Advanced Access Content System (AACS)

Pre-recorded Video 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 0.91
February 17, 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.

Intellectual Property

Implementation of this specification requires a license from AACS LA LLC.

Contact Information

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

- 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 LLC web site is http://www.aacsla.com.

This page is intentionally left blank.

Table of Contents

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

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. This document (the *Pre-recorded Video Book*) specifies additional details for using the system to protect audiovisual content distributed on pre-recorded (read-only) storage media. Specifications covering other storage types, transmission media and formats are expected to be available in the future (see Section 1.6 below).

The use of this specification and access to the intellectual property and cryptographic materials required to implement it will be the subject 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 addition to the general objectives described in the *Introduction and Common Cryptographic Elements* book of this specification, the use of AACS for protecting pre-recorded video content was designed to meet the following specific criteria:

- Provide robust protection for both off-line playback and optional enhanced uses enabled via on-line connection.
- Provide for extended and extensible usage (e.g. jukebox storage, pay for copy).
- Independent of physical storage format to the degree possible.
- Compliant players can authenticate that content came from an authorized, licensed replicator.

Figure 1-1 presents an informative overview of the system, as used for protecting pre-recorded high-definition video content. Actual details and requirements of system operation are described in subsequent chapters.

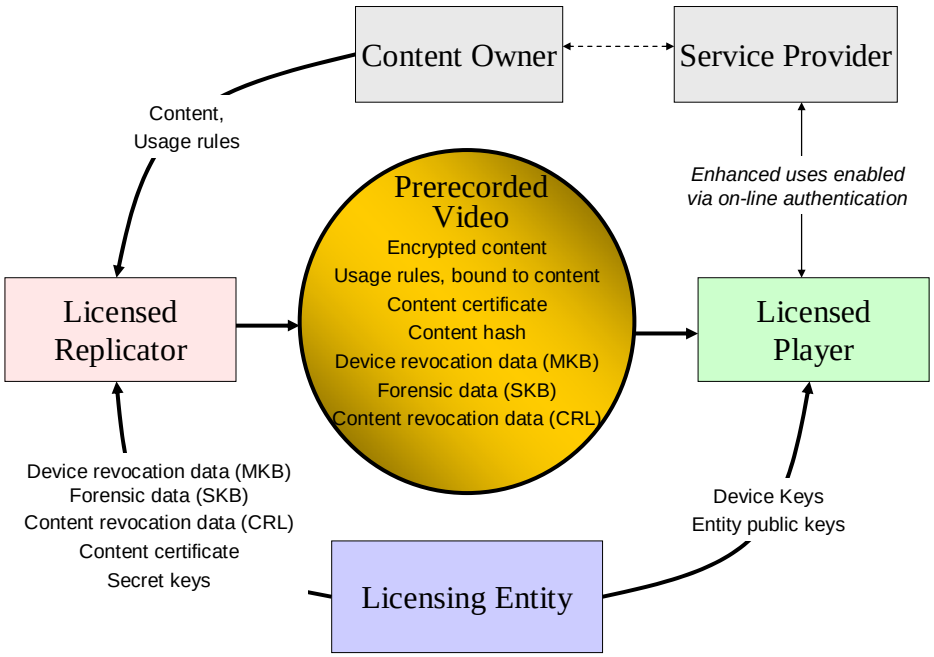

Figure 1-1 – System Overview (Informative)

The owner of audiovisual content that is to be protected provides that content and an associated set of usage rules to a licensed replicator.

The licensing entity provides to the licensed replicator device revocation data in the form of a Media Key Block (MKB). As described in the *Introduction and Common Cryptographic Elements* book of this specification, the MKB will enable players, each using its set of device keys, to calculate the Media Key. If a set of device keys is compromised in a way that threatens the integrity of the system, an updated MKB can be provided by the licensing entity that will cause a device with the compromised set of device keys to calculate a different key than is computed by devices containing unrevoked device keys. In this way, the compromised device keys are "revoked" by the new MKB.

The licensing entity also provides to the licensed replicator content revocation data in the form of a Content Revocation List (CRL), and a Content Certificate, both cryptographically signed by the AACS LA. The Certificate identifies the content and includes a cryptographic hash thereof (based on a series of hashes), which the licensing entity receives from the replicator prior to signing the Certificate. The CRL identifies content that has been signed and contains a valid certificate but has since been revoked and therefore should not be accessed by a compliant player.

The licensing entity also provides to the licensed replicator device variation data in the form of a Sequence Key Block (SKB), and secret keys, called Media Key Variants, based on both the device variation data and Media Key.

The licensed replicator encrypts the content to be protected, using one or more secret keys of its own choosing, called Title Keys. The replicator encrypts the Title Keys in one or more keys derived from the Media Key. For usage rules that can be modified by a compliant recorder, the replicator denotes one of the Title Keys as the primary Title Key and cryptographically binds that Title Key to those usage rules to prevent any malicious alteration of those usage rules. The encrypted content, usage rules, MKB, SKB, CRL and Content Certificate are all pre-recorded onto the storage medium.

The AACS LA provides secret device keys and sequence keys to licensed manufacturers for inclusion in compliant playback devices/applications (device key sets and sequence key sets are typically unique per device/application). When a storage medium with protected content is placed in a compliant player, the player uses its Device Keys to process the MKB and calculate the corresponding Media Key. Using the Media Key

and its Sequence Keys to process the SKB, the device will calculate a particular Media Key Variant. Assuming the given set of Device Keys has not been revoked, the calculated key will be one of the keys used by the licensed replicator. The player uses an inverse procedure to that used by the licensed replicator to derive a Title Key. The player also uses the Title Key to provide access to the protected content in a compliant manner.

The AACS LA also provides its entity public keys to licensed manufacturers for inclusion in compliant devices/applications. Prior to accessing content that has been signed as described in this book, a compliant player reads the Content Certificate and CRL from the pre-recorded medium, and uses the entity public keys to verify their integrity. If either verification fails, access is aborted.

The compliant player keeps the CRL in non-volatile storage, unless it already has a more up-to-date list. Using the most up-to-date CRL, the player checks to see if the content is revoked, and if it is, access is aborted. During playback, the compliant player calculates a series of content hashes using the same method used by the replicator. If the player's calculated hash values differ at any point from the replicator-stored values, access is aborted.

The system enables certain enhanced uses of pre-recorded video content, at the election of the content owner, through the use of a robust on-line connection. For example, a home video server might connect with a service provider to obtain authorization to make a protected local copy of a given pre-recorded Title for "jukebox" purposes. Such authorization might be provided free-of-charge to the owner of the optical media, with any additional authorized copies incurring a charge. Thus, this and other enhanced uses may entail business interaction between content owners and service providers, as indicated by the dashed line in the figure above.

1.3 Organization of this Document

This document is organized as follows:

- Chapter 1 provides an introduction and overview.
- Chapter 2 describes procedures related to the authentication and revocation of pre-recorded content.
- Chapter 3 describes procedures for the production (encryption) and off-line playback (decryption) of protected pre-recorded content.
- Chapter 4 describes the Sequence Key Block.
- Chapter 5 describes how devices can perform a Managed Copy of content protected by AACS.

1.4 References

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

- AACS LA, *License agreement*
- *Introduction and Common Cryptographic Elements* book.
- *Specification of Exclusive XML Canonicalization*, http://www.w3.org/TR/xml-exc-c14n/#sec-Specification.

1.5 Document History

This document version 0.91 supersedes version 0.90 dated April 13, 2005. It contains editorial improvements since the 0.90 version, plus the following changes:

- Modifications to the Content Certificate
- Introduction of a second AACS LA Key Pair for the purpose of signing Content Certificates
- Clarified scope of how content signing technology can be used on encrypted and in the clear content; and requirements on devices for verifying that signature
- Addition to CRL of a revocation record type for Managed Copy Server Certificates
- Binding method for usage rules has been changed from a MAC based binding to use cryptographic signatures
- Addition of Chapter 5 to describe Managed Copy

1.6 Future Directions

With its advanced, robust cryptography, key management and renewal mechanisms, it is expected that this technology will develop and expand, through additions to this specification, to address content protection for additional storage types, application formats and usage models, as authorized by AACS LA.

1.7 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.

1.8 Terminology

Except where specifically noted otherwise, this document uses the same terminology as described in the *Introduction and Common Cryptographic Elements* book of this specification.

1.9 Abbreviations and Acronyms

Except where specifically noted otherwise, this document uses the same abbreviations and Acronyms as described in the *Introduction and Common Cryptographic Elements* book of this specification.

Chapter 2
Content Revocation

2 Introduction

This chapter describes a robust mechanism whereby content on individual media can be revoked to prevent playback of unauthorized content. This is accomplished by applying cryptographic signatures to authorized content and storing those signatures on the media with the content. The signature is validated before allowing playback. A Content Revocation List (CRL) is also embedded onto media and then stored in non-volatile memory by players and contains a list of content that contains a valid signature but has since been revoked. The license agreement describes the conditions under which such revocation can occur. Figure 2-1 presents an overview of the approach, and details are provided in subsequent sections.

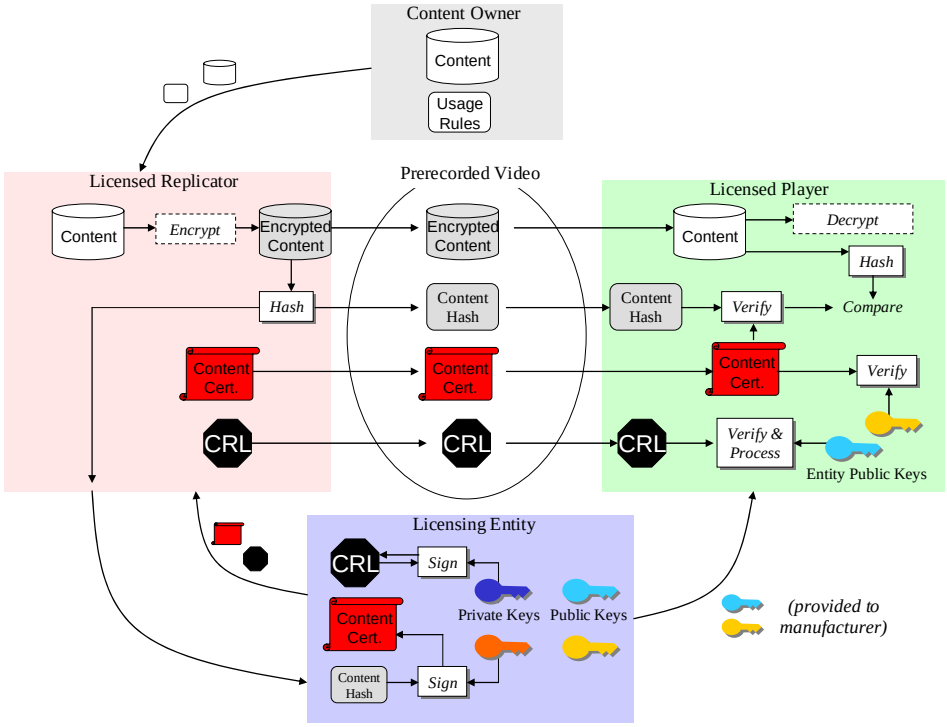

Figure 2-1 – Content Validation and Revocation Overview

2.1 Scope

The content verification and revocation scheme described in this chapter can be applied to both AACS Encrypted and Unencrypted data. In this chapter, AACS Encrypted shall mean encrypted as described in this specification and as applied to content formatted as defined in the format specific books of this specification. Unencrypted shall mean content which could have been AACS Encrypted but was not, and is completely unprotected, i.e., in the clear. Any content to which the AACS content verification rules can apply, whether AACS Encrypted or Unencrypted, is referred to in this chapter as Certifiable Content. Content that conforms to this scheme is referred to as Certified. The format specific books of this specification shall define requirements, if any, for verifying the content signature that are not provided in this specification book.

- Certifiable Content that is AACS Encrypted must conform to this scheme.
- Certifiable Content that is Unencrypted but on the same media as AACS Encrypted content must also conform to this scheme.
- Certifiable Content that is completely Unencrypted is not required under this license to conform to this scheme.

AACS licensed players must verify that the signature is correct as defined in Section 2.6 before providing access to all content (AACS Encrypted or Unencrypted) which has been signed as described in this chapter, including signed content that is completely Unencrypted. It is anticipated that format specific books will be added to this specification which will define how to apply this scheme to other pre-recorded video formats which do not utilize AACS encryption as it is defined in this specification.

2.2 Content Signing infrastructure

Licensed Players will contain the Entity Public Keys that will be used as the root of trust for validating content signatures. Media containing content signed in accordance with this scheme will contain the following items:

- Content Certificate
- Content Hash Table
- Content Revocation List

The Content Certificate and Content Hash Table are together used to validate the authenticity of the content and prevent playback of that content if the signature is not valid. The Content Revocation List is a mechanism to prevent playback of content that contains a valid signature but is not valid content.

The Content Certificate, Content Hash Table, and Content Revocation List must be stored on the pre-recorded media with the signed content. The details and file formats for storing this information is contained in the format specific books of this specification.

2.3 Content Hash Table

Replicators shall include with each *Certified Content* that they produce a Content Hash Table (CHT). The CHT consists of a series of cryptographic hash values calculated by the replicator, which cover the complete content, including any unencrypted content included on the same media with encrypted content. For storage media that contain more than one physical layer, the storage medium may contain a separate Content Hash Table for each layer. Details of CHT format, calculation and storage are specific to each supported format, and are provided in the Format-specific books of this specification.

The inclusion of a Content Hash Table, together with the use of its digest in signing the Content Certificate, enables licensed products to verify a correspondence between the Content Certificate and the pre-recorded content with which it is used. If this verification fails, access to such content shall be aborted. When providing access to Certified Content, the licensed product shall calculate the series of content hash values using the same algorithm used by the replicator, and shall abort such access if at any time a calculated hash value does not match the corresponding hash value provided by the replicator in the CHT. Unless the licensed product has a robust means of detecting a change of the pre-recorded storage medium, it must re-verify that the hash of the CHT is the same as the Content Hash Table Digest contained in the Content Certificate whenever a CHT is re-read from the medium.

2.4 Content Certificate

Licensed replicators shall include with any Certified Content that they produce, a signed Content Certificate covering that content. The Content Certificate is stored as unencrypted data on the same storage medium as the Certified Content, as described for each supported content format elsewhere in this specification. For storage media that contain more than one physical layer, the storage medium may contain a separate Content Certificate for each layer. Where a storage medium contains Certified Content in more than one content format, a Content Certificate shall be included for each format. Table 2-1 shows the format of the Content Certificate.

Table 2-1 – Content Certificate for Pre-recorded 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	Number_of_HashUnits							
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 … 26+L-1	Format_Specific_Section Reserved for definition and possible extension in Adaptation books							
26+L : 33+L	Content Hash Table Digest #1							
…	…							
26+L+(N-1)*8 … 33+L+(N-1)*8	Content Hash Table Digest #N							
34+L+(N-1)*8 : 73+L+(N-1)*8	Signature Data							

Note: L is the length determined by Lengh_Format_Specific_Section. See the description below.

Each Content Certificate includes:

- A 1-byte Certificate Type value, where 00_{16} shall be used to indicate a first-generation AACS Content Certificate
- A 4-byte Total_Number_of_HashUnits field indicates the total number of Hash Unites on the optical media.
- A 1-byte Total_Number_of_Layers field indicates the total number of layers on the optical media.
- A 1-byte Layer_Number field indicates the layer of the optical media for which this Content Certificate is created. NOTE: This field may not be used by every Format. Additional details on the meaning and use of this field may be provided in the Format specific books of this specification.
- A 4-byte Number_of_HashUnits field indicates the number of Hash Units on the layer for which this Content Certificate is created. NOTE: This field may not be used by every Format. Additional details on the meaning and use of this field may be provided in the Format specific books of this specification.
- A 2-byte Number_of_Digests field indicates the number of Content Hash Table Digests contained within the Content Certificate. NOTE: Additional details on the meaning and use of this field may be provided in the Format specific books of this specification.
- A 2-byte Applicant ID, assigned by the AACS LA. Each adopter that will be submitting requests to AACS LA to create Content Certificates will be assigned a unique Applicant ID.
- A 4-byte Content Sequence Number assigned by the AACS LA to uniquely identify the Certified Content amongst that applicant's content. The combination of the Applicant ID and the Content Sequence Number is referred to as the *Content Certificate ID.* In other words, the Content Certificate ID is a 6-byte number.
- A 2-byte Minimum CRL Version value, assigned by the AACS LA to indicate the minimum Content Revocation List Version number that must accompany the Certified Content.
- A 2-byte Length_Format_Specific_Section that specifies the length of the subsequent Format_Specific_Section section. If this value is zero, then the length of Format_Specific_Section is 2 to maintain byte alignment and the 2 bytes shall be treated as Reserved.
- A Format_Specific_Section. The length of this section and the fields contained within it are defined separately in the adaptation books for each Format utilizing AACS. The combination of the Length_Format_Specific_Section field and the Format_Specific_Section field shall always be a multiple of 4 bytes.
- A series of 8-byte Content Hash Table Digests, containing the digests of the Content Hash Tables. The digest consists of the least significant 64 bits of the resulting digest from SHA-1 as described in the *Introduction and Common Cryptographic Elements* book of this specification.
- A 40 byte Signature Data, calculated using the Entity Private Key, over the entire data up to and including Content_Hash_Table_Digest#N .

The licensed replicator shall obtain a signed Content Certificate from AACS LA for the content to be stored on the medium, and where appropriate with a separate Content Certificate for each layer. The resulting Content Certificate(s) and Content Hash Table(s) will be included with the content on the pre-recorded medium. Section 2.5 contains additional details on what content must be included in the creation of the Content Certificate. The AACS LA will provide to the licensed replicator a current version of the Content Revocation List which will also be included on the pre-recorded medium. The creation of the Content Certificate is performed by the licensed replicator. The signing of the certificate is performed by a secure facility operated by AACS LA. The licensed replicator will submit the certificate to the secure facility and receive the signed certificate back from that facility.

The AACS LA provides its Entity Public Keys (which corresponds to the Entity Private Keys) to each licensed manufacturer for inclusion in each licensed device or application produced. Licensed products shall treat the Entity Public Keys as Integrity Required, as defined in the license agreement. Prior to providing access to Certified Content, licensed products shall verify the signature of the Content Certificate. If at any point in the process the verification fails, such access shall be aborted. Unless the licensed product has a robust means of detecting change of the pre-recorded storage medium, it must, whenever a Content Certificate is re-read from the medium, either re-verify the signature of the Certificate or robustly verify that the Certificate is the same as

one whose signature it already verified, before using the Content Hash Table Digest contained therein for comparisons with the CHT.

Prior to providing access to Certified Content, licensed products shall also read and process a Content Revocation List having a List Version value equal to or greater than the Minimum CRL Version value, as described below.

2.5 Creating Content Certificate

A licensed replicator shall create the content certificate by the following procedure.

1. The digests of the individual units of content are computed as follows:

   $C_d = [\text{SHA-1}(\text{Hash\_Unit})]_{lsb\_64}$

Where Hash_Unit is defined in the format specific Pre-recorded books of this specification and SHA-1 is the SHA hashing function as defined in *Introduction and Common Cryptographic Elements* book.

All Hash_Units on the media must be included in this computation. In the case where some of the Hash_Units on the media are encrypted and others are not encrypted, the digest of the unencrypted Hash_Units must also be included in the CHT. For Hash_Units that are encrypted, C_d shall be calculated after encryption of those Hash_Units.

2. Each instance of C_d is stored in one or more Content Hash Tables (CHT) as defined in the format specific Pre-recorded books of this specification.
3. The digest of each Content Hash Table is computed as follows:

   $CHT_d = [\text{SHA-1}(\text{CHT})]_{lsb\_64}$

4. Each instance of CHT_d is stored in a Content Certificate (CC) and the Content Certificate is cryptographically signed as follows:

   $CC_{sig} = \text{AACS\_Sign}(\text{AACS\_CC}_{priv}, \text{CC})$

With AACS_Sign as defined in *Introduction and Common Cryptographic Elements* book and the format and layout of the Content Certificate as defined in Section 2.4. The private key, denoted AACS_CC_{priv}, is an additional private key of the AACS LA that is used only for the purpose of signing the Content Certificate.

This step will be performed at a secure facility operated by the AACS LA where AACS_CC_{priv} is securely stored. The previous steps are all performed by the licensed replicator. This process is demonstrated in Figure 2-2.

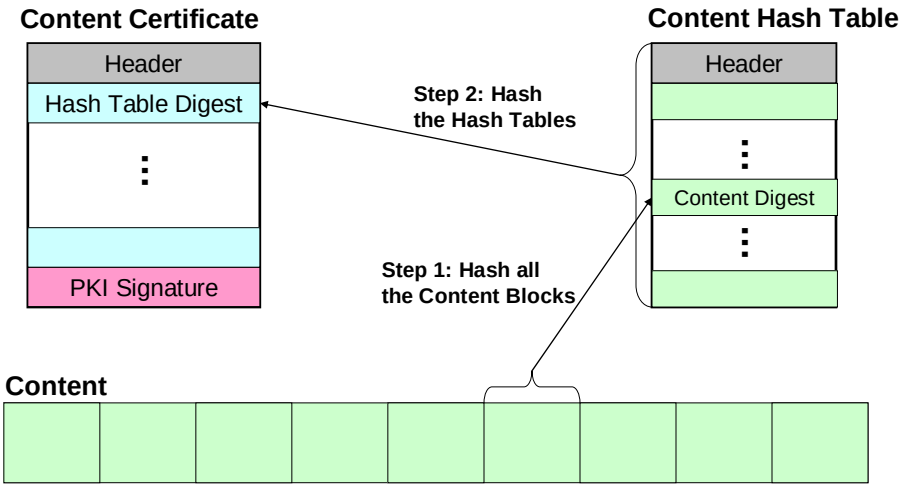

Figure 2-2 – Content Signing Process

2.6 Verifying Content Certificate

As a condition of playing, copying or other use of content stored on a pre-recorded media as defined in the relevant format specific books of this specification, a licensed product shall verify the integrity of that content by the following procedure, once for each starting use of that content.

1. Selects a subset of hash units randomly from all content hash units, from which the content hash value is calculated. The licensed product can select the subset of hash units using either of the following two procedures:
   a) Selects 7 hash units randomly from all content hash units
   b) Selects the first hash unit and additionally selects at least 1% of the remaining content hash units from the position where playback begins until the end of the Title, where the hash units are randomly selected and evenly distributed throughout all content hash units. As an example, the licensed product could use a pseudo randomly generated value modulo 100 to determine which one of the next 100 hash units to verify.

   All digests in the CHT must be included in this selection process.
2. Calculates hash value for each of the selected content hash units.

   $C_d = [\text{SHA-1}(\text{Hash\_Unit})]_{lsb\_64}$

   Where Hash_Unit is defined in the Format-specific Pre-Recorded books of this specification and SHA-1 is the SHA hashing function as defined in *Introduction and Common Cryptographic Elements* book.
3. Reads Content Hash Table (or tables), which includes the content hash values corresponding to the selected content hash units and calculates the Content Hash Table digest.

   $CHT_d = [\text{SHA-1}(\text{CHT})]_{lsb\_64}$
4. Reads Content Certificate, which includes the Content Hash Table digest (or digests) that should match the digests calculated in step 3.

5. Verifies the Signature of the Content Certificate, where the Content Hash Table digest (or digests) has been replaced with the digests calculated in step 3.

   AACS_Verify(AACS_CC$_{pub}$, CC$_{sig}$, CC)

   With AACS_Verify as defined in *Introduction and Common Cryptographic Elements* book. The public key, denoted AACS_CC$_{pub}$, is an additional public key of the AACS LA that is used only for the purpose of verifying the Content Certificate. All licensed players are required to store this AACS LA's additional public key in a way that resists malicious modification.
6. The licensed product will not proceed with content playback if the signature fails to correctly verify.
7. For each selected hash unit, verifies that the calculated C_d from step 2 is equal to the corresponding C_d that was contained in the selected Content Hash Tables. The licensed product will not proceed with content playback if the digest of any of the selected hash units fails to match the expected digest value.

In addition, the player computes the cryptographic hash of the usage rules stored on the media, C_{ur}. The licensed product will not proceed with content playback if C_{ur} is not equal to the associated usage rules hash value present in the content certificate.

C_{ur} = SHA-1(Usage_Rules)

This procedure may be performed either before or after playback begins. If it is performed after playback begins and the selection process described in step 1.a is utilized, steps 1 through 7 shall be completed within the first 300 seconds of playback. If it is performed after playback begins and the selection process described in step 1.b is utilized, steps 3 through 6 shall be completed within the first 30 seconds of playback.

2.7 Content Revocation List (CRL)

Under certain circumstances described in the license agreement, Certified Content may be revoked. When this occurs, corresponding revocation information is added to a Content Revocation List, which the AACS LA signs using its Entity Private Key and provides to all licensed replicators. Licensed replicators shall include the most recent CRL on each pre-recorded medium that they produce containing Certified Content, in a manner consistent with normal production cycles as described in the license agreement, and as described for each supported content format elsewhere in this specification. Table 2-2 shows the format of the CRL.

Table 2-2 – Content Revocation List for Pre-recorded Video

Byte \ Bit	7	6	5	4	3	2	1	0	
0	List Type: 0_{16}				(reserved)				CRL Header
1	List Version								
2									
3	Number of Segments (N)								
4 : 7	Segment Size #1 (S_1)								CRL Segment #1
8 : 4+S_1-41	Revocation Record Set #1								
4+S_1-40 : 4+S_1-1	Entity Signature #1								
…	…								…

X : X+3	Segment Size #N (S_N)	CRL Segment #N
X+4 : X+S_N-41	Revocation Record Set #N	
X+S_N-40 : X+S_N-1	Entity Signature #N	

A CRL shall consist of:

- A 4-bit List Type value, where 0_{16} shall be used to indicate a first-generation AACS Content Revocation List
- A 2-byte List Version value, which is assigned by the AACS LA to identify the version number of the CRL, and shall start at 0000_{16} and increase by an increment of one with each new version
- A 1-byte Number of Segments field, which indicates the number of CRL Segments that follow, and shall be at least 1.
- One or more variable-size CRL Segments, each consisting of the following:
    - A 4-byte Segment Size field, which indicates the size of the CRL Segment in bytes, starting with the Segment Size field itself, and ending with the Entity Signature field of that CRL Segment. The Segment Size value for the first segment (CRL Segment #1) shall be no more than 128K bytes less the CRL Header.
    - A variable-size Revocation Record Set field, which consists of zero or more Revocation Records, as described below.
    - A 40-byte Signature field, which contains the result of the AACS LA signing all fields above the Signature field, including the CRL Header and previous CRL Segments, using the Entity Private Key

Prior to providing access to Certified Content, licensed products shall read at least the CRL Header and CRL Segment #1 of the CRL provided on the pre-recorded medium with that content, and use the Entity Public Key to verify the Entity Signature found in that first segment. Licensed products that allocate more than the minimum non-volatile storage for the CRL shall also read subsequent CRL Segments, up to the amount of storage provided, if present, and verify their corresponding Entity Signature. NOTE: when reading more than one CRL Segment, only the signature of the last Segment must be checked since that signature includes all previous fields including previous Segments and the CRL Header. Licensed products shall verify that the CRL's List Version value is equal to or greater than the Minimum CRL Version value of the Content Certificate. If any of the above-mentioned verifications fails, such access shall be aborted.

The signature of a CRL segment is verified as follows:

AACS_Verify(AACS_LA_{pub}, CRL_Seg_{sig}, CRL_Seg)

With AACS_Verify and AACS_LA_{pub} as defined in *Introduction and Common Cryptographic Elements* book.

A licensed product must retain in non-volatile storage, at least the List Version and the Revocation Record Set #1 of the CRL data with the highest valued List version which it encounters and has verified. Therefore, when a CRL (or a portion thereof) is successfully verified as described above, the licensed product shall compare its List Version value to the List Version value of its persistently stored CRL data, if any. If

- no CRL data was previously stored, or

- if the List Version value of the previously stored CRL data is lower than that of the newly read CRL, or
- if the List Version values are the same but the newly read CRL data is larger (more complete) than the previously stored CRL data and the product is able to store more CRL data than it previously stored,

then the licensed product shall replace the previously stored CRL data, if any, with the newly read CRL data. Then, using its persistently stored CRL data, the licensed product shall examine the list to see if the Content Certificate ID to be accessed is revoked and if so, such access shall be aborted.

When persistently storing CRL data, licensed products shall have at least 128K bytes of non-volatile storage for that purpose. That size is sufficient to store the List Version field of the CRL and the Revocation Record Set field of CRL Segment #1. Licensed products that allocate more than 128K bytes of non-volatile storage shall store other fields of the CRL which fit in the additional storage, including the Revocation Record Set fields of other CRL Segments, if present. Where a licensed product persistently stores some but not all Revocation Record Set fields of a CRL, it shall give priority to storing the Revocation Record Set fields of the lowest-numbered CRL Segments. In all cases, licensed products shall treat CRL List Version and Revocation Record Set fields as Integrity Required, as defined in the license agreement. Table 2-3 shows the format of a Revocation Record for Content Certificate ID.

Table 2-3 – Revocation Record for Content Certificate ID

Bit / Byte	7	6	5	4	3	2	1	0
0	Record_Type: 00_{16}				Range (bits 11-8)			
1	Range (bits 7-0)							
2 : 7	Content Certificate ID							

A Revocation Record for Content Certificate ID shall consist of:

- A 4-bit Record Type value, where 00_{16} shall be used to indicate a Revocation Record for Content Certificate ID.
- A 12-bit Range value indicates the range of revoked Content Certificate ID's starting from the ID contained in the record. A value of zero in the Range field indicates that only one ID is being revoked.
- A 6-byte Content Certificate ID value (the concatenation of the 2-byte Applicant ID and the 4-byte Content Sequence Number) indicating the lowest numbered Content Certificate ID to be revoked.

The CRL is also used to contain revocation information for Managed Copy Servers. Licensed Players that support the Managed Copy features described in Chapter 5 shall check the persistently stored CRL for revocation entries that correspond to the Managed Copy Server Certificate that they received during communication with the Managed Copy Server as defined in Chapter 5 when requesting authorization to make a Managed Copy. Table 2-4 shows the format of a Revocation Record for Managed Copy Server Certificate ID.

Table 2-4 – Revocation Record for Managed Copy Server Certificate ID

Bit / Byte	7	6	5	4	3	2	1	0
0	Record_Type: 01_{16}				Range (bits 11-8)			
1	Range (bits 7-0)							

2 : 7	Managed Copy Server Certificate ID

A Revocation Record for Managed Copy Server Certificate ID shall consist of:

- A 4-bit Record Type value, where 01_{16} shall be used to indicate a Revocation Record for Managed Copy Server Certificate ID.
- A 12-bit Range value indicates the range of revoked Managed Copy Server Certificate ID's starting from the ID contained in the record. A value of zero in the Range field indicates that only one ID is being revoked.
- A 6-byte Managed Copy Server Certificate ID value indicating the lowest numbered ID to be revoked.

If a licensed product encounters a Revocation Record with a Record_Type value it does not recognize, the record shall be ignored. Other valid records in the CRL shall still be processed, however. Revocation Records within a CRL will not necessarily occur in order of their Content Certificate ID field values.

This page is intentionally left blank.

Chapter 3
Content Encryption and Decryption

3 Introduction

This chapter specifies the procedures for encryption and decryption of pre-recorded video content protected by AACS. Figure 3-1 presents an overview, and the remainder of the chapter describes the encryption and decryption processes in detail.

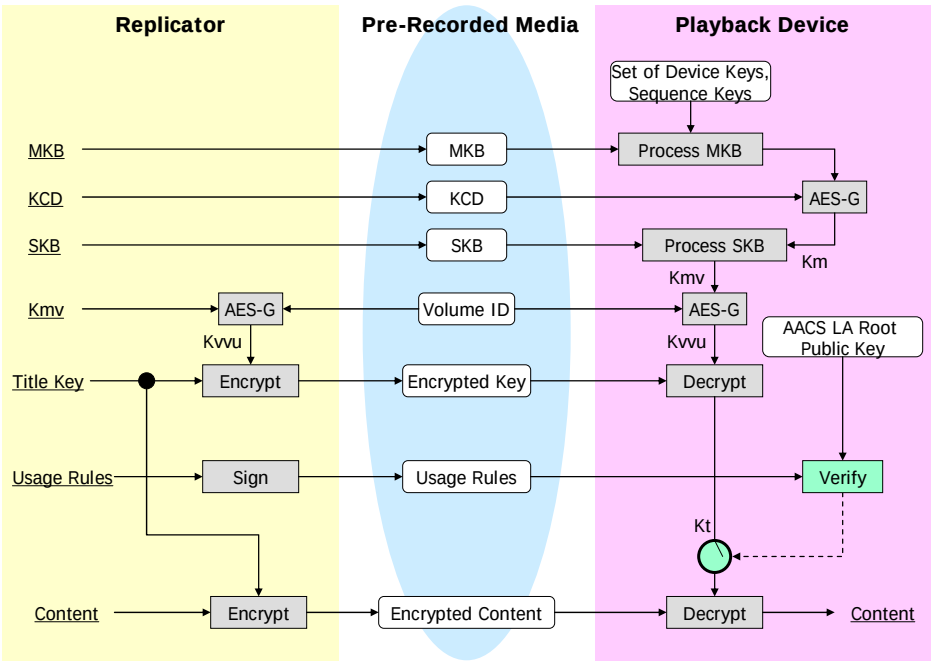

Figure 3-1 – Encryption and Decryption Overview

3.1 Content Encryption (General)

The owner of content that is to be protected provides the content in the form of one or more Titles, and their associated Usage Rules, to the licensed replicator.

The licensed replicator shall select a secret, random Title Key for each Title to be protected. Each Title Key shall be used to encrypt the content of its corresponding Title, as specified for each supported content format elsewhere in this specification. At the replicator's discretion, a given Title may be encrypted using the same Title Key for all instances of pre-recorded media, or different Title Keys may be used for different instances.

The licensed replicator shall also assign a unpredictable (e.g., random) identifier to the protected Title or set of protected Titles to be included together on a pre-recorded medium. This identifier, referred to as the Volume ID, is used as a safeguard against "bit-by-bit copying" of protected content, and is therefore stored on the pre-recorded medium in a manner that cannot be duplicated by consumer recorders, as specified for each supported storage format elsewhere in this specification. At the licensed replicator's discretion, the same Volume ID may be used for all instances of pre-recorded media containing a given protected Title or set of protected Titles, or different values may be assigned for different instances. Note: The use of different Volume IDs for the same set of protected Titles in combination with downloaded content that is bound using the Content Binding method as described in Section 5.5 of the *Introduction and Common Cryptographic Elements* book of this specification will limit the downloaded content to use with media which share the same Volume ID.

For each protected Title or set of protected Titles to be included together on a pre-recorded medium, the AACS LA provides to the licensed replicator a Media Key Block (MKB), a secret Media Key, a Sequence Key Block (SKB), and a corresponding set of secret Media Key Variants. The Media Key Block will enable all compliant devices, each using their set of secret Device Keys, to calculate the same Media Key as described in the *Introduction and Common Cryptographic Elements* book of this specification. If a set of Device Keys is compromised in a way that threatens the integrity of the system, an updated MKB can be released that will cause a device with the compromised set of Device Keys to calculate a different Media Key than is computed by the remaining compliant devices. In this way, the compromised Device Keys are "revoked" by the new MKB.

For each protected Title, the licensed replicator calculates a cryptographic hash of the Media Key and/or the Media Key Variants and the Volume ID, and uses the result to encrypt the Title's Title Key. The encrypted content, Encrypted Title Keys, signed Usage Rules and MKB are stored on the pre-recorded medium as specified for each supported storage/content format elsewhere in this specification.

3.2 Content Decryption (General)

The AACS LA provides a set of 253 secret Device Keys, denoted K_{d_0},K_{d_1},…,K_{d_n-1}, to the licensed manufacturer for inclusion into each compliant device or application produced. Device Key sets may either be unique per licensed product, or used commonly by multiple products; the license agreement describes the details and requirements associated with these two alternatives. A licensed product shall treat its Device Keys as highly confidential, as defined in the license agreement.

The licensed product reads the MKB from the pre-recorded medium, and uses its Device Keys to process the MKB and thereby calculate the Media Key, as described in the *Introduction and Common Cryptographic Elements* book of this specification. If the given set of Device Keys has not been revoked, then the calculated Media Key will be the same Media Key that was used by the licensed replicator as described above.

The AACS LA also provides a set of 256 Sequence Keys to the licensed manufacturer for inclusion in each compliant device or application produced. Like Device Keys, Sequence Keys may be either unique per licensed product, or used in common by multiple products. If a product's Device Keys are unique, then its Sequence Keys will also be unique; if a product's Device Keys are in common, then its Sequence Keys will also be in common. In either event, the licensed product reads the SKB from the prerecorded medium and uses its Sequence Keys and its previously calculated Media Key to calculate its Media Key Variant.

For each protected Title the licensed product then calculates a cryptographic hash of the calculated Media Key and/or the Media Key Variants and the Volume ID, and uses the result to decrypt the Title's Encrypted Title Key. The result is then used to decrypt the Title, as specified for each supported format elsewhere in this specification.

The licensed product verifies that the Content Certificate ID has not been included on a revocation list before allowing playback of that Title.

Playback of AACS Content shall only be performed using the Title Keys and Volume ID which are read from the media as defined in the applicable adaptation book. Except where otherwise provided for in these specifications, the values used to enable playback of AACS content (e.g. Title Keys and Volume ID) shall be discarded upon removal of the instance of media from which they were retrieved. Any derived or intermediate cryptographic values shall also be discarded.

3.3 Calculating the Volume Unique Keys

The Volume Unique Key (K_{vu}) and/or the Volume Variant Unique Keys (K_{vvu}) are used to encrypt and decrypt the Title Keys stored on the pre-recorded media, in a manner that is described in the given Format-specific book of this specification.

The Volume Unique Key is calculated as follows:

1. Calculate the Media Key (K_m):

The K_m, and the Media Key Block are delivered directly to licensed replicators. The licensed replicator embeds the MKB onto the media to enable licensed players to derive K_m.

2. Calculate the Volume Unique Key (K_{vu}):

The licensed replicator chooses a Volume ID (ID_v) to be placed on the pre-recorded media and calculates a Volume Unique Key (K_{vu}) as follows:

$$K_{vu} = \text{AES-G}(K_m, ID_v)$$

where AES-G represents the AES-based one-way function defined in the *Introduction and Common Cryptographic Elements* book.

The Volume Variant Unique Key is calculated as follows:

1. Calculate the Media Key Variant (K_{mv}):

The K_{mv}'s, and the Sequence Key Block are delivered directly to licensed replicators. The licensed replicator embeds SKB onto the media to enable licensed players to derive its particular K_{mv}, as described in Chapter 4

2. Calculate the Volume Variant Unique Key (K_{vvu}):

The licensed replicator calculates a Volume Variant Unique Key (K_{vvu}) as follows:

$$K_{vvu} = \text{AES-G}(K_{mv}, ID_v).$$

3.4 AACS Encryption on Pre-Recorded Media

The following steps detail the minimum procedures for encrypting AACS content on pre-recorded media as illustrated in Figure 3-1 above.

1. Generate the Title Key

The licensed replicator generates a statistically unique 128-bit Title Key (K_t).

2. Encrypt the content

The Title Key is used to encrypt the Content (C) as follows:

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

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

3. Sign the content

The encrypted content is signed using the procedure defined in Section 2.5.

4. Encrypt the Title Key(s)

The Title Key(s) is encrypted (K_{te}) as follows:

$$K_{te} = \text{AES-128E}(K_u, K_t)$$

where AES-128E represents encryption by the AES algorithm in ECB mode as defined in the *Introduction and Common Cryptographic Elements* book and the K_u's is one of the volume unique keys defined in Section 3.3. There may be more than one encryption at this step, if a Title Key is encrypted in all of the Volume Variant Unique Keys.

5. Transfer the data

The licensed replicator stores the encrypted Title Keys (K_{te}), Usage Rules, Content Certificate, and the encrypted AACS content to the pre-recorded media in the location and manner as specified for each supported Format elsewhere in this specification.

3.5 AACS Decryption on Pre-Recorded Media

The following steps detail the minimum procedures for decrypting AACS content on pre-recorded media as illustrated in Figure 3-1 above.

1. Decrypt the Title Key(s)

The Title Key(s) is decrypted (K_t) as follows:

$$K_t = \text{AES-128D}(K_u, K_{te})$$

where AES-128D represents decryption by the AES algorithm in ECB mode as defined in the *Introduction and Common Cryptographic Elements* book and K_u is one of the volume unique keys defined in Section 3.3.

2. Verify content is not revoked

The Content Certificate ID is checked for revocation status as defined in Section 2.7. If the content has been revoked, the content shall not be decrypted nor played.

3. Verify content signature

The encrypted content is verified as defined in Section 2.6. If the verification fails, the content shall not be further decrypted nor played.

4. Decrypt the content

The Title Key is used to decrypt the Content (C) as follows:

$$C = \text{AES-128CBCD}(K_t, C_e)$$

where AES-128CBCD represents decryption by the AES algorithm in CBC mode as defined in the *Introduction and Common Cryptographic Elements* book.

Steps 3 and 4 can be performed in parallel as defined in Section 2.6.

Chapter 4
Sequence Key Block

4 Introduction

This chapter describes the use of device Sequence Keys and the Sequence Key Block (SKB) on pre-recorded media. Fundamentally, AACS protection depends on Device Keys and the tree-based Media Key Block, which allows unlimited, precise revocation without danger of collateral damage to innocent devices. Because of the inherent power of the revocation of the AACS system, it is possible that attackers may forgo building clones or non-compliant devices and instead devote themselves to attacks where they try to hide the underlying compromised device(s). These attacks are both more expensive and more legally risky for the attackers, because the attacks require them to have an active server serving either content keys or the content itself, on an instance-by-instance basis.

It is possible that non-technical means could stop these “anonymous attack” servers. Nonetheless, AACS has developed a technology to accurately determine the underlying compromised devices that these servers are using, so the AACS revocation mechanism can be brought to bear. The technology is called Sequence Keys and the Sequence Key Block.

Sets of Sequence Keys are assigned to individual devices by the AACS LA out of a matrix of keys. The AACS LA also assigns the Sequence Key Blocks to be used on the pre-recorded media. In this respect, the key management aspects of Sequence Keys are identical to the technology developed by 4C Entity, LLC, called Content Protection for Recordable Media (CPRM). Sequence Key Blocks are very similar to CPRM Media Key Blocks; the only differences arise from the different ciphers used (AES instead of C2). However, unlike CPRM MKBs, the AACS SKBs are not part of the fundamental cryptographic protection of the content. The fundamental protection of AACS is the Media Key; the SKB merely allows different variants of the Media Key to be calculated by different devices.

The remainder of this section describes the Sequence Key Block in detail.

4.1 Sequence Key Block Principles

This is an informative section intended to give an overview of how Sequence Key Blocks work.

The AACS licensing agency will generate Sequence Keys organized in a large matrix. The matrix has 256 columns and not more than 65,536 rows. Each cell is a different Sequence Key. A single device has one key in each column. Thus, each device has 256 Sequence Keys.

Attackers would prefer to use already-compromised Sequence Keys if they could, so that no new forensic information could be deduced by the licensing agency. Therefore, it is important that compromised keys are no longer usable by the attackers. The problem is that many thousands of devices might share a single compromised key. Therefore, revocation of a single key is impossible. On the other hand, revocation of a unique *set* of keys is very possible; in fact, that is precisely what the SKB achieves. The fundamental principle is that no two devices have many keys in common, so even if the system has been heavily attacked and a significant fraction of the Sequence Keys is compromised, all innocent devices will have many columns in which they have uncompromised keys. The purpose of the Sequence Key Block is to give all innocent devices a column they can use to calculate the correct answer, while at the same time preventing compromised devices (who have compromised keys in all columns) from getting to the same answer.

In an SKB there are actually many correct answers, one for each variation in the content. For the purpose of explanation, however, it is helpful to imagine that a single SKB is producing a single answer. We will call that answer the *output key*. Then the SKB mechanism is completely identical to the CPRM/CPPM mechanism.

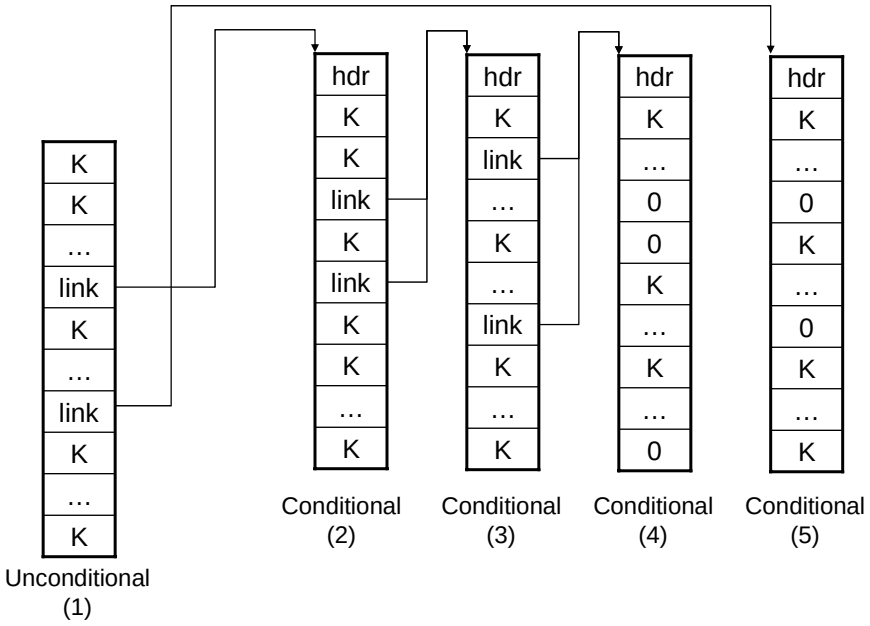

Figure 4-1 – Example Sequence Key Block

As shown in Figure 4-1 above, the SKB begins with a first column, called the "unconditional" column. By "column", we mean a column of Sequence Keys in the matrix will be used to encrypt. (To be precise, the key used to encrypt is derived from the Sequence Key, not the Sequence Key itself.) The first column will have an encryption of the output key (denoted 'K' in the figure) in every uncompromised Sequence Key's cell. Devices that do not have compromised keys in that column immediately decrypt the output key. Devices, both innocent and otherwise, that do have compromised keys instead decrypt a key called a *link key* that allows them to process a further column in the SKB. To process the further column they need both the link key and their Sequence Key in that column. Thus the subsequent columns are called "conditional" columns because they can only be processed by the device if it were given the necessary link key in a previous column.

The subsequent additional conditional columns are produced the same way as the first column: They will have an encryption of the output key in every uncompromised Sequence Key's cell. Devices with a compromised key will get a further link key to another column instead of the output key. However, after some number of columns depending on the actual number of compromised keys, the AACS licensing agency will know that only compromised devices would be getting the link key—all innocent devices would have found the output key in this column or in a previous column. At this point, rather than encrypting a link key, the agency encrypts a 0, and the SKB is complete. All innocent devices will have decrypted the output key, and all compromised devices have ended up decrypting 0.

How do the devices know they have a link key versus the output key? The short answer is they do not, at least not at first. Each conditional column has a header of known data ($DEADBEEF_{16}$) encrypted in the link key for that column. The device decrypts the header with the key it currently has. If the header decrypts correctly, the device knows it has a link key and processes the column. If it does not decrypt correctly, the device knows it has either the output key or a link key for a different column. When the device reaches the end of the SKB without decrypting 0, it knows it must have an output key. Note that this device logic allows the licensing agency to send different populations of devices to different columns by having more than one link key output from a single column. For example, in the figure, column (1) links to both column (2) and column (5). This flexibility can help against certain types of attacks.

The preceding description is equally accurate for an AACS SKB as it is for a CPRM/CPPM Media Key Block, with the exception that in the AACS SKB there is not a single output key, but multiple output keys called *Variant Data*.

4.2 Calculation of the Media Key Variant Data

4.2.1 Sequence Keys

Each AACS compliant device capable of playing pre-recorded content is given a set of secret Sequence Keys when manufactured. These are in addition to the Device Keys that all AACS devices require. These Sequence Keys are provided by the AACS LA and are for use in processing the Sequence Key Block. The result of the calculation is Variant Data which is then combined with the Media Key from the Media Key Block to generate the Media Key Variant, as explained in section 4.3. Key sets may either be unique per device, or used commonly by multiple devices. The AACS license agreement describes the details and requirements associated with these two options.

Each device receives 256 64-bit Sequence Keys, which are referred to as K_{s_i} (i=0,1,…,255). For each Sequence Key there is an associated Column and Row value, referred to as C_{s_i} (i=0,1,…,n-1) and R_{s_j} (j=0,1,…,m-1) respectively. Column and Row values start at 0. For a given device, no two Sequence Keys will have the same associated Column value (in other words, a device will have at most one Sequence Key per Column). It is possible for a device to have some Sequence Keys with the same associated Row values.

A device uses a Sequence Key K_{s_i} together with the Media Key K_m to calculate the Media Sequence Key K_{ms_i} as follows:

$$K_{ms\_i} = \text{AES-G}(K_m, K_{s\_i} \,||\, 0302153EE3EC7524_{16})$$

The Media Sequence Keys serve the role that the device keys in CPRM. In other words, the device does not use its Sequence Key directly to decrypt; instead, it combines it with the Media Key first as shown above. That means that a given SKB is associated with a given MKB (because the SKB depends on the Media Key for correct processing).

A device shall treat its Sequence Keys as highly confidential, and their associated Row values as confidential, as defined in the AACS license agreement.

4.2.2 Sequence Key Block (SKB)

The SKB is generated by the AACS LA and allows all compliant devices, each using their set of secret Sequence Keys and the Media Key, to calculate the Variant Data, D_v, which in turn allows them to calculate the Media Key Variant. If a set of Sequence Keys is compromised in a way that threatens the integrity of the system, an updated SKB can be released that causes a device with one or more compromised sets of Sequence Keys to calculate invalid Variant Data. In this way, the compromised Sequence Keys are “revoked” by the new SKB.

An SKB is formatted as a sequence of contiguous Records. Each Record begins with a one-byte Record Type field, followed by a three-byte Record Length field. The Record Type field value indicates the type of the Record, and the Record Length field value indicates the number of bytes in the Record, including the Record Type and the Record Length fields themselves. Record lengths are always multiples of 4 bytes. The Record Type and Record Length fields are never encrypted. Subsequent fields in a Record may be encrypted (by the AES cipher in ECB mode), depending on the Record Type.

Using its Sequence Keys, a device calculates D_v by processing Records of the SKB one-by-one, in order, from first to last. Except where explicitly noted otherwise, a device must process every Record of the SKB. The device must not make any assumptions about the length of Records, and must instead use the Record Length field value to go from one Record to the next. If a device encounters a Record with a Record Type field value it does not recognize, it ignores that Record and skips to the next. For some Records, processing will result in the calculation of a D_v value. Processing of subsequent Records may update the D_v value that was calculated previously. After processing of the SKB is completed, the device uses the most recently calculated D_v value as the final value for D_v.

If a device correctly processes an SKB using Sequence Keys that are revoked by that SKB, the resulting final D_v will have the special value $00000000000000000000_{16}$. This special value will never be an SKB’s correct final D_v value, and can therefore always be taken as an indication that the device’s Sequence Keys are revoked. Device behavior in this situation is implementation defined. As an example, a device could exhibit a special

diagnostic code, as information to a service technician. If at any point, a device calculates a D_v value of zero, it should discontinue processing of the SKB and may conclude that it has been revoked.

The following subsections describe the currently defined Record types, and how a device processes each.

4.2.2.1 Verify Media Key Record

Table 4-1 – *Verify Media Key* Record Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	Record Type: 81_{16}							
1	Record Length: 000014_{16}							
2								
3								
4	Verification Data (D_x)							
…								
19								

A properly formatted SKB shall have exactly one *Verify Media Key* Record as its first Record. Bytes 4 through 19 of the Record contain the ciphertext value

D_x = AES-128E (K_m, $0123456789ABCDEF_{16}$ || $XXXXXXXXXXXXXXXX_{16}$)

where $XXXXXXXXXXXXXXXX_{16}$ is an arbitrary 8-byte value, and K_m is the correct final Media Key value.

The presence of the *Verify Media Key* Record in an SKB is mandatory, but the use of the Record by a device is not mandatory. The device may use the *Verify Media Key* Record to make sure that it has the correct Media Key for processing the SKB. The Media Key comes from calculation based on the separate Media Key Block. Since MKBs and SKBs are associated, the device can, in effect, verify it has the correct MKB/SKB pair. The device action in the case of a mismatched MKB/SKB pair is manufacturer-specific. In any event, the device will not be able to process the content correctly.

If everything is correct, the device should observe the condition:

$[AES\_128D(K_m, D_x)]_{msb\_64} == 0123456789ABCDEF_{16}$

where K_m is the Media Key value.

4.2.2.2 Nonce Record

Table 4-2 – *Nonce* Record Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	Record Type: 03_{16}							
1	Record Length: 000014_{16}							
2								
3								
4	Nonce number (X)							
…								

19	

A properly formatted SKB shall have exactly one *Nonce* Record. The nonce number X is used in the Variant Data calculation as described below. The nonce record will always precede the Calculate Variant Data Record and the Conditionally Calculate Variant Data Records in the SKB, although it may not immediately precede them.

4.2.2.3 Calculate Variant Data Record

Table 4-3 shows the format of a *Calculate Variant Data* Record.

Table 4-3 – *Calculate Variant Data* Record Format

	Bit / Byte	7	6	5	4	3	2	1	0
	0	Record Type: 01_{16}							
	1	Record Length							
	2								
	3								
	4	Reserved							
	…								
	7								
	8	Column							
	9								
	10	Generation: 0001_{16}							
	11								
	12	Reserved							
	…								
	19								
Encrypted Key Data	20	Encrypted Variant Data for Row 0 (D_{ke_0})							
	…								
	29								
	30	Encrypted Variant Data for Row 1 (D_{ke_1})							
	…								
	39								
	40	.							
	…	.							
	Length-1	.							

A properly formatted SKB shall have exactly one *Calculate Variant Data* Record. Devices must ignore any *Calculate Variant Data* Records encountered after the first one in an SKB. The use of the reserved fields is currently undefined, and they shall be ignored. The Generation field shall contain 0001_{16} for the first generation. The Column field indicates the associated Column value for the Sequence Key to be used with this Record, as described below. Bytes 20 and higher contain Encrypted Key Data (possibly followed by some padding bytes at the end of the Record, not shown in Table 4-3). The first ten bytes of the Encrypted Key Data correspond to Sequence Key Row 0; the next ten bytes correspond to Sequence Key Row 1; and so forth.

Before processing the Record, the device checks that both of the following conditions are true:

Generation == 0001_{16}

and

the device has a Sequence Key with associated Column value C_{d_i} == Column, for some i.

If either of these conditions is false, the device ignores the rest of the Record.

Otherwise, using the value i from the condition above, the value X from the Nonce Record, and r = R_{d_i}, c = C_{d_i}, the device calculates:

$D_v = [\text{AES-G}(K_{ms\_i}, X \oplus f(c,r))]_{msb\_80} \oplus D_{ke\_r}$

where K_{ms_i} is the device's i-th Media Sequence Key's value and D_{ke_r} is the 80-bit value starting at byte offset r × 10 within the Record's Encrypted Key Data. f(c,r) represents the 128-bit value:

$f(c,r) = 0000_{16} \,||\, c \,||\, 0000_{16} \,||\, r \,||\, 0000000000000000_{16}$

where c and r are left-padded to lengths 16 bits, by prepending zero-valued bits to each as needed. The resulting D_v becomes the current Variant Data value.

It is not necessary for a first generation device to verify that Record Length is sufficient to index into the Encrypted Key Data. First generation devices are assured that the Encrypted Key Data contains a value corresponding to their Device Key's associated Row value.

4.2.2.4 Conditionally Calculate Variant Data Record

Table 4-4 shows the format of a *Conditionally Calculate Variant Data* Record.

Table 4-4 – *Conditionally Calculate Variant Data* Record Format

	Bit / Byte	7	6	5	4	3	2	1	0
	0	Record Type: 82_{16}							
	1	Record Length							
	2								
	3								
Encrypted Conditional Data (D_{ce})	4	$DEADBEEF_{16}$ (encrypted)							
	…								
	7								
	8	Column (encrypted)							
	9								
	10	Generation: 0001_{16} (encrypted)							
	11								
	12	Reserved							
	…								
	19								
Doubly Encrypted Variant Data	20	Doubly Encrypted Variant Data for Row 0 (D_{kde_0})							
	…								
	29								
	30	Doubly Encrypted Variant Data for Row 1 (D_{kde_1})							
	…								
	39								
	40	.							
	…	.							

	Length-1	.

A properly formatted SKB may have zero or more *Conditionally Calculate Media Key* Records. Bytes 4 through 19 of the Record contain Encrypted Conditional Data (D_{ce}). If decrypted successfully, as described below, bytes 4 through 7 contain the value $DEADBEEF_{16}$, bytes 8-9 contains the associated Column value for the Sequence Key to be used with this Record, and bytes 10-11 contain a Generation value of 0001_{16} for the first generation. Bytes 20 and higher contain Doubly Encrypted Variant Data (possibly followed by some padding bytes at the end of the Record, not shown in Table 4-4). The first ten bytes of the Doubly Encrypted Key Data correspond to Sequence Key Row 0, the next ten bytes correspond to Sequence Key Row 1, and so forth.

Upon encountering a Conditionally Calculate Variant Data Record, the device first calculates its current Media Key Variant, as follows:

$K_{mv} = AES\text{-}G(K_m, D_v \| 041826fa7749_{16})$

Where D_v is its current Variant Data calculated from a previous Calculate Variant Data Record or Conditionally Calculate Variant Data Record.

Using its current K_{mv} value, the device calculates Conditional Data (D_c) as:

$D_c = AES\text{-}128D(K_{mv}, D_{ce})$.

Before continuing to process the Record, the device checks that all of the following conditions are true:

$[D_c]_{msb\_32} == DEADBEEF_{16}$

and

$[D_c]_{79:64} == 0001_{16}$

and

the device has a Sequence Key with associated Column value $C_{d\_i} == [D_c]_{95:80}$ for some i.

If any of these conditions is false, the device ignores the rest of the Record.

Otherwise, using the value i from the condition above, X from the Nonce Record, the device's current Variant Data D_{vold}, and $r = R_{d\_i}$, $c = C_{d\_i}$, the device calculates:

$D_{vnew} = [AES\text{-}G(K_{ms\_i}, X \oplus f(c,r))]_{msb\_80} \oplus D_{vold} \oplus D_{kde\_r}$

where D_{kde_r} is the 80-bit value starting at byte offset $r \times 10$ within the Record's Doubly Encrypted Key Data, f(c,r) represents the 128-bit value:

$f(c,r) = 0000_{16} \| c \| 0000_{16} \| r \| 0000000000000000_{16}$

where c and r are left-padded to lengths 16 bits, by prepending zero-valued bits to each as needed. The resulting D_{vnew} becomes the current Variant Data value.

This record is always a multiple of 4 bytes.

4.2.2.5 End of Sequence Key Block Record

Table 4-5 – *End of Sequence Key Block* Record Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	Record Type: 02_{16}							
1	Record Length							
2								
3								
4	Signature Data							
…								
Length-1								

A properly formatted SKB shall contain an *End of Sequence Key Block* Record. When a device encounters this Record it stops processing the SKB, using whatever D_v value it has calculated up to that point as the final D_v for that SKB.

The End of Sequence Key Block Record contains the AACS LA's signature on the data in the Sequence Key Block up to, but not including, this record. Devices may ignore the signature data. However, if any device checks the signatures and determines that the signature does not verify or is omitted, it must refuse to use the Variant Data.

The length of this record is always a multiple of 4 bytes.

4.3 Calculation of the Media Key Variant from the Variant Data

When the device has finished processing the SKB, and if it has not been revoked, it will have an 80-bit valid Variant Data D_v. The device calculates the Media Key Variant from the Variant Data as follows:

$$K_{mv} = \text{AES-G}(K_m, D_v \,||\, 041826fa7749_{16})$$

In addition, the low-order 10 bits of the Variant Data identify the Variant Number for the device to use in playing the content, from 0 to 1023. This number usually denotes the particular Title Key file the device should use to decrypt the content, although the meaning and use of the Variant Number is format-specific.

Chapter 5
Managed Copy of Prerecorded Content

5 Introduction

Content protected by AACS includes an offer for the consumer to make at least one additional copy of that content after receiving appropriate authorization. That copy can be up to a full resolution "bit for bit" copy of the original content and can also include other offers where only certain portions of the original content is included in the copy. With some exceptions, all content protected by AACS includes this offer. Additional details on those exceptions are documented in the compliance rules portion of the Adopters Agreement. There may be additional offers available and for the purposes of this chapter, the term "Managed Copy" means a copy of the content that has been made subject to external authorization using the process defined in this chapter.

For the sake of clarity in this section, the precise definition of relevant terms is given as follows:

Content ID	Defined in Chapter 5 of the *Introduction and Common Cryptographic Elements* book
Default URL	A URL provided by AACS LA to be used for locating a Managed Copy Server for media which does not contain a valid Managed Copy URL. The Default URL is embedded into Managed Copy Machines for this purpose. The actual Default URL to be embedded into a Managed Copy Machine is available on the AACS LA website. NOTE to Adopters: It may be advisable to support the ability to update the embedded URL in the Managed Copy Machine.
Managed Copy Output Technology or (MCOT)	As defined in the Compliance Rules
Managed Copy Machine	or (MCM) is the consumer software or hardware which performs a Managed Copy. It may be tied to a Licensed Player, or it may exist as a standalone application – e.g. as part of a home media server
Managed Copy Server	or (MCS) is the remote computer that provide authorization to MCM's to make a Managed Copy. The appropriate MCS for a particular Title will be identified by an URL that will be contained on the media to be copied
Managed Copy Unit	or (MCU) is a particular offer that is made available as a part of the offers retrieved from the MCS or which reside on the media
PMSN	Defined in Chapter 5 of the *Introduction and Common Cryptographic Elements* book as the Pre-recorded Media Serial Number
Serial Number	This is a string provided to the managed copy server to identify the particular disc being copied. If the disc includes one, this is the PMSN. Otherwise, it is a human readable equivalent to a PMSN, entered by the User.

Figure 5-1 presents an overview of the Managed Copy process and the remainder of this chapter describes this process in detail.

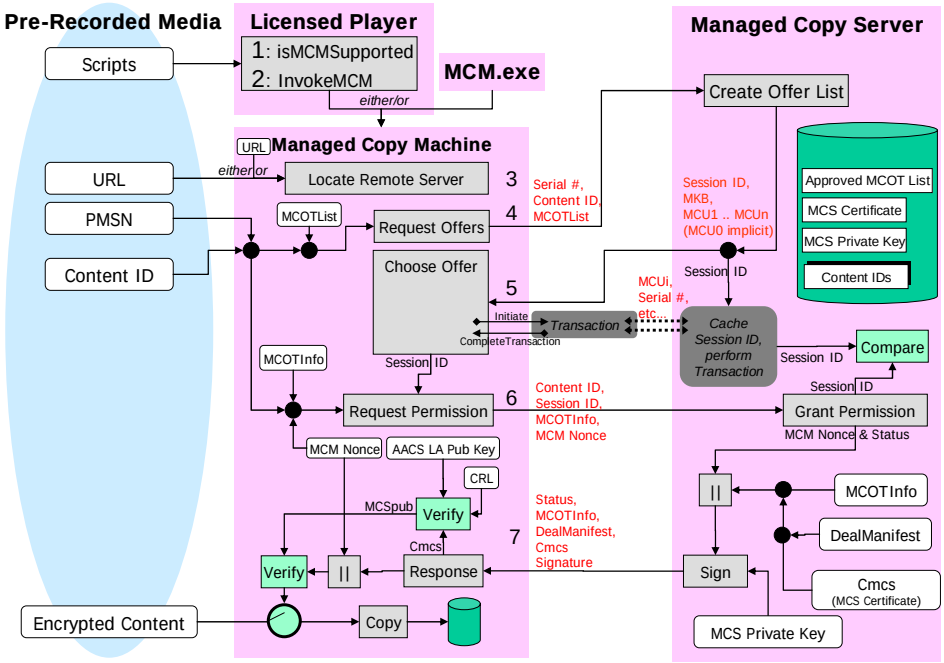

Figure 5-1 – Managed Copy Overview

When making a Managed Copy, the MCM must first connect to a MCS to obtain authorization. The URL contained on the media identifies the MCS to be used for obtaining this authorization. In the event that no valid Managed Copy URL is contained on the media, the MCM uses the Default URL to locate the MCS to be used. The MCM must provide to the MCS the Content ID contained on the media. The Format specific books of this specification define where these items are contained on the media.

The MCM can either be activated directly as a standalone application or it can be invoked via the menuing system contained within the scripts on the media to be copied. Assuming the MCM is activated via the menuing system, the MCM will follow the steps outlined below. If it is being activated as a standalone application, then the MCM will being at step 3.

1. The menuing system calls the API "IsMCMSupported" to determine if the Licensed Player contains the ability to make a Managed Copy. If the response is false, then terminate process.
2. The menuing system calls the API "InvokeMCM" which will transfer control to the MCM
3. The MCM uses the URL contained on the media to identify which MCS will be used to obtain authorization to make the Managed Copy
4. The MCM formulates a "Request Offer" message as described in Section 5.5.1 below, to be sent to the MCS as a means to request what Managed Copy offers are available.
5. The MCS formulates the list of Managed Copy offers that are available and sends them to the MCM using the AACS defined web service.
6. The MCM displays the Managed Copy offers to the user, using either its own custom display, an application referenced in the offers message, or an XSLT-generated web page. Subsequent interactions between the MCM and the MCS to exchange account information and any financial transactions are outside the scope of this specification.

7. Once the user has selected an offer and completed any required transactions with the MCS, the MCM sends a “Request Permission” message as described in Section 5.5.2 below, to the MCS with a Nonce value that can be used to protect against replay attacks on the “yes/no” response message from the MCS.
8. The MCS verifies the correctness of the values contained in the Request Permission message by comparing them to the values contained in previous transactions and if they are correct and all conditions have been met, then the MCS formulates a cryptographically secure response to the MCM that will indicate authorization to make the Managed Copy as described in Section 5.5.3 below.
9. The MCM will verify the integrity of the response message and if all conditions are met as described in Section 5.5.4 below, then the Managed Copy will be performed.

Sections 5.1 to 5.6 provide additional details on the process of making a Managed Copy and are normative unless otherwise specified.

Sections 5.7 and subsequent describe an implementation example and are informative unless otherwise specified. These sections are included to provide sufficient background and description to enable implementation of a Managed Copy infrastructure but are not meant to dictate a particular implementation.

5.1 Managed Copy Machine Initiation

AACS does not specify whether or not the MCM is integrated with the AACS Licensed Player and does not require that all Licensed Players support making a Managed Copy. AACS does define two normative API’s to facilitate the initiation of a Managed Copy as a part of the menuing system of the disc.

`IsMCMSupported`	A Boolean function which returns true if the player can invoke a managed copy server, otherwise it must return false. If the player is designed to support a managed copy machine, but none is available, it must return false.
`InvokeMCM`	A function which invokes a managed copy machine, if one is supported by the player.

The precise syntax of these APIs is format specific and will be defined in the format specific books of this specification. The general functions are as listed here. All AACS Licensed Players must support the equivalent of the IsMCMSupported call. Support for the InvokeMCM call is only required for Licensed Players that also contain an MCM. The menuing system is format specific and is defined in the format specific books of the specification. The scripts that implement the menuing system shall not call the InvokeMCM function if the IsMCMSupported call returns false.

The MCM may also be activated as a standalone application and is not required to be integrated with the menuing system on the media.

5.2 Connection Protocol

All of the normative communication between the MCM and the MCS are performed using the AACS defined web services interface as presented (see Appendix C).

The initial message sent from the MCM to the MCS is described in Section 5.5.1. MCMs shall use the following Application Programming Interface between the AACS secure layer in the MCM and the not-necessarily-secure Internet application layer in the MCM. This API is shown in Figure 5-2:

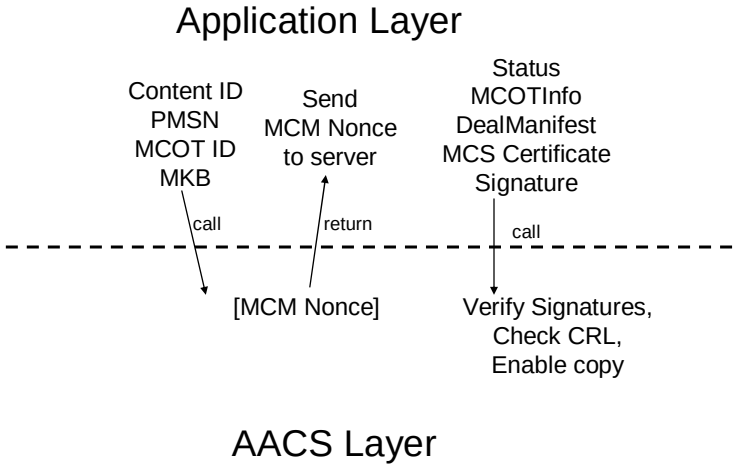

Figure 5-2 –Transaction Protocol API

5.3 Managed Copy Account Transactions

A managed copy may involve a financial and/or account setup transaction. The protocol used to exchange any such information is outside of the scope of this specification. However, two hooks are provided within the managed copy web services to enable the content provider to insert methods for collecting this information from the consumer. These hooks are defined in the Managed Copy Offer XML Schema (see Appendix A). They are:

- associating an encapsulated web service client with each offer;
- including links to a transaction web page.

These methods are roughly equivalent, since they both involve execution of a program defined by the MCS on the MCM, within a standard execution environment (e.g. iHD, BD-J or browser), and communicating transaction results to the MCM through an AACS defined API.

In both cases the Session ID in the Managed Copy Offer XML Schema (see Appendix A) may be used by the MCS to correctly associate each financial and/or account transaction with the correct AACS-specified managed copy message.

5.3.1 Encapsulated web service clients

A URI is included in the Managed Copy Offer XML Schema (see Appendix A) which may be used to encapsulate the required account or financial transactions for the specified offer. If used, this "financialApplicationURI" URI must point to a file (for example, a .JAR or an .ACA file) containing a client

application (BD-J or iHD, respectively). This client application would know how to engage the MCS-specific web service for these transactions.

The MCM retrieves the referenced file and executes the enclosed application. Once this application has finished, the last action it takes is to execute the required CompleteTransaction method as described in Section 5.3.3 below. This provides the MCM with the necessary information needed to complete the managed copy operation.

This method enables the MCM to be written without specific knowledge of the web service framework required for the MCS encountered.

5.3.2 Links to a transaction web page

In addition to the financialApplicationURI URI as described in Section 5.3.1, the Managed Copy Offer XML schema associates with each offer an financialHTMLURL URL. Regardless of how the offers are rendered for the user, the MCM can choose to consummate financial and/or account transactions using the page pointed to by this link.

The MCM launches a component browser passing to it the financialHTMLURL URL. When the transaction is complete, the last action the pages HTML must instruct the browser to do is to execute the CompleteTransaction method as described in Section 5.3.3 below. This provides the MCM with the necessary information needed to complete managed copy operation.

5.3.3 Managed copy machine object

The financial and/or account transactions required to acquire a managed copy approval are not normative to AACS. For this reason, they are executed within either a client application or a browser.

In order for these hosted components described in 5.3.1 and 5.3.2 to provide the MCM with the status of their actions, the MCM must expose to the selected component the CompleteTransaction method. This method can be implemented in a variety of ways:

- If the encapsulated web service client is used, then the method should be implemented as part of a client application method appropriate to the format (e.g. iHD or BD-J);
- If links to a transaction web page are used, then the method can be implemented as either a Java applet or an ActiveX control – whichever is most appropriate.

This API is synchronous, which means that a call of the API returns after the process of API is completed. Coupon, MCOT, MCUi, Status, and MCOTParams are set in variables in an MCM which are prepared for the purpose.

The managed copy machine may use the ‘render’ element of the XML offers schema (see Appendix A) to display the managed copy offers. This element contains the URI of a content provider application which can, for example, display the offers in a manner matching the look and feel of the disc menuing system.

In order for the interactive layer scripts to have access to the XML offers object, the MCM must load the contents of the received XML offers object into a managed copy ‘offers’ property.

So a managed copy machine must include two additions to the AACS object – a CompleteTransaction method and an offers string property.

Property: String Offers

Where:

Offers	Contains the XML object “Offers” returned from the managed copy server using the RequestOffer message (see 5.5.1 and Appendix C).

Functional Property: void CompleteTransaction

Syntax:

```
void CompleteTransaction( Coupon, MCOT, MCUi, Status, MCOTParams );
```

Where:

Coupon	A string uniquely identifying the financial or account transaction. If no financial or account transaction has been completed, Coupon must be a null string.
MCOT	A string identifier of the managed copy output technology selected for the managed copy, as defined in the AACS Compliance rules.
MCUi	Managed Copy Unit. A string containing the ID of the particular offer that was selected as a part of the transaction. If no offer was selected, the MCUi must be a null string..
Status	an optional string containing further information on the transaction. Informative: For example, if the transaction failed, Status may contain information about why that transaction failed.
MCOTParams (optional)	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.

5.4 MCS Certificate

The Managed Copy Server must have a MCS Certificate and a MCS Private Key which can be used to validate the Permission Response messages to MCM's. Table 5-1 shows the format of the MCS Certificate.

Table 5-1 –Managed Copy Server Certificate

Byte \ Bit	7	6	5	4	3	2	1	0
0	Certificate Type: 03_{16}							
1	(reserved)							
2 … 3	Length: $005C_{16}$							
4 … 9	MCS ID							
10 … 11	(reserved)							
12 : 51	MCS Public Key							
52 : 91	Signature Data							

Each MCS Certificate includes:

- A 1-byte Certificate Type value, where 03_{16} shall be used to indicate a first-generation AACS MCS Certificate
- A 2-byte Length field which indicates in bytes, the length of the MCS Certificate including the signature data.
- A 6-byte MCS ID field which will be a unique identifier for each Managed Copy Server and will be assigned by AACS LA.
- A 40 byte Managed Copy Server Public Key.
- A 40 byte Signature Data, calculated using the Entity Private Key, over the entire data up to and including MCS Public Key.

For future compatibility, when verifying the signature of the MCS Certificate the Length field should be used to locate the Signature Data field rather than a fixed offset.

5.5 Managed Copy Messages

5.5.1 Request Offer

When a MCM is seeking to make a Managed Copy, the "Request Offer" message shall be the first message sent to the MCS. The Request Offer message is a web service message as given in Appendix C, which uses the offer schema described in Appendix A. It contains the following information:

Serial Number (optional)	The Serial Number may be sent to the MCS to identify the specific instance of media for which the copy is being requested. The MCS can use the Serial Number to determine what offers remain available for this media. If the PMSN is included on the disc, then the Serial Number is the PMSN, and it must be sent to the MCS as part of the Request Offer message. If the PMSN is not included on the disc, then the Serial Number does not need to be passed to the MCS as part of the Request Offer Message.
Cid	Content ID. This must be provided to the MCS since it is needed to identify the content, and therefore the offers which are available.
MCOTList	This is an array of Managed Copy Output IDs (MCOT IDs) that are supported by the MCM. Each of the formulated offers (or MCU's) that are returned will specify which MCOT will be used as the output technology for that offer.

The response from this request is an XML object containing the offers available for this particular disc. The contents of that XML object are described in Appendix A.

The managed copy machine can display the offers in one of three distinct ways – 1) using its own custom display based upon the XML schema (see Appendix A), 2) using the optional XSLT provided with the offers to create an HTML representation of the offers, or 3) downloading and unpacking the optional archive file included in the offers message, and using that application to render the offers.

Note that in addition to the offers, this XML object includes an optional updated MKB, as indicated for all online services (see Chapter five of the *Introduction and Common Cryptographic Elements* book - Uses of On-line Connections).

5.5.2 Request Permission

Once the appropriate offer has been selected, the MCM sends a Request Permission message to the MCS. The Request Permission message is a web service message as given in Appendix C, which uses the permission schema described in Appendix B. The Request Permission message is executed synchronously with the Permission Response returned as described in Section 5.5.4. It contains the following information:

Cid	Content ID. This must be provided to the MCS since it is needed to identify the content, and therefore the offers which are available.
session id	Contains the Session ID that was returned in the XML object in response to the Request Offers message. This Session ID is used by the MCS to correlate this Request Permission message to any transactions that occurred as a result of selecting a particular offer, as described in Section 5.3.
MCOTInfo	(Optional) information sent to the MCS which is MCOT specific. For example, if the output technology requires a license to be created by a server which is not local to the consumer device, then the MCOTInfo field may contain information about the consumer device used in creating that license.
mcmnonce	Managed copy machine generated nonce. This will be used in processing the permission

response message to prevent replay attacks. The MCM shall retain a cached copy of the mcmnonce for comparison with the nonce value received back from the MCS in the Permission Response message.

5.5.3 Permission Response Creation

When the MCS receives a Request Permission message, the contents of the message are compared to the information received in the initial Request Offer message and any subsequent transactions that occurred. If all the information is correct and the conditions have been satisfactorily met, the MCS will compose a "Permission Response" message to be sent to the MCM. The Permission Response message is a web service message as given in Appendix C, using the permission schema described in Appendix B. It contains the following information:

status	Indicates whether or not permission has been granted to make the copy. This status field shall only be used by the AACS Layer after all message integrity checks have been completed. This status field can also be used to facilitate the Application Layer's ability to determine the authorization status.
MCOTInfo	(Optional) information sent to the MCM which is output technology specific. For example, if the output technology requires a license to be created by a server which is not local to the consumer device, then the MCOTInfo field might contain the license that is required by the MCOT.
dealmanifest	(Optional) the deal manifest will contain format specific information that corresponds to the MCOT and the MCU that was selected such that the MCM can determine exactly what needs to be copied when performing the copy to the destination media.
Mcmnonce	The same nonce that was sent to the MCS by the MCM in the Request Permission message.
signature	The Managed Copy Server (MCS) shall apply a cryptographic signature to the message which can be used by the AACS Layer to determine if the copy has or has not been authorized. This value shall be computed in the following manner: AACS_Sign(MCS_{pri}, permissionSignedContent) With AACS_Sign as defined in *Introduction and Common Cryptographic Elements* book and where MCS_{pri} is a ECDSA Private Key that has been provided by AACS LA to the MCS and where the data being signed consists only of the bytestream of the canonical serialization of the Managed Copy Permission XML Schema element permissionSignedContent and it's direct descendants and does not include the associated tags from the web service message (see Appendix B). In order to produce the contents of the permissionSignedContent element as a byte array for the AACS_Sign method, the canonical form of permissionSignedContent will be produced as UTF-8 bytes according to the Exclusive XML Canonicalization specification (http://www.w3.org/TR/xml-exc-c14n/#sec-Specification).
MCScert	The MCS Certificate is sent to the MCM and after it has been validated by the MCM, then the MCM shall use the public key contained within the MCS Certificate to verify the signature of the Permission Response message.

5.5.4 Permission Response Validation

When the MCM receives the Permission Response message from the MCS, it shall determine of the requested copy has been authorized using the process outlined below:

1. Verify the integrity of the MCS Certificate using the following procedure and shall refuse to allow the Managed Copy process to continue if the signature fails to verify.

AACS_Verify(AACS_LA$_{pub}$, Signature Data, MCS$_{cert}$)

With AACS_Verify as defined in *Introduction and Common Cryptographic Elements* book.

2. Verify the MCS Certificate has not been revoked using the procedure described in Section 2.7 above. If the MCS Certificate has been revoked, the MCM shall refuse to allow the Managed Copy process to continue.
3. Verify the integrity of the Permission Response Message using the following procedure and shall refuse to allow the Managed Copy process to continue if the signature fails to verify.

   AACS_Verify(MCS$_{pub}$, Signature Data, permissionSignedContent)

   With AACS_Verify as defined in *Introduction and Common Cryptographic Elements* book and where MCS$_{pub}$ is contained in the MCS$_{cert}$ and where the data which was signed consists only of the bytestream of the canonical serialization of the Managed Copy Permission XML Schema element permissionSignedContent and it's direct descendants and does not include the associated tags from the web service message (see Appendix B).

   In order to produce the contents of the permissionSignedContent element as a byte array for the AACS_Sign method, the canonical form of permissionSignedContent will be produced as UTF-8 bytes according to the Exclusive XML Canonicalization specification (http://www.w3.org/TR/xml-exc-c14n/#sec-Specification).
4. Verify that mcmnonce is the same nonce value that was transmitted to the MCS in the Request Permission message and shall refuse to allow the Managed Copy process to continue if the nonce values are not the same.
5. Determine if authorization to make the Managed Copy has been granted by verifying that the Status field of the Permission Response message is equal to true.

5.6 Making a Managed Copy

Once the MCM has validated the Permission Response, the copy can be made by the selected MCOT. The copy must be bound to the destination media using a binding method defined by the MCOT.

The MCOTInfo and Deal Manifest that is returned by the MCS in the Permission Response message will contain any MCOT specific information required by the selected MCOT to successfully bind the content to the destination media.

5.7 Informative Section: Components of a Managed Copy Architecture

This informative section is provided as a technical context for the requirements outlined in the normative sections, above. It describes some assumptions about the kind of architecture which may evolve around managed copy and the likely role of the normative managed copy components in that architecture.

For example,

- It is assumed that there will not be one monolithic managed copy server. Content providers will engage a variety of managed copy service providers - different service providers for different titles, for different titles at different times, or for the same title at the same time.
- Since permission granting for a managed copy is a more security sensitive operation, we have assumed that such permission granting servers might not be collocated with the 'store front' service server.
- It is further assumed that users will want to use a single account for multiple services, rather than having to set up an credit card or debit account with each and every managed copy service. For that reason, we consider it likely that the account manager will be a separate server, with multiple service servers accessing that same account.

What follows is an informative description of each component in this architecture.

5.7.1 The AACS Compliant Disc

An AACS compliant disc may contain the URL to identify the managed copy server. As noted above, the MCM also contains a Default URL to be used for any disc that does not contain this URL. The storage location and encoding of this URL can be format specific, but it must be available through an API to the AACS layer of the managed copy machine.

In the event that AACS compliant discs appear on the market prior to managed copy services, the URL stored on the AACS compliant disc may be set to a content provider owned URL which will deference to the managed copy service server chosen by the content provider for that disc. In this way changing the managed copy service server for a particular disc can easily done from a single chokepoint.

5.7.2 The AACS Compliant Player

There are two ways that a managed copy can be initiated. It may be started as part of the consumer's interactive experience with the menuing system of the disc, or it can be performed from a stand alone application.

Although it is optional for an AACS compliant player to enable a managed copy, it must expose the AACS specified API for determining if managed copy can be initiated from the player.

5.7.3 The Managed Copy Machine

This component is required because all products which support managed copy must, at a minimum, support the communication between the managed copy machine and the managed copy service server which makes rendering of managed copy offers possible.

The managed copy machine (MCM) is the consumer software which initiates a managed copy. It may be tied to a player, and/or it may exist as a standalone application – e.g.) as part of a home media server which includes an AACS-compliant drive.

Since AACS must support a wide variety of future managed copy scenarios, we make some rather limited assumptions about the nature of the target managed copy output technologies (MCOTs): other than they are AACS approved MCOTs, as listed in the AACS compliance rules.

All of the technologies currently approved bind the content to the MCOT locally, not from a server. However, future technologies may bind the content to the MCOT from a server, and that server could be collocated with the permission granting service.

This means two things:

- The permission granting mechanism is not required to be co-located with the binding operation, and
- this permission granting mechanism would only be required for binding mechanisms which are not collocated with the permission service.

In either case, a managed copy machine must be collocated with the drive in order to facilitate the managed copy operation. Such a machine has two logical components – one mandatory and one optional – based upon whether the binding operation is local to the AACS drive.

5.7.3.1 The Application Layer of the Managed Copy Machine

The application layer of the managed copy machine manages the communication with the managed copy service server, providing that server with the required information in the AACS specified format so that the server can render the managed copy offers to the consumer. For this reason, it must be provided in all products which support AACS managed copy.

This application layer of the managed copy machine is what is optionally launched by a AACS-compliant player.

5.7.3.2 The AACS Layer of the Managed Copy Machine

This component of the managed copy machine is required if binding of the content to the media takes place in the consumer product. Otherwise, it is not required.

If binding of the content is performed on separate machine from where the permission is granted, then a secure means of communication must be established between these two entities. This communication is described in detail in The Managed Copy Service Server, below.

5.7.4 The MCOT Transcryptor

Once permission has been granted to perform a managed copy, a transcription and rebinding operation can take place. If this transcription and rebinding occurs on the target device, a MCOT transcryptor is required. This is a software component which knows how to take decrypted AACS content and re-encrypt it in the approved MCOT, mapping rights correctly, according to the AACS compliance rules.

AACS places no requirements on how players, managed copy machines and MCOT transcryptors interoperate. However, it is assumed that a consumer may have multiple MCOTs on their license product. This leads to two general architectures:

- An optional player tied to a single managed copy machine, which in turn is tied to a single MCOT transcryptor.
- An optional player tied to a single managed copy machine, which in turn supports multiple MCOT transcryptors.

The second model would appear to be more consumer friendly, but we have defined an architecture which supports both models.

5.7.5 The Managed Copy Service Server

When a managed copy takes place, nominally, four essential steps occur. The first two steps are:

1. the user is provided with a list of managed copy offers
2. the user accepts one of these offers

The managed copy service server manages these first two steps. It provides the user with the list of offers and manages the response of the user to the offers. Since this process is a closed loop between the managed copy service server and itself, the majority of functioning of a managed copy service server is beyond the scope of this specification.

In the communication between the managed copy machine and the managed copy service server, the permission granting communication is a pass through to the correct managed copy permission server for the specified title.

5.7.6 The Managed Copy Permission Server

The final steps of a managed copy action are:

3. a permission request is made on the offer
4. permission is granted, and the managed copy occurs

In AACS managed copy, permission to perform the managed copy is provided by a network service. In the case when that the permission granting process and the rebinding process are separate, AACS defines a required mechanism for the managed copy permission to be securely communicated.

This page is intentionally left blank.

A Appendix
Managed Copy Offer XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.aacsla.com/2006/02/managedOffer"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns:tns="http://www.aacsla.com/2006/02/managedOffer"
     elementFormDefault="qualified">
     <xs:element name="offers">
          <xs:complexType>
               <xs:sequence>
                    <xs:element ref="tns:offer" minOccurs="1"
                         maxOccurs="256" />
               </xs:sequence>
               <xs:attribute name="MKB" use="optional" type="xs:string" />
               <xs:attribute name="render" use="optional" type="xs:anyURI" />
               <xs:attribute name="sessionId" use="required"
                    type="xs:string" />
               <xs:attribute name="version" use="required"
                    type="xs:decimal" />
          </xs:complexType>
     </xs:element>

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

```
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="url" type="xs:anyURI" />
        <xs:element name="language" type="xs:NCName" />
        <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="description" final="restriction">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:maxLength value="65536" />
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="MCOT" type="xs:string" />
        <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="availability">
            <xs:complexType>
                <xs:attribute name="end" use="optional" />
                <xs:attribute name="start" use="required" />
            </xs:complexType>
        </xs:element>
        <xs:element name="price" type="xs:string" />
        <xs:element name="financialApplicationURI" type="xs:anyURI" />
        <xs:element name="financialHTMLURL" type="xs:anyURI" />
        <xs:element name="ID" type="xs:string" />
</xs:schema>
```

B Appendix
Managed Copy Permission XML Schema

```
<?xml version="1.0"?>
<xs:schema
     targetNamespace="http://www.aacsla.com/2006/02/managedPermission"
     xmlns:tns="http://www.aacsla.com/2006/02/managedPermission"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="qualified">
     <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="tns:dealManifest"
                                        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="dealManifest">
          <xs:complexType>
               <xs:sequence>
                    <xs:any namespace="##other" processContents="lax" />
               </xs:sequence>
          </xs:complexType>
     </xs:element>
     <xs:element name="mcmNonce" type="xs:string" />
</xs:schema>
```

This page is intentionally left blank.

C Appendix
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:tns="http://www.aacsla.com/2006/02/managedCopyService"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns:aacsmp="http://www.aacsla.com/2006/02/managedPermission"
     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/managedPermission"
                    schemaLocation="aacs_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="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="aacsoffer: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>
```