Advanced Access Content System (AACS)

CBHD Pre-recorded Book

Intel Corporation
International Business Machines Corporation
Microsoft Corporation
Panasonic Corporation
Sony Corporation
Toshiba Corporation
The Walt Disney Company
Warner Bros.

Revision 0.90
Interim Specification
October 7, 2008

Preface

Notice

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.

Table of Contents

List of Figures

List of Tables

This page is intentionally left blank.

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 CBHD Pre-recorded Book*) specifies additional details for using the system to protect audiovisual content distributed on CBHD-ROM medium.

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

In the *CBHD Pre-recorded Book*, the following procedures of Content Encryption and Decryption are described that are required to protect AACS pre-recorded video content. This document is provided as a detailed description of procedures and data structures that are specified for the use of the AACS technology on CBHD-ROM medium.

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 a CBHD-ROM medium.
- Chapter 3 provides a detailed description of the AACS components which reside in the Data Area of a CBHD-ROM medium.
- Chapter 4 provides a detailed description of the protection scheme for the data which specifically belong to the CBHD-Video.
- Chapter 5 describes Managed Copy specifically belong to the CBHD-Video.
- Chapter 6 describes the implementation and realization of the Sequence Key.

1.4 References

- *VIDEO Specifications for China Blue High Definition Disc (CBHD)*
- *DVD Specifications for China High Density Read-Only Disc, Part 1: PHYSICAL SPECIFICATIONS,* Version 10.0.
- *DVD Specifications for China High Density Read-Only Disc, Part 2: FILE SYSTEM Profile for China,* Revision 1.0.
- *DVD Specifications for China High Density Read-Only Disc, Part 3: VIDEO Profile for China,* Revision 1.0.
- *AACS Introduction and Common Cryptographic Elements*.
- *AACS Pre-recorded Video Book.*
- *CBHD Media Mark Specification, for CE System* Revision 0.90.
- *CBHD Media Mark Specification, for PC System* Revision 0.90.

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

Except where specifically noted otherwise, this document uses the same terminology which is defined in the references.

- *CBHD Specifications* means *VIDEO Specifications for China Blue High Definition Disc* and relevant format specification books.
- A CBHD-Video means the CBHD-Video that complies with the definitions in the *CBHD Specifications*.
- A CBHD-Video Content means the CBHD-Video content that complies with the definitions in the *CBHD Specifications*.
- A CBHD-Video ROM medium means a CBHD-Video disc specified in the *CBHD Specifications*.
- A Video Disc means a CBHD-Video ROM medium.
    - Each side of a double sided CBHD-Video ROM medium is considered as one disc.
- A CBHD-Video Content is called AACS-Protected if it is protected by the AACS content protection scheme.
- An AACS-Protected CBHD-Video ROM medium means a CBHD-Video ROM medium which contains an AACS-Protected CBHD-Video content.
- An AACS Disc means an AACS-Protected CBHD-Video ROM medium.
- A Sequence Key Section means a combination of a Cell Block and a corresponding Interleaved Block in a P-EVOB which serves Sequence Key playback.
- Control Data Section means Control data section defined in the *CBHD Specifications*.
- Control Data Zone means Control data zone defined in the *CBHD Specifications*.
- Copyright Data Section means Copyright data section defined in the *CBHD Specifications*.
- Copyright Protection Information means Copyright protection information defined in the *CBHD Specifications*.
- Copyright Protection System Use Section means the Physical Sectors which are reserved for copyright protection system use, located from PSN $1E800_{16}$ to $1F7FF_{16}$, defined in the *CBHD Specifications*.
- Data Segment means Data segment defined in the *CBHD Specifications*.
- Physical Sector means Physical sector defined in the *CBHD Specifications.*
- Lead-in Area means System Lead-in area defined in the *CBHD Specifications*.

1.7 Abbreviations and Acronyms

- AACS Advanced Access Content System
- A_PCK Audio Pack
- ANF Advanced Navigation File
- APS Analog Protection System
- APSTB Analog Protection System Trigger Bits
- APS_VF APS Validity Flag
- BCA Burst Cutting Area
- BIFO Binding Information
- BURS Binding for Usage Rule Set
- CBHD China Blue High Definition Disc
- CC Content Certificate
- CCI Copy Control Information
- CGMS Copy Generation Management System
- CHMI Content Hash Management Information
- CH_PTR Content Hash Pointer
- CHT Content Hash Table
- CPI Content Protection Information
- CRL Content Revocation List
- DKF Directory Key File
- DOT Digital Only Token
- DOT_VF DOT Validity Flag
- EVOB Enhanced Video Object
- EVOBU Enhanced Video Object Unit
- GCI_PKT GCI packet
- HLI_PCK Highlight Information Pack
- ICT Image Constraint Token
- ICT_VF ICT Validity Flag
- ID Identifier

- ILVU Interleaved Unit
- KCD Key Conversion Data
- KEY_VF Key Validity Flag
- KMI Key Management Information
- LSB/lsb Least Significant Bit
- MAC Message Authentication Code
- MCM Managed Copy Machine
- MCMF Managed Copy Manifest File
- MCS Managed Copy Server
- MKB Media Key Block
- MSB/msb Most Significant Bit
- NHA Number of Hashes of ANFs
- NHV Number of Hash Values
- NV_PCK Navigation Pack
- PCCI Primitive CCI
- PCCI_VF PCCI Validity Flag
- P-EVOB Primary Enhanced Video Object
- P-MKB Partial Media Key Block
- PMSN Pre-recorded Media Serial Number
- PRFG Priority Flag
- PSN Physical Sector Number
- REL Right Expression Language
- SEG_KEY_PTR Segment Key Pointer
- SEG_NO Segment Key Number
- SKB Sequence Key Block
- SKBF Sequence Key Block File
- SKF Segment Key File
- SKG Segment Key Group
- SKR Segment Key Range

- SKT Segment Key Table
- SKTS Segment Key Table Set
- SKU Segment Key Unit
- SKUN Segment Key Unit Number
- SP_PCK Sub-picture Pack
- SRM System Renewability Message
- TITLE_KEY_PTR Title Key Pointer
- TKF Title Key File
- TUF Title Usage File
- URMI Usage Rule Management Information
- UR_PTR Usage Rule Pointer
- URS Usage Rule Set
- UR_VF Usage Rule Validity Flag
- V_PCK Video Pack

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 CBHD-Video ROM medium. The CBHD-ROM format is specified by the *CBHD Specifications*.

A reader of this chapter is assumed to be familiar with the CBHD-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 CBHD-Video ROM medium is shown in Figure 2-1.

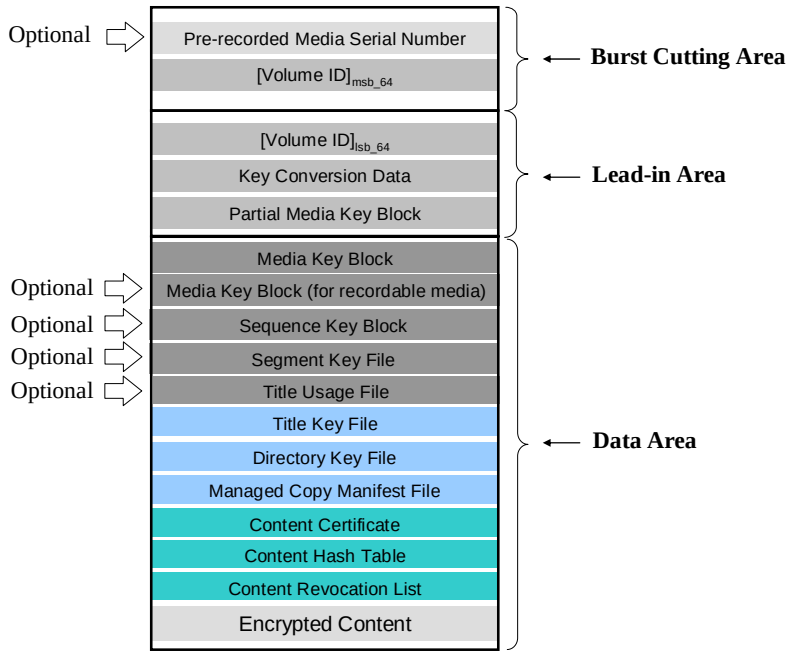

Figure 2-1: Physical Layout of AACS Components on a CBHD-ROM

- 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.

The following subsections describe the details.

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 *CBHD Media Mark Specification, for CE System*. 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 CBHD drive shall 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 CBHD-ROM

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

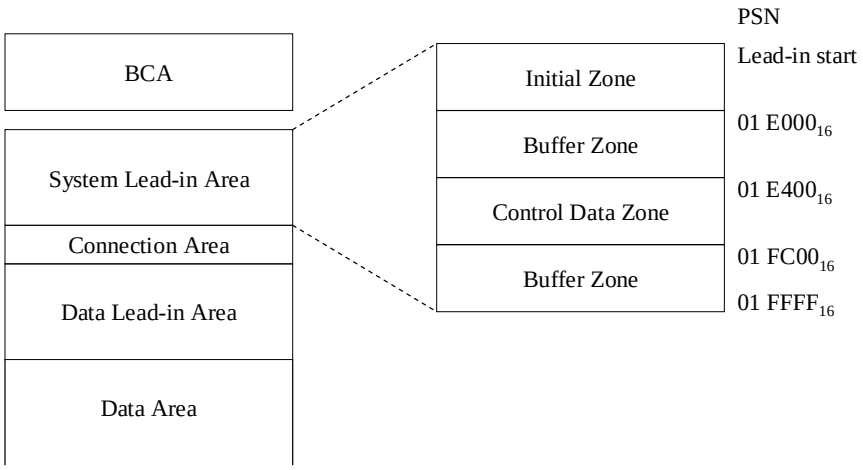

Figure 2-2: Structure of BCA and Lead-in Area of a CBHD ROM Medium

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 00_{16}. A Data Segment in a Copyright Data Section may contain copyright information. Otherwise, the Data Segment shall be filled with 00_{16}.

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. See the *CBHD Media Mark Specification, for CE/PC System* for usage of the field of Reserved for Copyright Protection System Use in the Copyright Protection Information.

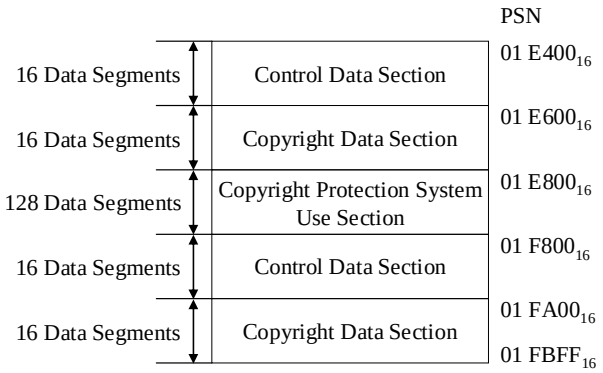

Figure 2-3: Structure of a Control Data Zone

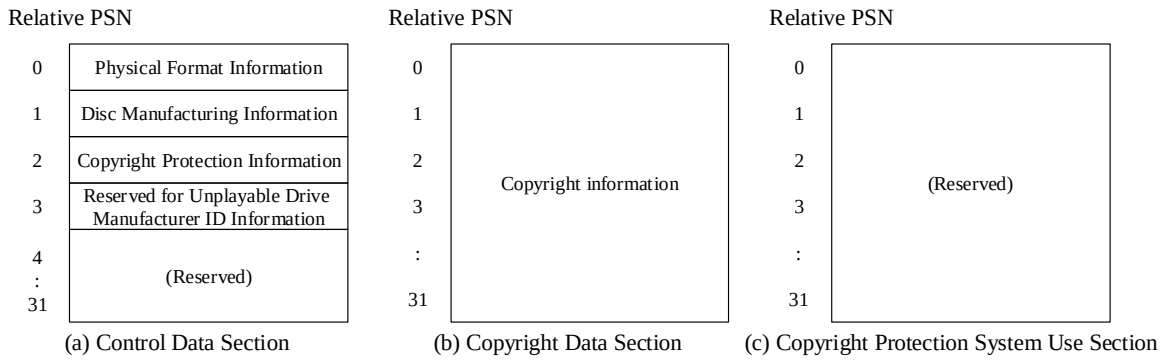

Figure 2-4: Structure of a Data Segment in a Control Data Zone

Table 2-1: Copyright Protection Information in a CBHD-Video ROM medium

Bit / Byte	7	6	5	4	3	2	1	0
0	Copyright protection system type							
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 CBHD-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 leaves some residual bytes. The residual shall be filled with zero.

The size of a P-MKB shall be stored in the 33rd 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.

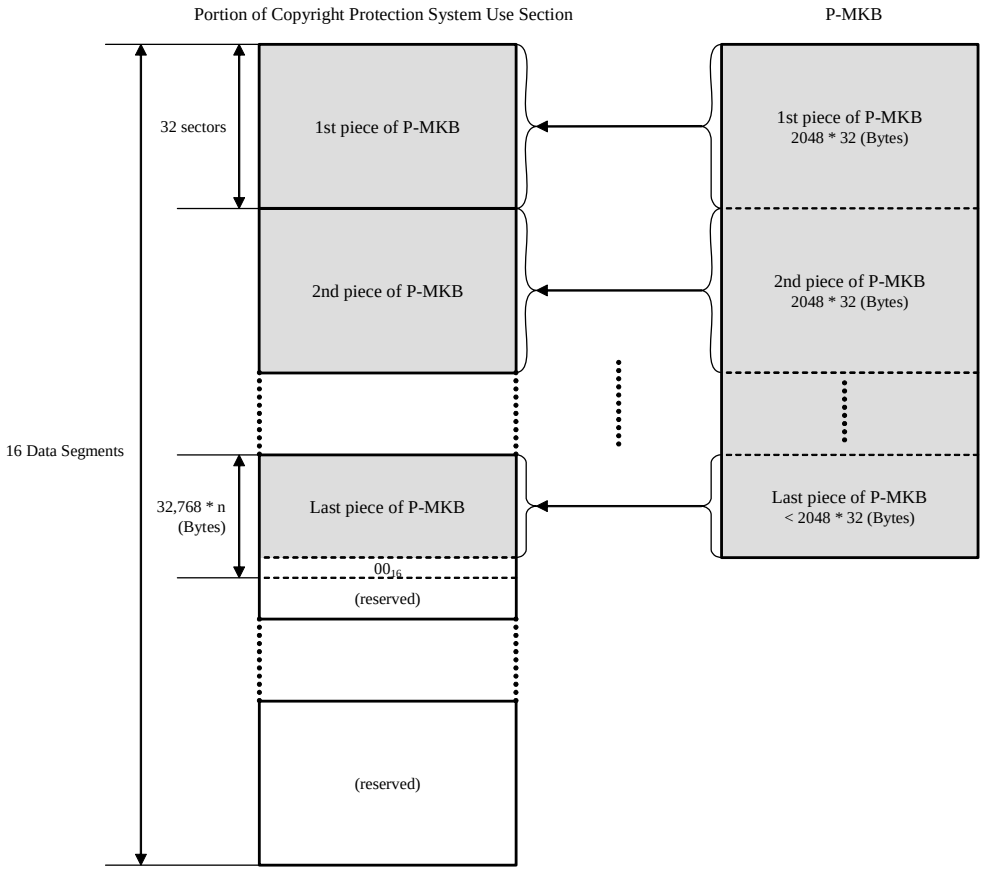

Figure 2-5: Example of storing P-MKB on Lead-in Area of a CBHD-Video ROM medium

2.3.3 Volume Identifier

Each side of an AACS-Protected CBHD-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 CBHD-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 40_{16} for a CBHD-ROM medium.
- Unique Number of 12 bytes. This value shall be assigned by a content participant in order to identify a volume of a CBHD-Video content.

The reserved fields shall be filled with 00_{16}.

In an AACS-Protected CBHD-Video ROM medium, 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 *CBHD Media Mark Specification, for CE/PC System.*

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 a BCA Record 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 BCA Record ID. The latter shows the length in byte of the remaining data in the Record. A Player uses the BCA Record 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 CBHD-Video ROM medium. 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								

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* and additionally defined in this book which are stored in the Data Area of an AACS-Protected CBHD-Video ROM medium. The CBHD-ROM format is specified by the *CBHD Specifications*.

A reader of this chapter is assumed to be familiar with the CBHD-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 are stored in the Data Area with some of them being optional: a Media Key Block (MKB), a Sequence Key Block File (SKBF), a Segment Key File (SKF), a Title Key File (TKF), a Title Usage File (TUF), a Content Certificate, Content Hash Tables (CHTs), a Content Revocation List (CRL), a Directory Key File (DKF), a Managed Copy Manifest File (MCMF) and encrypted contents. CBHD-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.

As defined in the *AACS Introduction and Common Cryptographic Elements* and the *AACS Pre-recorded Video Book*, AACS LA root private/public key, denoted as $AACS\_LA_{priv}$/$AACS\_LA_{pub}$, is used to calculate/verify the signature of each AACS component which is provided by AACS LA. For an AACS Protected CBHD-Video ROM medium, two types of AACS LA root private/public key are used. One is common AACS LA root private/public key and it is used only for CRL. The other is China specific AACS LA root private/public key and it is used for the other AACS components, e.g. MKB including DRL and HRL, SKB, Drive Certificate and Host Certificate. With regard to the Content Certificate, common AACS_CC private/public key, denoted as $AACS\_CC_{priv}$/$AACS\_CC_{pub}$, is used for an AACS Protected CBHD-Video ROM medium.

Some AACS components may be delivered to a title creation process after regions for the components have already been allocated. For convenience of title creation, it is allowed that superfluous data

follow the substantial data of the following AACS components and their backups: “MKBROM.AACS”, “MKBRECORDABLE.AACS”, “SKBF.AACS” and “CONTENT_REVOCATION_LIST.AACS”. For instance, MKBROM.AACS may have a residual at the end of the file. It is not allowed to append such a residual to any other data whose format is described in this book.

There are some reserved fields in data formats defined hereafter. Note that a reserved field shall be filled with 0_2 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.

3.3 Sequence Key Block and Segment Key Files

An AACS-Protected CBHD-Video ROM medium shall have at most 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], are 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 6 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 00_{16}.

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

Table 3-2: Format of Segment Key File

Byte \ Bit	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 00_{16}.

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:
  $[\text{AES-128CBCD}(K_{vvu}, C_{ekd})]_{msb\_64} = 0123456789ABCDEF_{16}$,
  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 is decrypted by the following Segment Key.
  - Segment Key of 16 bytes.

In a Segment Key Unit field, the portion C_{kd} from the 1st byte to the 656th 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*.

A CBHD-Video ROM medium shall have an SKBF and an SKF if it contains a P-EVOB which has Sequence Key Sections (See Chapter 6). Conversely, no SKBF and no SKF shall be present on a CBHD-Video ROM medium which contains no P-EVOB with Sequence Key Sections.

3.4 Title Key File

An AACS Disc shall have one and only one Title Key File (TKF) in which each Title Key data is encrypted by AES-128E with the Volume Unique Key (K_{vu}). Title Key File on an AACS Disc shall reside in the "AACS" directory. The name "VTKF.AACS" is reserved for the TKF. 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.
- PLAYLIST_NAME of 12 bytes is reserved for future use and this field shall be filled with FF_{16}.
- VERN of 4 bytes which indicates the version number of the Title Key File. This value shall be 0 for the current version.
- 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 00_{16}.
    - 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 = 000_2, all bytes of this field shall be filled with FF_{16}.
- All the Reserved fields in a TKF shall be filled with 00_{16}.

- The 16-byte field of TKF MAC. This field stores the CMAC value of the data ranging from the 1st to the 2464th 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:

  0_2: the corresponding Title Key is not available

  1_2: the corresponding Title Key is available

- BIND_TYPE of 3 bits is reserved for future use and this field shall be set to 000_2.

3.5 Title Usage File

This section describes the format of Usage Rule. Usage Rules are stored in a Title Usage File (TUF). TUF is optional, which means that a Disc may not have a TUF on it. A Usage Rule Pointer in the CPI field in a P-EVOB may indicate a Usage Rule Set in the Title Usage File. If a Usage Rule Pointer is valid, at least one TUF shall exist on the Disc. The Usage Rule Pointer in a P-EVOB shall not change within the P-EVOB. P-EVOBs are allowed to share the same Usage Rule Set. Figure 3-1 shows an example of Usage Rule application to EVOBs.

Title Usage File shall reside in the "AACS" directory if it exists on an AACS Disc. The name "VTUF.AACS" is reserved for the TUF. 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.

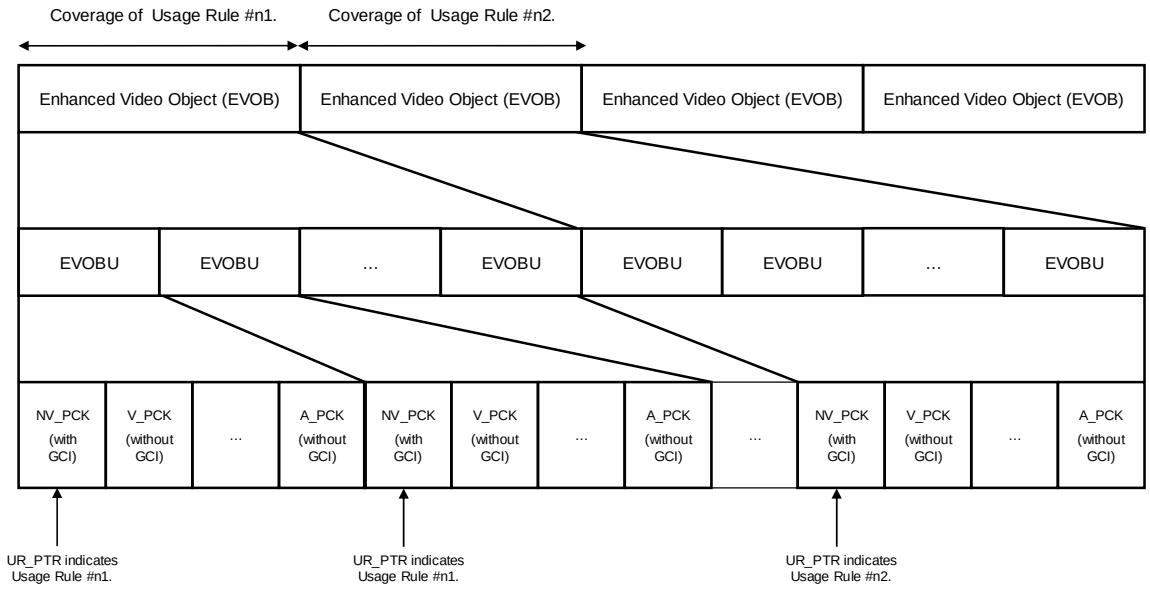

Figure 3-1: An Example of Coverage of Usage Rules

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, excluding the BURS fields and TUF MAC, is contained in Content Hash Table #2.
- VERN of 2 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 is reserved for future use and this field shall be filled with FF_{16}.
- 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 3-8.
- A Binding for Usage Rule Set (BURS) field corresponds to one and only one Usage Rule Set in order. This field is reserved for future use and shall be filled with 00_{16}.
- 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 00_{16}.

Before using a TUF, a Player shall verify the TUF MAC. If the verification fails, the TUF shall not be used. In this case, the Player shall go to Stop State before playback of an EVOB with which the TUF is associated starts.

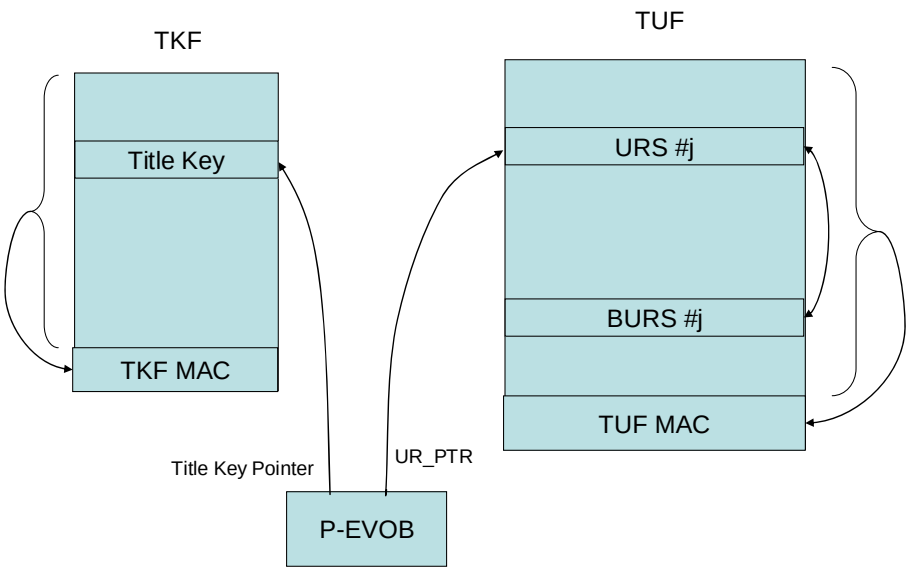

Figure 3-2: The Relationship between a BIFO and Binding MAC field and a Title Key

Table 3-8 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.

Table 3-8: 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 shall be currently set to 0000_{16}.
- 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 00_{16} or 10_{16}. 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 is.

00_{16}: Ignore the Usage Rule and play back the EVOB.

10_{16}: 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 = 00_{16}.

ii) UR_ID is recognizable.

iii) UR_ID is not recognizable & UR_TYPE = 10_{16}.

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. All the recognizable Usage Rules shall be applied to the EVOB, and if two or more Usage Rules of the same priority which contradict each other are to be applied, only the last one survives. 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).

A description by a Right Expression Language (REL) may optionally be included in a TUF as a Usage Rule. UR_ID for REL Usage Rule shall be 000003_{16}. 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 3-9.

Table 3-9: 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:

00_2: REL_STRING is a description of right in a REL.

01_2: REL_STRING is the name of a file which contains a description of right in a REL.

10_2: Reserved.

11_2: 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 0000_{16}.
- REL_STRING of N_S bytes. This field contains a file name if the preceding STRING_TYPE is 01_2. This field contains a right description if the preceding STRING_TYPE is 00_2. For the reserved values of the STRING_TYPE, this field shall not exist.

Usage Rule of Output Control Bits is one of the Usage Rules. UR_ID for Usage Rule of Output Control Bits shall be 000004_{16}. 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 3-10.

Table 3-10: 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.

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 3-11.

Table 3-11: Format of Content Certificate on an AACS Disc for CBHD-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	(msb)	CCSS ID					(lsb)	Sequence Number 1
17	Sequence Number 1	(msb)						
18	Timestamp							
19	(lsb)	Sequence Number 2						
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

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 shall be set to 01_{16} for CBHD-Video ROM medium.
- Layer_Number of 1 byte which shall be 00_{16} for an AACS-Protected CBHD-Video ROM medium.
- Number_of_Digests of 2 bytes which stores 0002_{16}.
- Length_Format_Specific_Section of 2 byte which shall be $000E_{16}$.
- 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-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). The details of CHT #2 are described in Subsection 3.7.2.
- All the reserved fields shall be filled with 0_2.

3.7 Content Hash

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.

3.7.1 Content Hash Table #1

The name “CONTENT_HASH_TABLE1.AACS” is reserved for the CHT #1 for P-EVOBs. CHT #1 has the format shown in Table 3-12. CHT #1 contains the eight-byte hash values of all the EVOBUs in P-EVOBs on an AACS Disc.

Table 3-12: Format of Content Hash Table #1

Byte \ Bit	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.
- Reserved fields shall be filled with 00_{16}.
- A series of 8-byte Hash Values of EVOBUs, each of which stores the hash value calculated from the corresponding EVOBU. 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 is encrypted or not, which means that this field stores the hash value after Pack basis AACS encryption and a Player need not decrypt the EVOBU 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 6) shall also be compiled into CHT #1.

The total number of the EVOBUs on an AACS Disc shall not exceed 500000. If a Player encounters an EVOBU 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 3-13. The CHT #2 on an AACS Disc contains the eight-byte hash value of Directory Key File, the twenty -byte hash value of Managed Copy Manifest and the twenty-byte hash value of the TUF on the Disc.

Table 3-13: Format of Content Hash Table #2 on an AACS Disc

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

8 : 15	Hash of Directory Key File
16 : 35	Hash of MNGCPY_MANIFEST
36 : 55	Hash of VTUF.AACS
56 : 40055	Reserved
40056 : 40059	Number of Hashes of ANFs (NHA)

CHT #2 consists of the following fields:

- The eight-byte hash value of DKF, Directory Key File. See Section 3.9 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 Directory Key, 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 Chapter 5 for 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”, excluding the BURS fields and TUF MAC. If the VTUF.AACS is absent, this field shall be filled with FF_{16}.
- The Number of Hashes of ANFs (NHA) of 4 bytes. This field shall be filled with 00000000_{16}.
- All the Reserved fields shall be filled with FF_{16}.

3.8 Content Revocation List

An AACS 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 Directory Key File

An AACS Disc shall store one and only one Directory Key File (DKF). The DKF shall reside in the “AACS” directory. The name “DKF.AACS” is reserved for the Directory Key File. Directory Key File has the format shown in Table 3-14. Directory Key is defined for future use and the usage is not described in this version of this book.

Table 3-14: 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*. Note that the Volume Unique Key is the one calculated using the Media Key which is derived from the MKB on the Disc. Encrypted Directory Key (K_{DIRe}) field shall be filled with 00_{16} if VERN is identical to 0.
- All the reserved fields shall be filled with 00_{16}.

3.10 Managed Copy Manifest

An AACS Disc shall store one and only one Managed Copy Manifest File (MCMF). The MCMF shall reside in the "AACS" directory. The name "MNGCPY_MANIFEST.XML" is reserved for the MCMF. The detailed format of MCMF is described in Chapter 5.

3.11 Boot Sequence

Firstly, AACS-Compliant Player shall decide the disc to be played back is an AACS Disc or not. If the AACS-Compliant drive for the Player is able to read the Volume ID, the disc shall be treated as an AACS Disc. Otherwise, the disc shall not be treated as AACS Disc and the Player shall not apply the technology of this book to the playback of the disc unless otherwise stated.

Before using each AACS component, associated signature, MAC and Hash value shall be verified. If the verification fails, the system immediately shall go to Stop State. 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/VTKF.AACS so that the AACS module is able to 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 6.)
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/VTUF.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.

3.12 Backups

Most AACS components (except the “MKBRECORDABLE.AACS”) have their 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 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.
- /AACS_BAK/VTUF.AACS (if it 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.

An AACS Disc may have the following backup file for Directory Key File:

- /AACS_BAK/DKF.AACS.

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

Chapter 4
Protection of a CBHD-Video Content on a Medium

4 Protection of a CBHD-Video Content on a Medium

4.1 Introduction

This section describes the details of encryption and decryption of a CBHD-Video Content on a CBHD-Video ROM medium. The CBHD-Video format is specified by the *CBHD Specifications*.

This section defines the protection format of an EVOB which is specified in the CBHD-Video format. 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.

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. See the *CBHD 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 00_{16}.

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:

  00_2: Neither the Segment Key Pointer nor the Title Key Pointer is valid.

  01_2: The Segment Key Pointer is valid.

  10_2: The Title Key Pointer is valid.

  11_2: Reserved.

- The reserved field shall be filled with 0_2.
- 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 0009_{16}, 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 00_2, 01_2 or 11_2, this field shall be filled with 00_{16}.
- 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 6. If KEY_VF is 00_2, 10_2 or 11_2, this field shall be filled with 00_{16}.

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 00000009_{16}, 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 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 1_2. Otherwise, this field shall be equal to 0_2. If URS_NUM in the TUF is zero, UR_VF shall be 0_2.
- 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 $000_2 \,||\, 009_{16}$, 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 0_2, this field shall be filled with 1_2.

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 1_2. Otherwise, this field shall be 0_2. If PCCI_VF is equal to 0_2, the value of PCCI is regarded as 000_2, i.e. Copy Freely, regardless of the value which is assigned to PCCI.
- 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 1_2. Otherwise, this field shall be equal to 0_2. If APS_VF is equal to 0_2, the value of APSTB is regarded as 000_2, i.e. APSTB is OFF, regardless of the value which is assigned to APSTB.
- 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 1_2. Otherwise, this field shall be 0_2. If ICT_VF is equal to 0_2, the value of ICT is regarded as 0_2, regardless of the value which is assigned to ICT.
- 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 1_2. Otherwise, this field shall be equal to 0_2. If DOT_VF is equal to 0_2, the value of DOT is regarded as 0_2, i.e. the analog output of the decoded video is allowed, regardless of the value which is assigned to DOT.
- The reserved fields shall be filled with 0_2.

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 which contains this CCI. When the PCCI_VF is equal to 0_2, the PCCI field shall be 000_2.

  000_2: Copy Freely

  100_2: Copy One Generation

  010_2: No More Copies

  110_2: Copy Never

  011_2: Encryption Plus Non-Assertion (This status means that the encrypted EVOB is allowed to be copied freely without re-distribution.)

  Other Combinations: Reserved.

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

  000_2: APSTB is OFF.

  001_2: Type 1 of APS1 is ON.

  010_2: Type 2 of APS1 is ON.

  011_2: Type 3 of APS1 is ON.

  110_2: APS2 is ON.

  111_2: APS2 is ON.

  Other Combinations: Reserved.

- ICT of 1 bit which indicates the status of the analog output for the EVOBU which contains this CCI. When the ICT_VF is 0_2, the ICT field shall be equal to 0_2.

  0_2: High Definition Analog Output in High Definition Analog Form.

  1_2: High Definition Analog Output in the form of Constrained Image.

- DOT of 1 bit: This flag indicates the Analog Output status. If this flag is 1_2, an analog output of the decoded video of the EVOB is not allowed.

- The reserved field shall be filled with 0_2.

Note that CCI does not change in one EVOB. Each field in CCI shall be identical throughout one EVOB.

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. An EVOB shall be protected by Content Hash on an EVOBU basis. If the Player encounters an unprotected EVOB on an AACS Disc, the system immediately shall go to Stop State. 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 contains an encrypted Pack, the EVOBU shall have a valid TITLE_KEY_PTR in the CPI field. Otherwise, the EVOBU shall not have a valid TITLE_KEY_PTR and the KEY_VF shall be 0_2. A Title Key shall not be changed within one P-EVOB. A Title Key may be shared by plural EVOBs. Figure 4-1 shows an example of Title Key assignment to EVOBs.

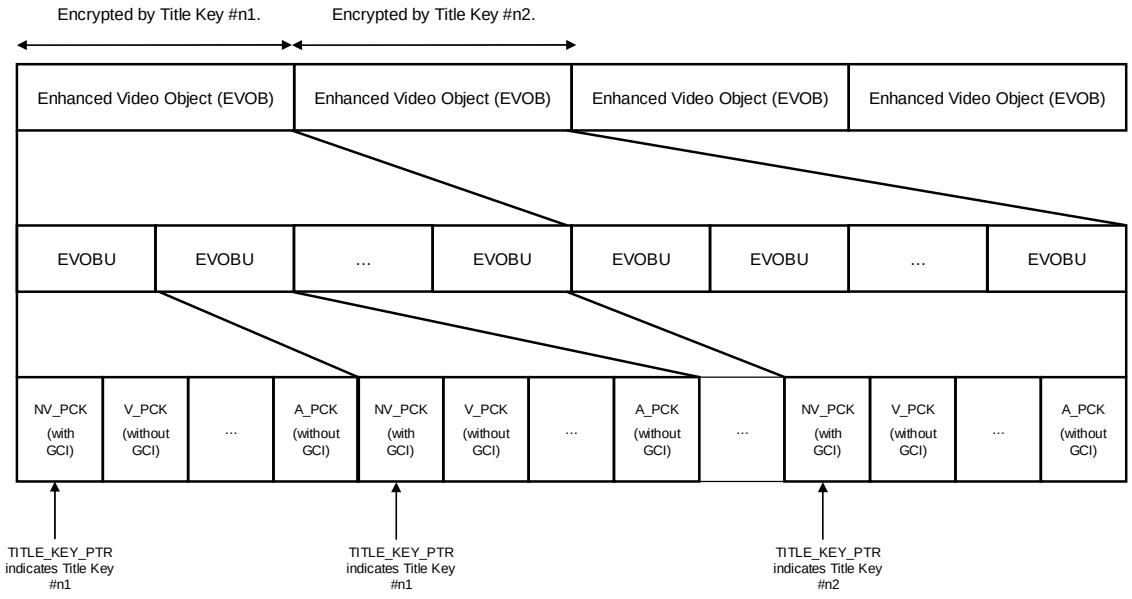

Figure 4-1: An Example of Title Key assignment to EVOBs

4.3.1 Pack Types

The following Packs are allowed to be encrypted and the other Packs are not allowed to be encrypted:

- Video Pack
- Audio Pack

- Sub-picture Pack
- Highlight Information Pack
- Real-time Text Pack

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:

$K_c = AES\text{-}G\ (K_t, D_{tk} \parallel CPI_{lsb\_96})$,

where AES-G denotes the one-way function based on the AES algorithm defined in the *AACS Introduction and Common Cryptographic Elements*. Note that Title Key Data (D_{tk}) is just data between the 85^{th} byte and the 88^{th} byte of Pack, inclusive.

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 01_2. Otherwise, the PES_scrambling_control shall be 00_2.

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 *CBHD Specifications*							

	20		PES_scrambling _control	
	21 : 83	Data defined in the *CBHD Specifications*		
	84 : 87	Title Key Data (D_{tk})		
	88 : 127	Data defined in the *CBHD Specifications*		
Encrypted Portion (1920 bytes)	**128 : 2047**	Encrypted Content		

Table 4-8 shows the Pack encryption format for Highlight Information Pack. For each Highlight Information 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. Note that every Highlight Information Pack on an AACS Disc is encrypted as long as KEY_VF is 01_2 or 10_2, that is, if the Title Key or the Segment Key is valid.

Each Highlight Information 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:

$K_c = AES\text{-}G\ (K_t, D_{tk} \parallel CPI_{lsb\_96})$,

where AES-G denotes the one-way function based on the AES algorithm defined in the *AACS Introduction*

and Common Cryptographic Elements. Note that Title Key Data (D_{tk}) is just data between the 85th byte and the 88th byte of Pack, inclusive.

The Encrypted Portion is encrypted as follows:

C_e = AES-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 Highlight Information Pack

	Bit / Byte	7	6	5	4	3	2	1	0
Unencrypted Portion (128 bytes)	**0 : 83**	Data defined in the *CBHD Specifications*							
	84 : 87	Title Key Data (D_{tk})							
	88 : 127	Data defined in the *CBHD Specifications*							
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 within 300 seconds after playback starts. Unless all the 7 hash values are respectively equal to the corresponding hash values stored in CHT #1, playback shall be stopped.

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-EVOB, the Player chooses one EVOBU 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 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.

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 as long as UR_PTR in the EVOB is valid. 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. If the KEY_VF is 00_2, decryption is unnecessary for the current EVOBU. If the KEY_VF is 10_2, 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 01_2 or if the current Pack is an Highlight Information Pack, 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 00_2 and if the current Pack is not an Highlight Information Pack, 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*.

4.3.5 Verification of Hash

As mentioned above, there are two Content Hash Tables recorded on an AACS Disc. Before verification of the hashes, the Content Certificate file shall be at first verified. 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.

Chapter 5

Managed Copy

5 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>
</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 contains at most one <serverList>.
serverUri	This indicates a Managed Copy Server. Multiple URIs may be recorded in a serverList. A Managed Copy Machine shall attempt to connect to one of the

available URIs. If the URI is not specified or the servers are not available, the URI held in the MCM shall be used. The method of choice of Managed Copy Server is proprietary to each Managed Copy Machine.

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.

The following is an example of DealManifest, which is sent back from the Managed Copy Server as part of Permission Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<dealManifest xmlns=”http://www.aacsla.com/2006/02/hdmcDealManifest”>
  <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"/>
  <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"/>
</dealManifest>
```

The semantics is as follows:

dealManifest — A collection of resources on an AACS Disc. This element shall have an <MCUi> element. 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 shall copy, provided that the copy operation is allowed by a Managed Copy Server, all the items listed in an <dealManifest>. One <dealManifest> has 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 B for the schema for DealManifest which is to be included in Managed Copy Permission. The schema for Managed Copy Permission is shown in Appendix C. For Managed CopyWeb Service Description, see Appendix D.

Note that APIs, IsMCMSupported and InvokeMCM, which are defined in *AACS Pre-recorded Video Book* is not supported in this book.

This page is intentionally left blank.

Chapter 6
Sequence Key

6 Sequence Key

6.1 Introduction

This section describes how a Player behaves for a VTS with Sequence Key. If an AACS Disc has an 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 *shall 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 6.2 describes the mechanism and the Player behavior for the Sequence Key. A Sequence Key Section continues in a time span. The time span has a minimum length which is defined in the *CBHD 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.

Playback of a Sequence Key Section shall be seamless as long as the conditions for seamless playback described in the *CBHD Specifications*. Furthermore, a transition from an SKR to another SKR *shall 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 *CBHD Specifications*.

6.2 Sequence Key for 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 shall be equal to 11_2. See the conceptual image of a Sequence Key Section in Figure 6-1. 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 *CBHD Specifications*. The meaning of the KEY_VF values is as follows:

- 00_2: SEG_KEY_PTR is not valid, 01_2: 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 *CBHD Specifications*. The SEG_KEY_PTR indicates an entry of the SKT. The value of the SEG_KEY_PTR runs from 1 to 192. See Table 6-1 for the bit assignment for the reserved field in C_PBI.

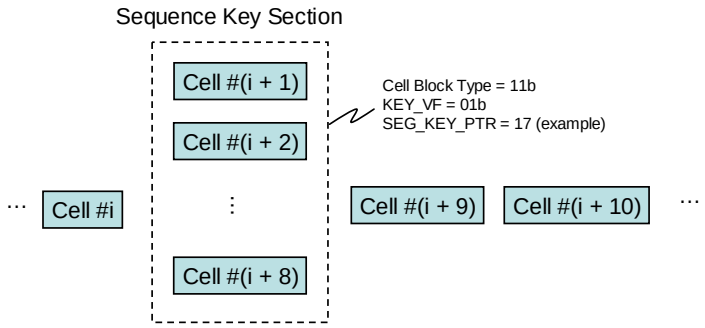

Figure 6-1: An Image of Sequence Key Section

Table 6-1: Bit Assignment for the Reserved Field of C_PBI

Bit / Byte	7	6	5	4	3	2	1	0

0	KEY_VF	SEG_KEY_PTR
1	SEG_KEY_PTR	Reserved

6.2.1 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 is 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.

6.2.2 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 *CBHD Specifications*. The value of KEY_VF of an ILVU in an Interleaved Block corresponding to a Sequence Key Section shall be 01_2. 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 *CBHD Specifications*. If the value of KEY_VF equals 01_2, however, the SML_AGLI field is reserved for SML_SEQI. Table 6-2 and Table 6-3 together describe the format of SML_SEQI. In addition, SML_SEQ_Cn_DSTA = $7FFFFFFFFFFF_{16}$ for n not equal to SEG_NO*, which means those fields are all invalid. Therefore, it always holds that SML_SEQ_C9_DSTA = $7FFFFFFFFFFF_{16}$. 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 6-2: 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 6-3: 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]							

6.2.3 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. $7FFFFFFFFFF_{16}$. 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 the *CBHD 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.

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"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.aacsla.com/2006/02/hdmcManifest" 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"/>
</xs:sequence>
<xs:attribute name="Id" type="xs:string" use="optional"/>
        <!-- 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:schema>
```

B Schema for DealManifest for CBHD

Suppose that this schema is stored in the file named “hdmc_deal_manifest.xsd”:

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

<!-- dealManifest -->
<xs:element name="dealManifest">
<xs:complexType>
<xs:sequence>
        <xs:element ref="hddm:MCUi"/>
        <xs:element ref="hddm:item" maxOccurs="65536"/>
</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>

<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>
```

C XML Schema for Managed Permission

```
<?xml version="1.0"?>
<xs:schema targetNamespace=”http://www.aacsla.com/2006/02/hdmcManagedPermission”
xmlns:hdmp="http://www.aacsla.com/2006/02/hdmcManagedPermission"
xmlns:hddm="http://www.aacsla.com/2006/02/hdmcDealManifest"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xs:import namespace=”http://www.aacsla.com/2006/02/hdmcDealManifest”
schemaLocation="hdmc_deal_manifest.xsd"/>
<xs:element name="permission">
<xs:complexType>
<xs:sequence>
        <xs:element name="permissionSignedContent">
        <xs:complexType>
        <xs:sequence>
                <xs:element ref="hdmp:status" minOccurs="1" maxOccurs="1"/>
                <xs:element ref="hdmp:MCS_MCOTInfo" minOccurs="0" maxOccurs="1"/>
                <xs:element ref="hddm:dealManifest" minOccurs="0" maxOccurs="1"/>
                <xs:element ref="hdmp:mcmNonce" minOccurs="1" maxOccurs="1"/>
                <xs:element ref="hdmp:MCUi" minOccurs="1" maxOccurs="1"/>
                <xs:element ref="hdmp:sessionId" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        </xs:complexType>
        </xs:element>
        <xs:element ref="hdmp:signature" minOccurs="1" maxOccurs="1"/>
</xs:sequence>
</xs:complexType>
```

```
</xs:element>
<xs:element name="status" type="xs:boolean"/>
<xs:element name="signature" type="xs:base64Binary"/>
<xs:element name="MCS_MCOTInfo" type="xs:base64Binary"/>
<xs:element name="mcmNonce" type="xs:string"/>
<xs:element name="MCUi" type="xs:string"/>
<xs:element name="sessionId" type="xs:string"/>
</xs:schema>
```

D Managed Copy Web Service Description

```
<?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:hdmcs="http://www.aacsla.com/2006/02/managedCopyService"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:hdmp="http://www.aacsla.com/2006/02/hdmcManagedPermission"
xmlns:aacsoffer="http://www.aacsla.com/2006/02/managedOffer"
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/hdmcManagedPermission"
schemaLocation="hdmc_managed_permission.xsd"/>
<xs:import namespace="http://www.aacsla.com/2006/02/managedOffer"
schemaLocation="aacs_copy_offer.xsd"/>

<xs:element name="RequestOffers">
<xs:complexType>
<xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1" name="cid" type="xs:string"/>
        <xs:element minOccurs="0"
```

```
                maxOccurs="8192" name="mcotList" type="hdmcs:ArrayOfMCOTs"/>
        <xs:element minOccurs="0"
                maxOccurs="unbounded" name="SerialNumber" type="xs:string"/>
        <xs:element minOccurs="1"
                maxOccurs="1" name="mcmNonce" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:complexType name="ArrayOfMCOTs">
<xs:sequence>
        <xs:element minOccurs="1"
                maxOccurs="1" name="mcotID" nillable="true" type="xs:string"/>
        <xs:element minOccurs="0"
                maxOccurs="8192" name="mcotMinorIDList" type="hdmcs:ArrayOfString"/>
</xs:sequence>
</xs:complexType>

<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="aacsoffer:offers"/>
```

```
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="RequestPermission">
<xs:complexType>
<xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1" name="MCUi" type="xs:string"/>
        <xs:element minOccurs="1" maxOccurs="1" name="sessionID" type="xs:string"/>
        <xs:element minOccurs="0"
                maxOccurs="1" name="MCM_MCOTInfo" type="xs:base64Binary"/>
        <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='hdmp:permission'/>
</xs:sequence>
</xs:complexType>
</xs:element>

</xs:schema>

</wsdl:types>

<wsdl:message name="RequestOffersSoapIn">
```

```
        <wsdl:part name="parameters" element="hdmcs:RequestOffers"/>
</wsdl:message>

<wsdl:message name="RequestOffersSoapOut">
        <wsdl:part name="parameters" element="hdmcs:RequestOffersResponse"/>
</wsdl:message>

<wsdl:message name="RequestPermissionSoapIn">
        <wsdl:part name="parameters" element="hdmcs:RequestPermission"/>
</wsdl:message>

<wsdl:message name="RequestPermissionSoapOut">
        <wsdl:part name="parameters" element="hdmcs:RequestPermissionResponse"/>
</wsdl:message>

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

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

E Additional requirement for carriage of SRM

E.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.

E.2 SRM (System Renewability Message)

E.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.

E.2.2 SRM for DTCP

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.