Interface Control Document
Harmonic Scrambler to Key Management Server

Version 2.4

1 Table of Contents

2 List of Tables

3 Bibliography

HTTP Live Streaming [Online] / auth. Pantos R // IETF. - Apple Inc., 2-April, 2010. - Draft 3. - http://www.computer.org/cms/Computer.org/ComputingNow/homepage/2011/1211/T_MM1_TheMPEGDASHStandard.pdf.

HTTP Live Streaming Overview [Online] / auth. Apple Inc // Developer Apple. - Apple Inc, 25-March, 2010. - http://developer.apple.com/iphone/library/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/Introduction/Introduction.html.

Hypertext Transfer Protocol -- HTTP/1.1 [Online] / auth. Fielding R. [et al.] // IETF. - IETF, June, 1999. - http://datatracker.ietf.org/doc/rfc2616/.

IEEE Std 1003.1-2008 [Online] / auth. IEEE Std 1003 Committee // IEEE. - IEEE, 2008. - http://www.opengroup.org/onlinepubs/9699919799/.

Microsoft PlayReady Content Access Technology [Online] / auth. Microsoft Corporation // Microsoft PlayReady. - July 2008. - http://download.microsoft.com/documents/uk/mediumbusiness/products/silverlight/MicrosoftPlayReadyContentAccessTechnologyWhitepaper.pdf.

Portable encoding of audio-video objects The Protected Interoperable File Format (PIFF) [Online] / auth. Bocharov John A [et al.] // Micorosoft Web Site. - 9-March, 2010. - http://go.microsoft.com/?linkid=9682897.

SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) [Online] / auth. Gudgin Martin [et al.] // w3c. - w3c, 27-April, 2007. - 1.2. - http://www.w3.org/TR/soap12-part1/.

The Transport Layer Security (TLS) Protocol Version 1.2 [Online] / auth. Dierks T. and Rescorla E. // IETF. - IETF, August, 2008. - 1.2. - http://datatracker.ietf.org/doc/rfc5246/.

Error! Reference source not found. [Online] / auth. ISO/IEC DIS 23009-1 2011 // www.itscj.ipsj.or.jp. - http://www.itscj.ipsj.or.jp/sc29/open/29view/29n12313t.doc

4 Abbreviations

Abbreviation	Meaning
EMS	Element Management System
HTTP	Hyper Text Transfer Protocol
ICD	Interface Control Document – This document
KMS	Key Management Server
PIFF	Protected Interoperable File Format
SOAP	Simple Object Access Protocol
URI	Uniform Resource Identifier
UUID	Universally Unique Identifier
DASH	Dynamic Adaptive Streaming over HTTP
CW	Control Word in DVB Simulcrypt Specification
CWG	Control Word Generator in DVB Simulcrypt Specification

5 Revision History

Revision	Date	Updated by	Changes/Remarks
0.1	28th June 2010	Moore Macauley	Initial Draft

0.2	2nd July 2010	Moore Macauley	Internal comments included
0.3	9th July 2010	Moore Macauley	Corrected the spelling of Algorithm in the CreateKeySession and GetKeySession calls. Fixed the descriptive text in section 7.2.3 Corrected the data type for resourceId in the GetKeySession request. Removed optional for resourceId for GetKeySession, DestroyKeySession and InvalidKeySession Clarified the parameters returned by GetClientParameters to highlight those used for Apple and those used for Microsoft. Added a field to GetClientParameters to specify the length of the systemData field Clarified the expected usage of the systemData field in the GetClientParamters Response Clarified that this interface requires a key and not a key seed as used by some Microsoft implementations. Added a Minimum Implementation section.
0.4	19th July 2010	Moore Macauley	Updated all the references in the document so resourceId is correctly identified as a string throughout the document. Removed the keyURI from the GetClientParameters call and moved it in the GetKey call to allow for full flexibility in the key URI in the playlist file for Apple HTTP Live Streaming. This makes the GetClientParameters call optional for Apple HTTP Live Streaming and the document was updated to reflect this particularly the ping pong diagrams. Added clarification on requirements on 1:1 redundant scramblers and the support of them by the Key Server. Added a section on multiple scramblers scrambling the same resource and the implications on the Key Server. Added clarification on the configuration required on the scrambler for the minimum implementation and the requirements on the Key Server for redundancy. Removed the DestroyKeySession call that was incorrectly left in the Minimum Implementation ping pong diagram.
1.0	19th August	Moore Macauley	Specified the option for an unencrypted SOAP connection using basic authentication for deployment on secured networks and for debugging purposes. Updated the keyId in the GetKey Response to be a UUID as it is now only used for Microsoft.
1.1	24th November 2010	Moore Macauley	Fixed a typo where the tables showing the parameters for the InvalidKeySession request and response were incorrectly labeled
2.0	5th June 2012	Jiong He	Added new API GetKeyAndSignalization to support MPEG DASH Common Encryption spec. Add chapter 12 Multi-DRM
2.1	30th July 2012	Jiong He	Added new API GetKeyAndSignalization to chapter 11 Minimum Implementation. It was missed by mistake.

2.2	16th August 2012	Jiong He	Make input parameter "drmContent" of API GetKeyAndSignalizationRequest as required because scrambler must pass drmContentId which is part of drmContent as input to KMS.
2.3	22nd May 2013	Brian Lai	Added multi-DRM support over HLS
2.3d	7th Aug 2013	Brian Lai Elvis Fan	Added commonEncryptionParam under GetKeyAndSignalizationResponse
2.3e	19th Aug 2013	Brian Lai Elvis Fan	Added contenKey back to GetKeyAndSignalizationResponse to maintain backward compatibility with KMS2.2
2.4	11 Feb 2014	Elvis Fan Brian Lai	Added SsSignalizationType in GetKeyAndSignalizationRequest to support Smooth Streaming encryption with key rotation. Fixed minor errors.

6 Harmonic Key Management Server interface and interaction

This document provides a description of the interface and interaction between Harmonic scramblers that may also include a transcoding function and a Key Management Server (KMS). It is provides the interface *definition* and the *implementation details.* Two scenarios are included; live channel transcoding and encryption and offline file transcoding and encryption. The description in this Interface Control Document (ICD) focuses on just the interface between a KMS and the transcoders. Interfaces from the KMS toward the client side and the authentication/entitlement entities are assumed to be the separate responsibility of the DRM system which includes the KMS and are not relevant for the interaction of the transcoders with the KMS.

7 Interface Description

This document will define the interface used by an MPEG-2 Scrambler and the associated Key Management Server in a conditional access system. The protocol used will be SOAP and should be implemented over TLS (Dierks, et al., 2008) using mutual authentication, for debugging purposes or on secured networks the option to use a clear text connection using basic authentication should also be supported. This interface should be implemented or supported on all transcoder or session streaming devices that must encrypt video streams or files being delivered over unsecured networks. Anticipated usage is as follows:

- Microsoft Smooth Streaming (Bocharov, et al., 2010)
- Apple HTTP Live Streaming (Pantos, 2010)
- MPEG Dynamic Adaptive Streaming over HTTP (DASH) (ISO/IEC DIS 23009-1)

The interface must adhere requirements for the above mentioned content protection methods. The Scrambler will follow the recommendations and requirements as specified in the relevant specification.

Figure 1 provides a simplified view of a typical system scenario.

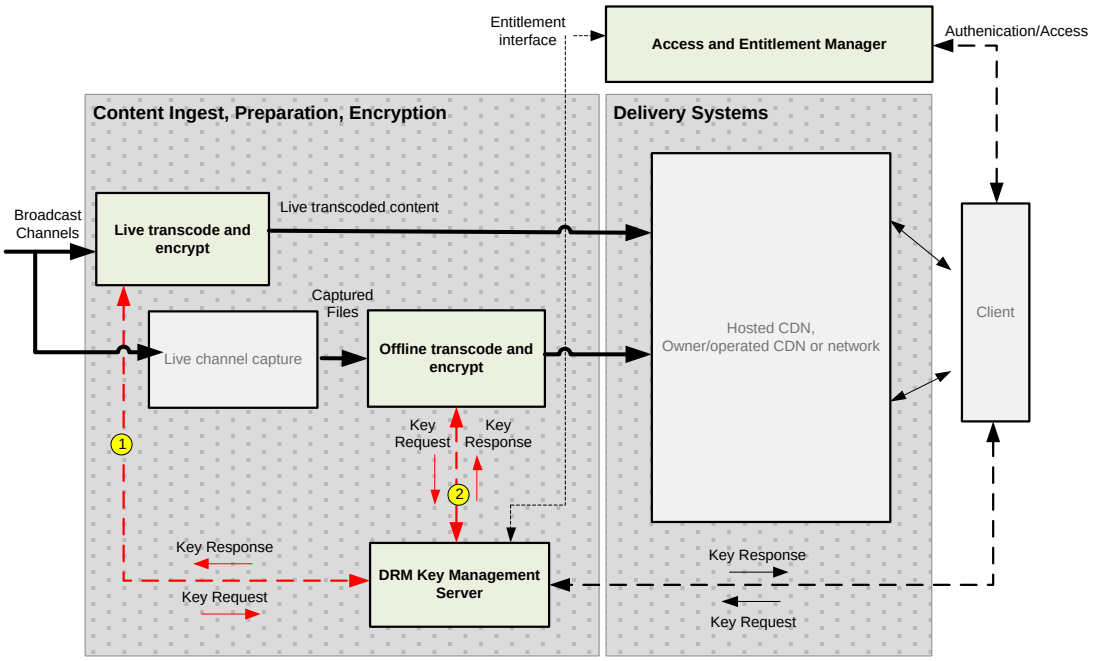


Figure 1. System view – simplified interface diagram

7.1 General interface interactions

Referring to figure 1, interface #1 is for live transcoders (with scramblers) interface #2 is for offline transcoders (with scramblers) for file-based scrambling. The basic operation of both interfaces is identical:

- Key sessions are established
- Key sessions are terminated
- Keys are requested by scrambler
- Keys are delivered by KMS

The system has three basic components:

1 A Scrambler – The Scrambler is responsible for scrambling the actual data.
2 A Key Management Server – The Key Management Server is responsible for managing key sessions on Key Servers.
3 A Key Server – The Key Server is responsible for generating keys.

7.2 Monitoring and Alarming

Both the KMS and the Transcoder/scrambler must provide alarms on failures related to this interface. The alarms should be raised to their respective monitoring/management system or they must provide SNMP traps which can be received by a customer or 3rd party Element Management System (EMS).

7.3 Key Session Management

A key session is established whenever the transcoder/scrambler wishes to encrypt either a new live channel or asset.

7.3.1 Create Key Session

This allows the creation of a key session on the KMS Server.

Parameter Name	Parameter Type	Parameter Description
resourceId	String	The resource id of the asset / live channel (AKA ContentID) that is to be encrypted. Must be ≤128 characters in length.
requestorId	UUID	This is the UUID of transcoder / scrambler requesting the key session
assetType	String	Defines the asset type must be one of: • VOD • LIVE
encryptionType	String	Defines the type of encryption being used must be one of: • PIFF • HTTP_STREAMING • DASH
EncryptionAlgorithm	String	The encryption algorithm to be used must be one of: • AES-CBC • AES-CTR
keyLength	Integer	Must be 128
cryptoPeriod	Integer	The length in playback time (in seconds) the key will be used for. It should be noted that this may not equated to wall clock time as a scrambler may scramble faster or slower than real time. A value of zero indicates that only 1 key will be requested on this session.
opaqueData	String	Private data required by the Key Management Server to determine the entitlement criteria for this asset.

Table 1: CreateKeySessionRequest Parameters

Parameter Name	Parameter Type	Parameter Description
returnCode	OperationReturnCode	Possible return values: • OPERATION_SUCCESS • STANDBY • ASSET_ID_DUPLICATE • UNSUPPORTED_ENCRYPTION • UNSUPPORTED_ASSET_TYPE • NO_RESOURCES • OPAQUE_DATA_INVALID • UNKNOWN_ERROR
errorMessage	String	Error message of operation

keySessionURI	String	The HTTP/S URI of the SOAP processor which the transcoder will contact to service key requests. The format of URI should follow the RFC 2396 standard.

Table 2: CreateKeySessionResponse Parameters

7.3.2 List Key Session

This allows the listing of the currently established key sessions.

Parameter Name	Parameter Type	Parameter Description
requestorId (optional)	UUID	When included limits the returned list of session to those established by the given requestorId

Table 3: ListKeySessionRequest Parameters

Parameter Name	Parameter Type	Parameter Description
returnCode	OperationReturnCode	Possible return values: • OPERATION_SUCCESS • STANDBY • UNKNOWN_REQUESTOR • UNKNOWN_ERROR
errorMessage	String	Error message of operation
resourceIds	ArrayOfResourceIds	Holds the resourceIds that match the list criteria

Table 4: ListKeySessionResponse Parameters

7.3.3 Get Key Session

This queries the parameters of a key session.

Parameter Name	Parameter Type	Parameter Description
resourceId	string	The resourceId associated with the Key Session to be fetched

Table 5: GetKeySessionRequest Parameters

Parameter Name	Parameter Type	Parameter Description
returnCode	OperationReturnCode	Possible return values: • OPERATION_SUCCESS • STANDBY • UNKNOWN_RESOURCE • UNKNOWN_ERROR
errorMessage	String	Error message of operation
resourceId	String	The resource id of the asset / live channel that is to be encrypted. Must be ≤128 characters in length.
requestorId	UUID	This is the UUID of transcoder / scrambler

		requesting the key session
assetType	String	Defines the asset type must be one of: • VOD • LIVE
encryptionType	String	Defines the type of encryption being used must be one of: • PIFF • HTTP_STREAMING • DASH
EncryptionAlgorithm	String	The encryption algorithm to be used must be one of: • AES-CBC • AES-CTR
keyLength	Integer	Must be 128
cryptoPeriod	Integer	The length (seconds) in playback time the key will be used for. It should be noted that this may not equated to wall clock time as a scrambler may scramble faster or slower than real time. A value of zero indicates that only 1 key will be requested on this session.
opaqueData	String	Private data required by the Key Management Server to determine the entitlement criteria for this asset.
keySessionURI	String	The http URI of the SOAP processor which the transcoder will contact key sever for key requests. KMS will decide which key server to use if it manages multiple key servers or whichever is geographically closer to the transcoder. The format of URI should follow the RFC 2396 standard.

Table 6: GetKeySessionResponse Parameters

7.3.4 Destroy Key Session

This allows key session to be destroyed after the scrambler has retrieved all of the required keys.

Parameter Name	Parameter Type	Parameter Description
resourceId	String	The resourceId associated with the Key Session to be destroyed

Table 7: DestroyKeySessionRequest Parameters

Parameter Name	Parameter Type	Parameter Description
returnCode	OperationReturnCode	Possible return values: • OPERATION_SUCCESS • UNKNOWN_RESOURCE

		• STANDBY • UNKNOWN_ERROR
errorMessage	String	Error message of operation
resourceId	String	The resourceId of the key session that was included in the DestroyKeySession Request

Table 8: DestroyKeySessionResponse Parameters

7.3.5 Invalid Key Session

This allows the Scrambler to inform the Key Management Server when a Key Server is not responding to key requests. The Key Management Server should assign a new Key Server for this session.

Parameter Name	Parameter Type	Parameter Description
resourceId	String	The resourceId associated with the Key Session to be destroyed

Table 9: InvalidKeySessionRequest Parameters

Parameter Name	Parameter Type	Parameter Description
returnCode	OperationReturnCode	Possible return values: • OPERATION_SUCCESS • UNKNOWN_RESOURCE • STANDBY • UNKNOWN_ERROR
errorMessage	String	Error message of operation
keySessionURI	String	The http URI of the SOAP processor which the transcoder will contact to service key requests. The format of URI should follow the RFC 2396 standard.

Table 10: InvalidKeySessionResponse Parameters

7.3.6 Heartbeat

This allows the scrambler to ensure that it has a connection to an active Key Session Management Server

Parameter Name	Parameter Type	Parameter Description
version	String	The is the version of this interface implemented by the scrambler. For this version it should be set to 2.0

Table 11: HeartbeatRequest Parameters

Parameter Name	Parameter Type	Parameter Description
returnCode	OperationReturnCode	Possible return values: • OPERATION_SUCCESS • UNSUPPORTED_VERSION • UNKNOWN_ERROR
errorMessage	String	Error message of operation
status	String	Possible return values:

		• ACTIVE • STANDBY A value of active implies that this Key Management Server is ready and willing to accept requests. A value of standby implies that this Key Management Server will not process requests other than heartbeat

Table 12: HeartbeatResponse Parameters

7.4 Key Provisioning

Keys are requested whenever the scrambler wishes for a new key associated with an established key session. These commands are sent on the key session between the scrambler and the session end point.

7.4.1 Get Client Parameters

This allows the scrambler to retrieve the static data associated with a key session.

Parameter Name	Parameter Type	Parameter Description
resourceId	String	The resourceId associated with the Key Session which is queried.

Table 13: GetClientParametersRequest Parameters

Parameter Name	Parameter Type	Parameter Description
returnCode	OperationReturnCode	Possible return values: • OPERATION_SUCCESS • UNKNOWN_RESOURCE • UNKNOWN_ERROR
errorMessage	String	Error message of operation
resourceId	String	The resource id of the asset / live channel that is to be encrypted. Must be ≤128 characters in length.
systemId (omitted Apple) (required Microsoft)	UUID	The information to be included in the Protection System Specific Header Box for PIFF. This is required for Microsoft Smooth Streaming and should not be specified for Apple HTTP Live Streaming. Is should be set to 9A04F079-9840-4286-AB92-E65BE0885F95 for Microsoft PIFF.
systemDataLength (omitted Apple) (required Microsoft)	Integer	The length of the systemData included. This is required for Microsoft Smooth Streaming and should not be specified for Apple HTTP Live Streaming.
systemData	Bytes[systemDataLength]	The information to be included in the

(omitted Apple) (required Microsoft)	Protection System Specific Header Box for PIFF. The Scrambler will take the given bytes and insert them directly into the data field of the Protection System Specific Header Box. This is required for Microsoft Smooth Streaming and should not be specified for Apple HTTP Live Streaming.

Table 14: GetClientParametersResponse Parameters

7.4.2 Get Key

This allows a key to be retrieved.

Parameter Name	Parameter Type	Parameter Description
resourceId	String	The resourceId associated with the Key Session on which a key is to be generated.
time	Integer	This is the time in seconds the scrambler is requesting a key for. For a file based scrambler this will be the offset from the start of the file. For a live channel scrambler this will be the POSIX time (IEEE Std 1003 Committee, 2008) representing the middle of the segment. If the segments are less than 2 seconds than they should be grouped together to form a block that covers at least 2 seconds and the time specified should be the middle of the block.

Table 15: GetKeyRequest Parameters

Parameter Name	Parameter Type	Parameter Description
returnCode	OperationReturnCode	Possible return values: • OPERATION_SUCCESS • UNKNOWN_RESOURCE • UNKNOWN_ERROR
errorMessage	String	Error message of operation
key	Byte[keyLength]	This is the key used to encrypt the content. Note: This interface does not include the key seed paradigm offered in the Microsoft PIFF Encryptor tool a complete key must be returned.
keyId (Required Microsoft) (Omitted Apple)	UUID	This is the key Id used by the client to retrieve the given key
keyURI	String	The http URI of the client will contact to

(Required Apple) (Omitted Microsoft)	retrieve the key; it will be inserted by the Scrambler into the Playlist. The format of URI should follow the RFC 2396 standard.

Table 16: GetKeyResponse Parameters

7.4.3 GetKeyAndSignalization

This allows the scrambler to retrieve common key, keyId and multiple DRM data.

Parameter Name	Parameter Type	Multiplicity	Parameter Description
scheduledKey	ScheduledKeyType	0..N	It holds key\keyId pair(s) generated by scrambler or received by scrambler from a 3rd party key server via DVB Simulcrypt interface. If only time is specified in ScheduledKeyType but not key\keyId, key\keyId are generated and managed by KMS.
drmList	DrmListType	0..1	The list of DRM systems that shall be addressed by this request. If omitted the configured list of DRM(s) managed by KMS will be assumed.
drmContent	DrmContentType	1	Content management and security profile information.

Table 17: GetKeyAndSignalizationRequest

Parameter Name	Parameter Type	Multiplicity	Parameter Description
returnCode	OperationReturnCode	1	Possible return values defined in table 26.
errorMessage	String	0..1	Detailed error message of operation, e.g., which DRM system ID is invalid, etc
scheduledKey	ScheduledKeyType	0..N	The associated time(s) and content key\keyId pair(s) to be used for content protection. It is omitted in case the request failed.
contentKey	ContentKeyType	0..1	**Deprecated. For backward compatibility only.** The first content key\keyId pair to be used for content protection. It is omitted in case the request failed.
signalization	DrmSignalizationType	0..1	The list of multiple DRM signalization data.
commonEncry	CommonEncryptionPara	0..1	The Common Encryption

ptionParam	mType	Parameter for each output format. e.g. encryption mode for HLS

Table 18: GetKeyAndSignalizationResponse

7.4.3.1 Simple Types defined in GetKeyAndSignalization

7.4.3.1.1 ScheduledKeyType structure

Field name	Field type	Multiplicity	Description
time	xsd:unsignedLong	1	This is the time in seconds the scrambler is requesting a key for. For a file based scrambler this will be the offset from the start of the file. For a live channel scrambler this will be the POSIX time (IEEE Std 1003 Committee, 2008) representing the middle of the segment. If the segments are less than 2 seconds than they should be grouped together to form a block that covers at least 2 seconds and the time specified should be the middle of the block.
contentKey	ContentKeyType	0..1	A content key provided by the encoder. In that case, no key will be generated for this request. If omitted, then it is generated or retrieved by KMS.

Table 19: ScheduledKeyType structure

7.4.3.1.2 ContentKeyType structure

Field name	Field type	Multiplicity	Description
keyId	UUID	0..1	The unique identifier of the content key, provided as a normalized UUID.
key	xsd:base64binary	0..1	The content key. Mandatory in the response if the KMS has the key generation role. Mandatory in the request if the Scramble has this role.
iv	xsd:base64binary	0..1	The initialization vector. If

			omitted, Scrambler should use IV as it deems fit.

Table 20: ContentKeyType structure

7.4.3.1.3 DrmListType structure

Field name	Field type	Multiplicity	Description
drm	DrmInfoType	1..N	The list of addressed DRMs. If the list is empty, then the list of configured DRM will be assumed.

Table 21: DrmListType structure

7.4.3.1.4 DrmInfoType structure

Field name	Field type	Multiplicity	Description
drmSystemId	UUID	1	The unique identifier of the DRM system, provided as a normalized UUID.
drmName	xsd:string	0..1	The readable name for the DRM system. Not mandatory, but may help at configuration time.
drmMetadata	xsd:base64binary	0..1	The input DRM specific metadata.

Table 22: DrmInfoType structure

7.4.3.1.5 DrmContentType structure

Field name	Field type	Multiplicity	Description
drmContentId	xsd:string	1	Identifier of the content ID allocated by the CMS.
profile	SecurityProfileType	0..1	The set of properties for the management of the content key and the signalization. The security profile is mandatory when requesting a content key.

Table 23: DrmContentType structure

7.4.3.1.6 SecurityProfileType structure

Field name	Field type	Multiplicity	Description
distributionMode	xsd:string	1	Defines the asset type must be one of: • VOD • LIVE
streamingMode	xsd:string	1	Defines the type of streaming

			mode being used must be: • DASH • HLS • SS
emi	xsd:unsignedShort	0..1	Defines encryption method indicator must be one of: 0x4022 (AES-128, CBC) 0x4024 (AES-128, CTR)
cryptoPeriod	xsd:unsignedInt	0..1	The length in playback time (in seconds) the key will be used for. It should be noted that this may not equated to wall clock time as a scrambler may scramble faster or slower than real time. A value of zero indicates that only 1 key will be requested.

Table 24: SecurityProfileType structure

7.4.3.1.7 OperationReturnCode

Base type	Enumeration	Description
xsd:string	OPERATION_SUCCESS	The request succeeded.
	UNDEFINED_DRM_SYSTEM_ID	One of the provided DRM system ID is not recognized in the system.
	UNDEFINED_STREAMING_MODE	The provided streaming mode is not known in the system.
	UNDEFINED_DISTRIBUTION_MODE	The provided distribution mode is not known in the system.
	INTERNAL_ERROR	A non business error occurred at server level.
	UNAVALAIBLE_SERVICE	One of the resource used by KMS is not available.
	UNDEFINED_ENCRYPTION_METHOD	When the provided EMI does not match a valid encryption method or if the DRM does not support it.
	INVALID_ DRM_METADATA	Invalid DRM metadata.

Table 25: OperationReturnCode

7.4.3.1.8 DrmSignalizationType structure

Field name	Field type	Multiplicity	Description
dash	DashSignalizationType	0..N	MPEG-DASH signalization as a MPD content protection XML

			section and/or PSSH box data.
hls	HlsSignalizationType	0..N	HLS signalization URI and/or header XML section
ss	SsSignalizationType	0..N	SS Signalization as a manifest protection XML section and/or PSSH box data.

Table 26: DrmSignalizationType structure

7.4.3.1.9 DashSignalizationType structure

Field name	Field type	Multiplicity	Description
drmSystemId	UUID	1	The unique identifier of the DRM system, provided as a normalized UUID.
drmName	xsd:string	0..1	The readable name for the DRM system. Not mandatory, but may help at configuration time.
manifestHeader	xsd:anyType	0..1	The content protection section to be inserted into the MPD.
psshBox	PsshType	0..1	Pssh signalization information blob for each DRM to be inserted in the content, as a base64 string.

Table 27: DashSignalizationType structure

7.4.3.1.9.1 PsshType structure

Field name	Field type	Multiplicity	Description
data	xsd:base64binary	1	DRM specific block to be inserted in the pssh box, provided as a base64 string.

Table 28: PsshType structure

7.4.3.1.10 HlsSignalizationType

Field name	Field type	Multiplicity	Description
drmSystemId	UUID	1	The unique identifier of the DRM system, provided as a normalized UUID.
drmName	xsd:string	0..1	The readable name for the DRM system. Not mandatory, but may help at configuration time.
variantPlaylistTag	xsd:string	0..N	Additional tag to be inserted in variant playlist for the DRM system
indexPlaylistTag	xsd:string	0..N	Additional tag to be inserted in index playlist for the DRM system

keyAttribute	HlsKeyAttributeType	0..N	Attribute to be added /overriden in EXT-X-KEY tag.

Table 29: HlsSignalizationType structure

7.4.3.1.10.1 HlsKeyAttributeType

Field name	Field type	Multiplicity	Description
attributeName	xsd:string	1	The name of the attribute
attributeValue	xsd:string	1	The value of the attribute. Note the value of attribute can be quoted-string or non-quoted string.

Table 30: HlsKeyAttributeType structure

7.4.3.1.11 SsSignalizationType structure

Field name	Field type	Multiplicity	Description
drmSystemId	UUID	1	The unique identifier of the DRM system, provided as a normalized UUID.
drmName	xsd:string	0..1	The readable name for the DRM system. Not mandatory, but may help at configuration time.
manifestHeader	xsd:anyType	0..1	The protection section to be inserted into the manifest.
psshBox	PsshType	0..1	Pssh signalization information blob for each DRM to be inserted in the content, as a base64 string.

Table 31: SsSignalizationType structure

7.4.3.1.12 CommonEncryptionParamType structure

Field name	Field type	Multiplicity	Description
hls	HlsCommonEncryptionParamType	0..1	Common EncryptionParam for HLS

Table 32: CommonEncryptionParamType structure

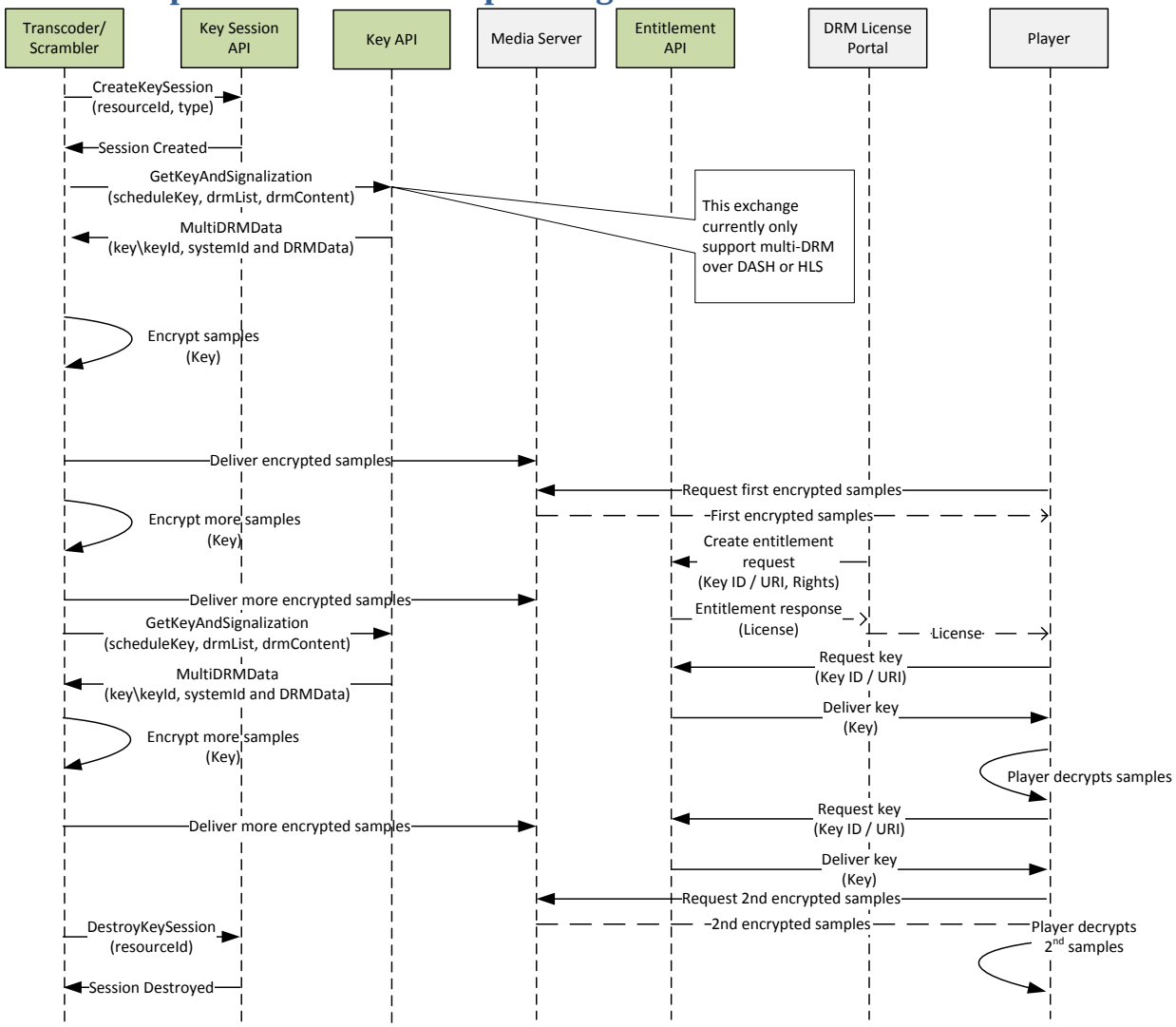
7.4.3.1.13 HlsCommonEncryptionParamType structure

Field name	Field type	Multiplicity	Description
HlsEncryptionMode	xsd:string	0..1	If present, defines the method of encryption mode being used. Must be: • AES-128-CBC • AES-128-CTR

			• SAMPLE-AES

Table 33: HlsCommonEncryptionParamType structure

8 Full Implementation –Example Usage



9 Redundancy

The following discusses how the interface deals with redundancy of the components of the system.

9.1 Scrambler

The scrambler has two possible operation modes for redundancy:

- N : M – Where a backup device will be provisioned when the primary device fails
- 1 : 1 – Where both a primary and backup device are actively scrambling the same content.

9.1.1 N for M

In this case the following is the expected flow:

1 The primary device fails
2 The backup device is provisioned with the same resourceId as the primary
3 The backup device connects to the Key Management server and queries for a list of active sessions associated with its resourceId
4 The backup device queries each session to find the URI for the Key Server connection.
5 The backup device connects to the Key Server and issues a Get Client Parameters request
6 The backup device issues a Get Key request to the Key Server
7 The backup device starts scrambling and outputting video content.

9.1.2 1 for 1

In this case both scramblers share the same resourceId and both are scrambling making key requests to the Key Server this means that the Key Server will have two different IP addresses requesting keys for the same session. In the event of the failure of one there is no required action on the part of the second device.

In the case of 1:1 a heartbeat mechanism is required between the primary and back up to designate a master which should be responsible for key session creation and destruction. It is possible that a redundancy switch occurs during the period of existence of the key session in which case different machines may create and destroy the key session.

9.2 Key Management Server

The scrambler has two possible operation modes for redundancy:

- Active : Active – Where there are two server both of which can handle session requests. In this case queries of either server must be identical.
- Active : Standby – Where the backup server will not respond to queries.

9.2.1 Active : Active

The Scramblers are configured with the IP addresses of both Key Management Servers and choose one at random. In the event that the Key Management Server does not respond to a request then the scrambler tries the same operation on the second Key Management Server

9.2.2 Active : Standby

The Scramblers are configured with the IP addresses of both Key Management Servers and choose one at random. If the operation return status from the Key Management Server is standby then the scrambler uses the other server.

9.3 Key Server

Key Servers are considered a pooled resource. In the event that a Scrambler is unable to get a key from a Key Server it sends an Invalid Key Session message to the Key Management server which should respond with a new Key Server to use to continue the session.

In the event that the Key Management Server is unavailable or that it is unable to provide a new Key Server then the Scrambler have two options:

- Pause processing and continue retrying with the Key Management Server until a valid Key Server is available and provides a new key. (This is only practicable in the case of file assets.)
- Continue Scrambling with the last valid key and keep retrying with the Key Management Server until a valid Key Server is available and provides a new key. In the event that the Scrambler does not have a valid key then depending on the user specified option it will either:
    - Output the data in the clear
    - Output nothing.

10 Multiple Scramblers

In this case multiple scramblers are working together to encrypt the same content this may be employed where a customer requires low asset creation time or in the live case where a single scrambler cannot achieve the necessary throughput.

In this case each scrambler will be configured with the same requestorId and the same resoureId for the asset and will contact the same Key Management Server. One of the scramblers will be designated master for the job and will do the Key Session creation; all the remaining scramblers will attempt GeyKeySession requests until such time as they are provided with a response. Once the scrambling job has been completed the master scrambler will issue the command to destroy the key session.

11 Minimum Implementation

In order to allow fast integration and reduced time to market, there is a minimum implementation of this interface. This interface removes the requirement for session management however it requires:

- Manual configuration on the Scrambler and Key Server
- Redundancy of the Key Server must be offered via a virtual IP address or other suitable method not covered by the document.
- Seamless scaling of the Key Servers.

The minimum implementation reduces the interface to 3 calls as it assumes a key session has already been configured manually on the Key Server. These are:

- GetClientParameters (Required for Microsoft Smooth Streaming)
- GetKey
- GetKeyAndSignalization (Required for MPEG DASH Streaming)

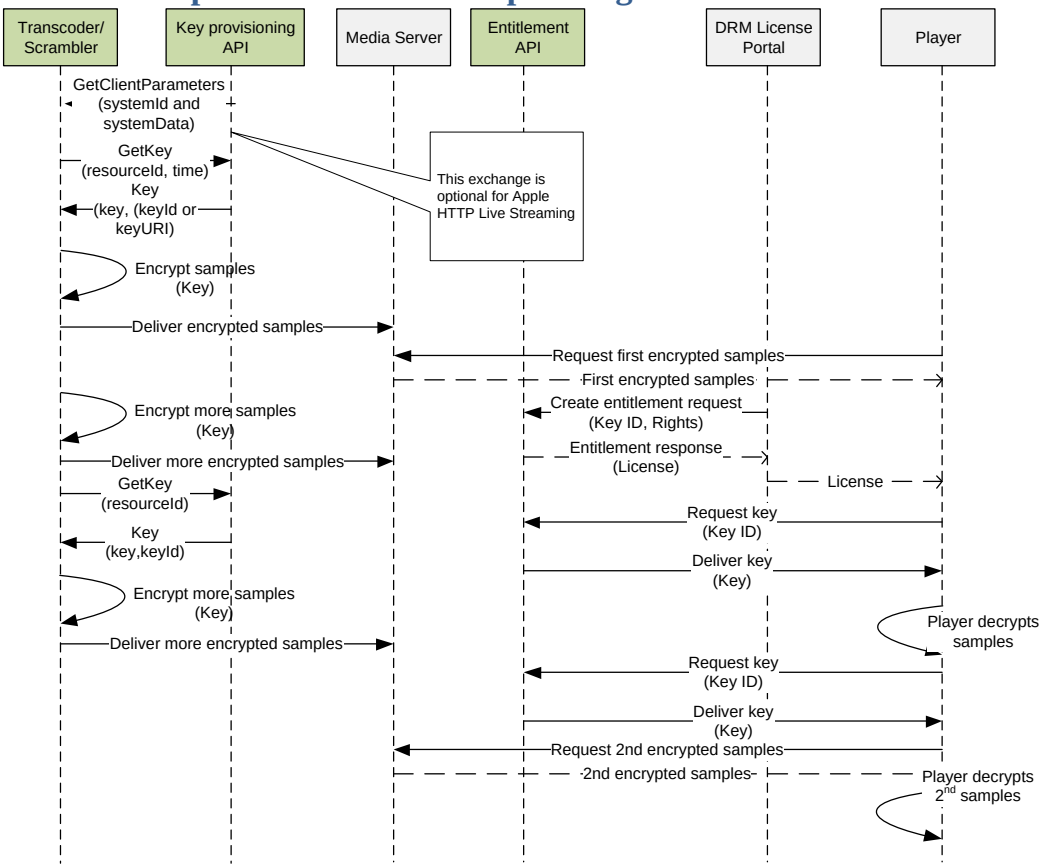
Configuration on the Scrambler will allow the setting of the following parameters:

- Key Server end point (Single entry in RFC 2396 format)
- resourceId
- encryptionAlgorithm
- keyLength

NOTE:

1. In case of one centralized KMS working with multiple DRM systems (e.g., one DRM for Smooth, one DRM for HLS, one DRM for MPEG-DASH), the minimum implementation doesn't support this use case by default because GetKey doesn't distinguish encryptionType (i.e., PIFF, HLS, DASH) which is only available in Key Session.
   One workaround is to have KMS return different keyURIs for different encryption types. For example:
   - Ex1: http://<server>/HLS/KeyService
   - Ex2: http://<server>/Smooth/KeyService
   - Ex3: http://<server>/DASH/KeyService

11.1 Minimum Implementation – Example Usage



12 Multi-DRM

12.1 Multi-DRM with one centralized KMS for key generation and synchronization

Figure 2 provides a simplified view of a typical Multi-DRM system scenario where one centralized KMS is responsible for key generation and synchronization.

This is a typical use case where CMS, encoder, scrambler and key servers co-exist in a Multi-DRM ecosystem. In this case, the scrambler will only interface with one centralized KMS component instead of interfacing with multiple DRM systems directly as shown by interface#1 in Figure 2.

For MPEG-DASH Common Encryption protection scheme, this centralized KMS generates common key(s) and keyId(s) to be utilized by one or more DRM systems to enable decryption of the same file using different DRM systems. GetKeyAndSignalization is designed specifically for MPEG-DASH Common Encryption key provisioning, and is extended to support multi-DRM for HLS streaming. Implement key session if full implementation is desired.

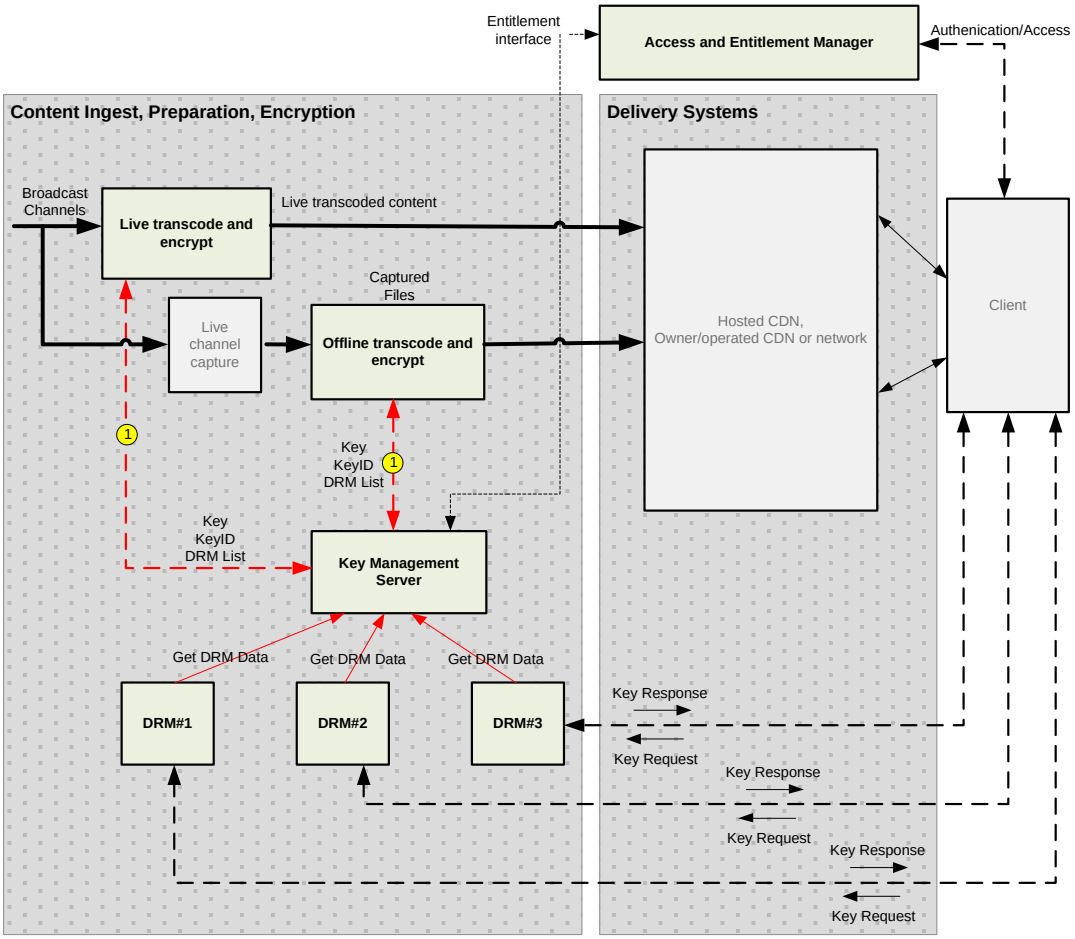


Figure 2. System view – Multi-DRM with centralized KMS for key generation and synchronization

12.2 Multi-DRM with Scrambler responsible for key generation and synchronization

Figure 3 provides a simplified view of a typical Multi-DRM system scenario where Scrambler is responsible for key generation and synchronization.

This is typical use case where CMS does not have a Multi-DRM solution in the ecosystem or customer wants to re-use CAS content keys (i.e., Control Words as in DVB Simulcrypt specification) generated by Broadcast\IPTV Mux (CAS world) for OTT content (DRM world).

In this case, Scrambler will interface with multiple DRM systems directly as shown by interface#2.

There are two ways for Scrambler to generate keys:

1. Scrambler(s) get CW generated by external CWG as shown in interface#1 in Figure 3. This CWG can be a Broadcast\IPTV Mux in CAS world. The CW is going to be utilized as encryption key in KMS 2.0 interface. This option can be used for N:M and 1:1 scrambler redundancy mode.

2. Scrambler provides internal CWG module to generate CW to be utilized as encryption key in KMS 2.0 interface. Interface#1 in Figure 3 will not be involved in this case. This option can't be used for 1:1 scrambler redundancy mode.

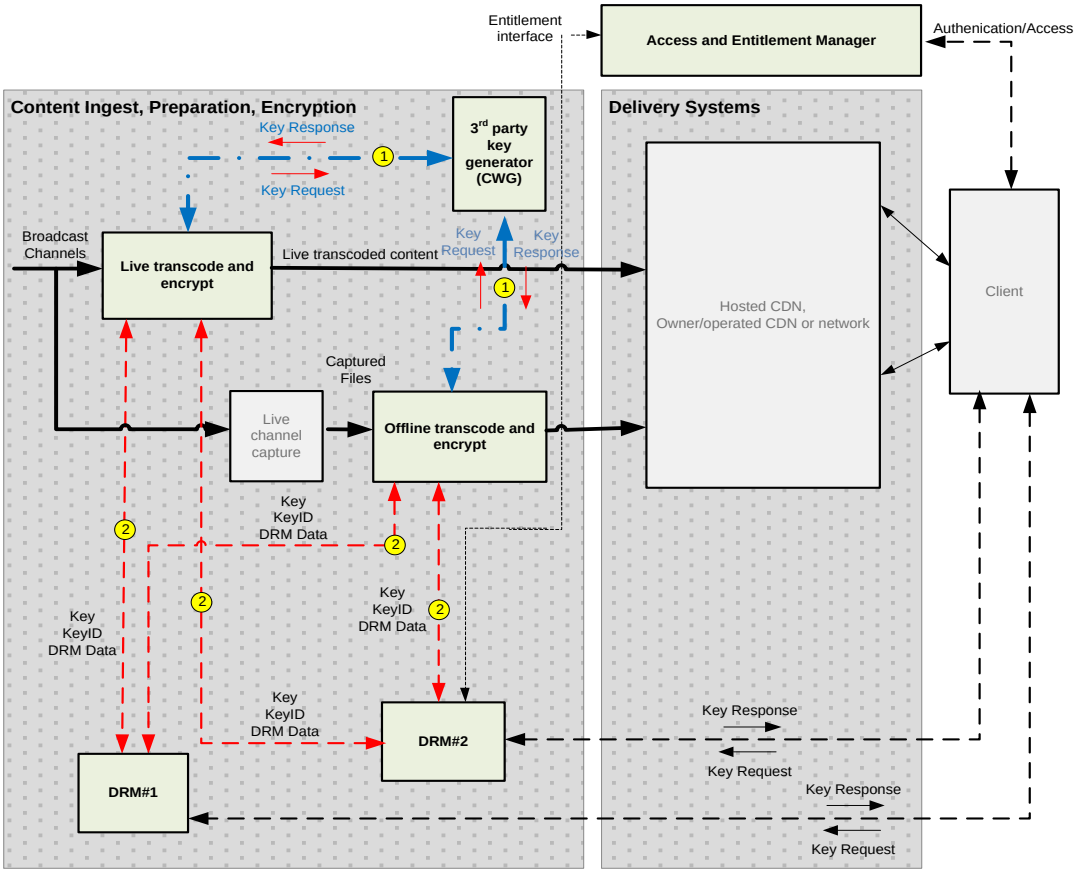


Figure 3. System view –Multi-DRM with Scrambler responsible for common key generation and synchronization

12.3 Multi-DRM – Example Usage