SKB #n.
- Six SKBs. See the *AACS Pre-recorded Video Book* for the detailed format of an SKB. The order of the records shall be as follows: Verify Media Key Record, Nonce Record, Calculate Variant Data Record, (Conditionally Calculate Variant Data Records: Optional), End of SKB Record.
- The reserved fields shall be filled with 00h.

The format of SKF is shown in Table 3-2.

Table 3-2: Format of Segment Key File

Bit / Byte	7	6	5	4	3	2	1	0
0 : 11	SKF_ID							
12 : 15	HDV_SKF_SIZE							
16 : 31	Reserved							
32 33	VERN							
34 : 39	Reserved							
40 : 671783	SKG #1							
671784 : 1343527	SKG #2							
40 + 671744*2 : 39 + 671744*5	⋮							

40 + 671744*5 **:** **39 + 671744*6**	SKG #6
40 + 671744*6 **:** **4032511**	Reserved

The SKF contains the following fields:

- SKF_ID of 12 bytes which is an identifier of a Segment Key File. The value shall be "DVD_HD_V_SKF" with character set code of ISO/IEC 646:1983 (a-characters).
- HDV_SKF_SIZE of 4 bytes which indicates the size of the SKF.
- VERN of 2 bytes which is the version number of the Segment Key File. This field shall be 0 for the current version of SKF.
- Six SKG fields. The format of an SKG field is shown in Table 3-3, where the offset in the table is relative. The length of SKG field is 671744 bytes.
- The reserved field shall be filled with 00h.

Table 3-3: Format of SKG Field

Bit **Byte**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0 **:** **655**	SKU #0							
656 **:** **1311**	SKU #1							

1312 **:** **656*1023 - 1**	⋮
656*1023 **:** **656*1024 - 1**	SKU #1023

Each SKG field consists of 1024 Segment Key Unit fields. Note that the index of the SKU fields start from 0. Table 3-4 shows the format of an SKU field. The offsets of the fields in an SKU are relative.

Table 3-4: Format of Segment Key Unit Field

Encrypted Portion (656 bytes) (C_{ekd})	**0 : 15**	Verification Data ($0123456789ABCDEF_{16}$ || $XXXXXXXXXXXXXXXX_{16}$) (where $XXXXXXXXXXXXXXXX_{16}$ is any value of 8 bytes)	
	16 : 19	SEG_NO for Segment Key #1	Segment Key Entry #1
	20 : 35	Encrypted Segment Key #1	
	36 : 39	SEG_NO for Segment Key #2	Segment Key Entry #2
	40 : 55	Encrypted Segment Key #2	
	56 … 635	⋮	
	636 : 638	SEG_NO for Segment Key #32	Segment Key Entry #32
	640 : 655	Encrypted Segment Key #32	

A Segment Key Unit field contains the following fields:

- Verification Data (V_{vvu}) of 16 bytes. This data may be used to verify the calculated Volume Variant Unique Key which is used to decrypt Encrypted Segment Keys in this file. A Player may check the validity of the Volume Variant Unique Key K_{vvu} before it processes decryption of the subsequent encrypted portion:
  $[\text{AES-128CBCD}(K_{vvu}, C_{ekd})]_{msb_64} = 0123456789ABCDEFh,$
  where AES-128CBCD denotes decryption by the AES algorithm in the CBC mode defined in the *AACS Introduction and Common Cryptographic Elements*.
- A series of 20-byte Segment Key Entries. There are 32 Segment Key Entries. Each Segment Key Entry consists of the following fields:
  - SEG_NO of 4 bytes which indicates a segment (from 1 to 8) in the corresponding Sequence Key Section. The segment can be decrypted by the following Segment Key.
  - Segment Key of 16 bytes.

In a Segment Key Unit field, the portion C_{kd} from the 0-th byte to the 655-th byte, which contains the Verification Data and a series of Segment Key Entries, is encrypted as follows:

$$C_{ekd} = \text{AES-128CBCE}(K_{vvu}, C_{kd}),$$

where AES-128CBCE denotes encryption by the AES algorithm in the CBC mode defined in the *AACS Introduction and Common Cryptographic Elements*.

An HD DVD-Video ROM medium shall have an SKBF and an SKF if it contains a P-EVOB which has Sequence Key Sections (See Chapter 7). Conversely, no SKBF and no SKF shall be present on a HD DVD-Video ROM medium which contains no P-EVOB with Sequence Key Sections.

3.4 Title Key File

An AACS Disc shall have at least one Title Key File (TKF) in which each Title Key data is encrypted by AES-128E with K_u. K_u is the Volume Unique Key (K_{vu}). Title Key Files on an AACS Disc shall reside in the "AACS" directory. The name "VTKF.AACS", "VTKF%%%.AACS" and "ATKF%%%.AACS" are reserved for the TKFs, where %%% runs from 000 to 999. "VTKF.AACS" is for a Category 1 Disc which is defined in the *HD DVD-Video Specifications*. "VTKF.AACS" is also used for Category 3 Disc if the Disc is played back in Standard Content Playback State. For a Category 2 or 3 Disc, a TKF other than "VTKF.AACS" is associated with a Playlist. Two or more Playlists are not allowed to share the same TKF. There is a name convention: "VPLST%%%.XPL" shall be accompanied by "VTKF%%%.AACS".

“APLST%%%.XPL” shall be accompanied by “ATKF%%%.AACS”. A Title Key File has the format shown in Table 3-5.

Table 3-5: Format of Title Key File

Byte \ Bit	7	6	5	4	3	2	1	0	
0 : 11	TKF_ID								Header
12 : 15	HD_VTKF_SIZE								
16 : 27	PLAYLIST_NAME								
28 : 31	Reserved								
32 33	VERN								
34 : 127	Reserved								
128	BIFO for Title Key #1								Title Key Entry #1
129 : 131	Reserved								

132 : 147	Encrypted Title Key #1 (K_{te_1})	
148 : 163	Binding MAC #1 (BM_1)	
164	BIFO for Title Key #2	Title Key Entry #2
165 : 167	Reserved	
168 : 183	Encrypted Title Key #2 (K_{te_2})	
184 : 199	Binding MAC (BM_2)	
…	…	
2396	BIFO for Title Key #64	Title Key Entry #64
2397 : 2399	Reserved	
2400 : 2415	Encrypted Title Key #64 (K_{te_64})	

2416 : 2431	Binding MAC (BM_{64})	
2432 : 2463	Reserved	
2464 : 2479	TKF MAC	

- TKF_ID of 12 bytes which is an identifier of the Title Key File. The value shall be "DVD_HD_V_TKF" with character set code of ISO/IEC 646:1983 (a-characters).
- HD_VTKF_SIZE of 4 bytes which indicates the end address of the Title Key File. The value shall be 2480, because the size of the Title Key File is fixed and has that length.
- VERN of 4 bytes which indicates the version number of the Title Key File. This value shall be 0 for the current version.
- PLAYLIST_NAME of 12 bytes which stores the name of the Playlist which this Title Key File accompanies. The name shall be "VPLST%%%.XPL" or "APLST%%%.XPL", where %%% runs from 000 to 999. The AACS module in a Player which plays back an Advanced Content shall compare PLAYLIST_NAME with the name of a Playlist which is currently active. Unless the names are identical, the Title Keys in this TKF must not be used. Note that only one Playlist is active at one time. For a Standard Content which involves no Playlist for playback, this field shall be filled with FFh. An implication of this is that the AACS module in a Player should know which contents the Player plays back, an Advanced Content or a Standard Content.
- A series of Title Key Entries of 36 bytes. Each Title Key Entry consists of the following fields:
    - BIFO of 1 byte which indicates the Binding Information for each Title Key. The format of BIFO is shown in Table 3-6.
    - A 3-byte reserved field which shall be filled with 00h.
    - A Encrypted Title Key of 16 bytes which is encrypted as follows: K_{te_i} = AES-128E(K_{vu}, K_{t_i}), where AES-128E denotes encryption by the AES algorithm in the ECB mode defined

in the AACS *Introduction and Common Cryptographic Elements*, K_{t_i} is a Title Key and K_{vu} is the Volume Unique Key defined in the *AACS Pre-recorded Video Book*.

- Binding MAC of 16 bytes which stores the MAC value which is specified by BIND_TYPE in the BIFO above. In the case that BIND_TYPE = 000b, all bytes of this field shall be filled with FFh.

- All the Reserved fields in a TKF shall be filled with 00h.
- The 16-byte field of TKF MAC. This field stores the CMAC value of the data ranging from the 0th to the 2463rd byte of the Title Key File. The key for the CMAC calculation is the Volume Unique Key (K_{vu}).

Table 3-6: Format of Binding Information

7	6	5	4	3	2	1	0
AV_FLG	BIND_TYPE			Reserved			

The Binding Information shall consist of the following fields:

- AV_FLG of 1 bit which shall be set as follows:

  0b: the corresponding Title Key is not available

  1b: the corresponding Title Key is available

- BIND_TYPE of 3 bits. The meaning of the value is as follows:

  000b The corresponding Title Key is bound only to the Volume ID (ID_v). The Binding MAC field shall be filled with FFh.

  001b The corresponding Title Key is bound to the Pre-recorded Media Serial Number (PMSN), which means that the Binding MAC is calculated as follows:

  K_t = AES-128D(K_{vu}, K_{te}),

  MAC = CMAC(K_t, PMSN).

  010b The corresponding Title Key is bound to the Device Unique Nonce (DUN), which means that the Binding MAC is calculated as follows:

K_t = AES-128D(K_{vu}, K_{te}),

MAC = CMAC(K_t, DUN).

011b The corresponding Title Key is bound to both PMSN and DUN, which means that the Binding MAC is calculated as follows:

K_t = AES-128D(K_{vu}, K_{te}),

MAC = CMAC(K_t, DM), where DM = AES-G(DUN, PMSN).

100b The corresponding Title Key is bound to the Temporary Nonce (TN), which means that the Binding MAC is calculated as follows:

K_t = AES-128D(K_{vu}, K_{te}),

MAC = CMAC(K_t, TN).

Others Reserved.

The BIND_TYPEs for the Title Keys used to decrypt contents on an AACS Disc usually are 000b. The other BIND_TYPEs may be used to protect contents stored in Persistent Storages. Before using a Title Key, a Player shall verify the associated Binding MAC. If the verification fails, the Title Key must not be used.

3.5 Title Usage File

This section describes the format of Usage Rule. Usage Rules are stored in a Title Usage File (TUF). A Usage Rule Pointer in the CPI field in a P/S-EVOB may indicate a Usage Rule Set in the Title Usage File. The Usage Rule Pointer in a P/S-EVOB shall not change within the P/S-EVOB. Some P/S-EVOBs may share the same Usage Rule Set. Figure 3-1 shows an example of Usage Rule application to EVOBs.

Title Usage Files shall reside in the “AACS” directory if they exist on an AACS Disc. The names “VTUF.AACS”, “VTUF%%%.AACS” and “ATUF%%%.AACS” are reserved for the TUFs, where %%% runs from 000 to 999. “VTUF.AACS” is for a Category 1 Disc which is defined in the *HD DVD-Video Specifications*. “VTUF.AACS” is also used for Category 3 Disc if the Disc is played back in Standard Content Playback State. For a Category 2 or 3 Disc, a TUF other than “VTUF.AACS” accompanies a Playlist. Two or more Playlists are not allowed to share the same TUF. There is a name convention: “VPLST%%%.XPL” shall be accompanied by “VTUF%%%.AACS”. “APLST%%%.XPL” shall be accompanied by “ATUF%%%.AACS”. A Title Usage File has the format shown in Table 3-7. The size of a Title Usage File

shall be equal to or less than 64 KB. A Title Usage File contains a set of Usage Rule Sets. In the table, X_v denotes the length of the v-th Usage Rule Set.

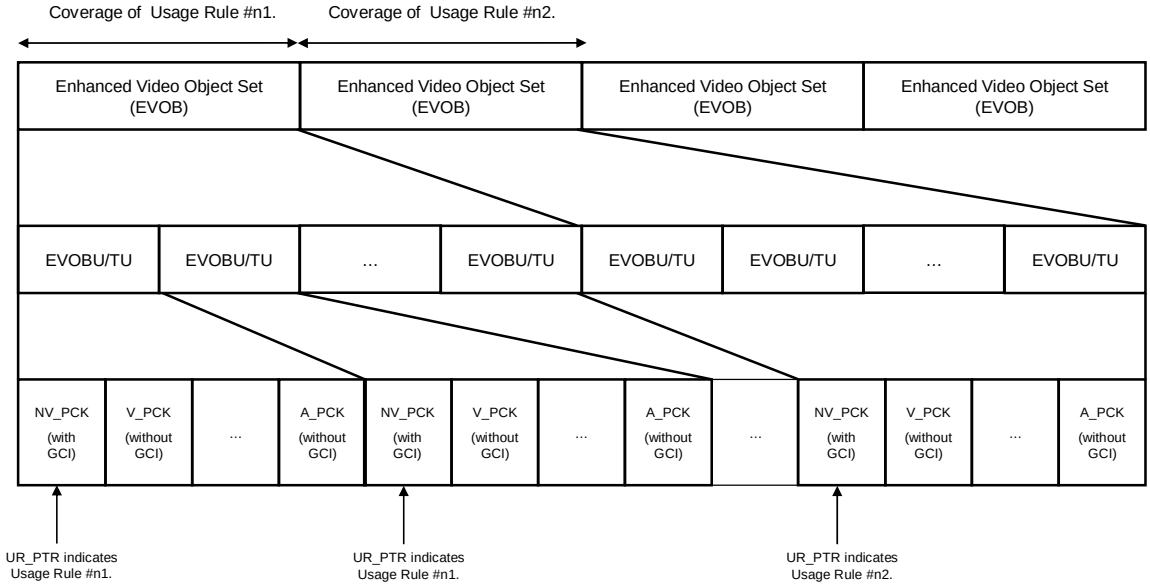

Figure 3-1: An Example of Coverage of Usage Rules

Table 3-7: Format for Title Usage File

Byte \ Bit	7	6	5	4	3	2	1	0
0 : 11	URF_ID							
12 : 15	HD_VURF_SIZE							
16	URS_NUM (N)							
17 : 20	HASH_SIZE							
21 22	VERN							
23 : 34	PLAYLIST_NAME							
35 : 127	Reserved							
128 : $127 + X_1$	Usage Rule Set (URS) #1							
$128 + X_1$:	Usage Rule Set (URS) #2							

$127 + X_1 + X_2$	
$128 + X_1 + X_2$: $127 + \Sigma_{v=1}^{N} X_v$	URS (#3 … #N)
$128 + \Sigma_{v=1}^{N} X_v$: $127 + 17*1 + \Sigma_{v=1}^{N} X_v$	Binding for Usage Rule Set (BURS) #1
:	⋮
$111 + 17*N + \Sigma_{v=1}^{N} X_v$: $127 + 17*N + \Sigma_{v=1}^{N} X_v$	BURS #N
$128 + 17*N + \Sigma_{v=1}^{N} X_v$: $143 + 17*N + \Sigma_{v=1}^{N} X_v$	TUF MAC

A Title Usage File consists of the following fields:

- URF_ID of 12 bytes which is an identifier of a Title Usage File. The value shall be "DVD_HD_V_TUF" with character set code of ISO/IEC 646:1983 (a-characters).
- HD_VURF_SIZE of 4 bytes which indicates the size in bytes of the Title Usage File. This size shall be equal to or less than 64 KB = 65536 bytes.

- URS_NUM of 1 byte which indicates the number of the Usage Rule Sets in the Title Usage File. This value shall be an integer which is equal to or more than zero and less than 2^8 = 256.
- The HASH_SIZE field of 4 bytes contains the size of the TUF excluding the Binding for Usage Rule Set fields and the TUF MAC. The hash value of the first HASH_SIZE bytes of the TUF is contained in the Content Hash Table #2.
- VERN of 4 bytes which indicates the version number of the Title Usage File. The value shall be 0 for the current version.
- PLAYLIST_NAME of 12 bytes which stores the name of the Playlist which is accompanied by this Title Usage File. The name shall be "VPLST%%%.XPL" or "APLST%%%.XPL", where %%% runs from 000 to 999. The AACS module in a Player which plays back an Advanced Content shall compare PLAYLIST_NAME with the name of a Playlist which is currently active. Unless the names are equal, the Usage Rule Sets in this TUF must not be adopted. Note that only one Playlist is active at one time. For a Standard Content which involves no Playlist for playback, this field shall be filled with FFh. An implication of this is that the AACS module in a Player should know which content the Player plays back, an Advanced Content or a Standard Content.
- A Usage Rule Set is a set of Usage Rules. If URS_NUM is equal to 0, there exists no Usage Rule Set. The format of a Usage Rule Set is described in Table 3-9.
- A Binding for Usage Rule Set (BURS) field corresponds to one and only one Usage Rule Set in order. This field consists of the following fields in order: a BIFO field of 1 byte and a Binding MAC field of 16 bytes. The format of a BIFO is shown in Table 3-6. Here, however, AV_FLG in BIFO shall always be 0b. The Binding MAC field stores the MAC value of the Usage Rule Set. The format of BURS is shown in Table 3-8. See also Figure 3-2 for reference.
- TUF MAC of 16 bytes. The field shall contain the CMAC value of the entire TUF except the TUF MAC field itself. The CMAC value is calculated using the Volume Unique Key (K_{vu}).

The reserved field in TUF shall be filled with 00h. Before using a TUF, a Player shall verify the TUF MAC. If the verification fails, the TUF shall not be used.

Table 3-8: Format of BURS

Bit / Byte	7	6	5	4	3	2	1	0

0	BIFO
1 : 16	Binding MAC

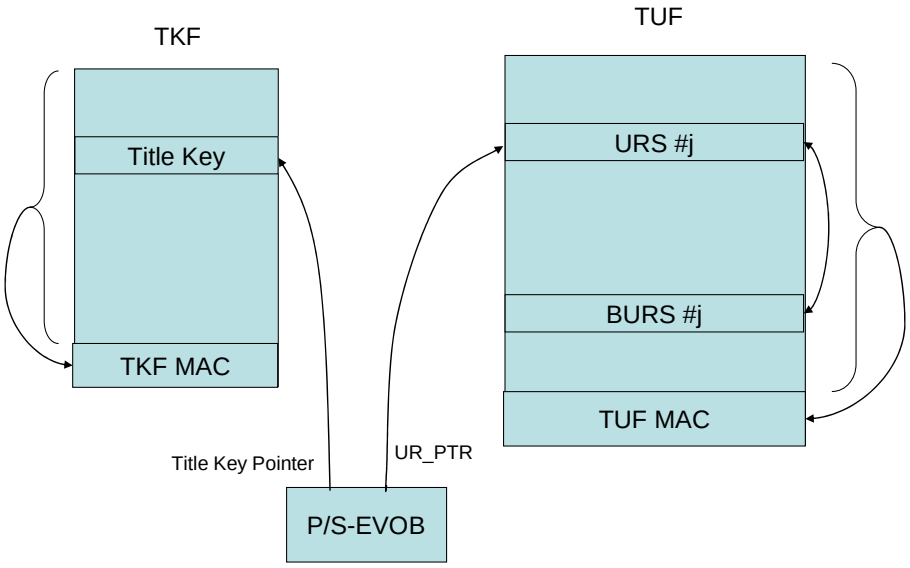

Figure 3-2: The Relationship between a BIFO and Binding MAC field and a Title Key

According to the BIND_TYPE in the BIFO, the Binding MAC field in the BURS stores the following value, where {URS} denotes the data of the corresponding URS:

000b Bound to the Volume ID:

The Binding MAC field shall be filled with 00h.

001b Bound to the Pre-recorded Media Serial Number (PMSN):

The Binding MAC field stores CMAC(PMSN, {URS}).

010b Bound to the Device Unique Nonce (DUN):

The Binding MAC field stores CMAC(DUN, {URS}).

011b Bound to both PMSN and DUN:

The Binding MAC field stores CMAC(DM, {URS}), where DM = AES-G(DUN, PMSN).

100b Bound to the Temporary Nonce

The Binding MAC field stores CMAC(TN, {URS}).

Others. Reserved.

Table 3-9 shows the format of Usage Rule Set. UR_PTR in an EVOB indicates the ordinal number of a URS in the TUF. A Player shall apply a Usage Rule in the URS to the EVOB if the Player recognizes the UR_ID for the Usage Rule. And, before the Usage Rule is applied, the BURS for the URS shall be verified, that is, the CMAC value shall be calculated and compared with the Binding MAC value. If they are not identical, the application of a Usage Rule in the URS is not allowed.

Table 3-9: Format of Usage Rule Set

Byte \ Bit	7	6	5	4	3	2	1	0	
0 1	URS_VERSION								Usage Rule Set Header
2 : 5	URS_SIZE (S_E)								
6 : 9	UR_NUM (N)								
10 : 12	UR_ ID #1								Usage Rule #1

13	UR_TYPE #1	
14 : 17	UR_SIZE #1 (S_1)	
18 : $S_1 + 9$	UR_BODY #1	
$S_1 + 10$: $S_1 + 12$	UR _ID #2	Usage Rule #2
$S_1 + 13$	UR_TYPE #2	
$S_1 + 14$: $S_1 + 17$	UR_SIZE #2 (S_2)	
$S_1 + 18$: $S_1 + S_2 + 9$	UR_BODY #2	
:	⋮	…
$S_E - S_N$: $S_E - S_N + 2$	UR_ ID #N	Usage Rule #N
$S_E - S_N + 3$	UR_TYPE #N	
$S_E - S_N + 4$: $S_E - S_N + 7$	UR_SIZE #N (S_N)	

$S_E - S_N + 8$: $S_E - 1$	UR_BODY #N	

A Usage Rule Set consists of the following fields:

- URS_VERSION of 2 bytes. This field serves for distinction of the same Usage Rule Set which has different BURS. Suppose there are two Usage Rule Sets the Usage Rules in which are identical. If the Usage Rule Sets are to have different Binding, this version field is used to distinguish the two Usage Rule Sets.
- URS _SIZE of 4 bytes which indicates the size of the Usage Rule Set. This size includes the size of URS_VERSION (2 bytes) and URS_SIZE (4 bytes) itself.
- UR_NUM of 4 bytes which indicates the number of Usage Rules in the Usage Rule Set.
- UR_ID of 3 bytes is an identifier of UR_BODY. If a Player recognizes the UR_ID, the Player shall process the Usage Rule stored in the following UR_BODY. If the Player does not recognize the UR_ID, the Player shall read the following UR_TYPE and decide its behavior. A Usage Rule Set shall not contain two or more Usage Rules which have the same UR_ID.
- UR_TYPE of 1 byte. The value stored in this field shall be 00h or 10h. Other values are reserved. Suppose a Player does not recognize the preceding UR_ID. Then, UR_TYPE tells a Player what the behavior for the Usage Rule should be.

  00h: Ignore the Usage Rule and play back the EVOB.

  10h: Immediately go to Stop State.

- UR_SIZE of 4 bytes which indicates the size in bytes of the Usage Rule including the UR_ID, the UR_TYPE and the UR_SIZE itself.
- UR_BODY is a description of usage for contents.

A Player shall check all the Usage Rules in a Usage Rule Set associated with an EVOB. The priorities of Usage Rules are decided as follows:

i) UR_ID is not recognizable & UR_TYPE = 00h.

ii) UR_ID is recognizable

iii) UR_ID is not recognizable & UR_TYPE = 10h.

The larger the number is, the higher the priority is. If two Usage Rules have the same priority, the earlier it appears in a URS, the lower the priority is. Note that a Player shall process all the Usage Rules if all the Usage Rules in a URS are recognizable or ignorable (i.e. i) above).

CCI for Update is one of Title Usage Rules. Table 3-10 shows the format of the Usage Rule of CCI for Update. The UR_ID of CCI for Update shall be 000001h and the UR_SIZE shall be 10.

Table 3-10: Format of the Usage Rule of CCI for Update

Bit / Byte	7	6	5	4	3	2	1	0
0	PCCI			APSTB			ICT	DOT
1	Reserved							PRFG

The CCI field consists of the following fields:

- PCCI of 3 bits which stores the status of primitive copy control information for the EVOB associated with this Usage Rule Set.

  000b: Copy Freely

  100b: Copy One Generation

  010b: No More Copies

  110b: Copy Never

  011b: Encryption Plus Non-Assertion. This status means that the encrypted EVOB can be copied freely without re-distribution.

- APSTB of 3 bits which indicates a status of analog protection for the EVOB associated with this Usage Rule Set:

  000b: APSTB is OFF.

  001b: Type 1 of APS1 is ON.

  010b: Type 2 of APS1 is ON.

011b: Type 3 of APS1 is ON.

110b: APS2 is ON.

111b: APS2 is ON.

Other Combinations: Reserved.

- ICT of 1 bit which indicates the status of the analog output for the EVOB associated with this Usage Rule Set.

  0b: High Definition Analog Output in High Definition Analog Form allowed.

  1b: High Definition Analog Output in the form of Constrained Image allowed.

- DOT of 1 bit: This flag indicates the Analog Output status. If this flag is 1b, an analog output of the decoded video of the EVOB is not allowed.
- The reserved field shall be filled with 0b.
- Priority Flag (PRFG) of 1 bit which indicates which CCI is valid: the one embedded in the associated EVOB or this one.

  0b: The CCI embedded in the EVOB is valid.

  1b: The CCI in this Usage Rule Set is valid and overrides the embedded CCI.

Time-Based Conditions are supported by an AACS-Compliant Player which supports a Secure Clock. Support of a Secure Clock by an AACS-Compliant Player is optional. The implementation of a Secure Clock may vary. It may be implemented based on a secure Internet connection which is proprietary to each Player as long as the AACS Compliance Rules for a Secure Clock are satisfied. Time-Based Conditions are a Title Usage Rule. The format of Time-Based Conditions is shown in Table 3-11. The UR_ID of time-based conditions shall be 000002h. A Player which supports no Secure Clock *must not* recognize a Usage Rule of Time-Based Conditions. The UR_SIZE of Time-Based Conditions is 18.

Table 3-11: Usage Rule of Time-Based Conditions

Bit / Byte	7	6	5	4	3	2	1	0
0	PP_FLG	PERMISSION_PERIOD						
1								

2 : 5	VA_FLG	VALID_AFTER
6 : 9	VB_FLG	VALID_BEFORE

Time-Based Conditions consist of the following fields:

- PP_FLG of 1 bit. This flag indicates the following PERMISSION_PERIOD field is valid or not. The PERMISSION_PERIOD is valid if and only if this flag is 1b.
- The 15-bit field of PERMISSION_PERIOD which indicates the "period" attribute defined in the AACS *Introduction and Common Cryptographic Elements*. The unit of time is hour.
- VA_FLG of 1 bit. This flag indicates the following VALID_AFTER field is valid or not. The VALID_AFTER is valid if and only if this flag is 1b.
- The field of VALID_AFTER of 4 bytes which indicates the "after" attribute defined in the AACS *Introduction and Common Cryptographic Elements*. The unit of time is hour and the zero time reference is zero o'clock of the 1st of July in 2005 (GMT).
- VB_FLG of 1 bit. This flag indicates the following VALID_BEFORE field is valid or not. The VALID_BEFORE is valid if and only if this flag is 1b.
- The field of VALID_BEFORE of 4 bytes which indicates the "before" attribute defined in the AACS *Introduction and Common Cryptographic Elements*. The unit of time is hour and the zero time reference is zero o'clock of the 1st of July in 2005 (GMT).

Suppose that a Player supports a Secure Clock. The Player must not play back the EVOB if at least one of the three time-based conditions, PERMISSION_PERIOD, VALID_AFTER and VALID_BEFORE, is valid and if all the valid conditions are not satisfied. In this case, the Player shall create an instance of a system event, Aacs TimeBased. The system event shall be fired even during trick play. The Title Timeline shall pass in accordance with the playback mode, i.e. forward/backward scan or normal playback, without playing back the concerned EVOB. The default action for the event immediately makes the Player go to Stop State. Event Object for Aacs TimeBased is AacsTimeBased which has "TIMEBASED" as Name and "aacs_timebased" as Value. Event Type of AacsTimeBased is "aacs_timebased". AacsTimeBased is a

cancelable Event. Conversely, if none of the three time-based conditions is valid, or if all the valid conditions are satisfied, the Player shall decrypt and play back the EVOB.

During an event process of AacsTimeBased, the Title Timeline progresses if the associated EVOB is Soft Synchronized or if the EVOB is associated to a Soft-Sync Application. Note that the EVOB shall not be played back while the Title Timeline progresses in this case. Otherwise, the Title Timeline shall hold during the event process. If an Advanced Application does not cancel the default action for the event, the Player shall go to Stop State.

The UR of Time-Based Conditions is valid only for an EVOB which belongs to an Advanced VTS. If a Player encounters in a Standard VTS an EVOB which has the Time-Based Conditions, the Player shall immediately go to Stop State, because no event can be handled in playback of a Standard VTS.

A description by a Right Expression Language (REL) may be included in a TUF as a Usage Rule. UR_ID for REL UR shall be 000003h. The UR_SIZE is $N_S + 11$, where N_S is the value stored in the STRING_LENGTH field. A Player which does not recognize the UR_ID shall ignore the Usage Rule. The format of REL Usage Rule is shown in Table 3-12.

Table 3-12: Format of REL Usage Rule

Bit / Byte	7	6	5	4	3	2	1	0
0	STRING_TYPE		Reserved					
1 2	STRING_LENGTH (N_S)							
3 : $2 + N_S$	REL_STRING							

The REL Usage Rule consists of the following fields:

- STRING_TYPE of 2 bits:

  00b: REL_STRING is a description of right in a REL.

01b: REL_STRING is the name of a file which contains a description of right in a REL.

10b: Reserved.

11b: Reserved.

- A reserved field of 6 bits.
- STRING_LENGTH of 2 bytes which stores the length, N_S, of the following REL_STRING field. For the reserved value of the preceding STRING_TYPE, this field shall store 0000h.
- REL_STRING of N_S bytes. This field contains a file name if the preceding STRING_TYPE is 01b. This field contains a right description if the preceding STRING_TYPE is 00b. For the reserved values of the STRING_TYPE, this field shall not exist.

Usage Rule of Output Control Bits are one of the Usage Rules. UR_ID for Usage Rule of Output Control Bits shall be 000004h. The UR_SIZE is 24. Currently, no Player shall recognize this Usage Rule. The format of Usage Rule of Output Control Bits is shown in Table 3-13.

Table 3-13: Format of Output Control Bits

Bit / Byte	7	6	5	4	3	2	1	0
0 : 15	Output Control Bits							

- The field of Output Control Bits contains Output Control Bits. See the Compliance Rules for Output Control Bits and the semantics.

A URS is associated with an EVOB, not with a Title in this Book. Here, the reason is briefly explained: The concept of Title lies in the application layer in the *HD DVD-Video Specifications*. In fact, a Title is a node in the DOM tree which is obtained by means of parsing the Playlist by XML Parser. Therefore, the Titles are information which resides outside of the AACS module. On the other hand, in general, the AACS module in a Player can recognize the lower level structures such as EVOB or TUF, and it can directly

process them. Thus, EVOB-TUF binding is more secure than Title-TUF binding. This is why EVOB-TUF binding is adopted in this Book.

3.6 Content Certificate

An AACS Disc shall store one and only one Content Certificate file. The Content Certificate file on an AACS Disc shall reside in the "AACS" directory. The name "CONTENT_CERT.AACS" is reserved for the Content Certificate file. A Content Certificate file has the format shown in Table 3-14.

Table 3-14: Format of Content Certificate on an AACS Disc for HD DVD-Video

Byte \ Bit	7	6	5	4	3	2	1	0
0	Certificate Type: 00_{16}							
1	Reserved							
2 : 5	Total_Number_of_HashUnits							
6	Total_Number_of_Layers							
7	Layer_Number							
8 : 11	Reserved							
12 13	Number_of_Digests							
14 15	Applicant ID							
16 … 19	Content Sequence Number							

20 21	Minimum CRL Version
22 23	Reserved
24 25	Length_Format_Specific_Section
26 : 39	Reserved
40 : 59	Content Hash Table Digest #1
60 : 79	Content Hash Table Digest #2
80 : 119	Signature Data

Content Certificate ID for an AACS Disc is a combination of the 2-byte field of Applicant ID and the 4-byte field of Content Sequence Number. See the *AACS Pre-recorded Video Book* for Content Certificate ID and CRL. The meaning of the fields in the Content Certificate on an AACS Disc is described in the AACS *Pre-recorded Video Book* except following fields:

- A 4-byte Total_Number_of_HashUnits field indicates the total number of hashes in CHT #1 and CHT #2 on the optical media.
- A 1-byte Total_Number_of_Layers field indicates the total number of layers on the optical media. For an HD DVD ROM medium and a DVD ROM media, this field shall store 01h.
- Layer_Number of 2 bytes which shall be 0000h for an AACS-Protected (HD) DVD-Video ROM.
- Number_of_Digests of 2 bytes which stores 0002h.
- Length_Format_Specific_Section of 2 byte which shall be 000Eh.

- Content Hash Table Digest #1 of 20 bytes which contains the SHA-1 hash value of the Content Hash Table #1 (CHT #1). CHT #1 contains all the hash values of the Hash Units in the P/S-EVOBs on the Disc. The details of CHT #1 are described in Subsection 3.7.1.
- Content Hash Table Digest #2 of 20 bytes which contains the SHA-1 hash value of the Content Hash Table #2 (CHT #2). CHT #2 contains hash values of the navigation data, i.e. all the XML document files and all the ECMAScript files on the Disc. In addition, CHT #2 contains the hash values of TKFs and TUFs. The details of CHT #2 are described in Subsection 3.7.2.
- All the reserved fields shall be filled with 00h.

3.7 Content Hash

3.7.1 Content Hash Table #1

An AACS Disc shall store two Content Hash Tables (CHTs): Content Hash Table #1 (CHT #1) and Content Hash Table #2 (CHT #2). The CHTs on a Disc shall reside in the "AACS" directory. The name "CONTENT_HASH_TABLE1.AACS" is reserved for the CHT #1 for P/S-EVOBs. CHT #1 has the format shown in Table 3-15. CHT #1 contains the eight-byte hash values of all the EVOBU/TUs in P/S-EVOBs for a Standard Content or Advanced Contents on an AACS Disc.

Table 3-15: Format of Content Hash Table #1

Bit / Byte	7	6	5	4	3	2	1	0
0 : 3	Number of Hash Values (NHV)							
4 : 7	Reserved							
8 : 15	Hash Value of EVOBU #1							

16 : 23	Hash Value of EVOBU #2
…	…
8*NHV : 7 + 8*NHV	Hash Value of EVOBU #NHV

CHT #1 consists of the following fields:

- Number of Hash Values (NHV) of 4 bytes which indicates the total number of Hash Values in the CHT. The NHV shall not exceed 500000.
- A series of 8-byte Hash Values of EVOBUs or TUs, each of which stores the hash value calculated from the corresponding EVOBU or TU. The hash value is the lsb 64 bits of the SHA-1 hash value. The SHA-1 hash value shall be calculated regardless of whether the EVOBU/TU is encrypted or not, which means that a Player need not decrypt the EVOBU/TU before checking the hash value.

Note that hash values of EVOBUs in an ILVU in a P-EVOB for an Angle Block shall be contained in CHT #1. Hash values of EVOBUs in an ILVUs in a P-EVOB for a Sequence Key Section (See Chapter 7) shall also be compiled into CHT #1.

The total number of the EVOBU/TUs on an AACS Disc shall not exceed 500000. If a Player encounters an EVOBU/TU whose Content Hash Pointer (CH_PTR) exceeds the NHV, the Player shall immediately go to Stop State.

3.7.2 Content Hash Table #2

The name “CONTENT_HASH_TABLE2.AACS” is reserved for the CHT #2. The CHT #2 has the format shown in Table 3-16. The CHT #2 on an AACS Disc contains the eight-byte hash value of Configuration File “/ADV_OBJ/DISCID.DAT”, Directory Key File, the full 20-byte hash value of Managed Copy Manifest, the twenty-byte hash values of the TUFs on the Disc, the eight-byte hash values of all the XML Documents and all the ECMAScript codes on the Disc. See Section 6.3 for the Configuration File and the Directory Key File. The maximum number of Playlists for Video and Audio are 1000 respectively. And

there is a TUF for a Standard Content, “VTUF.AACS”. Therefore, the CHT #2 has 2001 hash entries for TUFs.

Table 3-16: Format of Content Hash Table #2 on an AACS Disc

Byte \ Bit	7	6	5	4	3	2	1	0
0 : 7	Hash of DISCID.DAT							
8 : 15	Hash of Directory Key File							
16 : 35	Hash of MNGCPY_MANIFEST							
36 : 55	Hash of VTUF.AACS							
56 : 20055	Hash of VTUF000.AACS – Hash of VTUF999.AACS							
20056 : 40055	Hash of ATUF000.AACS – Hash of ATUF999.AACS							
40056 : 40059	Number of Hashes of ANFs (NHA)							
40060 : 40067	Hash of ANF #1							

40068 : 40059 + 8*NHA	Hash of ANF #2 – Hash of ANF #NHA

CHT #2 consists of the following fields:

- The eight-byte hash value of Configuration File “DISCID.DAT” in the “ADV_OBJ” directory. The hash value is the least significant 64 bits of the calculated result of the SHA-1 function. For a Category 1 Disc, which has no Configuration File, this field shall be filled with FFh. The Configuration File is described in the *HD DVD-Video Specifications*. (See also Section 6.3.) In the boot sequence, an AACS-Compliant Player shall verify the hash value of Configuration File using this field.
- The eight-byte hash value of DKF, Directory Key File. See Section 6.3 for DKF. The hash value is the least significant 64 bits of the calculated result of the SHA-1 function. Before the AACS module in a Player uses the Configuration File, the hash value of the DKF shall be verified using the value stored in this field.
- Hash of MNGCPY_MANIFEST stores the 20-bytes value of the SHA-1 hash function applied to Managed Copy Manifest File “MNGCPY_MANIFEST.XML”. See 5.3 for Managed Copy Manifest File. Note that Managed Copy Manifest File shall not be protected in Encapsulation Format for Hash and that the Hash Pointer filed in the encapsulation format shall indicate the offset of this field, i.e. 16. The following Hash of ANF fields must not contain the hash value of Managed Copy Manifest File.
- The hash value of “VTUF.AACS” which is the calculated result of the SHA-1 function applied to the first HASH_SIZE bytes of “VTUF.AACS”, where HASH_SIZE is a field in the TUF. If the VTUF.AACS is absent, this field shall be filled with FFh.
- The hash values of VTUF000.AACS, VTUF001.AACS, …, VTUF999.AACS. They are the SHA-1 values. The hash function is applied to the first HASH_SIZE bytes of each TUF, where HASH_SIZE is a field in the TUF. If one of the VTUFs is absent, the corresponding field shall be filled with FFh.
- The hash values of ATUF000.AACS, ATUF001.AACS, …, ATUF999.AACS. They are the SHA-1 values. The hash function is applied to the first HASH_SIZE bytes of each TUF, where HASH_SIZE is a field in the TUF. If one of the ATUFs is absent, the corresponding field shall be filled with FFh.
- The Number of Hashes of ANFs (NHA) of 4 bytes. The word “ANF” is an abbreviation of “Advanced Navigation File” which means an XML Document or ECMAScript Codes. This number shall be equal to or more than zero and less than 20000. Each hash unit size of an encrypted ANF is equal to or less than

512 bytes. Therefore, the maximum total size of all the encrypted ANFs is about 10 MB. On the other hand, a non-encrypted ANF has only one hash entry in CHT #2. Thus, the maximum total size of all the ANFs may be more than 10 MB. See Section 4.4.2 for reference.

- The eight-byte hash values of the hash units in the ANFs. The hash value is the least significant 64 bits of the calculated result of the SHA-1 function. The number of the hash values is equal to the NHA.

3.8 Content Revocation List

An AACS-Protected Disc shall store one and only one Content Revocation List (CRL) file. The CRL shall reside in the "AACS" directory. The name "CONTENT_REVOCATION_LIST.AACS" is reserved for the CRL file. The format of a CRL file is defined in the *AACS Pre-recorded Video Book.*

3.9 Boot Sequence for Disc Application

In the Startup Sequence of Advanced Content, a Playlist shall have the name, "VPLST%%%.XPL" or "APLST%%%.XPL", where %%% runs from 000 to 999. On the other hand, the HD DVD-Video Specifications does not specify a filename which may be used as a Playlist in a Soft Reset. This Book puts a restriction, however, on a Playlist filename for a Soft Reset: A Playlist on a Disc which may be used as an argument of Playlist.load() shall have the name, "VPLST%%%.XPL" or "APLST%%%.XPL", where %%% runs from 000 to 999.

When a Playlist, say, VPLST011.XPL on a Disc is loaded in the boot sequence, the associated AACS components will be automatically read and processed by the AACS module. The following associated AACS components shall reside in the directory /AACS: VTKF011.AACS. And the following associated AACS components may reside in the directory /AACS: VTUF011.AACS. If the associated TUF is absent, there is no TUF for EVOBs to be played back in the Playlist. The following is one example of the boot sequence:

1. The AACS module at first reads and processes /AACS/MKBROM.AACS.
2. The AACS module then reads and processes /AACS/TKF011.AACS so that the AACS module may prepare Title Keys.
   - TKF MAC is verified.
3. If /AACS/SKB.AACS and /AACS/SKF.AACS exist, the AACS module processes them to yield six SKTs. (See Chapter 7.)

4. The AACS module verifies the signature of /AACS/CONTENT_CERT.AACS.
5. The AACS module checks integrity of /AACS/CONTENT_HASH_TABLE1 and /AACS/CONTENT_HASH_TABLE2, respectively, based on the hash values recorded in /AACS/CONTENT_CERT.AACS.
6. If /AACS/VTUF011.AACS exists, its integrity is checked using the hash value recorded in /AACS/CONTENT_HASH_TABLE2.
    - TUF MAC is verified.

If one of the steps fails in the preceding boot sequence, the system immediately goes to Stop State. After the preceding boot sequence for AACS components is successfully executed, the system proceeds to process the Playlist, /ADV_OBJ/VPLST011.XPL.

Before a Title Key in a TKF is used, the associated Binding MAC shall be checked. If it fails, the Title Key must not be used. When Before a URS in a TUF is used, the associated BURS shall be checked. If it fails, the URS must not be applied.

3.10 Backups

All the AACS components except the “MKBRECORDABLE.AACS” shall have its backup image in the directory “AACS_BAK”. An AACS-Compliant Player may use any of the backup components if it decides that it is not be able to correctly read the original components. The following is a list of the backup components:

- /AACS_BAK/MKBROM.AACS.
- /AACS_BAK/SKB.AACS and /AACS_BAK/SKF.AACS (if they exist).
- /AACS_BAK/VTKF.AACS (for the Standard Contents), /AACS_BAK/VTKF%%%.AACS, /AACS_BAK/ATKF%%%.AACS (if the corresponding Playlist exists).
- /AACS_BAK/VTUF.AACS (for the Standard Contents), /AACS_BAK/VTUF%%%.AACS, /AACS_BAK/ATUF%%%.AACS (if the corresponding Playlist exists).
- /AACS_BAK/CONTENT_CERT.AACS.
- /AACS_BAK/CONTENT_HASH_TABLE1.AACS.
- /AACS_BAK/CONTENT_HASH_TABLE2.AACS.

- /AACS_BAK/CONTENT_REVOCATION_LIST.AACS.
- /AACS_BAK/MNGCPY_MANIFEST.XML.

It is recommended that the original component and the corresponding backup component are recorded in a physically separated manner.

This page is intentionally left blank.

Chapter 4
Protection of an HD DVD-Video Content on a Medium

4 Protection of an HD DVD-Video Content on a Medium

4.1 Introduction

This section describes the details of encryption and decryption of an HD DVD-Video Content on an HD DVD-Video ROM medium. The HD DVD-Video format is licensed by the DVD Forum, which publishes the *HD DVD-Video Specifications*. The following 2 types of content are defined in the specification:

- Standard Content that consists of Navigation data and Video object data.
- Advanced Content that consists of Advanced Navigation, Primary/Secondary Video Set and Advanced Element.

Figure 4-1 shows an overview of elements of an HD DVD-Video Content which are targets of the AACS content protection. There are roughly two protection formats: One is a protection format for a P/S-EVOB. An EVOB may include an audiovisual content, an audio-only content or other contents. An EVOB is a series of Packs each of which has length of 2048 bytes. Encryption of an EVOB is performed on a Pack basis. The protection format for an EVOB is defined in Section 4.3. The other is a protection format for Advanced Resources. Advanced Resources are the sum of two data sets, Advanced Navigation and Advanced Element. Advanced Resources are recorded as files on an HD DVD-Video ROM medium, and, therefore, protection of Advanced Resources is performed on a file basis. A file which contains data of Advanced Resources is called Advanced Resource File (ARF) in this specification. The protection format of ARF is defined in Section 4.4.

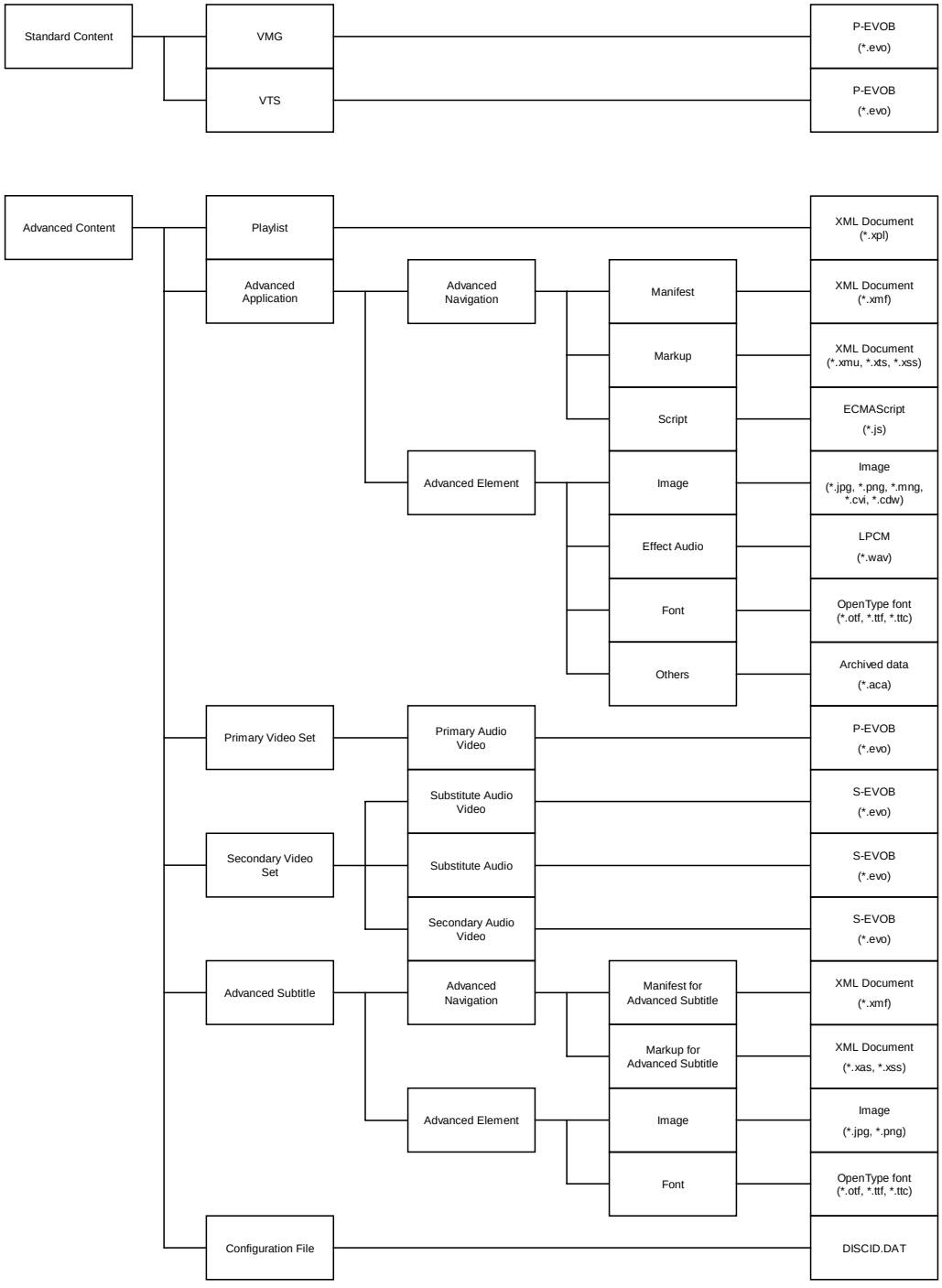

Figure 4-1: Protection Targets of an HD DVD-Video Content on an HD DVD-Video ROM Medium

4.2 Stored data values in CPI field

This section describes stored data values in a Content Protection Information (CPI) field. A CPI field is located in a GCI packet (GCI_PKT). A GCI_PKT is contained in an NV_PCK of an EVOBU in a P -EVOB or in the first Pack of TU of an S-EVOB. See the *HD DVD-Video Specifications* for the detailed location of CPI field in a GCI_PKT.

Table 4-1: Stored data values in Content Protection Information (CPI)

Bit / Byte	7	6	5	4	3	2	1	0
0	Key Management Information							
1								
2								
3								
4	Content Hash Management Information							
5								
6								
7								
8	Usage Rule Management Information							
9								
10	CCI_SS							
11								
12	CCI							
13								
14	Reserved							
15								

A CPI consists of the following values:

- Key Management Information (KMI) of 4 bytes which stores information of the key to encrypt the encrypted PCKs in the EVOB containing this CPI. The format of KMI is described on Table 4-2.

- Content Hash Management Information (CHMI) of 4 bytes stores information of the Content Hash for the EVOBU containing this CPI. The format of CHMI is described in Table 4-3.
- Usage Rule Management Information (URMI) of 2 bytes which indicates information of the Usage Rule for the EVOB containing this CPI. The format of URMI is described in Table 4-4.
- CCI_SS of 2 bytes which indicates the status of CCI for the EVOB containing this CPI. The format of CCI_SS is described in Table 4-5.
- CCI of 2 bytes which stores copy control information for the EVOB containing this CPI. The format of CCI is described in Table 4-6.
- A reserved field of 2 bytes which shall be filled with 00h.

Table 4-2: Stored data value in Key Management Information (KMI)

Bit / Byte	7	6	5	4	3	2	1	0
0	KEY_VF		Reserved					
1	TITLE_KEY_PTR							
2								
3	SEG_KEY_PTR							

The KMI consists of the following fields:

- Key Validity Flag (KEY_VF) of 2 bits which indicates the status of the Key Pointer fields:

  00b: Neither the Segment Key Pointer nor the Title Key Pointer is valid.

  01b: The Segment Key Pointer is valid.

  10b: The Title Key Pointer is valid.

  11b: Reserved.
- The reserved field shall be filled with 0b.
- Title Key Pointer value (TITLE_KEY_PTR) of 2 bytes which indicates the entry number of a Title Key in the Title Key File. This value shall be greater than 0 and equal to or less than 64. If TITLE_KEY_PTR is 0009h, for instance, the encrypted PCKs in the EVOB which contains this

KMI are encrypted by the Title Key (K_t) #9. When the KEY_VF is 00b, 01b or 11b, this field shall be filled with 00h.

- Segment Key Pointer (SEG_KEY_PTR) of 1 byte which indicates an entry in the Segment Key Table which contains a Segment Key. The usage of this field is described in Chapter 7. If KEY_VF is 00b, 10b or 11b, this field shall be filled with 0h.

Table 4-3: Stored data value in Content Hash Management Information (CHMI)

Bit / Byte	7	6	5	4	3	2	1	0
4 : 7	CH_PTR							

The CHMI consists of the following fields:

- Content Hash Pointer (CH_PTR) of 32 bits which indicates the number of an entry of the CHT #1. For example, if the CH_PTR is 00000009h, the hash value of the current EVOBU shall be identical to the value stored in the Hash Value of EVOBU #9 in the CHT #1. CH_PTR shall be more than zero and equal to or less than 500000.

Note that the following conditions hold on an AACS Disc:

a) CH_PTR is unique.

b) 1 <= CH_PTR <= NHV.

c) Let p be an integer such that 1 <= p <= NHV. There exists an EVOBU or a TU whose CH_PTR is identical to p.

Table 4-4: Stored data value in Usage Rule Management Information (URMI)

Bit / Byte	7	6	5	4	3	2	1	0
8	UR_VF	UR_PTR						

9	

The URMI shall consist of the following fields:

- Usage Rule Validity Flag (UR_VF) of 1 bit which indicates the status of the Usage Rule Pointer field. When the Usage Rule Pointer field is valid, this field shall be 1b. Otherwise, this field shall be equal to 0b. If URS_NUM in the TUF is zero, UR_VF shall be 0b.
- Usage Rule Pointer (UR_PTR) of 15 bits which indicates the ordinal number of a Usage Rule Set in the Title Usage File. If UR_PTR is equal to 000b || 009h, for example, usage of the EVOB which contains this URMI shall be governed by the Usage Rule Set #9. It shall hold that 1 <= UR_PTR <= 255. When the UR_VF is set to 0b, this field shall be filled with 1b.

Table 4-5: Stored data value in CCI_SS

Bit / Byte	7	6	5	4	3	2	1	0
10	PCCI_VF	APS_VF	ICT_VF	DOT_VF	Reserved			
11	Reserved							

The CCI_SS shall consist of the following fields:

- PCCI Validity Flag (PCCI_VF) of 1 bit which indicates validity of the PCCI field in the CCI. If the PCCI field is valid, this field shall be equal to 1b. Otherwise, this field shall be 0b.
- APS Validity Flag (APS_VF) of 1 bit which indicates validity of the APSTB field in the CCI. If the APSTB field is valid, this field shall be equal to 1b. Otherwise, this field shall be equal to 0b.
- ICT Validity Flag (ICT_VF) of 1 bit which indicates validity of the ICT field in the CCI. If the ICT field is valid, this field shall be equal to 1b. Otherwise, this field shall be 0b.
- DOT Validity Flag (DOT_VF) of 1 bit which indicates validity of the DOT field in the CCI. If the DOT field is valid, this field shall be equal to 1b. Otherwise, this field shall be equal to 0b.
- The reserved fields shall be filled with 0b.

Table 4-6: Stored data value in CCI

Bit / Byte	7	6	5	4	3	2	1	0
12	PCCI			APSTB			ICT	DOT
13	Reserved							

The CCI shall consist of the following fields:

- PCCI of 3 bits which stores the status of primitive copy control information for the EVOBU/TU which contains this CCI. When the PCCI_VF is equal to 0b, the PCCI field shall be 000b.

  000b: Copy Freely

  100b: Copy One Generation

  010b: No More Copies

  110b: Copy Never

  011b: Encryption Plus Non-Assertion (This status means that the encrypted EVOB can be copied freely without re-distribution.)

- APSTB of 3 bits which indicates a status of analog protection for the EVOB:

  000b: APSTB is OFF.

  001b: Type 1 of APS1 is ON.

  010b: Type 2 of APS1 is ON.

  011b: Type 3 of APS1 is ON.

  110b: APS2 is ON.

  111b: APS2 is ON.

  Other Combinations: Reserved.

- ICT of 1 bit which indicates the status of the analog output for the EVOBU/TU which contains this CCI. When the ICT_VF is 0b, the ICT field shall be equal to 0b.

  0b: High Definition Analog Output in High Definition Analog Form allowed.

  1b: High Definition Analog Output in the form of Constrained Image allowed.

- DOT of 1 bit: This flag indicates the Analog Output status. If this flag is 1b, an analog output of the decoded video of the EVOB is not allowed.

- The reserved field shall be filled with 0b.

If both Main Video and Sub Video are displayed, the CCI setting shall be equivalent to that of the Main Video.

4.3 Protection format for EVOB

This section describes the protection format for an EVOB. An EVOB is protected by content encryption on a Pack basis using a Title Key. The encrypted Title Key is stored in the Title Key File which is described in Chapter 3. Each encrypted Pack of an EVOB is able to be decrypted by the Title Key which is indicated by TITLE_KEY_PTR of the CPI field. If an EVOBU/TU contains an encrypted Pack, the EVOBU/TU shall have a valid TITLE_KEY_PTR in the CPI field. Otherwise, the EVOBU/TU shall not have a valid TITLE_KEY_PTR and the KEY_VF shall be 0b. A Title Key shall not be changed within one P/S-EVOB. A Title Key may be shared by plural EVOBs. Figure 4-3 shows an example of Title Key assignment to EVOBs.

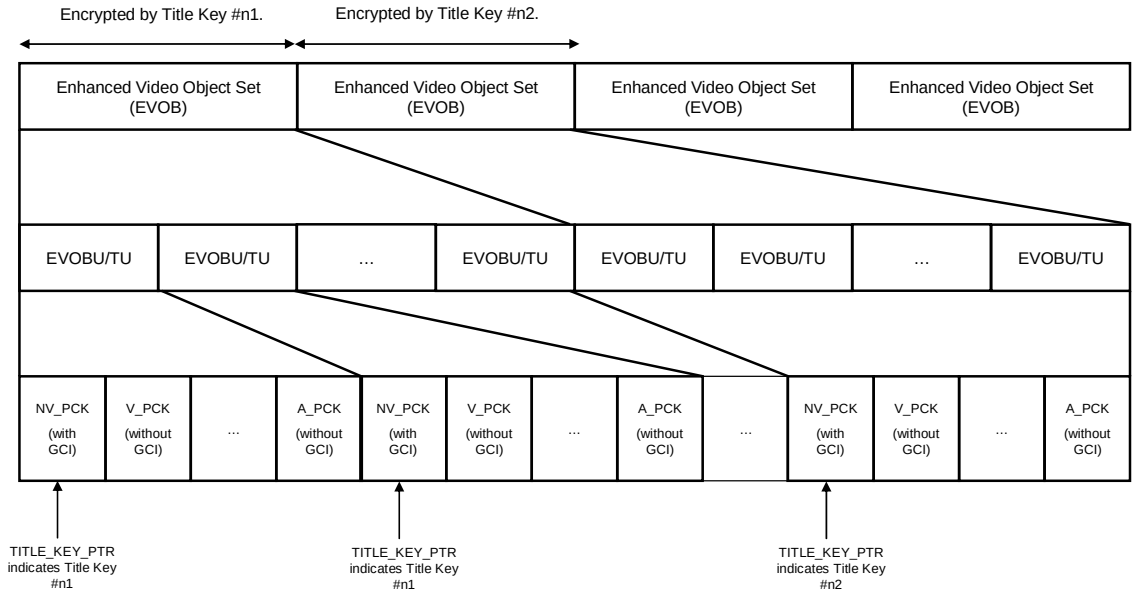

Figure 4-2: An Example of Title Key assignment to EVOBs

4.3.1 Pack Types

This section lists the Pack types which may be protected. A P-EVOB which belongs to a Standard Content consists of the following 5 types of Pack:

- Navigation Pack (NV_PCK)
- Video Pack (V_PCK)
- Audio Pack (A_PCK)
- Sub-picture Pack (SP_PCK)
- Highlight Information Pack (HL_PCK)

NV_PCK is not allowed to be encrypted.

A P-EVOB which belongs to an Advanced Content consists of the following 7 types of Pack:

- Navigation Pack (NV_PCK)
- Main Video Pack (VM_PCK)
- Sub Video Pack (VS_PCK)
- Main Audio Pack (AM_PCK)
- Sub Audio Pack (AS_PCK)
- Sub-picture Pack (SP_PCK)
- Advanced Pack (ADV_PCK)

NV_PCK and ADV_PCK are not allowed to be encrypted. Note that an encrypted ARF may be recorded as a file on an AACS Disc or multiplexed into an ADV_PCK in a P-EVOB. The same is true for an ARF with MAC.

An S-EVOB which belongs to an Advanced Content consists of the following 5 types of Pack:

- Navigation Pack (NV_PCK)
- Main Video Pack (VM_PCK)
- Sub Video Pack (VS_PCK)
- Main Audio Pack (AM_PCK)
- Sub Audio Pack (AS_PCK)

The first Pack in EVOBU/TU is not allowed to be encrypted.

4.3.2 Pack Encryption

Table 4-7 shows the Pack encryption format. For each encrypted Pack, the first 128 bytes are called the Unencrypted Portion and the remaining 1920 bytes are called the Encrypted Portion. The Unencrypted Portion is unencrypted and the Encrypted Portion is encrypted.

Each Encrypted Pack is encrypted by a 128-bit Content Key (K_c). The Content Key (K_c) is calculated by a 128-bit Title Key (K_t), a 32-bit Title Key Data (D_{tk}) and the least significant 96 bits of the CPI field in the GCI_PKT as follows:

$K_c = AES\text{-}G\ (K_t, D_{tk} \| CPI_{lsb_96})$,

where AES-G denotes the one-way function based on the AES algorithm defined in the *AACS Introduction and Common Cryptographic Elements*.

The Encrypted Portion is encrypted as follows:

$C_e = AES\text{-}128CBCE(K_c, C)$,

where AES-128CBCE denotes encryption by the AES algorithm in the CBC mode defined in the *AACS Introduction and Common Cryptographic Elements*. When a Pack is encrypted, the 2-bit PES_scrambling_control shall be 01b. Otherwise, the PES_scrambling_control shall be 00b.

Table 4-7: Encrypted Pack Format

	Bit / Byte	7	6	5	4	3	2	1	0
Unencrypted Portion (128 bytes)	**0 : 19**	Data defined in the *HD DVD-Video specification*							
	20			PES_scrambling _control					
	21 : 83	Data defined in the *HD DVD-Video specification*							

	84 **:** **87**	Title Key Data (D_{tk})
	88 **:** **127**	Data defined in the *HD DVD-Video specification*
Encrypted Portion (1920 bytes)	**128** **:** **2047**	Encrypted Content

Table 4-8 shows the Pack encryption format for HL_PCK. For each HL_PCK, the first 128 bytes are called the Unencrypted Portion and the remaining 1920 bytes are called the Encrypted Portion. The Unencrypted Portion is unencrypted and the Encrypted Portion is encrypted. Note that every HL_PCK on an AACS Disc is encrypted.

Each HL_PCK is encrypted by a 128-bit Content Key (K_c). The Content Key (K_c) is calculated by a 128-bit Title Key (K_t), a 32-bit Title Key Data (D_{tk}) and the least significant 96 bits of the CPI field in the GCI_PKT as follows:

$K_c = AES\text{-}G\ (K_t, D_{tk} \parallel CPI_{lsb_96})$,

where AES-G denotes the one-way function based on the AES algorithm defined in the *AACS Introduction and Common Cryptographic Elements*.

The Encrypted Portion is encrypted as follows:

$C_e = AES\text{-}128CBCE(K_c, C)$,

where AES-128CBCE denotes encryption by the AES algorithm in the CBC mode defined in the *AACS Introduction and Common Cryptographic Elements*.

Table 4-8: Encrypted Pack Format for HL_PCK

	Bit / Byte	7	6	5	4	3	2	1	0
Unencrypted Portion (128 bytes)	**0 : 83**	Data defined in the *HD DVD-Video specification*							
	84 : 87	Title Key Data (D_{tk})							
	88 : 127	Data defined in the *HD DVD-Video specification*							
Encrypted Portion (1920 bytes)	**128 : 2047**	Encrypted Content							

4.3.3 Content Hash Check for EVOBs

Before checking Content Hash of P-EVOBs on an AACS Disc, the integrity of Content Hash Table #1 shall be verified based on the Content Certificate File. There are two ways of content hash checking. One is to check 7 hash values which are randomly chosen from all the P-EVOBs on the disc. This content hash checking shall be performed before playback, and playback shall not start unless all the content hash values are respectively equal to the values stored in the CHT #1.

Another way of Content Hash Checking is to check randomly chosen 1% of the hash values while playback: Before starting playback, a Player loads the CHT #1 into the secure memory in the AACS module

with calculating the hash value of the CHT #1. After loading is finished, the hash value is verified using the Content Certificate. If the verification fails, the Player shall go to Stop State without playing back any content. At playback of a P/S-EVOB, the Player chooses one EVOBU/TU with probability 1/100 or more. The EVOBU has CH_PTR which indicates a hash entry of the CHT #1 in the secure memory. The Player calculates the hash value of the EVOBU/TU and compares it with the value stored in the hash entry of the CHT #1. If they are not equal, the Player shall immediately go to Stop State. Otherwise, playback of the P -EVOB continues.

In some Trick Play Modes such as Forward/Backward Scan, only a portion of an EVOBU is read from a Disc. For instance, the Player looks for the first I-Picture in an EVOBU and the found I-Picture is displayed. Then, the Player looks for the first I-Picture in the next EVOBU, without reading the rest of the current EVOBU. This kind of playback is sometimes called “I-Only-Playback”. In general, “I-Only-Playback” is for Forward Scan of the speed which is faster than 2x or for Backward Scan. A Player shall perform Content Hash Check for randomly chosen 1% of the EVOBUs which are *completely* read. This means that, in the “I-Only-Playback” as described above, no Content Hash Check is required.

The sources for P/S-EVOBs are listed as follows: Disc, Network Server, Persistent Storage and File Cache. Content Hash Check shall be performed only for P/S-EVOBs read from Disc.

4.3.4 Pack Decryption

Before decrypting an EVOB, an AACS-Compliant Player shall verify the Content Certificate and the MAC of the Title Usage File. If the verification fails, playback shall be aborted. The Content Hash Table shall also be verified according to the verification procedure defined in Section 4.3.3. If Content Hash Check fails, playback shall be aborted. Before using a Title Key, the Binding MAC associated with the Title Key shall be verified.

The following is an example of decryption of an EVOB:

Step1: Choose a Title Key. To choose a Title Key for the current EVOB, a Player checks the KEY_VF in the GCI_PKT which is located in the NV_PCK in EVOBU/TU. If the KEY_VF is 00b, decryption is unnecessary for the current EVOBU/ TU. If the KEY_VF is 10b, the Player chooses an Encrypted Title Key (K_{te}) in the Title Key File indicated by the TITLE_KEY_PTR, and it decrypts the Encrypted Title Key as defined in the *AACS Introduction and Common Cryptographic Elements*.

Step2: Calculate the Content Key. If the PES_scrambling_control of the current Pack is 01b or if the current Pack is an HL_PCK, the Player calculates a 128-bit Content Key (K_c) using Title Key (K_t), Title Key Data (D_{tk}) and the least significant 96 bits of the CPI field in the GCI_PKT as follows:

K_c = AES-G (K_t, D_{tk} || CPI_{lsb_96}),
where AES-G denotes a one-way function based on the AES algorithm defined in the *AACS Introduction and Common Cryptographic Elements*. If the PES_scrambling_control is 00b and if the current Pack is not an HL_PCK, decryption is unnecessary for the current Pack.

Step3: Decrypt the Pack. The Encrypted Portion (C_e) of the current Pack is decrypted as follows:
C = AES-128CBCD(K_c, C_e),
where AES-128CBCD denotes decryption by the AES algorithm in the CBC mode defined in the *AACS Introduction and Common Cryptographic Elements*.

4.4 Protection Formats for Advanced Resources

This section describes protection formats for Advanced Resources. The Advanced Resources consist of the following ARFs:

- XML Document Files for Playlist and Advanced Navigations
- ECMAScript Files for Advanced Navigations
- JPEG/PNG image files and MNG image files for Advanced Elements
- CVI/CDW image files for captured images
- LPCM/WAV files for Advanced Elements
- OpenType font files for Advanced Elements
- Archived data files for Advanced Elements

4.4.1 Protection Types for ARFs

As described in the Introduction of this book (See Section 1.2), there are five kinds of encapsulation formats: The first is Encapsulation Format for Hash, the second is Encapsulation Format for Encryption and Hash, the third is Encapsulation Format for MAC, the fourth is Encapsulation Format for Encryption and the fifth is Encapsulation Format for Non-Protected Advanced Element. Encapsulation Format for MAC may be applied to the following ARFs:

- JPEG/PNG images, captured images (CVI and CDW), MNG animations and fonts (OTF, TTF and TTC) on an AACS Disc or in a Persistent Storage.
- XML Documents (XPL, MMF, XMU, XTS, XAS, XSS, etc.) in a Persistent Storage.

- ECMAScript Codes (JS) in a Persistent Storage.

Encapsulation Format for Encryption may be applied to the following ARFs:

- JPEG/PNG images, MNG animations, LPCM/WAV effect audios, fonts (OTF, TTF and TTC) on an AACS Disc or in a Persistent Storage.
- XML Documents for Advanced Subtitles (XAS) in a Persistent Storage.
- ECMAScript Codes (JS) in a Persistent Storage.

Encapsulation Format for Hash may be applied to the following ARFs:

- XML Documents (XPL, MMF, XMU, XTS, XAS, XSS, etc.) on an AACS Disc.
- ECMAScript Codes (JS) on an AACS Disc.

Encapsulation Format for Encryption and Hash may be applied to the following ARFs:

- XML Documents for Advanced Subtitles (XAS) on an AACS Disc.
- ECMAScript Codes (JS) on an AACS Disc.

Encapsulation Format for Non-Protected Advanced Element may be applied to the following Advanced Elements:

- JPEG/PNG images, captured images (CVI and CDW), MNG animations, LPCM/WAV effect audios, fonts (OTF, TTF and TTC) on an AACS Disc or in a Persistent Storage.

An archived format for ARFs is defined in the *HD DVD-Video Specifications*. Although an archive file itself is an ARF, it *must not* be encapsulated. Each ARF contained in an archive file may be encapsulated by a protection format. Note that a Resource Search Pointer, which is defined in the *HD DVD-Video Specifications*, in an Archived File shall indicate the attribute of not an original ARF but of an encapsulated ARF. Each encapsulated ARF in an archived file has the MIME-type FFh. An ARF in an archive file, say, “Hallelujah.aca” is indicated by a URI such as “file:///dvddisc/ADV_OBJ/Hallelujah.aca/Kumbaya.jpg”. Note, however, that an AACS module in a Player does not see a directory structure for an archive file. That is, for instance, the file “Kumbaya.jpg” is regarded as a file directly under the directory “/ADV_OBJ”.

There is a possibility that no video is displayed on the screen and the screen is composed only of Advanced Elements. In such a case, the CCI setting for output shall be as follows:

- PCCI: 110b, Copy Never.
- APSTB: 000b, APS is OFF.
- ICT: 0b, High Definition Analog Output in High Definition Analog Form allowed.

- DOT: 0b.

4.4.2 Five Encapsulation Formats

Table 4-9 shows Encapsulation Format for Encryption, and Encapsulation Format for Encryption and Hash is shown in Table 4-10.

Table 4-9: Encapsulation Format for Encryption

Bit / Byte	7	6	5	4	3	2	1	0
0 : 3	FILE_ID							
4	Protection Type: 01_{16}							
5	Reserved							
6	TITLE_KEY_PTR							
7 : 10	Resource File Size (N_{fs})							
11 : N_{fs} + 282	Encrypted Data (D_e)							

Table 4-10: Encapsulation Format for Encryption and Hash

Bit / Byte	7	6	5	4	3	2	1	0
0 : 3	FILE_ID							

4	Protection Type: 11_{16}
5	Reserved
6	TITLE_KEY_PTR
7 : 10	Resource File Size (N_{fs})
11 : N_{fs} + 282	Encrypted Data (D_e)
N_{fs} + 283 : N_{fs} + 286	Hash Pointer #1
N_{fs} + 287 : N_{fs} + 290	Hash Pointer #2
N_{fs} + 291 : N_{fs} + 278 + 4*n	Hash Pointer #3 – Hash Pointer #(n – 1)
N_{fs} + 279 + 4*n : N_{fs} + 282 + 4*n	Hash Pointer #n

Table 4-9 shows Encapsulation Format for Encryption. Encapsulation Format for Encryption shall consist of the following fields:

- FILE_ID of 4 bytes which stores the characters “AACS” with the character set code of ISO646 (a-characters).
- Protection Type of 1 byte. Protection Type shall be 01h if the encapsulation format is Encapsulation Format for Encryption, and it shall be 11h if the encapsulation format is Encapsulation Format for Encryption and Hash.

- Title Key Pointer (TITLE_KEY_PTR) of 1 byte which indicates the entry number of a Title Key in the Title Key File. The value shall be greater than 0 and equal to or less than 64. If the TITLE_KEY_PTR is equal to 09h, the Resource Data is encrypted by the Title Key (K_t) #9 which is obtained as a decrypted value of Encrypted Title Key (K_{te}) #9.
- N_{fs}: Resource File Size of 4 bytes which indicates the size of the Resource Data. The size does not include the Resource File Name field of 272 bytes.
- Encrypted Data of (N_{fs} + 272) bytes which contains encrypted data of the Resource File Name field and the ARF data. The Resource File Name is the filename of the ARF followed by the ".AACS" extension. For instance, if the filename of the encapsulated ARF is "FOO.JPG", the Resource File Name is "FOO.JPG.AACS". The *HD DVD-Video Specifications* says that the maximum length of a filename is 255 bytes. Thus, the length of the Resource File Name is not greater than 260 bytes. See 3.3.2 of the *HD DVD-Video Specifications* for the character code set which can be used for a filename. The Resource File Name field shall be padded with 00h after the ".AACS" extension so that the length becomes 272 bytes.

Let D_{RFN} be the Resource File Name field and let D_{RD} be the ARF data. The Encrypted Data D_e is obtained as follows:

D_e = AES-128CBCE(K_t, D_{RFN} || D_{RD}),

where K_t denotes the Title Key which TITLE_KEY_PTR indicates and AES-128CBCE denotes encryption by the AES algorithm in the CBC mode defined in the *AACS Introduction and Common Cryptographic Elements*. If there is residual data, the size of which is less than 16 bytes, it shall be left unencrypted. Before using an ARF encapsulated in Encapsulation Format for Encryption, the Player shall verify if the Resource File Name stored in the Resource File Name field is identical to the filename of the encapsulated ARF followed by the ".AACS" extension. If they are not identical, the Player must not use the encapsulated ARF, and the Player shall behave as if the file did not exist. The Player will throw the exception of HDDVD_E_FILENOTFOUND or set error info of FILE_NOT_FOUND for the callback, etc. as is defined in the *HD DVD-Video Specifications*. If the exception is not caught, it makes the Player immediately go to Stop State.

- Encapsulation Format for Encryption and Hash has, in addition to Encapsulation Format for Encryption, n fields of the Hash Pointers each of which refers to the offset of an entry of the CHT #2, where n is the smallest integer such that (N_{fs} + 272)/512 <= n.

The k^{th} field of the Hash Pointer of 4 bytes indicates an entry in the CHT #2. The entry stores the SHA-1 hash value of Encrypted Data calculated as follows:

[SHA-1({ D_e }$_k$)]$_{lsb_64}$,

where {D}$_k$ denotes the range between the $512*(k-1)^{th}$ byte and the $(512*k-1)^{th}$ byte of the data D if (0 <) k

< n and $\{D\}_k$ denotes the range between the $512*(n-1)^{th}$ byte and the end of the data D if k = n. Before decrypting and using an ARF encapsulated in Encapsulation Format for Encryption and Hash, the AACS module in a Player shall verify randomly chosen 5% of the hash values of the ARF. If n <= 20, at least randomly chosen 1 hash value shall be verified. If the verification fails, the process of the ARF shall be aborted and the Player shall behave as if the file did not exist. The Player will throw the exception of HDDVD_E_FILENOTFOUND or set error info of FILE_NOT_FOUND for the callback, etc. In addition, the Player shall verify if the Resource File Name stored in the Resource File Name field is identical to the filename of the encapsulated ARF followed by the ".AACS" extension. If they are not identical, the Player must not use the encapsulated ARF, and the Player shall behave as if the file did not exist. The Player will throw the exception of HDDVD_E_FILENOTFOUND or set error info of FILE_NOT_FOUND for the callback, etc.

Table 4-11: Encapsulation Format for MAC

Bit / Byte	7	6	5	4	3	2	1	0
0 : 3	FILE_ID							
4	Protection Type: 02_{16}							
5	Reserved							
6	TITLE_KEY_PTR							
7 : 10	File Size (N_{fs})							
11 : 282	Resource File Name field							
283 : N_{fs} + 282	Resource Data							

N_{fs} + 283 : N_{fs} + 298	MAC of Resource Data

Table 4-12: Encapsulation Format for Hash

Bit Byte	7	6	5	4	3	2	1	0
0 : 3	FILE_ID							
4	Protection Type: 12_{16}							
5 6	Reserved							
7 : 10	File Size (N_{fs})							
11 : 282	Resource File Name field							
283 : N_{fs} + 282	Resource Data							
N_{fs} + 283 : N_{fs} + 286	Hash Pointer							

Encapsulation Format for MAC is shown in Table 4-11. Encapsulation Format for MAC shall consist of the following fields:

- FILE_ID of 4 bytes which stores the characters “AACS” with the character set code of ISO646 (a-characters).
- Protection Type of 1 byte. Protection Type shall be 02h if the encapsulation format is Encapsulation Format for MAC, and it shall be 12h if the encapsulation format is Encapsulation Format for Hash.
- The reserved field shall store 00h.
- N_{fs}: Resource File Size of 4 bytes which indicates the size of the Resource Data.
- The Resource File Name field of 272 bytes which stores the Resource File Name. The Resource File Name is the filename of the encapsulated ARF followed by the “.AACS” extension. The *HD DVD-Video Specifications* says that the maximum length of a filename is 255 bytes. Thus, the length of the Resource File Name is not greater than 260 bytes. See 3.3.2 of the *HD DVD-Video Specifications* for the character code set which can be used for a filename. The Resource File Name field shall be filled with 00h after the “.AACS” extension.
- Resource Data of N_{fs} bytes which contains data of the ARF.

Encryption Format for MAC has a field called MAC of Resource Data:

- A 16-byte field for MAC of Resource Data which stores the MAC of Resource File Name and Resource Data such that MAC = CMAC(K_t, D_{RFN} || D_{RD}), where K_t denotes the Title Key which TITLE_KEY_PTR indicates, D_{RFN} is the Resource File Name field and D_{RD} is the ARF data.

Before using an ARF encapsulated in Encapsulation Format for MAC, the AACS module in a Player shall verify the MAC value. If the verification fails, the Player must not use the ARF and shall behave as if the file did not exist. The Player will throw the exception of HDDVD_E_FILENOTFOUND or set error info of FILE_NOT_FOUND for the callback, etc. If the exception is not caught, it makes the Player immediately go to Stop State. In addition, the Player shall verify if the Resource File Name stored in the Resource File Name field is identical to the filename of the encapsulated ARF followed by the “.AACS” extension. If they are not identical, the Player must not use the encapsulated ARF, and the Player shall behave as if the file did not exist. The Player will throw the exception of HDDVD_E_FILENOTFOUND or set error info of FILE_NOT_FOUND for the callback, etc.

On the other hand, Encapsulation Format for Hash has the field of the Hash Pointer which refers to an entry of the CHT #2:

- The Hash Pointer of 4 bytes indicates the offset of an entry in the CHT #2. The entry stores the SHA-1 hash value of Encrypted Data calculated as follows:
  $[\text{SHA-1}(D_{RFN} \| D_{RD})]_{lsb_64}$.

Before using an ARF encapsulated in Encapsulation Format for Hash, the AACS module in a Player shall verify the hash values of the ARF. If the verification fails, the Player must not use the ARF and shall behave as if the file did not exist. The Player will throw the exception of HDDVD_E_FILENOTFOUND or set error info of FILE_NOT_FOUND for the callback, etc. If the exception is not caught, it makes the Player immediately go to Stop State. In addition, the Player shall verify if the Resource File Name stored in the Resource File Name field is identical to the filename of the encapsulated ARF followed by the ".AACS" extension. If they are not identical, the Player must not use the encapsulated ARF, and the Player shall behave as if the file did not exist. The Player may throw the exception of HDDVD_E_FILENOTFOUND or set error info of FILE_NOT_FOUND for the callback, etc.

Table 4-13: Encapsulation Format for Non-Protected Advanced Element

Bit / Byte	7	6	5	4	3	2	1	0
0 : 3	FILE_ID							
4	Protection Type: 21_{16}							
5	Reserved							
6	TITLE_KEY_PTR							
7 : 10	File Size (N_{fs})							
11 : 282	Encrypted File Name field							
283 : N_{fs} + 282	Resource Data							

Encapsulation Format for Non-Protected Advanced Element is shown in Table 4-13. It shall consist of the following fields:

- FILE_ID of 4 bytes which stores the characters “AACS” with the character set code of ISO646 (a-characters).
- Protection Type of 1 byte. Protection Type shall be 21h.
- The reserved field shall store 00h.
- N_{fs}: Resource File Size of 4 bytes which indicates the size of the Resource Data.
- The Encrypted File Name field of 272 bytes which stores the filename of the encapsulated ARF. The filename must not have the “.AACS” extension. See 3.3.2 of the *HD DVD-Video Specifications* for the character code set which can be used for a filename. The File Name field shall be filled with 00h after the filename. Let D_{RFN} be the Resource File Name field. The Encrypted File Name FNe is obtained as follows:
  FNe = AES-128CBCE(Kt, D_{RFN}),
  where Kt denotes the Title Key which TITLE_KEY_PTR indicates and AES-128CBCE denotes encryption by the AES algorithm in the CBC mode defined in the *AACS Introduction and Common Cryptographic Elements*.
- Resource Data of N_{fs} bytes which contains data of the ARF.

Before using an ARF encapsulated in Encapsulation Format for Non-Protected Advanced Element, the AACS module in the Player shall decrypt the Encrypted File Name field. Then, the Player shall verify if the filename stored in Encrypted File Name field is identical to the filename of the encapsulated ARF. If they are not identical, the Player must not use the encapsulated ARF, and the Player shall behave as if the file did not exist. The Player will throw the exception of HDDVD_E_FILENOTFOUND or set error info of FILE_NOT_FOUND for the callback, etc.

4.4.3 Verification of Hash and/or Decryption

As mentioned above, there are two Content Hash Tables recorded on an AACS Disc. Of these two, the CHT #2 stores the hashes of the TUFs, the hashes of the ARFs encapsulated in Encapsulation Format for Hash and the hashes of the ARFs encapsulated in Encapsulation Format for Encryption and Hash. See Table 3-16 for the format of CHT #2.

Before verification of the hashes, the Content Certificate file shall be at first verified. If the Disc to be played back belongs to the Category 1, the SHA-1 hash value of “VTUF.AACS” is calculated and the

value is compared with the value stored in the corresponding field in the CHT #2. If the values are not equal to each other, the verification fails and the rest of the playback process shall be aborted. If the Disc belongs to the Category 2 or 3, a Playlist to be played back is at first chosen according to the procedure described in the *HD DVD-Video Specifications*. The Playlist is accompanied by one and only one TKF and one and only one TUF. The integrity of the TUF shall be verified by means of the content hash, that is, the SHA-1 hash values of the TUF are calculated and compared with the stored values in the corresponding field in the CHT #2. If the verification fails, the rest of the playback process shall be aborted and the Player shall immediately go to Stop State.

Before using an ARF encapsulated in Encapsulation Format for Hash or in Encapsulation Format for Encryption and Hash, the AACS module in the Player shall check the integrity by means of the content hash.

Chapter 5
Downloading, Streaming and Online-Enabling

5 Downloading, Streaming and Online-Enabling

5.1 Introduction

An HD DVD-Video Player has the capability to download contents from the Internet. A Disc Application or a P-Storage Application controls the procedure of content downloading. What it downloads and when are controlled by XML Documents and ECMAScript Codes in the application. In the AACS content protection scheme, however, downloading or streaming of contents shall satisfy some conditions. Those conditions are described in Section 5.4. Some AACS APIs exposed to an application for downloading are also described there.

Streaming of audiovisual contents or audio-only contents is defined in the *HD DVD-Video Specifications*. There are two scenarios for streaming: One is scheduled streaming. In this scenario, when to start streaming is described in the Playlist. Streamed data is an S-EVOB and it may be encrypted in the format described in Section 4.3. The Title Key Pointer in the KMI indicates a Title Key in the active TKF, which is used to decrypt the streamed data. When the Playlist on an AACS Disc is active, the active TKF is the one on the Disc. If the Playlist is booted from a Persistent Storage, the active TKF is the one in the Persistent Storage associated with the Playlist. (See Section 6.2.2)

The other scenario is streaming on demand. In this scenario, for instance, an application calls, according to a user operation, a URL of a streaming content. The assumption here is that the streamed content is, at least partially, encrypted and that the application does not currently have the Title Key to decrypt it. The application should set beforehand the Title Key for the streamed content in the AACS module of the system. Section 5.4 describes how such a streaming scenario is dealt with in the AACS content protection scheme.

There is a usage called Online-Enabling, which is described in Section 5.1 of the *AACS Introduction and Common Cryptographic Elements*. As for the AACS scheme, an application may realize the Online-Enabling usage by means of the same API used in streaming on demand. How to realize the Online-Enabling usage is also described in Section 5.4.

Note that the Internet connection, streaming and Online-Enabling is not available if the Disc is played back in Standard Content Playback State. Thus, the AACS Object and its function properties are not available in the State.

5.2 Binding

A concrete protocol for downloading is defined by each application by means of XML documents and ECMAScript codes. There are, however, some conditions an application shall satisfy. One is the condition about Binding and Permissions. In order to download a content which has appropriate binding, a Player shall send some information to a server. The information can be obtained by an application through some APIs.

5.2.1 Binding and Permissions

See 5.6 in the *AACS Introduction and Common Cryptographic Elements* for binding allowed in the AACS content protection scheme. An HD DVD-Video Player supports the following five kinds of binding:

a. Binding to Medium: Data are bound to 128 bits of the Volume Identifier and 128 bits of the Pre-Recorded Media Serial Number. (See Section 2.3.3 for definition of Volume Identifier and see Section 2.4.4 for definition of Pre-Recorded Media Serial Number.)

b. Binding to Content: Data are bound to the Volume Identifier.

c. Binding to Medium & Device: Data are bound to the Media and the Device Unique Nonce (DUN). (See Section 3.4 for definition of the DUN.)

d. Binding to Content & Device: Data are bound to Content and the DUN.

e. Binding to a Temporary Nonce (TN) of 128 bits which is generated in the AACS module in a Player and is discarded by the module once used.

The DUN is an ID of 128 bits which is unique to each Player. This ID may be assigned by a Player manufacturer. Or, a Player may generate a GUID for the DUN when it is initialized for the first time. Once the ID is generated, it is stored in a persistent memory in a Player. The following condition shall hold for the DUN: An application as well as a user operation is not allowed to assign an arbitrary value to the DUN.

A TN is a nonce of 128 bits which is generated in the AACS module by an API call. A TN is used in an API call for TKF exchange in order to verify Binding MACs. Once a TN is used, the TN shall be discarded

by the AACS module. In addition, a TN shall be cleared from the AACS module when the following conditions hold:

1. There is another call for the API (the property TN of the AACS Object).
2. The Disc is ejected.
3. The Player loses power.
4. The Player goes into Stop State.
5. The boot sequence for an AACS Disc starts.
    a) The boot sequence is described in 6.2.2.2.

A Title Key is called TN-bound if it is bound to the Temporary Nonce. It is not allowed two or more EVOBs share the same TN-bound Title Key. A TN-bound Title Key shall be discarded or disabled when playback leaves an EVOB associated with the Title Key. That is, a TN-bound Title Key shall be discarded when the value of the Title Key Pointer is changed or the Title Key is used no more. A TN-bound Title Key shall not be discarded or disabled when playback is in Pause State or when playback is forward/backward scanned in the concerned EVOB.

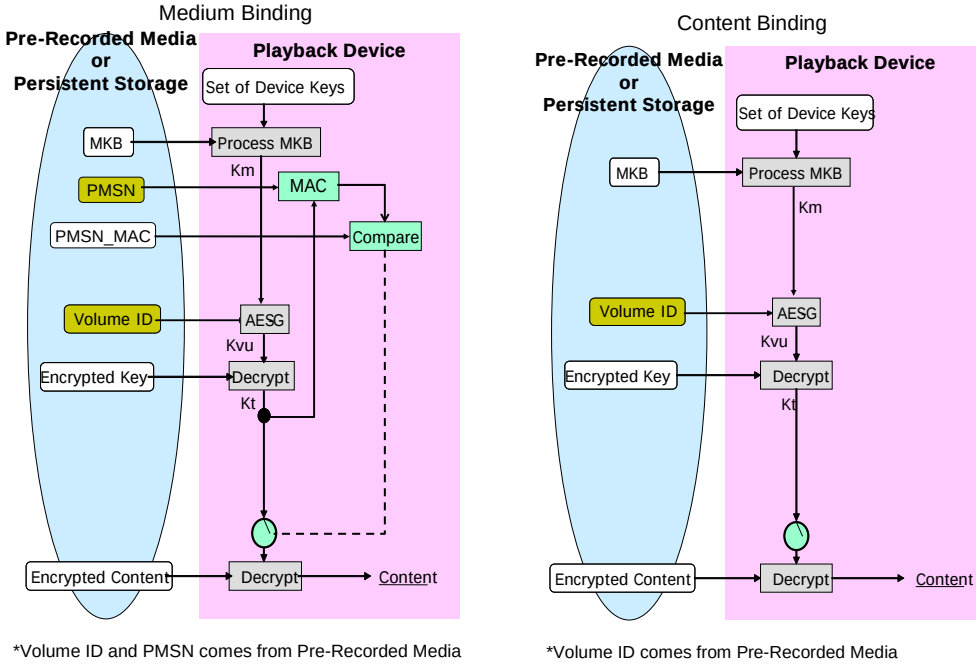

Figure 5-1: Key Retrieval Flows for Medium Binding and Content Binding

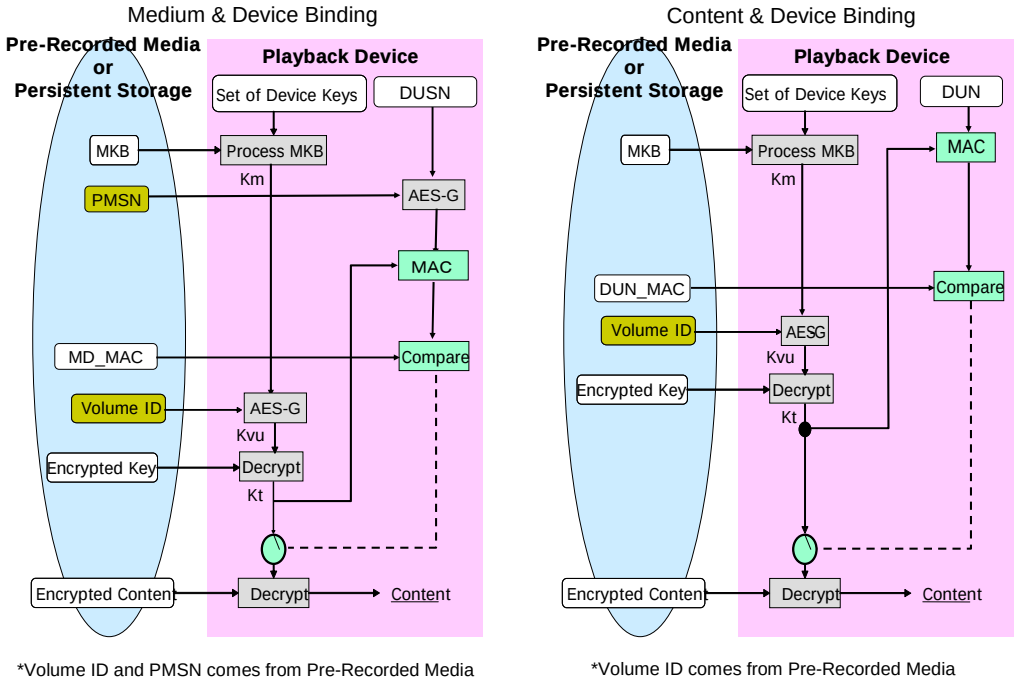

Figure 5-2: Key Retrieval Flows for Medium & Device Binding and Content & Device Binding

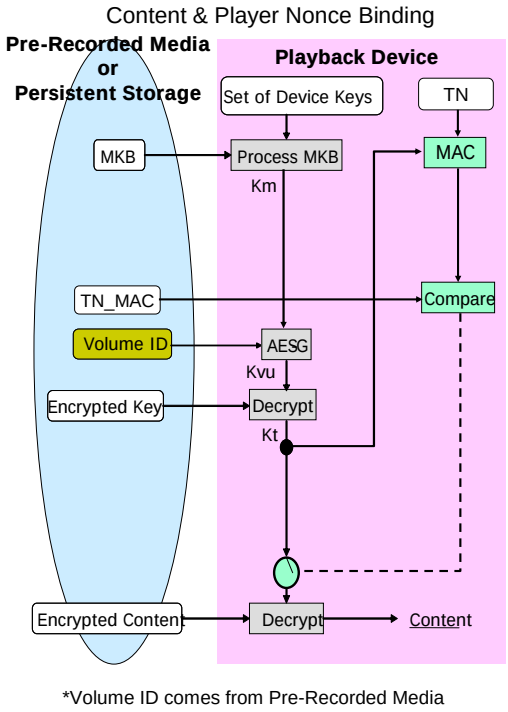

Figure 5-3: Key Retrieval Flows for Content & Temporary Nonce Binding

The key retrieval flows are shown in Figure 5-1, Figure 5-2 and Figure 5-3. In those figures of Medium Binding, PMSN_MAC denotes the MAC of PMSN, i.e. PMSN_MAC = CMAC(K_t, PMSN). DUN_MAC, MD_MAC and TN_MAC denote the following MACs respectively:

- DUN_MAC = CMAC(K_t, DUN),
- MD_MAC = CMAC(K_t, AES-G(DUN, PMSN)),
- TN_MAC = CMAC(K_t, TN).

Note that there are 64 Title Keys in the TKF file. Therefore, PMSN_MAC requires a Title Key which is used to calculate the MAC value. The same is true for the DUN_MAC, MD_MAC and TN_MAC. On the other hand, a Title Key is accompanied by a Binding Information which constrains use of the concerned Title Key. This is why a TKF file has the field of Binding MAC. Each Title Key can be decrypted using the Volume ID. The decrypted Title Key *must not*, however, be used to decrypt a content before the associated Binding MAC is checked and confirmed.

In the above figures, the Volume ID and PMSN shall be those which are read from the Disc. This does not mean these values shall be read directly from the disc every time they are necessary. The Volume ID and PMSN may be stored in the system memory of a Player until one of the following conditions holds:

1. The Disc is ejected.
2. The Player loses power.
3. The Player goes into Stop State.
4. The boot sequence for an AACS Disc starts.
    a) The boot sequence is described in 6.2.2.2.

A Player may also retain the decrypted Title Keys while playback continues. The decrypted Title Keys shall be, however, discarded if one of the following conditions holds:

1. The Disc is ejected.
2. The Player loses power.
3. The Player goes into Stop State.
4. The boot sequence for an AACS Disc starts.
    a) The boot sequence is described in 6.2.2.2.

A Permission for playback itself is in fact a TKF which resides on a Disc, in a Persistent Storage or in the File Cache in a Player. Only a Title Key which is bound to a TN is for an Instant Permission because the TN in the AACS module to which the Title Key is bound is discarded once the Binding MAC is resolved. In this context, a Binding MAC for TN, where BIND_TYPE = 100b, shall be verified every time the corresponding Title Key is used. Note that a Permission associated with a Title Key with other binding is considered to be “Basic” or “Cacheable”.

5.2.2 APIs used to Obtain the Binding Information

An application should send some information of an AACS Disc or a Player to a server in order to obtain a content, a TKF, etc. which has appropriate binding. A Player shall provide some APIs (properties) exposed to an application for that purpose: AACS.pmsn, AACS.volumeID, AACS.dun and AACS.tn. AACS.pmsn returns 128 bits of the PMSN of the current Disc in the tray. AACS.volumeID returns 128 bits of the Volume Identifier of the current Disc in the tray. AACS.dun returns the DUN which resides in the Player system. See Section 5.2.1 for explanation of the DUN. AACS.tn returns 128 bits of the Temporary Nonce.

In order to download a content which has appropriate binding, an application should send the appropriate information for binding to a server. Required information for each binding is listed in Table 5-1.

Table 5-1: Required Information for Each Binding

Binding Information	Medium	Content	Medium & Device	Content & Device	Temporary Nonce
Volume ID	YES	YES	YES	YES	YES
PMSN	YES	NO	YES	NO	NO
DUN	NO	NO	YES	YES	NO
TN	NO	NO	NO	NO	YES

All the information, i.e. the Volume Identifier, the Pre-Recorded Media Serial Number, the Device Unique Nonce and Temporary Nonce may be sent to a server in the clear: It is not necessary to use a secure protocol such as HTTPS in order to protect any of those data. Note that the Default Connection Protocol defined in Appendix B of the AACS Introduction and Common Cryptographic Elements shall be ignored for the on-line transaction for HD DVD-Video.

5.3 Managed Copy

There shall be a file, named “MNGCPY_MANIFEST.XML”, in the “AACS” directory. The file is called Managed Copy Manifest File (MCMF). It may contain URLs for Managed Copy Servers and it may contain resource descriptions for Managed Copy. The following is an example of description of MCMF:

```
<?xml version="1.0" encoding="UTF-8"?>

<mcManifest xmlns="http://www.aacsla.com/2006/02/hdmcManifest"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Id="Clockwork_Tomato_V1.0">

<Cid>apX7MvTl8Gir1cAM+bAy/Q==</Cid>
```

```
<serverList>
  <serverUri>http://www.xyz.com/mc/Clockwork_Tomato</serverUri>
  <serverUri>http://www.aacs.com/mc/Clockwork_Tomato</serverUri>
</serverList>

<offersNManifest>
  <offers version="1.25">
    <offer>
      <MCUi>3eff03</MCUi>
      <title>Clockwork_Tomato_CH1</title>
      <abstract>This is an abstract.</abstract>
      <description>This is a description.</description>
      <image>
         <url>http://www.foo-zoo.co.jp/images/foo.jpg</url>
         <title>A Thumbnail</title>
      </image>
      <language>jp</language>
      <MCOT>greatdrm</MCOT>
      <availability start="2005-10-31T00:00:00" end="2005-12-25T00:00:00"/>
      <price>1000yen</price>

  <financialApplicationURI>http://www.bank.co.jp/Clockwork_Tomato/app</financialApplication
URI>
      <financialHTMLURL>http://www.bank.com/Clockwork_Tomato</financialHTMLURL>
    </offer>
    <offer>
      <MCUi>3eff05</MCUi>
      <title>Clockwork_Tomato_Overview</title>
```

```
      <abstract>This is an abstract.</abstract>
      <description>This is a description.</description>
      <image>
         <url>http://foo-zoo.co.jp/images/zoo.jpg</url>
         <title>A Thumbnail</title>
      </image>
      <language>us-en</language>
      <MCOT>robustdrm</MCOT>
      <availability start="2005-10-31T01:01:01" end="2005-12-25T13:59:59"/>
      <price>$20</price>

  <financialApplicationURI>http://www.bank.com/Clockwork_Tomato/app</financialApplicationUR
I>
      <financialHTMLURL>http://www.bank.com/Clockwork_Tomato</financialHTMLURL>
    </offer>
  </offers>

  <dealManifest>
    <mcUnit>
      <MCUi>3eff03</MCUi>
      <item Id="4" Src="file:///dvddisc/ADV_OBJ/VPLST010.XPL"/>
      <item Id="1" Src="file:///dvddisc/HVDVD_TS/TITLE01.EVO"/>
      <item Id="3" Encapsulation="0" Src="file:///dvddisc/ADV_OBJ/wallpaper.jpg"/>
    </mcUnit>
    <mcUnit>
      <MCUi>3eff05</MCUi>
      <item Id="10" Src="file:///dvddisc/ADV_OBJ/OVERVIEW.XPL"/>
      <item Id="11" Src="file:///dvddisc/ADV_OBJ/INTRO.JPG"/>
      <item Id="13" Src="file:///dvddisc/HVDVD_TS/Overview.EVO"/>
```

```
    </mcUnit>
  </dealManifest>
</offersNManifest>


</mcManifest>
```

The semantics is as follows:

Cid	This is a 128-bit Content ID expressed in the base64 encoded format. See the *AACS Introduction and Common Cryptography Elements* for Content ID.
serverList	This consists of at most 16 serverUris. An MCMF may contain at most one <serverList>.
serverUri	This indicates a Managed Copy Server. Multiple URIs may be recorded. A Managed Copy Machine shall check validity of URIs. The checking order may be proprietary to each Managed Copy Machine.
offers	Offers for Managed Copy. See the *AACS Pre-recorded Video Book* for the <offers> element and the <offer> element. An MCMF may contain at most 256 <offer> elements. Each <offer> shall have an <MCUi> element, which shall correspond to the <MCUi> element in an <mcUnit> element. Note that the value “0” of the element <MCUi> is reserved for copy of the entire Disc. And note that the value “0” of <MCUi> need not be explicitly listed in <offers> or in <dealManifest>.
dealManifest	It consists of <mcUnit>s.

mcUnit	A bunch of resources on an AACS Disc. This element shall have an <MCUi> element. Two or more <mcUnit>s must not share the same <MCUi>. An <mcUnit> is associated with one <offer> if and only if the <MCUi> element in the <mcUnit> corresponds to the <MCUi> element in the <offer> element. A Managed Copy Machine will copy, provided that the copy operation is allowed by a Managed Copy Server, all the items listed in an <mcUnit>. An MCMF may contain at most 256 <mcUnit>s, and one <mcUnit> may have at most 65536 items.
Item	An item indicates a resource file. It shall have a Src attribute indicating a resource file and may have an Id attribute and an Encapsulation attribute. The Src attribute is a URI. The default value of the Encapsulation attribute is “true” which means the resource file is encapsulated.

See Appendix A for the schema for MCMF. MCMF is used by a Managed Copy Machine. A Managed Copy Machine shall verify the MCMF on a Disc before using it based on the full SHA-1 hash value which is contained in CHT#2 on the AACS Disc.

See the *AACS Pre-recorded Video Book* for the details of Managed Copy. A Managed Copy Server may return an empty offer with a Session ID. In that case, the Managed Copy Machine may show the user the offers listed in Managed Copy Manifest recorded on the Disc. And if the permission, which does not contain a dealmanifest in that case, is obtained from the server, the MCM copies all the items in an <mcUnit> which is designated by the <MCUi> element in the <offer> chosen by the user. If a Managed Copy Server returns a non-empty offers, the MCM must not use the offers and the dealmanifest in Managed Copy Manifest on the Disc.

Section 5.5.3 in the AACS Pre-recorded Video Book says the dealmanifest in a Permission Response message is proprietary to each format. The element <mcUnit> defines the format for the dealmanifest for HD DVD-Video. See Appendix B and C for Managed Copy Permission XML Schema and Managed Copy Web Service Description, respectively. Note that, for HD DVD/DVD Pre-recorded Media, Managed Copy Permission and Managed Copy Web Service Description shall follow those in Appendix B and C. See Appendix E for APIs which shall be implemented in a Managed Copy Machine which is able to process Advanced Navigations defined in the HD DVD-Video Specifications.

5.4 APIs

5.4.1 API for Streaming

An AACS-Compliant Player shall provide an AACS API for streaming on demand: i.e. changeTKF(tkf, mkb), where “tkf” is a TKF file and “mkb” is an MKB file. This API replaces the current Title Keys with the new Title Keys in the “tkf”. The file “mkb” is used to decrypt the new TKF. A Title Key in the “tkf” must not be used for content decryption unless its associated Binding MAC is checked and verified. The new TKF which is set in the AACS module by changeTKF() shall be discarded if one of the following conditions hold:

1. There is another call for changeTKF().
2. The Disc is ejected.
3. The Player loses power.
4. The Player goes into Stop State.
5. The boot sequence for an AACS Disc starts.
    a) The boot sequence is described in 6.2.2.2.

Streaming on demand is not scheduled in the Playlist, and the system does not currently have the Title Key to decrypt the streamed data. Before pulling the streamed data from a server, the application should download a TKF file which contains the Encrypted Title Key for the streamed S-EVOB. The application should also download an MKB file for decryption of the TKF. And then, the application should replace the TKF by the API changeTKF() in order to set in the AACS module an appropriate Title Key set containing a Title Key for the streamed S-EVOB. After the API call, the application may pull the streamed S-EVOB and play it back. No content hash check is required for a streamed S-EVOB. changeTKF() will throw an exception AACS_E_TKF if the TKF change fails. If the exception is not caught, it makes the Player go to Stop State.

5.4.2 APIs for Online-Enabling

The same API changeTKF() may also be used to realize the Online-Enabling scenario. Suppose that the AACS module in a Player system does not currently have the Title Key to decrypt a P/S-EVOB on a disc. When a user requests to play it back, an application connects to a server and downloads an appropriate TKF file and an MKB file for it. Then the application calls changeTKF() for replacement of the current TKF with the new one containing the Title Key for the Online-Enabled P/S-EVOB. After the API call, the application may play back the concerned P/S-EVOB on the Disc just by sending the EVOB to the Presentation Engine. A

Player shall perform content hash checking for the Online-Enabled P/S-EVOB based on the CHT #1 on the Disc.

5.4.3 Examples of Usage Scenarios of Online-Enabling

This section describes some examples of usage scenarios of Online-Enabling. The first example shows a scenario of Instant Permission. See Figure 5-4 for an image for an Online-Enabling scenario:

1. Suppose the current TKF has no key for EVOB2.
2. Application 1 suspends playback of the video before entering into EVOB2 and displays a menu which says “Do You Want to Buy …” or something like that.
3. If the user pushes the “Do Not Buy” button, playback jumps to EVOB3.
4. If the user chooses to buy playback of EVOB2, the application obtains the AACS properties, VolumeID and TN.
5. The application sends to a server which is preliminarily defined in the script the values of the Volume ID and the Temporary Nonce.
6. The server makes a TKF, where the TKF contains the Encrypted Title Key for EVOB2 which is bound to the Temporary Nonce.
   - Note that the server can decide whether it will allow the client Player to play back EVOB2 or not at this stage.
7. The server sends to the Player the new TKF and the MKB for it.
8. The Player calls the API, changeTKF(), with the new TKF and the MKB being set as the arguments.
9. The AACS module in the Player changes the current TKF with the new TKF on a successful process of the MKB and the new TKF.
   - In this occasion, the current TN in the AACS module is discarded. This prohibits the replay attack using the same TKF.
10. The application starts playback of EVOB2 which is now playable because the AACS module has the Title Key for it.
    - The Title Key is valid as long as playback stays in EVOB2.
11. Once playback leaves EVOB2, the Title Key for EVOB2 is discarded or disabled.

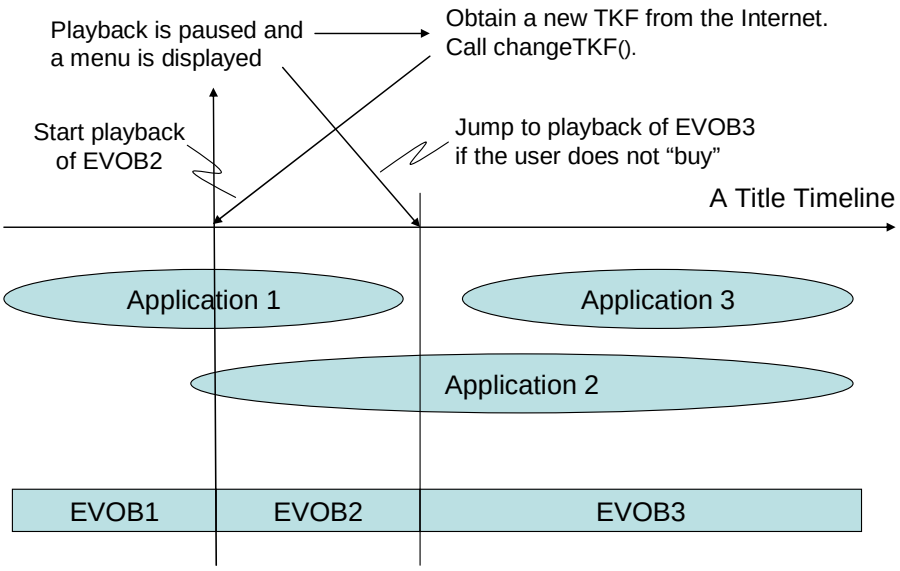

Figure 5-4: Image of an Online-Enabling Scenario

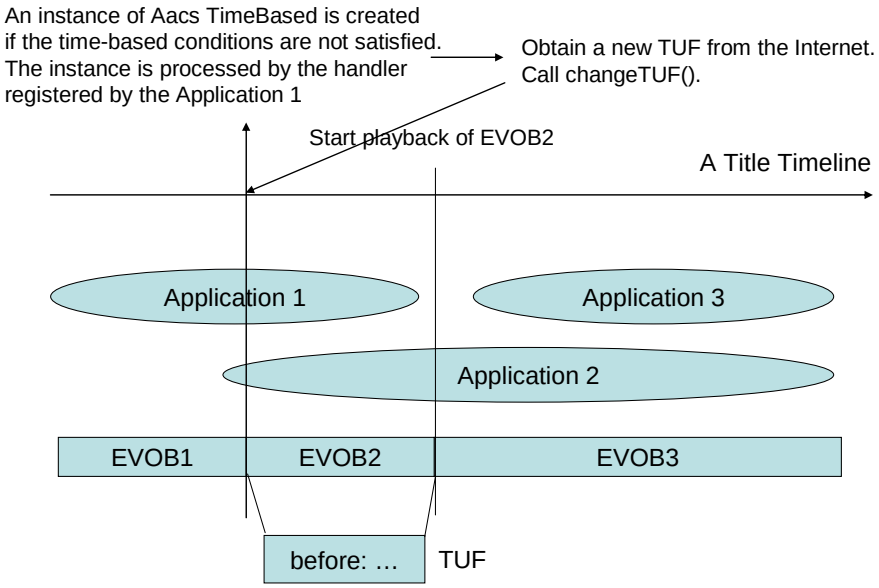

Figure 5-5: Image of an Online-Enabling Scenario

Another image of Online-Enabling scenario is shown in Figure 5-5. Suppose in this scenario that the Player supports a Secure Clock:

1. Suppose that the TUF associated with EVOB2 has only the Time-Based Condition of VALID_BEFORE.
2. Application 1 registers an event handler for the system event Aacs TimeBased.
3. Before playback of EVOB2, Application 1 calls the API, IsSecureClockSupported(). If the return value is **false**, playback jumps to EVOB3. Otherwise, go to Step 4.
   - IsSecureClockSupported() returns **true** if a secure clock exists in the AACS module. Otherwise, it returns **false**.
4. If the VALID_BEFORE condition holds, the AACS module allows decryption of EVOB2. Otherwise, go to Step 5.
5. The AACS module does not allow decryption of EVOB2 and creates an instance of the Event Object, AacsTimeBased.
6. The event is processed by the handler which has been registered by Application 1. In the handler, the Player proceeds to a procedure for obtaining an Instant Permission as described in the previous scenario example.
   - This time, the Player obtains a TUF and a Content Certificate in the format described in Table 6-1. The new TUF is replaced with the current TUF in the Player system by the API changeTUF(tuf, cc), where “tuf” denotes a new TUF and “cc” is the new Content Certificate. The new TUF has a UR of Time-Based Conditions such that PERMISSION_PERIOD = 8. UR_PTR in the EVOB2 points to the URS containing the UR. The URS, for instance, is bound to the Player by DUN.
   - The new TUF which is set in the AACS module by changeTUF() shall be discarded if one of the following conditions hold:
     a) There is another call for changeTUF().
     b) The Disc is ejected.
     c) The Player loses power.
     d) The Player goes into Stop State.
     e) The boot sequence for an AACS Disc starts
        i) The boot sequence is defined in 6.2.2.2.
7. Start playback of EVOB2.

Suppose a content owner creates a free “Movie Lovers Club” that offers special offers, discounts and rewards for customers who register online. For a particular movie, there is a director’s commentary video which is to be played back simultaneously with the Main Video and to overlay the Main Video with a chroma-key. The director’s commentary is free but only available to the club members. The director’s commentary is contained in a single S-EVOB for Sub Video. The playlist as shipped on the disc does not allow access to the extra scenes. The scenario works as follows:

1. An application asks if the user would allow an online connection. If the answer is no, only the normal Playlist is played back.
2. Otherwise, the application performs an online connection. If the user has already been a club member, the Player passes the membership ID to the server. Go to 4.
3. If the user is not a club member, the server shows a screen which asks if the user wants to join the club. The answer is yes the server issues a membership ID to the user.
4. The server sends the Player a bunch of AACS components and resources, which contains a new Playlist, a new Content Certificate file, a new TKF, a new TUF and XML Documents, ECMAScript Codes, images and animations. The Player stores the AACS components and the resources in a Persistent Storage.
   - The TUF and Title Keys in the TKF could be bound, for example, to the PMSN. This means that the user is allowed to copy this permanent permission represented by the bunch of AACS components and resources to other Players in his home --- in other words, the permission is associated with the individual physical disc.
5. Now with the latest Playlist, the director’s commentary video is allowed to be played.

5.5 AACS Object

All the APIs which are proprietary to AACS introduced in the preceding sections are member function of an Object, i.e. AACS Object. AACS Object is a member of Player Object defined in the *HD DVD-Video Specifications*, which means that AACS Object shall be added as a read only member to Player Object for an AACS-Compliant Player:

```
readonly AACS        aacs;
```

An access to any property or any property or any function property of AACS Object which occurs when a Player is not in AACS mode shall lead to the exception HDDVD_E_INVALIDCALL, i.e. the Player shall throw the exception.

Properties of the AACS Object	
	String pmsn;
	String volumeID;
	String dun;
	String tn;
	Boolean isSecureClockSupported;
	Boolean isMCMSupported;
	int secureHour;
	const unsigned int INVALIDKEY = 10001;

Function Properties of the AACS Object	
	void changeTKF(tkf: String, mkb: String);
	void changeTUF(tuf: String, cc: String);
	void setMACKey(keyno: unsigned int);
	void invokeMCM(callback : Function);

pmsn returns the PMSN of the AACS Disc currently played in the hexadecimal notation, i.e. as a String of 32 upper case letters. If the Disc has not PMSN, the value shall be filled with the character “Z”.

volumeID returns the Volume ID of the AACS Disc currently played in the hexadecimal notation, i.e. as a String of 32 upper case letters.

dun returns the DUN of the AACS-Compliant Player in the hexadecimal notation, i.e. as a String of 32 upper case letters.

tn returns a TN which is generated by the AACS-Compliant Player and is stored in the AACS module in the hexadecimal notation, i.e. as a String of 32 upper case letters.

isSecureClockSupported returns **true** if a secure clock exists in the AACS module. Otherwise, it returns **false**.

isMCMSupported returns **true** if the Player can call and invoke a Managed Copy Machine. Otherwise, it returns **false**.

secureHour returns the elapsed time since the 1st of July, 2005 (GMT) measured in hours if the Player supports a secure clock. If a secure clock is not supported by the Player, the return value is negative.

changeTKF changes the current TKF into the TKF given as the first argument. The second argument is an MKB which is required for the key derivation. This is a synchronous API and a call for this API may take a few seconds. The lengths of both of the arguments shall be strictly checked. If a Tilte Key in the new TKF is identical to the corresponding one in the current TKF, playback of an EVOB which is decrypted by the Title Key shall continue seamlessly, i.e. without interruption or picture disorder. Note that the field of PLAYLIST_NAME in the new TKF shall be compared to the current Playlist name, and if they do not match, the exception AACS_E_TKF shall be thrown. Note also that the TKF MAC shall be verified. If the verification fails, the exception AACS_E_TKF shall be thrown.		
Parameters	**tkf** of type **String**	This argument specifies the new Title Key File. This is a URL, whose length does not exceed 1024 characters as defined in the *HD DVD-Video Specifications*.
	mkb of type **String**	This argument specifies the new MKB used to decrypt the new Title Keys. This is a URL, whose length does not exceed 1024 characters as defined in the *HD DVD-Video Specifications*.
Return Value	**None**	
Exceptions	If TKF exchange does not succeed, an exception AACS_E_TKF is thrown. If the	

	exception is not caught, it makes the Player immediately go into Stop State. The value of the exception AACS_E_TKF is “AACS_E_TKF”.

changeTUF changes the current TUF into the TUF given as the first argument. The second argument is a Content Certificate in the format of Table 6-1 which is required to verify integrity of the new TUF. This is a synchronous API and a call for this API may take a few seconds. The lengths of both of the arguments shall be strictly checked. If a URS in the new TUF is identical to the corresponding one in the current TUF, playback of an EVOB which is associated with the URS shall continue seamlessly, i.e. without interruption or picture disorder. Note that the field of PLAYLIST_NAME in the new TUF shall be compared to the current Playlist name, and if they do not match, the exception AACS_E_TUF shall be thrown. Note also that the TUF MAC shall be verified. If the verification fails, the exception AACS_E_TUF shall be thrown.		
Parameters	**Tuf** of type **String**	This argument specifies the new Title Usage File. This is a URL, whose length does not exceed 1024 characters as defined in the *HD DVD-Video Specifications*.
	Cc of type **String**	This argument specifies the new Content Certificate used to verify integrity of the new Title Usage File. This is a URL, whose length does not exceed 1024 characters as defined in the *HD DVD-Video Specifications*.
Return Value	**None**	
Exceptions	If TUF exchange does not succeed, an exception AACS_E_TUF is thrown. If the exception is not caught, it makes the Player immediately go into Stop State. The value of the exception AACS_E_TUF is “AACS_E_TUF”	

setMACKey sets the index of a Title Key which is used to calculate the CMAC value when an XML Document, a captured image, etc. are saved. Note that the default index of a Title Key for calculation of the CMAC value is 1, which means that the CMAC value is calculated using the Title Key#1 before the first call for this API.		
Parameters	**keyno** of type **unsigned int**	This argument specifies the number of a Title Key.
Return Value	**None.**	

Exceptions	If the value of the argument **keyno** is out of range, that is, if the value is less than 1 or greater than 64, an exception AACS_E_SETMACKEY is thrown. If the exception is not caught, it makes the Player immediately go into Stop State. The value of the exception AACS_E_SETMACKEY is "AACS_E_SETMACKEY".

invokeMCM invokes an MCM (Managed Copy Machine) if an MCM is supported by the Player. If an MCM is not available, a call for this API yields an exception. This API is asynchronous. If a Player does not support MCM, the Player need not support this API.		
Parameters	**callback** of type **Function**	
	When the invoked MCM is terminated, this callback function is called: **void** callback(status);	
	Parameters	**status** of type **int**
		The process of MCM is successfully finished if and only if the value of status is equal to 0. If the call for an MCM fails or if the process of MCM fails, the value is non-zero.
Return Value	**None.**	
Exceptions	**HDDVD_E_INVALIDCALL** if an MCM is not available**.**	

Chapter 6
Protection of HD DVD-Video Contents in Persistent Storages

6 Protection of HD DVD-Video Contents in Persistent Storages

6.1 Introduction

As for Advanced Contents, there are places where elements of an HD DVD-Video content may possibly reside outside of an optical disc, i.e. Persistent Storages. A Persistent Storage may be, for instance, an HDD in a Player, an SD-Card, a USB memory, a Network Attached Storage and so on. For data categories allowed in a Persistent Storage, see 4.3.6.5 in the *HD DVD-Video Specifications*. Section 6.2 of this book describes data formats of an AACS-Protected content in a Persistent Storage and conditions required for the data formats.

A data access method for Persistent Storage is defined in the *HD DVD-Video Specifications*. A medium which serves as a Persistent Storage has a directory structure which is defined in the specification, and a directory for a content provider is distinguished by the name generated from a Provider ID which is a GUID given by the content provider. In order to prohibit an application from illegitimately using contents which belong to another content provider, the name of the provider directory shall be protected from access by another provider's applications. Protection of the directory name is mentioned in Section 6.3.

6.2 HD DVD-Video Contents in Persistent Storages

A Disc Application may use data in Persistent Storages. A Persistent Storage may contain an S-EVOB. A condition is assumed for the protection format for an S-EVOB in Persistent Storages used by a Disc Application. The condition is described in Section 6.2.1. A Persistent Storage may contain Advanced Elements such as Images, Effect Audio, Fonts and so on. It may also contain Advanced Navigations which consist of XML Documents and ECMAScript Codes. The same condition is assumed for the encapsulation format of Advanced Resources used by a Disc Application.

Update of contents in Persistent Storages may occur. This causes introduction of a brand-new user experience. Even a Playlist in a Persistent Storage may be updated. Such Playlist update implies update of content architecture itself including playback sequences. Playlist update shall satisfy some conditions defined by the AACS content protection scheme. Section 6.2.2 describes the conditions for Playlist Update.

An application may generate some data which are to be stored in Persistent Storages. Among those data, only the following categories of data are to be protected by AACS: Captured images and XML Documents. Note that no method is provided for protection of application-generated ECMAScript Codes. An HD DVD-Video Player does not provide a functionality of data-encryption. It provides instead a MAC API for protection against data manipulation. Captured images *may* be protected by means of MAC from malicious modification. An application-generated XML Documents *cannot* be encrypted. They *shall*, however, be protected by means of MAC. Copyright protection of an application-generated content is mentioned in Section 6.2.3.

The protection format for an S-EVOB in Persistent Storages is equivalent to that defined in Section 4.3. Note that no Content Hash Check shall be performed for playback of an S-EVOB in a Persistent Storage. The Encapsulation Format for Encryption and the Encapsulation Format for MAC may be used to protect ARFs in Persistent Storages. The Encapsulation Format for Encryption and Hash and the Encapsulation Format for Hash, however, *must not* be used.

6.2.1 Contents in Persistent Storages Used by a Disc Application

There is a condition assumed for the protection format or the encapsulation format of contents to be used by a Disc Application. When a Disc Application is active and executed, the Title Keys which are active and available for the application shall be those in a TKF on the Disc. This means that Title Key Pointer in KMI (See 4.2) and Title Key Pointer in the encapsulation format (See 4.4.2) shall indicate a Title Key in the TKF on the Disc which corresponds to the Playlist. And when a Disc Application is active, a TUF corresponding to the Playlist on the Disc shall be used, which means that UR_PTR in URMI indicates a Usage Rule Set in the TUF on the Disc as long as the UR_PTR is valid. If UR_PTR in an EVOB is valid and if the TUF associated with the Playlist is absent, the Player shall immediately go to Stop State. Note that no Content Hash Check shall be performed for playback of an S-EVOB in a Persistent Storage. It is assumed here that the APIs, changeTKF() or changeTUF(), are not called. If changeTKF() is successfully called, for instance, the new TKF shall be used in place of the TKF which is associated by the default name convention.

6.2.2 Playlist Update

6.2.2.1 Data in a Persistent Storage

Playlist Update may occur for an Advanced Content in order to update or upgrade the user experience. A Playlist in a Persistent Storage may be updated by a Disc. Following instructions issued by a Disc Application or a P-Storage Application, a Player may copy a Playlist on the Disc into a Persistent Storage. The Player may also copy some resources required for playback defined in the Playlist. Another way to update a Playlist in a Persistent Storage is as follows: Suppose a Player connects to a server on the Internet. Following instructions issued by a Disc application or a P-Storage Application, the Player downloads a Playlist from the server and stores it in a Persistent Storage. The Player may also download related resources from the server and store them into Persistent Storages. File Cache in a Player may also be used to store a Playlist and other resources in place of a Persistent Storage though File Cache cannot keep its data if the Player goes to Stop State for instance.

A Playlist is an XML file, the name of which is reserved by the *HD DVD-Video Specifications*. 1000 names from “VPLST000.XPL” to “VPLST999.XPL” are reserved for audiovisual contents. 1000 names “APLST000.XPL” to “APLST999.XPL” are reserved for audio-only playback. “VPLST000.XPL” and “APLST000.XPL” do not reside in a Persistent Storage, but they may reside on a Disc if they exist. Playlists in Persistent Storages shall be encapsulated in Encapsulation Format for MAC so that they are protected from data manipulation.

Each Playlist in a Persistent Storage shall be accompanied by an MKB file. Two or more Playlists *must not* share the same MKB file. Each Playlist in a Persistent Storage shall also be accompanied by a TKF. Two or more Playlists are *not* allowed to share the same TKF. And each Playlist in a Persistent Storage shall be accompanied by a TUF. Two or more Playlists are *not* allowed to share the same TUF. The SHA-1 hash values of the TUF shall be contained in a Content Certificate file which is signed by the AACS LA. A Playlist file, an associated MKB file, an associated TKF file, an associated TUF file and an associated Content Certificate file shall reside in the same directory in a Persistent Storage. There is a file name convention for the combination of these five files: Suppose, for instance, “VPLST007.XPL” resides in a Persistent Storage. It shall be accompanied by an MKB file “MKB007.AACS”, a TKF “VTKF007.AACS”, a TUF “VTUF007.AACS” and a Content Certificate file “VCC007.AACS” in the same directory. The file names “VCC%%%.AACS” and “ACC%%%.AACS” are reserved in Persistent Storages, where %%% runs from 000 to 999. “VCC%%%.AACS” is for an audiovisual content and “ACC%%%.AACS” is for an audio-only content.

Combinations of AACS components and relevant contents in a Persistent Storage for Playlist update are as follows:

a) An MKB, a TKF and a Playlist.

b) An MKB, a TKF, a Playlist and contents.

c) An MKB, a TKF, a Content Certificate, a TUF and a Playlist.

d) An MKB, a TKF, a Content Certificate, a TUF, a Playlist and contents.

Here, contents may include EVOBs, Advanced Navigations and Advanced Elements. The preceding combinations for Playlist update may also be used for File Cache.

Table 6-1 shows the format of Content Certificate file in a Persistent Storage.

Table 6-1: Format of Content Certificate File in a Persistent Storage

Byte \ Bit	7	6	5	4	3	2	1	0
0	Certificate Type: 40_{16}							
1	Reserved							
2 : 5	Total_Number_of_HashUnits = 00000001h							
6	Total_Number_of_Layers = 00h							
7	Layer_Number = 00h							
8 : 11	Number_of_HashUnits = 00000001h							
12 13	Number_of_Digests = 0001h							
14 15	Applicant ID							

16 … 19	Content Sequence Number
20 21	Minimum CRL Version
22 23	Reserved
24 25	Length_Format_Specific_Section = 000ah
26 : 35	Reserved
36 : 55	Hash of TUF
56 : 95	Signature Data

The field of Certificate Type stores the value of 40h. Content Certificate ID for Persistent Storage shall be revoked by Revocation Record in CRL whose Record_Type = 00h, where Content Certificate ID for Persistent Storage is a combination of the 2-byte field of Applicant ID and the 4-byte field of Content Sequence Number. See the *AACS Pre-recorded Video Book* for Content Certificate ID and CRL. Note that, in contrast to Minimum CRL Version of Content Certificate on a Disc (See **Error! Reference source not found.**), Minimum CRL Version of Content Certificate in a Persistent Storage must not be checked against the CRL Version stored in the Player. In addition to the fields described in the *AACS Pre-recorded Video Book*, the Content Certificate file in a Persistent Storage contains the following fields:

- The field of Hash of TUF of 20 bytes stores the SHA-1 hash value of the associated TUF. The SHA-1 function is applied to the first HASH_SIZE bytes of “VTUF%%%.AACS” or “ATUF%%%.AACS”, where HASH_SIZE is a field in the TUF. If the associated TUF is absent, this field shall be filled with FFh.

The reserved fields shall all be filled with 00h.

6.2.2.2 Boot Sequence

In the first place, a Player shall determine the mode in which it starts playback of a Disc. Assume hereafter that the Player detects an AACS Disc and that it starts playback in the AACS Mode. In addition, assume that the Disc belongs to Category 2 or Category 3. The Player reads the Configuration file "DISCID.DAT" in the "ADV_OBJ" directory as defined in the *HD DVD-Video Specifications*. The file format of the Configuration File is defined in the specification. If SEARCH_FLG in the Configuration File is 1b, the Player tries to find a Playlist on the Disc whose number is the largest and proceeds to a boot sequence of the Playlist.

Suppose hereafter that SEARCH_FLG is 0b. In this case, too, the Player tries to find a Playlist on the Disc whose number is the largest. After that, the Player continues to find a Playlist in Persistent Storages which has the largest number in the Persistent Storages. Then, the Player starts a boot sequence of the Playlist which has the largest number among the Playlists on the Disc and in the Persistent Storages. Each Persistent Storage has a Provider Directory which is specified by each content provider. The Player shall search Playlists in the Provider Directory in a Persistent Storage. The name of the Provider Directory is defined in the manner described in Section 6.3.

In the Startup Sequence of Advanced Content, a Playlist shall have the name, "VPLST%%%.XPL" or "APLST%%%.XPL", where %%% runs from 000 to 999. On the other hand, the HD DVD-Video Specifications does not specify a filename which may be used as a Playlist in a Soft Reset. This Book puts a restriction, however, on a Playlist filename for a Soft Reset: A Playlist which may be used as an argument of Playlist.load() shall have the name, "VPLST%%%.XPL" or "APLST%%%.XPL", where %%% runs from 000 to 999.

Since a Playlist is AACS-protected by means of MAC, it is encapsulated by the format described in Table 4-11 in 4.4.2. Note here that the Title Key Pointer in the header of "VPLST007.XPL", say, indicates a Title Key in "VTKF007.AACS". At the beginning of a boot sequence of the Playlist, the MAC of the Playlist file shall be checked using the Title Key. This means that, before the booting process, "MKB007.AACS" and "VTKF007.AACS" shall be processed by the AACS module in the Player system so that Title Keys in the TKF are available for the integrity check. Of course, the integrity of the TUF "VTUF007.AACS", if it exists, shall be verified based on the associated Content Certificate file "VCC007.AACS".

An HD DVD-Video Player has an initial boot sequence, i.e. Startup Sequence, in which the Player finds the latest Playlist for the Disc to be played back. The latest Playlist may reside in a Persistent Storage. In

the initial boot sequence, the Player shall automatically find the MKB file and the TKF which are associated with the Playlist and reside in the same directory. And if the associated TUF exists, the Player shall read it and read the Content Certificate file which is associated with the Playlist. Then, the Player shall pass them to the AACS module for processing. If the associated TUF is absent, there is no TUF for the EVOBs to be played back in the Playlist. The Player loads the Playlist in the Player system, and the boot sequence for a Playlist defined in the *HD DVD-Video Specifications* follows. Suppose the uri of the Playlist to be loaded is "file:///additional/'Base Path'/'Content ID'/VPLST011.XPL". (cf. 10.3.1 of the *HD DVD-Video Specifications*) The following is one example of the boot sequence:

1. The AACS module at first reads and processes "file:///additional/'Base Path'/'Content ID'/MKB011.AACS".
2. The AACS module reads and processes "file:///additional/'Base Path'/'Content ID'/TKF011.AACS" so that the AACS module may prepare Title Keys.
   - TKF MAC is verified.
3. If "file:///additional/'Base Path'/'Content ID'/TUF011.AACS" does not exist, go to 4. Otherwise:
   - The Player reads and verifies "file:///additional/'Base Path'/'Content ID'/VCC011.AACS".
   - The Player reads and verifies integrity of "file:///additional/'Base Path'/'Content ID'/TUF011.AACS" using the hash value contained in "file:///additional/'Base Path'/'Content ID'/VCC011.AACS".
     - TUF MAC is verified.
4. Proceed to process the Playlist.

If one of the steps, 1, 2 and 3 fails, the system immediately goes to Stop State. After the preceding boot sequence for AACS components is successfully executed, the system proceeds to process the Playlist, "file:///additional/'Base Path'/'Content ID'/VPLST011.XPL". Note that the associated AACS components shall reside in the appropriate directory before the concerned Playlist is loaded. The content provider is responsible for preparation of a Playlist and the associated AACS components. As mentioned before, possible combinations of data in a Persistent Storage or in File Cache for Playlist update are as follows:

1) An MKB, a TKF and a Playlist.
2) An MKB, a TKF, a Playlist and contents.
3) An MKB, a TKF, a Content Certificate, a TUF and a Playlist.
4) An MKB, a TKF, a Content Certificate, a TUF, a Playlist and contents.

When a boot sequence starts for a Playlist in File Cache, the Playlist is saved somewhere in the system before cleanup of File Cache and it is restored after the cleanup. The relevant AACS components, an MKB, a TKF, etc., shall also be saved and then restored by the Player.

Before a Title Key in a TKF is used, the associated Binding MAC shall be checked. If it fails, the Title Key must not be used. When Before a URS in a TUF is used, the associated BURS shall be checked. If it fails, the URS must not be applied.

In the boot sequence, the AACS module in the Player may keep the Title Keys in a secure manner. In that case, the AACS module in the Player shall be reset and the decrypted Title Keys shall be cleared from the system when the following conditions hold in the boot sequence:

i) A Player starts an initial boot sequence.

ii) The associated Disc is ejected.

iii) The Player loses power.

iv) Another boot sequence starts.

- This may be caused by a call of the API, Playlist.load().

Every Title Key in the TKF file is bound to the Volume Identifier of a Disc. The associated Disc means such a Disc.

When a P-Storage Application is active and executed, the Title Keys which are active and available for the application shall be those in a TKF in the Persistent Storage. This means that Title Key Pointer in KMI (See 4.2) and Title Key Pointer in the encapsulation format (See 4.4.2) shall indicate a Title Key in the TKF in the same directory which stores the Playlist. And when a P-Storage Application is active, a TUF corresponding to the Playlist in a Persistent Storage shall be used, which means that UR_PTR in URMI indicates a Usage Rule Set in the TUF in the Persistent Storage which is associated with the Playlist as long as the UR_PTR is valid. If UR_PTR in an EVOB is valid and if the TUF associated with the Playlist is absent, the Player shall immediately go to Stop State. It is assumed here that the APIs, changeTKF() or changeTUF(), are not called. If changeTKF() is successfully called, for instance, the new TKF shall be used in place of the TKF which is associated by the default name convention. Note that no Content Hash Check shall be performed for playback of an S-EVOB in a Persistent Storage.

6.2.2.3 Content Hash Check

As mentioned in Section 6.2.1, no Content Hash Check is required when an S-EVOB in a Persistent Storage is played back. There is, however, a case where playback of a P/S-EVOB on a Disc is defined in a

Playlist in a Persistent Storage. In such a case, content hash checking of the P/S-EVOB is necessary. The CHT #1 on the Disc has the hashes of all the EBOVU/TUs in the P/S-EVOB. A Player shall use the CHT #1 for content hash checking of the P/S-EVOB on the Disc according to the method described in Section 4.3.3.

And there is also a case where XML Documents or ECMAScript Codes on a Disc are used by a P-Storage Application. In this case, too, the integrity of the Advanced Navigation Files shall be verified in accordance with the scheme defined in Section 4.4. Hash values of ANFs on the Disc shall be verified.

6.2.3 APIs for Loading and Saving

See Annex Z of the *HD DVD-Video Specifications* for the APIs described in this section. There are some APIs for loading/saving XML Documents from/to an HD DVD-Video Disc, File Cache, Persistent Storages or a Network Server. Loading an XML Document, here, means that XML Parser reads the XML Documents for parsing and processing. According to the content protection policy described in Section 1.2.2, all the XML Documents and all the ECMASccript Codes shall be encapsulated. Therefore, loading an XML Document shall be accompanied by de-capsulation and decryption and/or verification of MAC/hash. The same is true for loading ECMAScript Codes.

There is, however, no specific API for loading ECMAScript Codes. The APIs for loading an XML Documents includes the followings: Application.link(uri: String), HTTPClient.getResponseXML(), Playlist.load(uri: String) and XMLParser.parse(uri: String, callback: Function). Note that there occurs loading of an XML Document when, for instance, the <link> element in a Markup become active. De-capsulation and decryption and/or verification of MAC/hash shall be performed every time a Markup is loaded. If the verification of MAC/hash fails, as is described in Section 4.4.2, the Player must not use the Markup and shall behave as if the Markup did not exist. The Player will throw the exception of HDDVD_E_FILENOTFOUND or set error info of FILE_NOT_FOUND for the callback, etc.

The API Playlist.load() makes a Soft Reset of the Player and starts a new boot sequence with the designated Playlist. If the designated Playlist is on the Disc, the boot sequence for a Disc Application shall be performed. And if the designated Playlist is in a Persistent Storage, the boot sequence for a P-Storage Application shall be performed. There may be a case where the designated Playlist resides in File Cache (API Managed Area). In this case, the same boot sequence as that for a P-Storage Application shall be performed. That is, a directory in API Managed Area is regarded as a directory in a Persistent Storage, and the same name convention as applied to AACS components in a Persistent Storage shall be applied to AACS components in API Managed Area.

There is an API which reads an XML Document from a buffer in Script Engine: XMLParser.parseString(string: XMLString). The AACS module does not assume the encapsulation format for this API, and thus the AACS module does not intervene in the behavior of this API. Therefore, it is *strongly recommended* that the API, XMLParser.parseString(), *should not* be used in AACS Mode.

Saving an XML Document, here, means that a DOM Document expanded in XML Parser is saved in File Cache, a Persistent Storage or a Network Server in the form of an XML Document. In this case, the CMAC value shall be calculated before the XML Document is written so that integrity of it is maintained. The calculated MAC shall be attached to the XML Document, which means that it shall be encapsulated in Encapsulation Format for MAC. The CMAC value shall be calculated using a Title Key set by setMACKey(). The same is true for saving ECMAScript Codes, but there is no specific API for saving ECMAScript Codes. The APIs for saving an XML Document includes the followings: HTTPClient.sendXML(doc: Document) and XMLParser.write(doc: Document, uri: String, encoding: int, callback: Function). If the designated Title Key is invalid, HTTPClient.sendXML() shall throw the exception AACS_E_INVALIDKEY. The value of the exception AACS_E_INVALIDKEY is “AACS_E_INVALIDKEY”. And, if the designated Title Key is invalid, XMLParser.write() shall call the callback function with setting the argument at AACS.INVALIDKEY.

There is two APIs for I/O which have a different meaning: FileIO.openTextFile() and FileIO.saveTextFile(). Although the former may be used to read an XML Document into a buffer in Script Engine, the XML Document is not parsed by Script Engine. No de-capsulation shall accompany FileIO.openTextFile(). Therefore, MAC verification and/or decryption must not accompany FileIO.openTextFile(). No DOM Document can be saved by FileIO.saveTextFile(). Therefore, no MAC calculation accompanies FileIO.saveTextFile(), and the saved text file is not encapsulated.

In the *HD DVD-Video Specifications*, there are three APIs related to image capturing: DrawingArea.capture(), MainVideo.capture() and MainVideo.changeImageSize(). They respectively have the MAC versions: DrawingArea.captureWithMAC(), MainVideo.captureWithMAC() and MainVideo.changeImageSizeWithMAC(). The argument of the DrawingArea.captureWithMAC() is a URI of a file to store the captured drawing image. The first argument of MainVideo.captureWithMAC() is a URI of a file to store the captured video image. The second argument of MainVideo.captureWithMAC() is a callback function. The first argument and the second argument of the callback function are respectively a status and a uri which specifies the file storing the captured image. Note that these three APIs are defined only when a Player is in AACS mode. Therefore, if they are called in non-AACS mode, an exception HDDVD_E_INVALIDCALL is thrown.

DrawingArea.captureWithMAC() saves the current drawing image in File Cache. Before writing the drawing image, the CMAC value of the image shall be calculated using the Title Key which is indicated by the API setMACKey(). The saved image, thus, shall be encapsulated in Encapsulation Format for MAC. The argument of DrawingArea.captureWithMAC() is a uri which specifies the file name to store the captured image. If the designated Title Key is invalid, DrawingArea.captureWithMAC() shall throw the exception AACS_E_INVALIDKEY.

MainVideo.captureWithMAC() saves the current Main Video image in File Cache. Before writing the captured image, the CMAC value of the image shall be calculated using the Title Key which is indicated by the API setMACKey(). The saved image, thus, shall be encapsulated in Encapsulation Format for MAC. If the designated Title Key is invalid, MainVideo.captureWithMAC() shall call the callback function with setting the first argument at AACS.INVALIDKEY.

MainVideo.changeImageSizeWithMAC() changes the scale of a captured Main Video image in File Cache which is encapsulated in Encapsulation Format for MAC. MainVideo.changeImageSizeWithMAC() shall guarantee the integrity of the scaled image. The first argument is a uri which specifies the source image file and the second argument is a uri which specifies the destination file to store the scaled image. The third argument and the fourth argument are the numerator and the denominator, respectively. The fifth argument is a callback function. The first argument and the second argument of the callback function are respectively a status and a uri of the destination file. The destination file will be a file encapsulated in Encapsulation Format for MAC where the Title Key for MAC calculation is what is set by setMACKey(). If the Title Key is invalid, MainVideo.changeImageSizeWithMAC() shall call the callback function with setting the first argument at AACS.INVALIDKEY.

6.3 Protection of Directory Name for a Content Provider in Persistent Storages

A Configuration File whose name is “DISCID.DAT” resides in the directory “ADV_OBJ” on a Disc. The format of the Configuration File is defined in the *HD DVD-Video Specifications*. An AACS Disc contains a Directory Key File in the “AACS” directory. The name “DKF.AACS” is reserved for the Directory Key File. The format of the Directory Key File is shown in Table 6-2.

Table 6-2: Format of Directory Key File

Byte \ Bit	7	6	5	4	3	2	1	0	
0 : 11	DKF_ID								Header
12 : 15	HD_VDKF_SIZE								
16 : 31	Reserved								
32 33	VERN								
34 : 47	Reserved								
48 : 63	Encrypted Directory Key (K_{DIRe})								

The Directory Key File consists of the following fields:

- DKF_ID of 12 bytes which is an identifier of the Directory Key File. The value shall be "DVD_HD_V_DKF" with character set code of ISO/IEC 646:1983 (a-characters).
- HD_VDKF_SIZE of 4 bytes which indicates the size of the Directory Key File. The value shall be 64.
- VERN of 2 bytes which indicates the version number of the Directory Key File. This value shall be 0 for the current version.

- Encrypted Directory Key (K_{DIRe}) of 16 bytes. The following equation holds: K_{DIRe} = AES-128E(K_{vu}, K_{DIR}), where AES-128E denotes encryption by the AES algorithm in the ECB mode defined in the *AACS Introduction and Common Cryptographic Elements*, K_{DIR} is a Directory Key and K_{vu} is the Volume Unique Key defined in the *AACS Pre-recorded Video Book*.

The reserved fields shall be filled with 00h.

Suppose an AACS Disc is inserted in a Player so that the Player will start playback in AACS Mode. In the initial boot sequence, the Player system automatically reads the Configuration File "/ADV_OBJ/DISCID.DAT" when the Player shall verify the integrity of the file by the Content Hash Check. The hash value of the Configuration File is contained in the CHT #2 on the AACS Disc. Suppose SEARCH_FLG is equal to 0b, which means the Player need search Playlists from Persistent Storages as well as from the Disc. The Player passes the Configuration File to the AACS module which has already generated the Volume Unique Key. The AACS module reads the Directory Key File when it shall check the integrity of the file by the Content Hash Check. The hash value of the DKF is contained in the CHT #2 on the AACS Disc. Then the AACS module decrypts the Directory Key (K_{DIR}) in order to obtain the PROVIDER_DIR value of 16 bytes as follows:
PROVIDER_DIR = AES-G(K_{DIR}, PROVIDER_ID).
PROVIDER_ID, which is defined as a GUID in the *HD DVD-Video Specifications*, is stored in the Configuration File.

According to the *HD DVD-Video Specification*, a medium which serves as a Persistent Storage shall have such a directory structure as shown in Figure 6-1. A Player system automatically creates the "HD_DVD" directory in the medium if it does not already exist, and it stores the medium information in the file "/HD_DVD/INFO.TXT". The Player system creates a Provider Directory if it does not already exist. The directory name shall be the GUID which expresses PROVIDER_DIR. The file "INFO.TXT" in the Provider Directory stores information about the content provider. It is not written by the system, but by applications. The Content ID directory is also created by applications. The directory name is the string of the GUID which expresses CONTENT_ID in Configuration File. According to the *HD DVD-Video Specifications*, the "INFO.TXT" file in the Content ID directory should contain information about the content. The "INFO.TXT" file is written by applications.

There may be some image files in the "PROVIDER_DIR" directory which are used as icons to show logos or explanations of the content provider or contents provided by the content provider. They may be used to display information of each provider and the contents in the management screen of a Player. Therefore, it is

recommended that they should not be encapsulated. See Sections 10.3 and 10.4 of the *HD DVD-Video Specifications*.

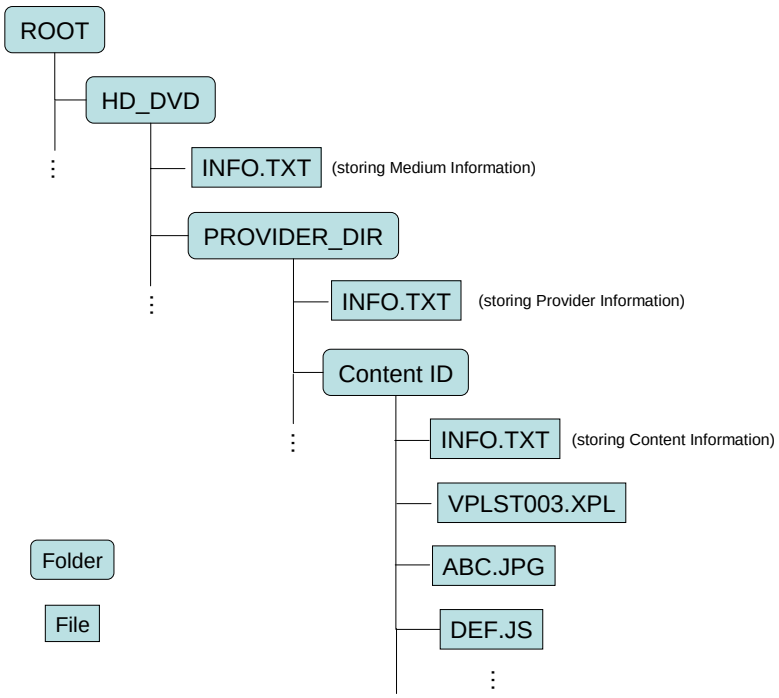

Figure 6-1: Directory Structure for Persistent Storage

Chapter 7
Sequence Key

7 Sequence Key

7.1 Introduction

This section describes how a Player behaves for a VTS with Sequence Key. If an AACS Disc has a SKBF, there shall be at most six SKRs on the Disc. See Section 3.3 for SKBF, SKR and other related terminologies. A P-EVOB in an SKR *must not* have more than 32 Sequence Key Sections. Each Sequence Key Section is an Interleaved Block like an Angle Block. The Player behavior at a Sequence Key Section is similar to that at an Angle Block. Section 7.2 describes the mechanism and the Player behavior for the Sequence Key in the Standard VTS. Section 7.3 describes the mechanism and the Player behavior for the Sequence Key in the Advanced VTS. Note that only a P-EVOB on a Disc may have Sequence Key Sections. A Sequence Key Section continues in a time span. The time span has a minimum length which is defined in the *HD DVD-Video Specifications*. That is, the duration of a Sequence Key Section shall be more than the minimum time interval.

Before processing a P-EVOB with Sequence Key Sections, the AACS module shall process the file "/AACS/SKBF.AACS" and the file "/AACS/SKF.AACS" and shall have six SKTs each of which consists of 32 pairs of an SEG_NO and a Segment Key. The order of the pairs is important. The order shall be kept in an SKT as they appear in the corresponding SKU field. The six SKTs shall be stored in the AACS module as one table called Segment Key Table Set (SKTS). The first 32 entries of the SKTS are identical to the entries of the first SKT, the next 32 entries of the SKTS are identical to the entries of the second SKT, …, the last 32 entries of the SKTS are identical to the entries of the sixth SKT. The order of the SKTs follows the order of the SKGs in the SKF. The entry number of the SKTS varies from 1 to 192 and that is the number which SEG_KEY_PTR in KMI of an EVOBU indicates. Note that SEG_NO runs from 1 to 8.

As mentioned in the preceding Chapters, there exists the case where playback starts by processing a Playlist in a Persistent Storage. In this case, an MKB which accompanies the Playlist is used, which means that MKB Process for the MKB may yield a Media Key different from the Media Key yielded by the MKB on the Disc. As long as Sequence Key concerns, thus, there are two implications: i) All the MKBs on a Disc which respectively accompany the corresponding Playlists shall yield the same Media Key. ii) The Media

Key thus yielded by processing an MKB on the Disc shall be used for SKB Process. Suppose a Disc has an SKBF. In general, the SKBF is processed to yield an SKTS when the Disc is inserted into the Player. The implication ii) above means that the same SKTS shall be used when a P-Storage Application plays back a P-EVOB on the Disc equipped with Sequence Key Sections.

Playback of a Sequence Key Section shall be seamless as long as the conditions for seamless playback described in the Annex K in the *HD DVD-Video Specifications*. Furthermore, a transition from an SKR to another SKR *must not* prohibit seamless playback if seamless transition from a P-EVOB in the first SKR to a P-EVOB in the second SKR is guaranteed by the conditions in the Annex K.

7.2 Sequence Key for Standard VTS

In a Standard VTS, a Sequence Key Section is defined as a Cell Block corresponding to an Interleaved Block in the P-EVOB. Each Cell which belongs to a Sequence Key Section has a mark, i.e. the Cell Block Type in C_CAT in C_PBI. If a Cell belongs to a Sequence Key Section, the Cell Block Type of the Cell must be equal to 11b. See the conceptual image of a Sequence Key Section in Figure 7-1. In the C_PBI field in a Cell of a Sequence Key Block, there is a field of 2 bits which is reserved for KEY_VF in the HD DVD-Video Specification. The meaning of the KEY_VF values is as follows:

- 00b: SEG_KEY_PTR is not valid, 01b: SEG_KEY_PTR is valid, others: Reserved.

And, in the C_PBI field in a Cell which belongs to a Sequence Key Block, there is a field of 8 bits which is reserved for SEG_KEY_PTR in the HD DVD-Video Specification. The SEG_KEY_PTR indicates an entry of the SKT. The value of the SEG_KEY_PTR runs from 1 to 192. See Table 7-1 for the bit assignment for the reserved field in C_PBI.

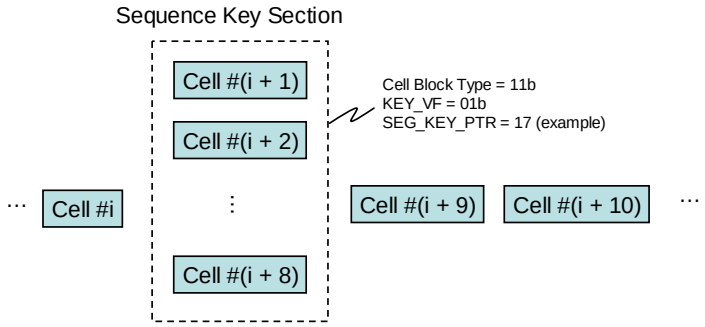

Figure 7-1: An Image of Sequence Key Section

Table 7-1: Bit Assignment for the Reserved Field of C_PBI

Byte \ Bit	7	6	5	4	3	2	1	0
0	KEY_VF		SEG_KEY_PTR					
1	SEG_KEY_PTR		Reserved					

7.2.1 Entering into a Sequence Key Section

If a Player encounters a Cell in PGCI which belongs to a Sequence Key Section, it shall execute the following sequence:

1. Disable the function of Angle Change before the ILVUs are read into Track Buffer.
2. Read the field SEG_KEY_PTR in C_PBI in the first Cell for the Sequence Key Section.
   a. The value of SEG_KEY_PTR runs from 1 to 192.
3. Look up the entry of SKT which the SEG_KEY_PTR indicates.
   a. Let the entry be the pair (SEG_NO*, SEG_KEY*).

4. Find the SEG_NO*-th Cell in the Sequence Key Section.
5. Read C_FEVOBU_SA in C_PBI of the Cell found in 4 above.
6. Jump to the ILVU pointed by C_FEVOBU_SA read in 5 above.
7. Decrypt the encrypted portion of the ILVU by the Segment Key SEG_KEY*.
   - Packs in the ILVU are decrypted in the same manner as defined in Section 4.3.4 with the Segment Key being used in place of the Title Key. That is, the Content Key (K_c) is calculated as follows:
     K_c = AES-G (SEG_KEY*, D_{tk} || CPI_{lsb_96}).
   - If the preceding SEG_KEY_PTR in C_PBI is valid and if SKB or SKF are absent, the Player shall immediately go into Stop State.

There may be a case where playback jumps into a Sequence Key Section. Before jumping into a Sequence Key Section and before the ILVUs are read into Track Buffer, the function of Angle Change shall be disabled.

7.2.2 Transitions among Interleaved Units in a Sequence Key Section

After entering into a Sequence Key Section, a Player decrypts and plays back ILVUs in the Interleaved Block for the Sequence Key Section, making transitions among the ILVUs. This section describes how to make the transitions. An ILVU in an Interleaved Block has two fields, KEY_VF (2 bits) and SEG_KEY_PTR (8 bits), reserved in the HD DVD-Video Specification. The value of KEY_VF of an ILVU in an Interleaved Block corresponding to a Sequence Key Section must be 01b. The meaning of the value of KEY_VF is the same as defined above. The meaning of the field SEG_KEY_PTR is also the same.

There is a field called SML_AGLI in the DSI of an ILVU. This is a field for the function of Angle Change defined in the *HD DVD-Video Specifications*. If the value of KEY_VF equals 01b, however, the SML_AGLI field is reserved for SML_SEQI. Table 7-2 and Table 7-3 together describe the format of SML_SEQI. In addition, SML_SEQ_Cn_DSTA = 7FFFFFFFFFFFh for n not equal to SEG_NO*, which means those fields are all invalid. Therefore, it always holds that SML_SEQ_C9_DSTA = 7FFFFFFFFFFFh. For n equal to SEG_NO*, SML_SEQ_Cn_DSTA stores the start address of the next ILVU. After playing back the current ILVU, the Player jumps to the next ILVU, decrypts the encrypted portion of the ILVU by the Segment Key SEG_KEY* and plays back the ILVU. Note that a Segment Key is used in place of a Title Key, which means that the Pack decryption scheme follows the description in Section 4.3.4 with the Title Key being replaced by the Segment Key. That is, the Content Key (K_c) is calculated as follows: K_c = AES-G

(SEG_KEY*, D_{tk} || CPI_{lsb_96}). If SEG_KEY_PTR in CPI in an ILVU is valid and if SKB or SKF are absent, the Player shall immediately go into Stop State.

Table 7-2: SML_SEQI

Field Names	Contents	Number of bytes
SML_SEQ_C1_DSTA	Address and size of destination ILVU in SEQ_C1	6 bytes
SML_SEQ_C2_DSTA	Address and size of destination ILVU in SEQ_C2	6 bytes
SML_SEQ_C3_DSTA	Address and size of destination ILVU in SEQ_C3	6 bytes
SML_SEQ_C4_DSTA	Address and size of destination ILVU in SEQ_C4	6 bytes
SML_SEQ_C5_DSTA	Address and size of destination ILVU in SEQ_C5	6 bytes
SML_SEQ_C6_DSTA	Address and size of destination ILVU in SEQ_C6	6 bytes
SML_SEQ_C7_DSTA	Address and size of destination ILVU in SEQ_C7	6 bytes
SML_SEQ_C8_DSTA	Address and size of destination ILVU in SEQ_C8	6 bytes
SML_SEQ_C9_DSTA	Address and size of destination ILVU in SEQ_C9	6 bytes
	Total	54 bytes

Table 7-3: SML_SEQ_Cn_DSTA

bit Byte	7	6	5	4	3	2	1	0
0	SEQ_C location	Destination address of SEQ_C #n [30…24]						
1	Destination address of SEQ_C #n [23…16]							
2	Destination address of SEQ_C #n [15…8]							
3	Destination address of SEQ_C #n [7…0]							
4	Size of destination ILVU of SEQ_C #n [15…8]							

5	Size of destination ILVU of SEQ_C #n [7…0]

7.2.3 Getting Out of a Sequence Key Section

At the last ILVU to play in the Interleaved Block, all the SML_SEQ_Cn_DSTA fields in the SML_SEQI shall be invalid, i.e. 7FFFFFFFFFFFh. Thus, a Player finds an invalid SML_SEQ_Cn_DSTA for n = SEG_NO*. Then, the Player enables again the function of Angle Change and continues to play the next Contiguous Block or the next Sequence Key Section.

Note, for a Sequence Key Section, that the System Parameters such as SPRM3 or SPRM28 are not referred to by a Player and that the value of any of the System Parameters is not changed by a Player. There are some restrictions on the number of Angles in a Angle Block (See 5.1.3.5.2 in the *HD DVD-Video Specifications*). The number of Angles is, of course, independent of the number of Sequence Key Segments in a Sequence Key Section which is equal to 8.

7.3 Sequence Key for Advanced VTS

Sequence Key is realized almost in the same manner as described in the previous section. The difference is that an Advanced VTS does not have the Cell structure but a Playlist and TMAPs. TMAPI is an element of TMAP and is used to convert from a given presentation time inside an EVOB to the address of an EVOBU. A TMAPI consists of one or more EVOBU Entries. One TMAPI for one EVOB which belongs to a Contiguous Block shall be stored in one TMAP file. TMAPIs for EVOBs which belong to one Interleaved Block shall be stored in one TMAP file. A TMAP consists of a TMAP_GI, one or more TMAPI_SRPs (TMAPI Search Pointers), one or more TMAPIs and ILVUI if the TMAP is for an Interleaved Block.

A Sequence Key Section is an Interleaved Block. The TMAP for the Sequence Key Section contains 8 TMAPIs. TMAP_GI of the TMAP has an 8-bit field reserved for SEG_KEY_PTR, and there is a field reserved for KEY_VF in TMAP_TY of TMAP_GI. The value of KEY_VF shall be 01b, which means that the TMAP is for a Sequence Key Section. See Table 7-4 and Table 7-5 for the assignment of the SEG_KEY_PTR field in TMAP_GI and the KEY_VF field in TMAP_TY. The meaning of the KEY_VF values is the same as described in 7.2:

- 00b: SEG_KEY_PTR is not valid, 01b: SEG_KEY_PTR is valid, others: Reserved.

The angleNumber attribute of <Video> Element in <PrimaryAudioVideoClip> Element must not be specified, that is, it shall be omitted.

Table 7-4: TMAP_GI for AACS-Compliant Disc

Field Name	Contents	Number of Bytes
TMAP_ID	TMAP Identifier	12
TMAP_EA	End Address of TMAP	4
Reserved		2
TMAP_VERN	Version Number	2
TMAP_TY	Attribute of TMAP	2
Reserved		28
Reserved	Reserved for Interoperable VTS	5
TMAPI_Ns	Number of TMAPIs	2
ILVUI_SA	Start Address of ILVUI	4
EVOB_ATR_SA	Start Address of EVOB_ATR	4
Reserved		49
VTSI_FNAME	Filename of VTSI	255
SEG_KEY_PTR	SEG_KEY_PTR	1
ILVU_ENT_Ns	Number of ILVU_ENT in ILVB	4
Reserved		10
Total		384

Table 7-5: TMAP_TY for AACS-Compliant Disc

Bit / Byte	7	6	5	4	3	2	1	0
1	Application type				KEY_VF		ILVUI	ATR

2	Reserved	Angle

7.3.1 Entering into a Sequence Key Section

If a Player encounters a TMAP for a Sequence Key Section, it shall execute the following sequence:

1. Read the field SEG_KEY_PTR in TMAP_GI of the TMAP.
    a. The value of SEG_KEY_PTR runs from 1 to 192.
2. Look up the entry of SKT which the SEG_KEY_PTR indicates.
    a. Let the entry be the pair (SEG_NO*, SEG_KEY*).
3. Find the TMAPI #(SEG_NO*) in the TMAP using the TMAPI_SRPs.
    a. Read 1STREF_SZ of the EVOBU_ENT#1 in the TMAPI
4. Find the ILVU_ENT #(SEG_NO*) in the ILVUI in the TMAP.
    a. Read ILVU_ADR of the ILVU_ENT.
5. Jump to the ILVU using the ILVU_ADR read in 4 and the 1STREF_SZ read in 3.
6. Decrypt the encrypted portion of the ILVU by the Sequence Key SEG_KEY*.
    - Packs in the ILVU are decrypted in the same manner as defined in 4.3.4 with the Segment Key being used in place of the Title Key. That is, the Content Key (K_c) is calculated as follows: K_c = AES-G (SEG_KEY*, D_{tk} || CPI_{lsb_96}).
    - If the preceding SEG_KEY_PTR in TMAP_GI is valid and if SKB or SKF are absent, the Player shall immediately go into Stop State.

A Sequence Key Section looks like a Contiguous Block in a Playlist. In playback of an Advanced VTS, a Player shall play back a Sequence Key Section as defined in this section ignoring the value of the angleNumber attribute.

7.3.2 Transition among Interleaved Units in a Sequence Key Section

After entering into a Sequence Key Section, a Player decrypts and plays back ILVUs in the Interleaved Block for the Sequence Key Section, making transitions among the ILVUs. The data required for the transition and the mechanism for it are identical to those for the Standard VTS. See the Section 7.2.2 for the description.

7.3.3 Getting Out of a Sequence Key Section

At the last ILVU to play in the Interleaved Block, all the SML_SEQ_Cn_DSTA fields in the SML_SEQI shall be invalid, i.e. 7FFFFFFFFFFh. Thus, a Player finds an invalid SML_SEQ_Cn_DSTA for n = SEG_NO*. Then, the Player continues to play the next Contiguous Block or the next Sequence Key Section.

This page is intentionally left blank.

Appendices

A Schema for Managed Copy Manifest File

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:hdmc="http://www.aacsla.com/2006/02/hdmcManifest"
targetNamespace="http://www.aacsla.com/2006/02/hdmcManifest"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xs:element name="mcManifest">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="hdmc:Cid"/>
          <!-- Content ID -->
        <xs:element ref="hdmc:serverList" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="hdmc:offersNManifest" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="Id" use="optional" type="xs:string"/>
        <!-- The length of ID shall be no more than 256. -->
    </xs:complexType>
  </xs:element>

  <xs:element name="Cid" final="restriction">
    <xs:simpleType>
      <xs:restriction base="xs:base64Binary">
        <xs:maxLength value="24"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
```

```
  <!-- ISAN -->

<xs:element name="serverList">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="hdmc:serverUri" maxOccurs="16"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="serverUri" final="restriction">
  <xs:simpleType>
    <xs:restriction base="xs:anyURI">
      <xs:maxLength value="1024"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
  <!-- The length of URI shall not be more than 1024 -->

<xs:element name="offersNManifest">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="hdmc:offers"/>
      <xs:element ref="hdmc:dealManifest"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- offers -->
<xs:element name="offers">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="hdmc:offer" maxOccurs="256"/>
    </xs:sequence>
```

```
    <xs:attribute name="version" use="required" type="xs:decimal"/>
      <!-- The totalDigits shall not be more than 10 -->
    <xs:attribute name="render" use="optional" type="xs:anyURI"/>
      <!-- The length of URI shall not be more than 1024 -->
    <xs:attribute name="MKB" use="optional" type="xs:string"/>
      <!-- The MKB attribute must not exist for offers recorded on a Disc -->
    <xs:attribute name="sessionID" use="optional" type="xs:string"/>
      <!-- The sessionID attribute must not exist for offers recorded on a Disc -->
      <!-- The sessionID attribute is mandatory for offers returned by an MCS -->
      <!-- The length of sessionID shall not be more than 32 -->
  </xs:complexType>
</xs:element>

<xs:element name="offer">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="hdmc:MCUi"/>
      <xs:element ref="hdmc:title"/>
      <xs:element ref="hdmc:abstract"/>
      <xs:element ref="hdmc:description"/>
      <xs:element ref="hdmc:image"/>
      <xs:element ref="hdmc:language"/>
      <xs:element ref="hdmc:MCOT"/>
      <xs:element ref="hdmc:availability"/>
      <xs:element ref="hdmc:price" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="hdmc:financialApplicationURI"/>
      <xs:element ref="hdmc:financialHTMLURL"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="MCUi" final="restriction">
  <xs:simpleType>
    <xs:restriction base="xs:string">
```

```
      <xs:maxLength value="32"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
  <!-- The value “0” is reserved for copy of the whole Disc -->

<xs:element name="title" final="restriction">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="1024"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="abstract" final="restriction">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="4096"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="description" final="restriction">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="65536"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="image">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="hdmc:url"/>
```

```
      <xs:element ref="hdmc:title"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="url" final="restriction">
  <xs:simpleType>
    <xs:restriction base="xs:anyURI">
      <xs:maxLength value="1024"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
  <!-- The length of URI shall not be more than 1024 -->

<xs:element name="language" final="restriction">
  <xs:simpleType>
    <xs:restriction base="xs:NCName">
      <xs:maxLength value="256"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
  <!-- The maximum length of language shall be 256 -->

<xs:element name="MCOT" final="restriction">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="256"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
  <!-- The maximum length of MCOT shall be 256 -->

<xs:element name="availability">
  <xs:complexType>
```

```
    <xs:attribute name="start" type="xs:dateTime" use="required"/>
    <xs:attribute name="end" type="xs:dateTime" use="required"/>
    <!-- The maximum length of start and end shall be 32 -->
  </xs:complexType>
</xs:element>

<xs:element name="price" final="restriction">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="256"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
  <!-- The maximum length of price shall be 256 -->

<xs:element name="financialApplicationURI" final="restriction">
  <xs:simpleType>
    <xs:restriction base="xs:anyURI">
      <xs:maxLength value="1024"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
  <!-- The length of URI shall not be more than 1024 -->

<xs:element name="financialHTMLURL" final="restriction">
  <xs:simpleType>
    <xs:restriction base="xs:anyURI">
      <xs:maxLength value="1024"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
  <!-- The length of URI shall not be more than 1024 -->

<!-- dealManifest -->
```

```
  <xs:element name="dealManifest">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="hdmc:mcUnit" maxOccurs="256"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="mcUnit">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="hdmc:MCUi"/>
        <xs:element ref="hdmc:item" maxOccurs="65536"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="item">
    <xs:complexType>
      <xs:attribute name="Id" type="xs:positiveInteger" use="optional"/>
        <!-- Id shall be less than 2^32 -->
      <xs:attribute name="Src" type="xs:anyURI" use="required"/>
        <!-- The length of URI shall not be more than 1024 -->
      <xs:attribute name="Encapsulation" type="xs:boolean" use="optional" default="true"/>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

This page is intentionally left blank.

B Managed Copy Permission XML Schema for HD DVD and DVD

```
<?xml version="1.0"?>
<xs:schema targetNamespace="http://www.aacsla.com/2006/02/hdmanagedPermission"
xmlns:tns="http://www.aacsla.com/2006/02/hdmanagedPermission"
xmlns:hdmc="http://www.aacsla.com/2006/02/hdmcManifest"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<xs:import namespace="http://www.aacsla.com/2006/02/hdmcManifest"
schemaLocation="aacs_hdmcManifest.xsd"/>
<xs:element name="permission">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="permissionSignedContent">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="tns:status" minOccurs="1" maxOccurs="1" />
            <xs:element ref="tns:mcotInfo" minOccurs="0" maxOccurs="1" />
            <xs:element ref="hdmc:mcUnit" minOccurs="0" maxOccurs="1" />
            <xs:element ref="tns:mcmNonce" minOccurs="1" maxOccurs="1" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element ref="tns:MCScert" minOccurs="1" maxOccurs="1" />
      <xs:element ref="tns:signature" minOccurs="1" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="MCScert" type="xs:base64Binary"/>
```

```
<xs:element name="status" type="xs:boolean"/>

<xs:element name="signature" type="xs:base64Binary"/>

<xs:element name="mcotInfo" type="xs:base64Binary"/>

<xs:element name="mcmNonce" type="xs:string"/>

</xs:schema>
```

C Managed Copy Web Service Description for HD DVD and DVD

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions targetNamespace="http://www.aacsla.com/2006/02/managedCopyService"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:tns="http://www.aacsla.com/2006/02/managedCopyService"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:aacsmp="http://www.aacsla.com/2006/02/hdmanagedPermission"
xmlns:hdmc="http://www.aacsla.com/2006/02/hdmcManifest"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:documentation>Managed Copy Web Service</wsdl:documentation>

  <wsdl:types>
    <xs:schema elementFormDefault="qualified"
targetNamespace="http://www.aacsla.com/2006/02/managedCopyService">
      <xs:import namespace="http://www.aacsla.com/2006/02/hdmanagedPermission"
schemaLocation="aacs_hdmanaged_permission.xsd"/>
      <xs:import namespace="http://www.aacsla.com/2006/02/hdmcManifest"
schemaLocation="aacs_hdmcManifest.xsd"/>
      <xs:element name="RequestOffers">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="1" maxOccurs="1" name="cid" type="xs:string"/>
            <xs:element minOccurs="1" maxOccurs="1" name="mcotList" type="tns:ArrayOfString"/>
            <xs:element minOccurs="0" maxOccurs="1" name="SerialNumber" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
```

```
      <xs:complexType name="ArrayOfString">
        <xs:sequence>
          <xs:element minOccurs="0" maxOccurs="unbounded" name="string" nillable="true"
type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
      <xs:element name="RequestOffersResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="1" ref="hdmc:offers"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="RequestPermission">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="1" maxOccurs="1" name="cid" type="xs:string"/>
            <xs:element minOccurs="1" maxOccurs="1" name="sessionID" type="xs:string"/>
            <xs:element minOccurs="0" maxOccurs="1" name="mcotInfo"/>
            <xs:element minOccurs="1" maxOccurs="1" name="mcmNonce" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="RequestPermissionResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="1" ref='aacsmp:permission' />
          </xs:sequence>
        </xs:complexType>
```

```
    </xs:element>
  </xs:schema>
</wsdl:types>

<wsdl:message name="RequestOffersSoapIn">
  <wsdl:part name="parameters" element="tns:RequestOffers"/>
</wsdl:message>
<wsdl:message name="RequestOffersSoapOut">
  <wsdl:part name="parameters" element="tns:RequestOffersResponse"/>
</wsdl:message>
<wsdl:message name="RequestPermissionSoapIn">
  <wsdl:part name="parameters" element="tns:RequestPermission"/>
</wsdl:message>
<wsdl:message name="RequestPermissionSoapOut">
  <wsdl:part name="parameters" element="tns:RequestPermissionResponse"/>
</wsdl:message>

<wsdl:portType name="ServiceSoap">
  <wsdl:operation name="RequestOffers">
    <wsdl:input message="tns:RequestOffersSoapIn"/>
    <wsdl:output message="tns:RequestOffersSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="RequestPermission">
    <wsdl:input message="tns:RequestPermissionSoapIn"/>
    <wsdl:output message="tns:RequestPermissionSoapOut"/>
  </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="ServiceSoap" type="tns:ServiceSoap">
```

```
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="RequestOffers">
      <soap:operation
soapAction="http://www.aacsla.com/2006/02/managedCopyService/RequestOffers"
style="document"/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="RequestPermission">
      <soap:operation
soapAction="http://www.aacsla.com/2006/02/managedCopyService/RequestPermission"
style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:binding name="ServiceSoap12" type="tns:ServiceSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="RequestOffers">
```

```
      <soap12:operation
soapAction="http://www.aacsla.com/2006/02/managedCopyService/RequestOffers"
style="document"/>
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="RequestPermission">
      <soap12:operation
soapAction="http://www.aacsla.com/2006/02/managedCopyService/RequestPermission"
style="document"/>
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

</wsdl:definitions>
```

This page is intentionally left blank.

D Additional requirement for carriage of SRM

D.1 Introduction

This chapter describes the method to store an SRM (System Renewability Message) on an AACS Disc in the case where an SRM is to be stored on an AACS Disc.

D.2 SRM (System Renewability Message)

D.2.1 SRM for DTCP

The SRM for DTCP shall be stored as the file named “DTCP.SRM” in the root directory of the Data Area.

D.2.2 SRM for HDCP

The SRM for HDCP shall be stored as the file named “HDCP.SRM” in the root directory of the Data Area.

This page is intentionally left blank.

E McmAACS Object for Managed Copy Machine

If a Managed Copy Machine is capable of processing Advanced Applications defined in the *HD DVD-Video Specifications*, it shall support McmAACS Object:

readonly McmAACS mcmAacs;

The object shall have a function property named **completeTransaction** and a property named **offers**. See Chapter 5 in the *AACS Pre-recorded Video Book* for the detailed information:

<table>
<tr><td colspan="3">completeTransaction informs the MCM of completion of a transaction. See 5.3.3 in the AACS Pre-recorded Video Book for this API.</td></tr>
<tr><td rowspan="5">Parameters</td><td>Coupon of type String</td><td>This argument uniquely identifies the financial or account transaction. If no financial or account transaction has been completed, Coupon must be an empty string.</td></tr>
<tr><td>MCOT of type String</td><td>An ID of the managed copy output technology selected for the managed copy, as defined in the AACS Compliance Rules.</td></tr>
<tr><td>MCUi of type String</td><td>An ID which identifies a particular offer that was selected as a part of transaction.</td></tr>
<tr><td>Status of type String</td><td>A string containing further information on the transaction. If the transaction failed, Status may contain information about why that transaction failed.</td></tr>
<tr><td>MCOTParams of type String</td><td>A string value with additional information specific to the managed copy output technology to be used in customization of MCOTInfo to be sent in the RequestPermission message. If this parameter is an empty string, it means that no MCOTParams is passed to the MCM.</td></tr>
<tr><td>Return Value</td><td colspan="2">None.</td></tr>
</table>

Exceptions	If this API fails to inform the MCM of completion of transaction, an exception AACS_E_MCM is thrown. If the exception is not caught, it makes the Advanced Application immediately go into Stop State. The value of the exception AACS_E_MCM is "AACS_E_MCM".

offers of type **String**: This property provides the element <offers> in the XML document which is returned to this MCM by an MCS responding to a RequestOffer message. Note that the property contains the <offers> and </offers> tag. See 5.3.3 in the *AACS Pre-recorded Video Book*. This property shall return an empty string when no <offers> are returned by the MCS.