ATSC Recommended Practice: Digital Rights Management (DRM)

A/362:2026-04
14 April 2026

ATSC®, the Broadcast Standards Association®, is an international, non-profit organization developing voluntary standards and recommended practices for broadcast television and multimedia data distribution. ATSC member organizations represent the broadcast, professional equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries. ATSC also develops implementation strategies and supports educational activities on ATSC standards. ATSC was formed in 1983 by the member organizations of the Joint Committee on Inter-society Coordination (JCIC): the Consumer Technology Association (CTA), the Institute of Electrical and Electronics Engineers (IEEE), the National Association of Broadcasters (NAB), the Internet & Television Association (NCTA), and the Society of Motion Picture and Television Engineers (SMPTE). For more information visit www.atsc.org.

Note: The user's attention is called to the possibility that compliance with this document may require use of an invention covered by patent rights. By publication of this document, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. One or more patent holders have, however, filed a statement regarding the terms on which such patent holder(s) may be willing to grant a license under these rights to individuals or entities desiring to obtain such a license. Details may be obtained from the ATSC Secretary and the patent holder.

Implementers with feedback, comments, or potential bug reports relating to this document may contact ATSC at https://www.atsc.org/feedback/.

Revision History

Version	Date
A/362:2020 approved	17 January 2020
A/360:2022-03 (references to ATSC documents updated) A/360:2022-03 revision approved	31 March 2022 [date]
A/362:2023-02 revision approved	20 February 2023
A/362:2023-03 (references to ATSC documents updated) A/362:2024-03 Amendment No. 1 approved	28 March 2023 14 February 2024
A/362:2024-02 published	14 February 2024
A/362:2024-04 (references to ATSC documents updated) A/362:2024-04a (Title of Section 5.1.2 corrected and Section 5.4 removed, per A/362:2023-03 Amend. No. 1)	3 April 2024 15 May 2024
A/362:2025-07 (references to ATSC documents updated)	17 July 2025
A/362:2026-04 (references to ATSC documents updated)	14 April 2026

Table of Contents

Index of Figures

ATSC Recommended Practice: Digital Rights Management (DRM)

1. SCOPE

This Recommended Practice provides best industry practices for implementers of A/360 [8] and the security and content protection provisions of A/344 [9]

1.1 Introduction and Background

This document is a collection of material that was previously located in A/344 and A/360, and then published in Technology Group Report, "DRM Guidelines". This RP supersedes and replaces that document.

1.2 Organization

This document is organized as follows:

- Section 1 – Outlines the scope of this document and provides a general introduction.
- Section 2 – Lists references and applicable documents.
- Section 3 – Provides a definition of terms, acronyms, and abbreviations for this document.
- Section 4 – System overview
- Section 5 – Client processing
- Section 6 – Use of A/344 APIs
- Annex A – Example message flows
- Annex B – Example SLT

2. REFERENCES

All referenced documents are subject to revision. Users of this Standard are cautioned that newer editions might or might not be compatible.

2.1 Informative References

The following documents contain information that may be helpful in applying this Standard.

[1] IEEE: "Use of the International Systems of Units (SI): The Modern Metric System," Doc. SI 10, Institute of Electrical and Electronics Engineers, New York, NY.

[2] ISO/IEC: ISO/IEC 23001-7:2016, "Information technology — MPEG systems technologies — Part 7: Common encryption in ISO base media file format files."

[3] W3C: "Encrypted Media Extensions," W3C Recommendation 18 September 2017, World Wide Web Consortium.
https://www.w3.org/TR/encrypted-media;
a newer work in process is available at: https://w3c.github.io/encrypted-media/.

[4] W3C: "Media Source Extensions", W3C Recommendation 17 November 2016, World Wide Web Consortium.
https://www.w3.org/TR/media-source

[5] ATSC: "ATSC Standard: Signaling, Delivery, Synchronization and Error Protection," Doc. A/331:2026-04, ATSC, Washington, DC, 14 April 2026.

[6] ISO/IEC: "Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 1: MPEG media transport (MMT)," Doc. ISO/IEC 23008-

1:2017(E), International Organization for Standardization/ International Electrotechnical Commission, Geneva Switzerland.

[7] DASH-IF: "Guidelines for Implementation: DASH-IF Interoperability Points, Version 5, Part 6, Content Protection", DASH Industry Forum, Beaverton, OR. ISO/IEC: "ISO/IEC 23009–1:2022, Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats," International Organization for Standardization, Geneva. Publicly available: https://standards.iso.org/ittf/PubliclyAvailableStandards/c083314_ISO_IEC%2023009-1_2022(en).zip
Schemas and examples are available here: https://standards.iso.org/iso-iec/23009/-1/ed-5/en/.

[8] ATSC: "ATSC 3.0 Security and Service Protection," Doc. A/360:2026-04, ATSC, Washington, DC, 14 April 2026.

[9] ATSC: "ATSC 3.0 Interactive Content," Doc. A/344:2026-04, ATSC, Washington, DC, 14 April 2026.

[10] ISO/IEC: "ISO/IEC 14496-12:2015, "Information technology — Coding of audio-visual objects – Part 12: ISO base media file format," International Organization for Standardization, Geneva, 5th Edition, 15 December 2015.

[11] ATSC: "Technology Group Report: DRM Guidelines", ATSC, Washington, DC, 12 March 2019.
https://www.atsc.org/standards/other-technical-documents/

[12] ISO/IEC: "Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 1: MPEG media transport (MMT)," Doc. ISO/IEC 23008-1:2017(E), International Organization for Standardization/ International Electrotechnical Commission, Geneva, Switzerland.

3. DEFINITION OF TERMS

With respect to definition of terms, abbreviations, and units, the practice of the Institute of Electrical and Electronics Engineers (IEEE) as outlined in the Institute's published standards [1] should be used. Where an abbreviation is not covered by IEEE practice or industry practice differs from IEEE practice, the abbreviation in question will be described in Section 3.3of this document.

3.1 Compliance Notation

This section defines compliance terms for use by this document:

should – This word indicates that a certain course of action is preferred but not necessarily required.

should not – This phrase means a certain possibility or course of action is undesirable but not prohibited.

3.2 Treatment of Syntactic Elements

This document contains symbolic references to syntactic elements used in the audio, video, and transport coding subsystems. These references are typographically distinguished by the use of a different font (e.g., restricted), may contain the underscore character (e.g., sequence_end_code) and may consist of character strings that are not English words (e.g., dynrng).

3.2.1 Reserved Elements

One or more reserved bits, symbols, fields, or ranges of values (i.e., elements) may be present in this document. These are used primarily to enable adding new values to a syntactical structure without altering its syntax or causing a problem with backwards compatibility, but they also can be used for other reasons.

The ATSC default value for reserved bits is '1'. There is no default value for other reserved elements. Use of reserved elements except as defined in ATSC Standards or by an industry standards-setting body is not permitted. See individual element semantics for mandatory settings and any additional use constraints. As currently reserved elements may be assigned values and meanings in future versions of this Standard, receiving devices built to this version are expected to ignore all values appearing in currently reserved elements to avoid possible future failure to function as intended.

3.3 Acronyms and Abbreviations

The following acronyms and abbreviations are used within this document.

AMP Application Media Player
BMFF Base Media File Format
CDM Content Decryption Module
CENC Common Encryption
CID Content Identifier
DASH Dynamic Adaptive Streaming over HTTP
DRM Digital Rights Management
EME Encrypted Media Extensions
HEVC High Efficiency Video Codec
HTML Hypertext Markup Language
IEC International Electrotechnical Commission
IF Industry Forum
IOP InterOperability Points
ISO International Organization for Standardization
IV Initialization Vector
KID Key ID
MPD Media Presentation Description
MPEG Moving Picture Experts Group
NAL Network Abstraction Layer
RMP Receiver Media Player
ROUTE Real-Time Object Delivery over Unidirectional Transport
SHVC Scalable HEVC
SLT Service List Table
URI Uniform Resource Identifier
URL Uniform Resource Locator
URN Uniform Resource Name
UUID Universal Unique Identifier
VOD Video on Demand

W3C World Wide Web Consortium

3.4 Terms

The following terms are used within this document.

Broadcast Application – As defined in ATSC A/344 [9] **.**

reserved – Set aside for future use by a Standard.

4. SYSTEM OVERVIEW

This is a Recommended Practice that mainly addresses DRM operation covering ATSC3 ROUTE/DASH (A/331 [5]) and Interactive Content (A/344 [9]).

ATSC constrains DRM technology through DASH-IF IOP Security [7] (ROUTE/DASH), and A/331 [5] (MMT), which in turn relies on various ISO MPEG standards, notably including DASH [7] , MMT [12] , and Common Encryption [2] .

This section provides an overview of MPEG Common Encryption (CENC), W3C Encrypted Media Extensions (EME) and A/344 applicable APIs, which are essential to DRM operation on AMP and RMP respectively.

4.1 MPEG Common Encryption (CENC)

4.1.1 Introduction

Encryption, including the use of DRM systems, is addressed in DASH-IF IOP Security [7] (ROUTE/DASH) and A/331 [5] (MMT). This work is all based on MPEG Common Encryption defined in ISO 23001-7 [2] (“CENC”).

4.1.2 CENC Overview

The CENC protection scheme specifies encryption parameters that can be applied by a scrambling system, along with key mapping methods via common key identifier (KID). KIDs are for use by multiple DRM systems, such that the same encrypted version of a file can be decrypted by different DRM systems. The licensing and key acquisition is generally via proprietary information stored in metadata boxes of the ISO BMFF Segments – specifically, the Protection System Specific Header Box (‘`pssh`’).

The key advantage of CENC is that by providing a common way to encrypt content, it decouples the content encryption from the licensing and key acquisition and thus provides support for multiple DRM systems.

The CENC mechanism only encrypts media samples or parts thereof and leaves the ISO BMFF metadata such as the file and track structure boxes un-encrypted to enable players to recognize and read the file correctly and acquire any required license. CENC supports the encryption of NAL-based video encoding formats such as HEVC, thus offering sub-sample encryption capability, where only the video data of a sub-sample is encrypted, while the NAL header is not. This flexibility can be used to offer a free preview of the video, enable editing and processing of the video, or provide free access to some service components such as audio. By providing offsets to the encrypted byte ranges inside a sample in an ISO BMFF ‘mdat’ box, players can easily process the file and pass the encrypted chunks to the decryption module for decryption and playback.

For decryption to work, CENC provides the following information in the ISO BMFF (note that more commonly everything needed for decryption is (also) signaled in the MPD for ROUTE/DASH and in the security_properties_descriptor for MMT. See Section 5.1.2 or Section 5.2, respectively):

- Key Identifiers (KID): A key ID must be associated with every encrypted sample in a track even in the case that a single key is used for the whole track.
- Initialization Vectors (IV): The IV, a random number used to initialize an encryption function, is used for randomization and removal of semantics and is essential for strong protection. For every sample, the IV must be known in order to be able to construct the decryption key.
- License Acquisition Information: Information about license acquisition includes some common signaling and some aspects that are specific to each DRM system. The player needs to support at least one of the DRM systems that offer access to the encrypted tracks.

CENC defines a way to store the previous information in the ISO BMFF. The Key Identifiers may be provided:

- as opaque data in the DRM-specific content protection header, 'pssh',
- as the default_KID in the track encryption box "tenc', when a single key applies to the whole track,
- as content protection elements in the MPD (ROUTE/DASH) or the security_properties_descriptor (MMT), or
- as a key for a set of samples that share the same encryption key, provided in a sample grouping structure using the sample group description box 'sgpd'.

The IV for every sample is provided as part of the sample auxiliary information in the 'mdat' box or in the 'senc' box together with information about the position of the encrypted chunks.

The license acquisition information is provided as part of the protection system specific header box 'pssh', where each DRM system is identified by a SystemID. The 'pssh' box also provides a list of the provided Key Identifiers and opaque system-specific information that enables the acquisition of the keys identified by the supported key ids. For more information on license acquisition, see Section 5.

4.2 W3C Encrypted Media Extensions (EME) Overview

This section is specific to the Application Media Player (AMP) and does not necessarily apply to the Receiver Media Player (RMP). See A/344 [9] Section 7.

W3C Encrypted Media Extensions (EME) [3] specifies JavaScript APIs that enable a web application to facilitate the exchange of decryption keys. This is an interface between a device-resident DRM system agent, referred to as the Content Decryption Module (CDM), and a key source or license server located somewhere on the network. EME is based on the HTML5 Media Source Extensions (MSE) specification [4] , which enables adaptive bitrate streaming in HTML5 using DASH-IF IOP Security with MPEG-CENC (Common Encryption) [2] protected content. The architecture of EME is illustrated in Figure 4-1, which depicts the primary interactions of the EME workflow between the functional entities involved in the detection of encrypted content and the subsequent acquisition of license and key material, to enable content decryption and playout.

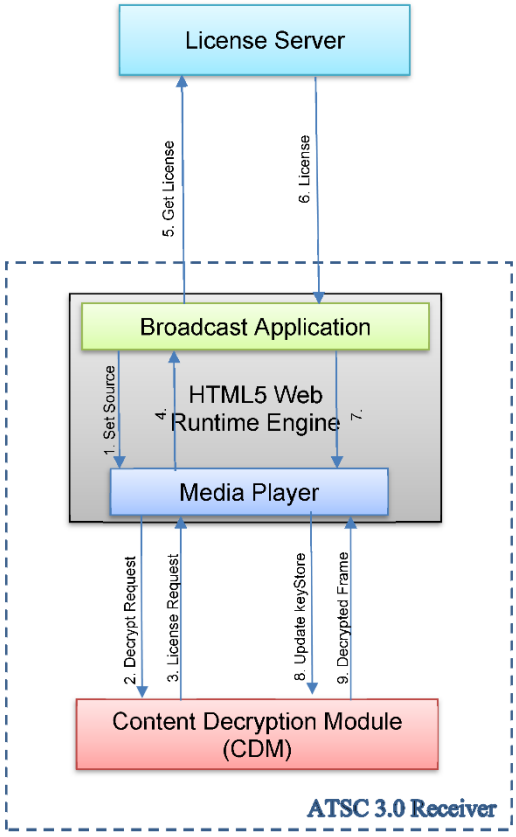

Figure 4-1 Encrypted Media Extensions workflow.

The principal objects in EME are MediaKeySession and MediaKeys. The Broadcast Application creates a MediaKeySession object, which represents the lifetime of a license and its key(s), by calling createSession() on the MediaKeys object. The Broadcast Application initiates the request for a license by passing the media data obtained in the encrypted event handler to the CDM. In turn, the CDM for the selected DRM system will generate a data blob (license request) and deliver it back to the app, which will then send that request to the license server. The returned license from the server is then passed by the Broadcast Application to the CDM, by using the update() method of the MediaKeySession. The CDM and/or the browser will use keys stored in the key session to decrypt media samples as they are encountered. The CDM may be either embedded in the web browser, or run in a trusted environment, depending on the required level of security, in passing the decrypted frames to a decoder.

4.3 A/344 Encrypted Media Extensions (EME) Overview

This section is specific to the Receiver Media Player (RMP) and does not necessarily apply to the Application Media Player (AMP). See A/344 [9] Section 7.

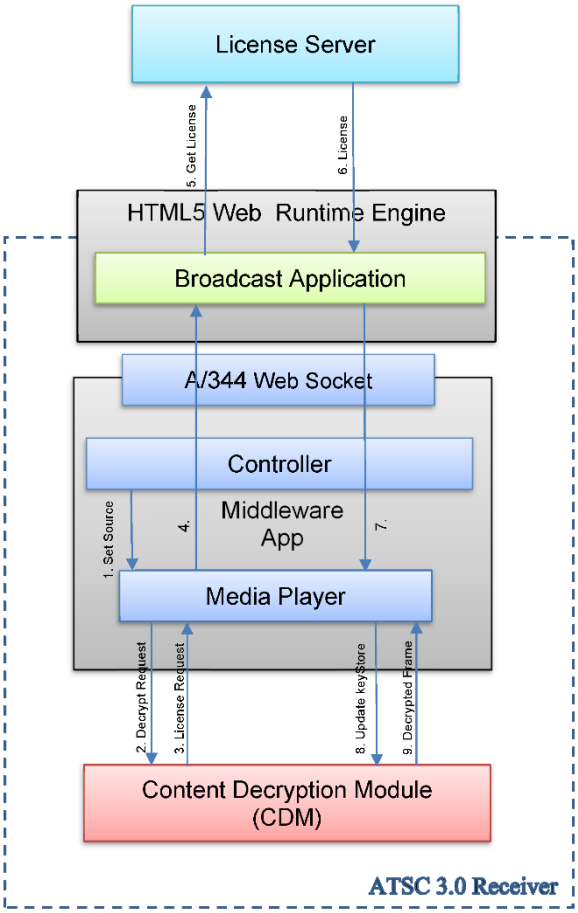

Figure 4-2 A/344 Web Socket workflow.

5. CLIENT PROCESSING FOR CENC AND DRM OPERATION

5.1 General Client Processing for CENC and DRM Operation

5.1.1 Introduction

This section describes the operation of a receiver when accessing CENC-protected media, independent of the transport protocol used for content delivery. The Common Encryption (CENC) framework for multiple DRM systems is used to protect ISO BMFF-formatted content. Protection system specific and proprietary signaling information delivered in two ways:

- in the signaling (MPD for ROUTE/DASH or security_properties_descriptor for MMT), or
- carried in-band in a protection system specific header box (‘pssh’).

The former is common today and is preferred; and the latter is required by CENC.

5.1.2 Basic CENC Operation in ROUTE/DASH and MMT

This section describes the basic mechanisms of how CENC-protected streaming content, protected by a DRM system and delivered by the ROUTE and MMT protocol, can be decrypted and played out. It describes, in the context of CENC and A/344 Web Socket APIs, the required interactions within the receiver and between the receiver and a Broadcast Application and/or a license server, for license and key acquisition and subsequent content decryption and playout.

In the first (see Section A.1), acquisition of the DRM license and content key by the CDM occurs during the program delivery via a Broadcast Application for a connected receiver.

In the second method (see Section A.2), acquisition of the DRM license and content key by the CDM occurs during the program delivery via the connected receiver.

In the third method (see Section A.3), acquisition of the DRM license and content key by the CDM occurs during the program delivery via an unconnected receiver.

In the fourth method (see Section A.4), acquisition of the DRM license and content key by the CDM occurs during the program delivery via a Broadcast Application for an unconnected receiver.

5.1.3 Solution Framework for DRM and CENC

ISO-IEC 23001-7 [2] represents the normative standard for common encryption in conjunction with ISO BMFF [10] , and includes the following technology components used for DRM protection of streaming media:

- Common encryption of NAL structure video and other media with one of the common encryption scheme types defined in A/360, Section 5.7.2 [8]
- Support for decryption of individual representations by one or more DRM systems (Note: Defined for ROUTE/DASH only)
- Key rotation to enable the change of the content encryption keys over time
- Signaling of `default_KID` and '`pssh`' parameters in the MPD or security_properties_descriptor

The primary DRM related signaling components and tools are as follows:

1) For ROUTE/DASH, the **`ContentProtection`** descriptor in the MPD which contains the URI for identifying the use of Common Encryption or specific DRM scheme(s) being used. For MMT, the system_UUID from the security_properties_descriptor.
2) Parameters of the '`tenc`' box, carried as part of protection scheme information in the movie box ('`moov`') of the Initialization Segment, which specify encryption parameters and `default_KID`. The `default_KID` information should also be carried out-of-band in the MPD or the security_properties_descriptor.
3) Signaling of common encryption sample auxiliary information in the form of initialization vectors and subsample encryption ranges, if applicable, using the 'senc' box as defined in ISO/IEC 23001-7 [2] , or via the `SampleAuxiliaryInformationSizesBox` ('`saiz`') and a `SampleAuxiliaryInformationOffsetsBox` ('`saio`').
4) '`pssh`' license acquisition data or keys for each DRM system in a format that is protection system specific. '`pssh`' may be stored in the Initialization Segment or in Media Segments. It should also be present in a **`cenc:pssh`** element in the MPD or the `pssh` element of the security_properties_descriptor. Note that while the presence of `pssh` information increases the payload size, it may allow faster parsing, earlier access, and addition of DRM systems without content modification.

Key rotation to enable modification over time in the entitlement for access to continuous live content. Details on how key rotation operates in the protection of broadcast DASH streaming content can be found in the DASH-IF IOP Security [7] . (Note: Defined for ROUTE/DASH only.)

5.2 MPD Support for Encryption and DRM Signaling

5.2.1 Overview

In addition to the ISO BMFF signaling of DRM-related items, these items are also commonly included in the MPD (and should be), which provides more timely delivery and faster decoding.

This starts with the **ContentProtection** element which is typically on the **AdaptationSet** element. There is one of these per DRM system, which may include generic CENC signaling as defined in CENC [2] . Within the **ContentProtection** element, there may be standardized license acquisition URL element (see DASH-IF IOP Security [7] **Laurl**); a mechanism for identifying the content to the DRM system, e.g. "content ID" (beyond **AssetIdentifier**); as well as DRM system-specific elements.

5.2.2 Use of the Content Protection Descriptor for mp4 Protection Scheme

As described in DASH-IF IOP, a **ContentProtection** descriptor with `@schemeIdUri` value of "`urn:mpeg:dash:mp4protection:2011`" indicates that the content is encrypted with the scheme as indicated in the `@value` attribute. The file structure of content protection schemes is specified in MPEG-DASH [7] , Section 5.8.5.2, and the `@value` is '`cenc`' in denoting the Common Encryption scheme. Such value for the `@schemeIdUri` of the **ContentProtection** descriptor along with `@cenc:default_KID` as defined within the "`urn:mpeg:cenc:2013`" extension namespace may be sufficient for the receiver to acquire a DRM license, or identify a previously acquired license that can be used to decrypt the Adaptation Set.

When the `@cenc:default_KID` is present for each Adaptation Set, it allows a player to determine if a new license needs to be acquired for each Adaptation Set by comparing their **default_KID**s with each other, and with the **default_KID**s of stored licenses. A player can simply compare these KID strings and determine what unique licenses are necessary without interpreting license information specific to each DRM system.

5.2.3 Use of Content Protection Descriptor for uuid Scheme

A UUID **ContentProtection** descriptor in the MPD may indicate the availability of a particular DRM scheme for license acquisition. An example is shown below:

```
<ContentProtection
schemeIdUri="urn:uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
value="DRMNAME version"/>
```

The `schemeIdUri` uses a UUID URN with the UUID string equal to the registered SystemID for a particular DRM system which can be found in the DASH identifier repository at: https://dashif.org/identifiers/content_protection/.

5.2.4 Protection System Specific Header Box in the MPD

A '`pssh`' box is defined by each DRM system for use with their registered SystemID, and is nominally stored in the movie box ('`moov`') and additionally may be present in the movie fragment box ('`moof`'). The same box should be stored in the MPD within a **ContentProtection** Descriptor for a UUID scheme using the extension element **cenc:pssh** in the "`urn:mpeg:cenc:2013`" namespace, as defined by ISO/IEC 23001-7 [2] . Carrying the **cenc:pssh** element and also the `cenc:default_KID` attribute as defined by the same "`urn:mpeg:cenc:2013`" extension namespace in the MPD can be useful in supporting key identification, license evaluation, and license retrieval before the availability of Initialization Segments for live content. This enables ATSC receivers, via the broadband network, to be able to acquire license requests prior to the start of the program. Also, spreading out license requests over time avoids potential overloading of the license server due to a high volume of simultaneous license requests from many viewers, starting when at an Initialization Segment containing license acquisition information in '`pssh`' becomes available.

With **cenc:default_KID** indicated in the mp4protection **ContentProtection** descriptor for each Adaptation Set, the DRM client in the receiver can determine whether:

- the associated decryption key for the program is available to the viewer (e.g., without purchase or subscription),
- the key has been downloaded, or
- which license the client may download before the `@availabilityStartTime` of the program, based on the `default_KID` of each Adaptation Set element selected.

5.2.5 Use of License Acquisition URL Descriptor in the MPD

A **Laurl** element may be added under the **ContentProtection** descriptor providing information for ATSC receivers to access the license server directly. An additional **licenseType** attribute is available defining the type of applicable license.

```
<dashif:Laurl licenseType="license-1.0">https://license-
server.com/license</dashif:Laurl>
```

ATSC receivers may POST the license request data from the CDM to the License Server to return a valid license. The returned license shall be passed to the CDM for the protected content to be decrypted by the given DRM System.

Additional **Laurl** elements may be added under the **ContentProtection** descriptor to provide more than one method to retrieve a valid license. ATSC3 receivers should use the **licenseType** to differentiate the different types.

- **license-1.0** Direct license acquisition by ATSC3 receivers and the URI scheme is a valid endpoint for access.\
  ```
  <dashif:Laurl licenseType="license-1.0">https://license-
  server.com/license</dashif:Laurl>
  ```
- **groupLicense-1.0** Provides a path for a group-based license. The URI may need to be parsed specific to that DRM System by the ATSC3 Receiver to access any local group licenses.
  ```
  <dashif:Laurl licenseType="groupLicense-1.0">file://groupLicense.lic
  </dashif:Laurl>
  ```
- **contentId-1.0** Provides information for the DRM specific content identifier used to generate the KIDs. The URI should be parsed to extract relevant information using REST based notation.
  ```
  <dashif:Laurl licenseType="contentId-
  1.0">file://contentId/superball2019</dashif:Laurl>
  ```

5.3 MMT Client Processing for CENC and DRM Operation

5.3.1 Overview

In addition to in-band ISO BMFF signaling of DRM-related items, these items should also be included in the security_properties_descriptor, providing a more-timely delivery and faster decoding. This starts with a loop for each Asset, and each Asset has an asset_id, scheme_code, and default_KID. Multiple DRM systems are supported for each Asset. For each DRM system, the system_UUID, license acquisition URLs, and protection system specific header information are provided.

5.3.2 MMT DRM Signaling

For MMT, protection system specific and proprietary signaling information is delivered as specified in ATSC A/331 [5] and ATSC A/360 [8] .

Similar to ROUTE/DASH DRM signaling, when default_KID is present for an Asset, it allows a player to determine if a new license needs to be acquired for each Asset by comparing their default_KID attributes with each other, and with the default_KID attributes of stored licenses. A player can simply compare these KID strings and determine what unique licenses are necessary without interpreting license information specific to each DRM system.

If license_info is present, a license type and an LA_URL for one license_info is added for ATSC receivers to POST the license request data from the CDM to the License Server to return a valid license.

If '`pssh`' box(s) are present for an Asset, they can be useful in supporting key identification, license evaluation, and license retrieval at the beginning of live content. This enables ATSC receivers, via the broadband network, to be able to acquire license requests prior to the start of the program.

6. USE OF A/344 APIS

This section describes how the DRM APIs defined in Section 9.15 of [9] are profiled to align with W3C EME APIs, however limiting the number of returned journeys. The formal syntax for the JSON notification, message, and response messages shown in examples below is described by the accompanying JSON schemas. In the event of any discrepancy between the syntax implied by the below examples and the accompanying JSON schemas, the syntax in the JSON schemas take precedence. Examples are shown as example JSON data structures.

The accompanying JSON schemas are located at https://www.atsc-schemas.org/atsc3.0/a362/20230220/

The key/value pairs used in the proprietary message structure should provide some minimum information that a Broadcast Application uses to manage the DRM-protected services.

```
<-- Notification

{
    "jsonrpc": "2.0",
    "method": "org.atsc.notify",
    "params": {
        "msgType": "DRM",
        "systemId": "urn:uuid:1077efec-c0b2-4d02-ace3-3c1e52e2fb4b",
        "service": "http://doi.org/10.5239/8A23-2B0",
        "message": [{"<system-specific-key>":"<value>"}]
    }
}

Application Request

--> {
    "jsonrpc": "2.0",
    "method": "org.atsc.drmOperation",
    "params": {
        "systemId": "urn:uuid:1077efec-c0b2-4d02-ace3-3c1e52e2fb4b",
        "service": "http://doi.org/10.5239/8A23-2B0",
        "message": [{"<system-specific-key>":"<value>"}]
```

```
    "id": 101
}
```

Below provides an example set of core pairs that are required to manage DRM message handling.

```
"message": [{
    "kid": "34e5db32-8625-47cd-ba06-68fca0655a72",
    "drmSessionId": "MzRlNWRiMzItODYyNS00N2NkLWJhMDYtNjhmY2EwNjU1YTcy",
    "drmMsgType": "<array_of_message_types>",
    "drmData": "base64-private-data",
}]
```

The `kid` pair should provide the information relating to the adaptation/Asset the notification relates to. Broadcast Applications may use this value to determine further track or Asset information from the MPD (ROUTE/DASH) or system_property_descriptor (MMT) and how the Broadcast Application should handle the request.

The `drmSessionId` is an optional pair that may be provided by the ATSC3 receiver, so reciprocal responses shall contain the same `drmSessionId` value in order to pair requests and responses together for the receiver.

The `drmMsgType` provides the type of message the `drmData` relates to and therefore how the Broadcast Application should react to the DRM Message Type. Further information on the different supported types is detailed in the sub-sections below but summarized here for convenience.

- `licenseRequest`
- `update`
- `generateRequest`
- `individualizationRequest`
- `individualizationResponse`
- `serverCertificateRequest`
- `serverCertificateResponse`

The `drmData` is a base64 encoded data format from the underlying DRM System. This provides the ability for the `drmData` to be transferred to license servers for processing.

Table 8.2 “JSON_RPC ATSC error Codes” of [9] defines a list of reserved error codes. The range of -32000 to -32099 is reserved for implementation-defined server errors and should be specific for the Content Protection system being used; however, there are some generic error codes that should be used. These are defined in Section 6.1.6.

6.1.1 Notifications for License Request

The DRM Notification with `drmMsgType` value of “licenseRequest” should be issued by the Receiver to the Broadcaster Application in order to request a license. The semantics for this notification should be as described in Table 6.1 and the syntax for this message is in the accompanying schema file org.atsc.notify-DRM.json. Additional semantic recommendations follow the table.

Table 6.1 License Request Semantics

Property Name	Use	Data Type	Short Description
jsonrpc	1	string	"2.0"
method	1	string	“org.atsc.notify”
params	1	object	
msgType	1	string	“DRM”
systemId	1	string	URI in the form “urn:uuid:…”
service	1	string	URI
message	1..n	object	
kid	1	string	
drmSessionId	1	string	
drmMsgType	1	string	“licenseRequest”
drmData	1	string	base64-encoded data

It is recommended that a single notification per **keyId** is raised for each **AdaptationSet** or Asset to the Media Player.

```
<-- Notification

{
    "jsonrpc": "2.0",
    "method": "org.atsc.notify",
    "params": {
        "msgType": "DRM",
        "systemId": "urn:uuid:1077efec-c0b2-4d02-ace3-3c1e52e2fb4b",
        "service": "http://doi.org/10.5239/8A23-2B0",
        "message": [{
            "kid": "34e5db32-8625-47cd-ba06-68fca0655a72",
            "drmSessionId": "MzRlNWRiMzItODYyNS00N2NkLWJhMDYtNjhmY2EwNjU1YTcy",
            "drmMsgType": "licenseRequest",
            "drmData": "……"
        }]
    }
}
```

One example is an MPD containing three adaptations or a system_property_descriptor containing three Assets, in this case, a Video and two Audio adaptations. The default Audio and Video is set to the Media Player, therefore two `licenseRequest` notifications are raised by the ATSC3 Receiver. If the second Audio is set to the Media Player, a new notification should be issued if it contains encrypted content, where the CDM does not find an applicable license.

Another example, for the ROUTE/DASH case, of when new notifications should be issued, is when an MPD period ends and a new period starts, or that the MPD changes and no licenses are found by the CDM.

ATSC3 receivers may hold a notification until a Broadcast Application has been loaded and subscribed to DRM notifications and then issue the notification.

6.1.2 Provide a License

A Broadcast Application should use the `licenseRequest` data to retrieve a valid license from the license server. The Broadcast Application may use the **`Laurl`** element where the **`licenseType`** has a value of **`contentId-1.0`**. The content identifier may be more useful to retrieve a license for all the supported adaptations or Assets, thus only making a single HTTPS call to the license service. The Broadcast Application should return the same license for each notification received if the `kid` is applicable to the same content identifier.

The Broadcast Application should use the `org.atsc.drmOperation` API and set the `drmMsgType` to `update` and place the base64 encoded license message into the `drmData` element.

The semantics for this request should be as described in Table 6.2 and the syntax for this message is in the accompanying schema file org.atsc.notify-DRM.json. Additional semantic recommendations follow the table.

Table 6.2 License Provision/Update Semantics

Property Name	Use	Data Type	Short Description
`jsonrpc`	1	string	"2.0"
`id`	1	integer	
`method`	1	string	"org.atsc.drmOperation"
`params`	1	object	
`systemId`	1	string	URI in the form "urn:uuid:…"
`service`	1	string	URI
`message`	1..n	object	
`kid`	1	string	
`drmSessionId`	1	string	
`drmMsgType`	1	string	"update"
`drmData`	1	string	base64-encoded data

The update response semantics should be as defined in Table 6.3 and the syntax defined in the accompanying schema file org.atsc.notify-DRM.json.

Table 6.3 Update Response Semantics

Property Name	Use	Data Type	Short Description
`jsonrpc`	1	string	"2.0"
`id`	1	integer	Matches the request id value
`result`	one of X	object	Empty, returned on successful request otherwise the error structure is returned
`error`	oneOf X		See Section 6.1.6

Examples:

```
--> Application Request

{
    "jsonrpc": "2.0",
    "method": "org.atsc.drmOperation",
    "params": {
```

```
        "systemId": "urn:uuid:1077efec-c0b2-4d02-ace3-3c1e52e2fb4b",
        "service": "http://doi.org/10.5239/8A23-2B0",
        "message": [{
            "kid": "34e5db32-8625-47cd-ba06-68fca0655a72",
            "drmSessionId": "MzRlNWRiMzItODYyNS00N2NkLWJhMDYtNjhmY2EwNju1YTcy",
            "drmMsgType": "update",
            "drmData": "<base64-encoded-licenseMessage>"
        }]
    },
    "id": 100
}

<-- Successful Response

{
    "jsonrpc": "2.0",
    "result": {},
    "id": 100
}

<-- unsuccessful Response

{
    "jsonrpc": "2.0",
    "error": {
        "code": 27,
        "message": "No Receiver capability"
    },
    "id": 100
}
```

6.1.3 Generate a License Request

There are use cases where additional **`AdaptationSets`** require a different license. These examples may require additional authorization. A Broadcast Application should read the MPD (ROUTE/DASH) or system_property_descriptor (MMT)and extract the PSSH of the required track and request to the ATSC receiver to generate request. The response should be no different than that of a normal notification.

The semantics for this API should be as described in Table 6.4 and the syntax for this message is in the accompanying schema file org.atsc.notify-DRM.json. Additional semantic recommendations follow the table.

Table 6.4 Generate Request Semantics

Property Name	Use	Data Type	Short Description
jsonrpc	1	stri“g	"2.0"
id	1	integer	
method	1	string	“org.atsc.drmOperation”
params	1	object	
systemId	1	string	URI in the form “urn:uuid:…”
service	1	string	URI
message	1..n	object	
kid	1	string	

		drmMsgType	1	string	“generateRequest”
		drmData	1	string	base64-encoded data

The update response semantics should be as defined in Table 6.5 and the syntax defined in the accompanying schema file org.atsc.notify-DRM.json.

Table 6.5 Generate Request Response Semantics

Property Name	Use	Data Type	Short Description
jsonrpc	1	string	"2.0"
id	1	integer	
result	oneOf X		Returned on successful request otherwise the error structure is returned
message	1	object	
kid	1	string	
drmSessionId	1	string	
drmMsgType	1	string	“licenseRequest”
drmData	1	string	base64-encoded license data
error	oneOf X		See Section 6.1.6

Examples:

```
--> Application Request

{
    "jsonrpc": "2.0",
    "method": "org.atsc.drmOperation",
    "params": {
        "systemId": "urn:uuid:1077efec-c0b2-4d02-ace3-3c1e52e2fb4b",
        "service": "http://doi.org/10.5239/8A23-2B0",
        "message": [{
            "kid": "34e5db32-8625-47cd-ba06-68fca0655a72",
            "drmMsgType": "generateRequest",
            "drmData": "<base64-encoded-psshData>"
        }]
    },
    "id": 101
}

<-- Response

{
    "jsonrpc": "2.0",
    "result": {
        "message": [{
            "kid": "34e5db32-8625-47cd-ba06-68fca0655a72",
            "drmSessionId": "MzRlNWRiMzItODYyNS00N2NkLWJhMDYtNjhmY2EwNjU1YTcy",
            "drmMsgType": "licenseRequest",
            "drmData": "……"
        }]
   },
   "id": 101
}
```

6.1.4 Notifications for Provisioning Request

Most DRM systems require unique device provisioning or certificates. For some systems, this is carried out in the factory or online. If device provisioning is required, the `individualizationRequest` notification is sent by the ATSC3 receiver, and the `individualizationResponse` request is sent by the application to provide the response.

The semantics for the `individualizationRequest` notification should be as described in Table 6.6 and the syntax for this message is in the accompanying schema file org.atsc.notify-DRM.json. Additional semantic recommendations follow the table.

Table 6.6 Generate Request Semantics

Property Name	Use	Data Type	Short Description
jsonrpc	1	stri“g	"2.0"
id	1	integer	
method	1	string	“org.atsc.drmOperation”
params	1	object	
systemId	1	string	URI in the form “urn:uuid:…”
service	1	string	URI
message	1..n	object	
kid	1	string	
drmMsgType	1	string	“generateRequest”
drmData	1	string	base64-encoded data

The example below shows how a provisioning request notification may be sent to the Broadcast Application for the ATSC3 receiver to be provisioned.

```
<-- Notification

{
    "jsonrpc": "2.0",
    "method": "org.atsc.notify",
    "params": {
        "msgType": "DRM",
        "systemId": "urn:uuid:1077efec-c0b2-4d02-ace3-3c1e52e2fb4b",
        "service": "http://doi.org/10.5239/8A23-2B0",
        "message": [{
            "drmMsgType": "individualizationRequest",
            "drmData": "<base64-endoded-individualization-data-request"
        }]
    }
}
```

When a provisioning request is sent to a server, the server should respond with a unique provisioning response. The response should be posted back to the ATSC3 receiver as shown below to complete the process.

The license request response semantics should be as defined in Table 6.7 and the syntax defined in the accompanying schema file org.atsc.drmOperation-simple-response.json.

Table 6.7 Generate Request Response Semantics

Property Name	Use	Data Type	Short Description
jsonrpc	1	string	"2.0"
id	1	integer	Matches the request id value
result	one of X	object	Empty, returned on successful request otherwise the error structure is returned
error	oneOf X		See Section 6.1.6

```
Application Method

--> {
    "jsonrpc": "2.0",
    "method": "org.atsc.drmOperation",
    "params": {
        "systemId": "urn:uuid:1077efec-c0b2-4d02-ace3-3c1e52e2fb4b",
        "service": "http://doi.org/10.5239/8A23-2B0",
        "message": [{
            "drmMsgType": "individualizationResponse",
            "drmData": "<base64-encoded-provisioning-response>"
        }]
    },
    "id": 101
}

<-- Response

{
    "jsonrpc": "2.0",
    "result": {},
    "id": 101
}
```

An ATSC3 receiver once provisioned should either raise a `serverCertificateRequest` or a `licenseRequest` notification to continue the process that triggered the provisioning request in the first place.

6.1.5 Notifications for Server Certificate Request

An additional layer that a receiver may support is server certificate as defined in [3] that provides an additional layer of protection between the ATSC3 Receiver and the CDM. If supported by a receiver, then the method outlined below should apply.

The server certificate request semantics should be as defined in Table 6.8 and the syntax defined in the accompanying schema file org.atsc.drmOperation-request.json.

Table 6.8 Server Certificate Request Notification

Property Name	Use	Data Type	Short Description
jsonrpc	1	string	"2.0"
method	1	string	"org.atsc.drmOperation"
params	1	object	
msgType	1	string	"DRM"
systemId	1	string	URI in the form "urn:uuid:…"
service	1	string	URI

	message	1..n	object	
	drmMsgType	1	string	“serverCertificateRequest”
	drmData	1	string	base64-encoded data

```
<-- Notification

{
    "jsonrpc": "2.0",
    "method": "org.atsc.notify",
    "params": {
        "msgType": "DRM",
        "systemId": "urn:uuid:1077efec-c0b2-4d02-ace3-3c1e52e2fb4b",
        "service": "http://doi.org/10.5239/8A23-2B0",
        "message": [{
            "drmMsgType": "serverCertificateRequest",
            "drmData": "<base64-endoded-serverCertficate-data-request"
        }]
    }
}
```

The **serverCertificate** request sent to a server, should respond with a **serverCertficiate** response. The response should be posted back to the ATSC3 receiver as shown below to complete the process.

The license request response semantics should be as defined in Table 6.9 and the syntax defined in the accompanying schema file org.atsc.notify-DRM.json.

Table 6.9 Server Certificate Response Request

Property Name	Use	Data Type	Short Description
jsonrpc	1	stri“g	"2.0"
id	1	integer	
method	1	string	“org.atsc.drmOperation”
params	1	object	
systemId	1	string	URI in the form “urn:uuid:…”
service	1	string	URI
message	1..n	object	
drmMsgType	1	string	“serverCertificateResponse”
drmData	1	string	base64-encoded data

```
Application Method

--> {
    "jsonrpc": "2.0",
    "method": "org.atsc.drmOperation",
    "params": {
        "systemId": "urn:uuid:1077efec-c0b2-4d02-ace3-3c1e52e2fb4b",
        "service": "http://doi.org/10.5239/8A23-2B0",
        "message": [{
            "drmMsgType": "serverCertificateResponse",
            "drmData": "<base64-encoded-serverCertificate-response>"
        }]
```

```
    },
    "id": 101
}

<-- Response

{
    "jsonrpc": "2.0",
    "result": {},
    "id": 101
}
```

6.1.6 Error Codes

See A/344 Section 8.3.3, A/344 Table 8.2 JSON-RPC ATSC Error Codes [9] .

Annex A: License Acquisitions Examples

A.1 LICENSE ACQUISITION – BROADCAST APPLICATION, CONNECTED

Figure A-1 is an example message flow illustrating the method whereby the `ContentProtection` descriptor in the MPD or the security_property_descriptor in MA3 messages is used to provide the affiliated metadata, such as the default KID to the CDM. This triggers the CDM to request and obtain the DRM license, and associated keys material.

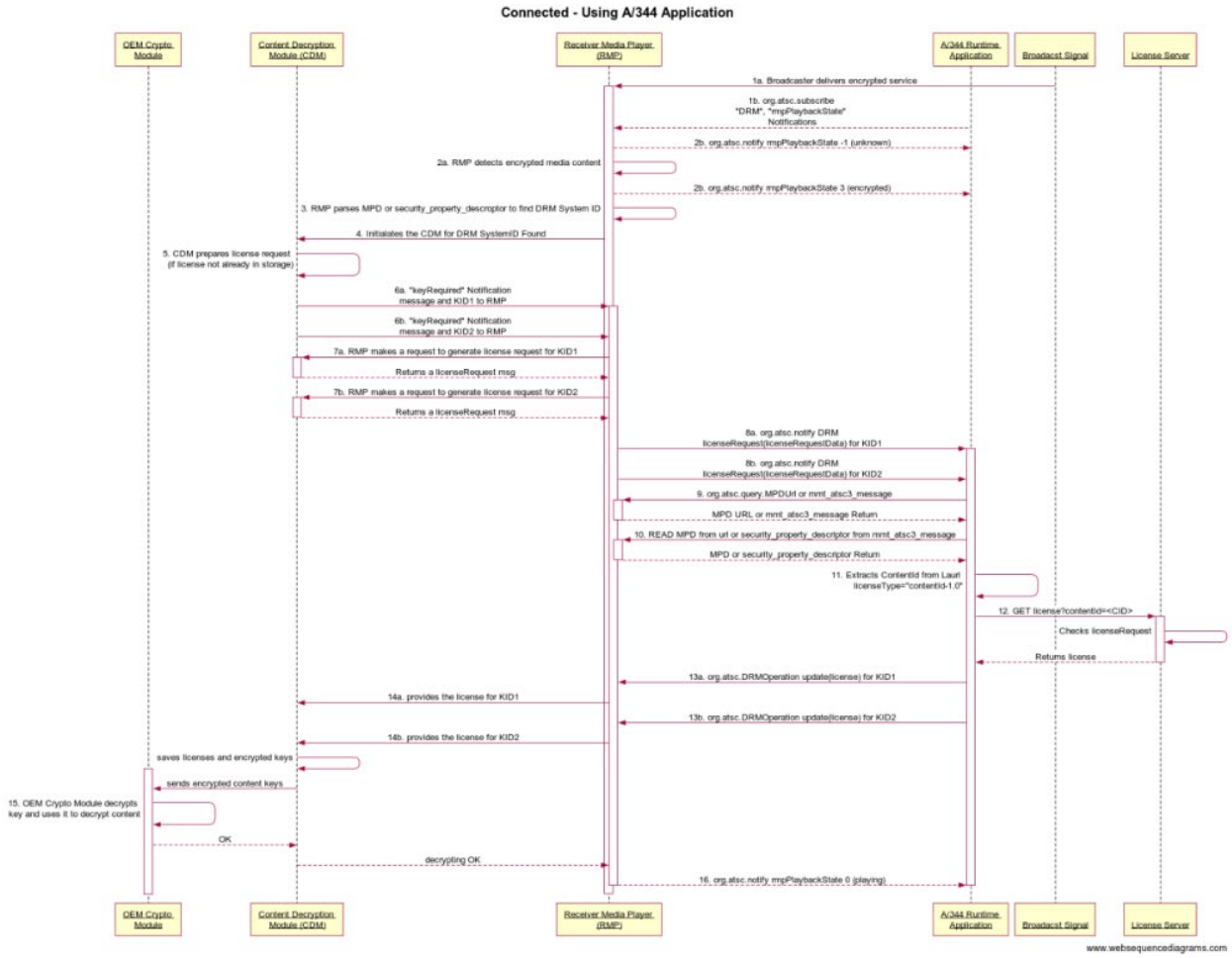

Figure A-1 License acquisition – Broadcast Application, connected.

1) A user selects a Service with encrypted audio and video. A Broadcast Application is also present in the definition of the Service, and the Receiver downloads and executes it when the package containing it has been retrieved and validated.
2) The RMP discovers the service is encrypted and sets rmpPlaybackStateChange to **3**.

3) The RMP parses the MPD in the Service Layer Signaling (SLS), specifically the `ContentProtection` element, or security_property_descriptor in MA3 messages to discover whether or not it lists a DRM System ID that the RMP supports. If one is found, then processing continues.
4) The RMP initiates the relevant CDM for the given DRM System ID.
5) The Content Decryption Module (CDM) in the Receiver associated with the DRM System ID checks to see whether the license key required to decrypt the video and audio is already in storage. In this example, the CDM determines that no pre-existing license for this content is available and prepares a license request for the key.
6) The CDM notifies the RMP that a key is required for the KID1 and KID2.
7) The RMP player needs a valid license request message from the CDM for both KID1 and KID2 to forward on to the Broadcast Application.
8) The RMP notifies the Broadcast Application using the DRM Notification API, passing the DRM system ID and a message understood by Broadcast Applications which support that system ID. The message in the notification would typically include the Key ID associated with this content, and a request for a license.
9) The Broadcast Application requires the content identifier associated with the KID1 and KID2 contained in the MPD or security_property_descriptor. The Broadcast Application uses the org.atsc.query.MPDUrl to retrieve the location of the MPD or the org.atsc.query.mmt_atsc3_message to retrieve the security_property_descriptor.
10) The Broadcast Application reads the MPD from the URL provided by the receiver or the security_property_descriptor from the MA3 messages provided by the receiver.
11) The Broadcast Application locates the elements containing the corresponding KID1 and reads the licenseType=contentId-1.0 value from the `Laurl` element.

```
<dashif:Laurl licenseType=”contentId-1.0”>
file://contentId/superball2019
</dashif:Laurl>
```

12) The Broadcast Application executes a process to retrieve a license for the requested content. This process involves interaction with the user, and interaction with a Web Server operated by the broadcaster and a License Server associated with the DRM system employed using the contentId. Interaction between the Broadcast Application and the License Server includes information the Broadcast Application needs to create a user interface to present to the user. The License Server determines that the user is entitled to access the requested content. The License Server delivers the license to the Broadcast Application using proprietary messaging.
13) The Broadcast Application issues the DRM Operation API with the DRM system ID and a message including the license for the content for each KID notification received by the RMP.
14) The RMP issues the license to the CDM for each of the KIDs.
15) The CDM receives the license, saves it, and uses it to derive the key needed to decrypt the content, sending that key to the OEM Crypto Module.
16) The OEM Crypto Module decrypts the enhancement layer and the user enjoys watching the service. The RMP player notifies the Broadcast Application with rmpPlaybackStateChange set to **0**.

A.2 LICENSE ACQUISITION – RECEIVER APPLICATION, CONNECTED

Figure A-2 is an example message flow illustrating the receiver retrieving the license from the license server directly using the details contained in the `Laurl`.

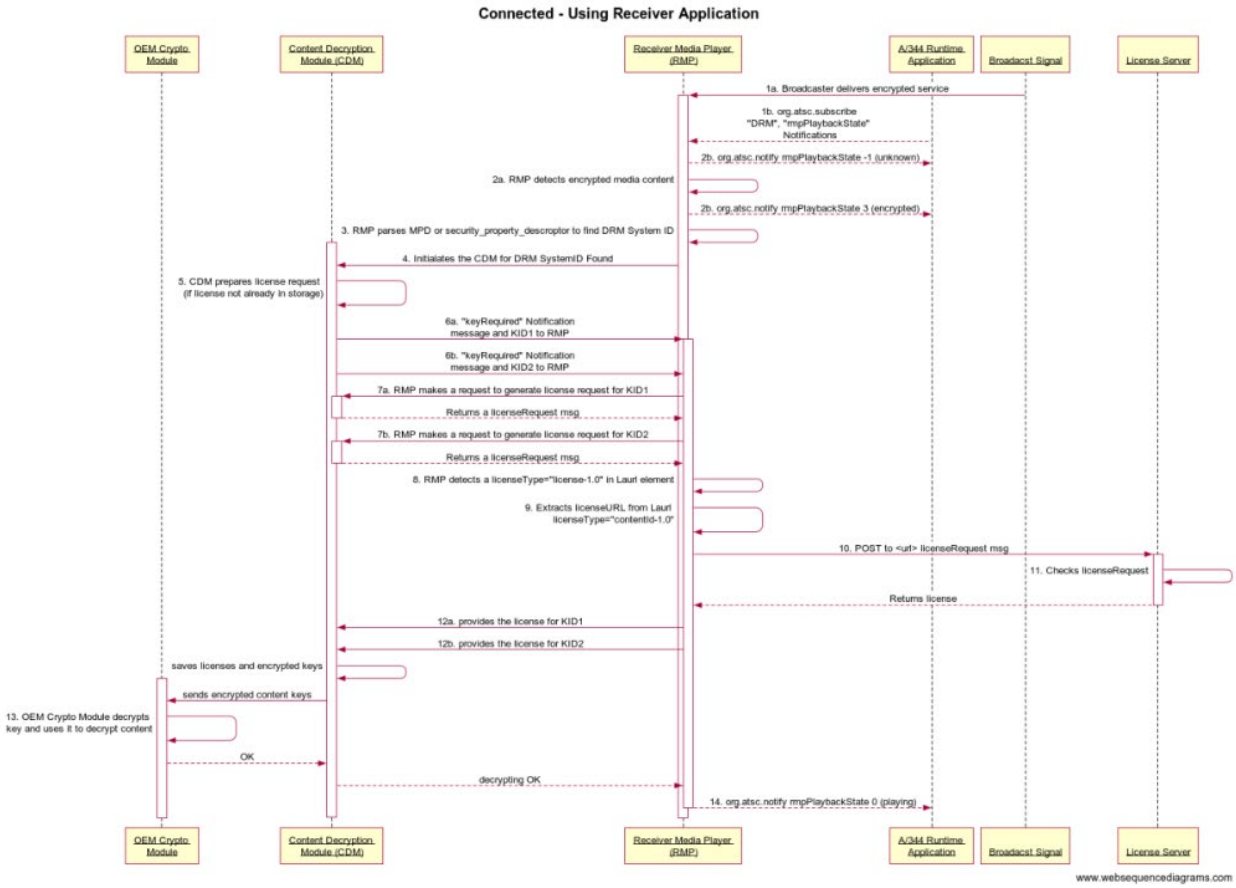

Figure A-2 License acquisition - Receiver Application, connected.

1) A user selects a Service with encrypted audio and video. A Broadcast Application is also present in the definition of the Service, and the Receiver downloads and executes it when the package containing it has been retrieved and validated.
2) The RMP discovers the service is encrypted and the RMP sets rmpPlaybackStateChange to **3**.
3) The RMP parses the MPD in the Service Layer Signaling (SLS), specifically the **`ContentProtection`** element, or the security_property_descriptor in MA3 messages to discover whether it lists a DRM System ID that the RMP supports. If one is found, then processing continues.
4) The RMP initiates the relevant CDM for the given DRM System ID.
5) The Content Decryption Module (CDM) in the Receiver associated with the DRM System ID checks to see whether the license key required to decrypt the video and audio is already in storage. In this example, the CDM determines that no pre-existing license for this content is available and prepares a license request for the key.
6) The CDM notifies the RMP that a key is required for the KID1 and KID2.

7) The RMP player needs a valid license request message from the CDM for both KID1 and KID2 to forward onto the Broadcast Application.
8) The RMP detects that the `licenseType`="license-1.0" is available and recognizes that a license can be fetched directly by the receiver.
9) The RMP extracts the URL value from the element.

```
<dashif:Laurl licenseType="license-1.0">
https://server.com:9873/getlicense?contentid=12345&amp;token=abcd
</dashif:Laurl>
```

10) The Broadcast Application executes a process to retrieve a license for the requested content. This process does not involve interaction with the user.
11) The License Server determines if the request is authentic. The License Server delivers the license to the Broadcast Application using proprietary messaging.
12) The RMP issues the license to the CDM for each of the KIDs.
13) The CDM receives the license, saves it, and uses it to derive the key needed to decrypt the content, sending that key to the OEM Crypto Module.
14) The OEM Crypto Module decrypts the enhancement layer and the user enjoys watching the service. The RMP player notifies the Broadcast Application with rmpPlaybackStateChange set to **0**.

A.3 LICENSE ACQUISITION – RECEIVER APPLICATION, UNCONNECTED

Figure A-3 is an example message flow illustrating the receiver retrieving the license locally on a receiver that is unconnected to the internet.

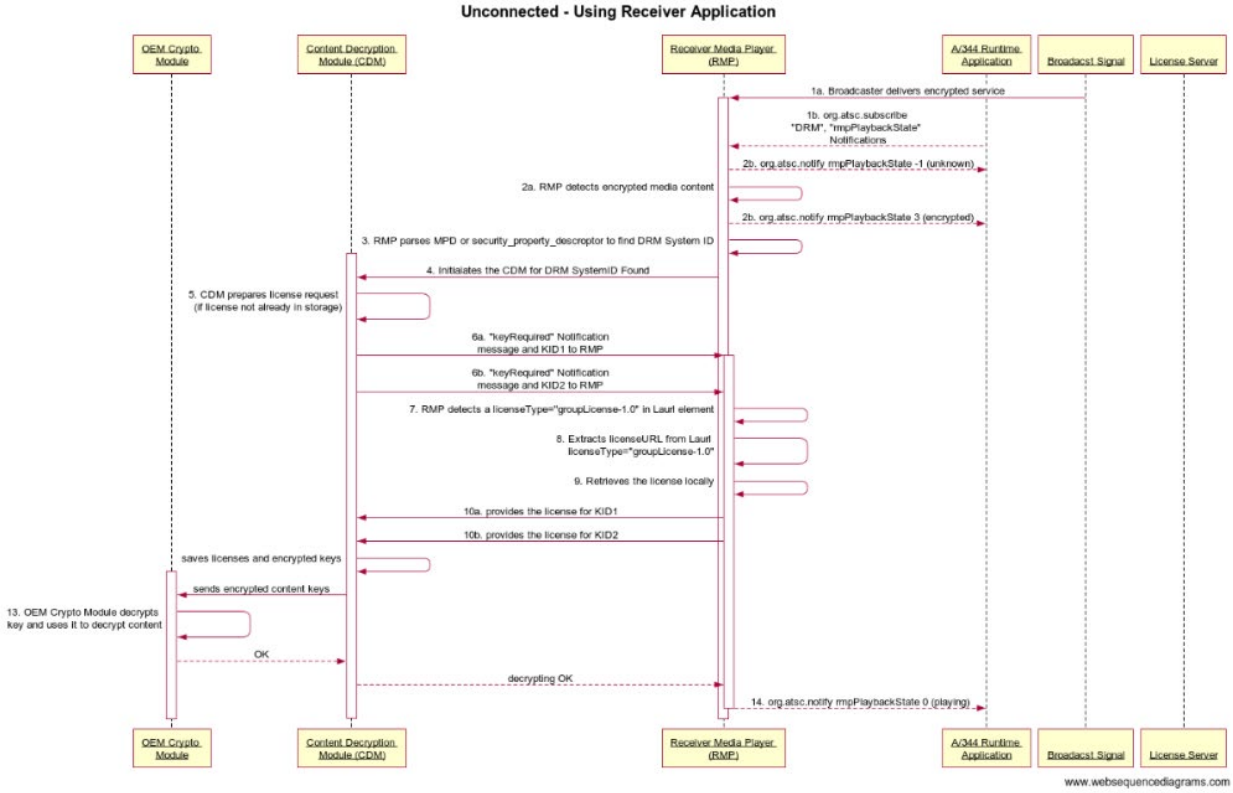

Figure A-3 License acquisition – Receiver Application, unconnected**.**

1) A user selects a Service with encrypted audio and video. A Broadcast Application is also present in the definition of the Service, and the Receiver downloads and executes it when the package containing it has been retrieved and validated.
2) The RMP sets rmpPlaybackStateChange to **-1**. The RMP discovers the service is encrypted and sets rmpPlaybackStateChange to **3**.
3) The RMP parses the MPD in the Service Layer Signaling (SLS), specifically the `ContentProtection` element, or the security_property_descriptor in MA3 messages to discover whether or not it lists a DRM System ID that the RMP supports. If one is found, then processing continues.
4) The RMP initiates the relevant CDM for the given DRM System ID.
5) The Content Decryption Module (CDM) in the Receiver associated with the DRM System ID checks to see whether the license key required to decrypt the video and audio is already in storage. In this example, the CDM determines that no pre-existing license for this content is available and prepares a license request for the key.
6) The CDM notifies the RMP that a key is required for the KID1 and KID2.
7) The RMP detects that the licenseType="groupLicense-1.0" is available and recognizes that a license can be fetched locally by the receiver.

8) The RMP extracts the URL value from the element.

```
<dashif:Laurl licenseType=”groupLicense-1.0”>
File://atsc3-license.lic
</dashif:Laurl>
```

9) The Broadcast Application locates the license locally. This process does not involve interaction with the user.
10) The RMP issues the license to the CDM for each of the KIDs.
11) The CDM receives the license, saves it, and uses it to derive the key needed to decrypt the content, sending that key to the OEM Crypto Module.
12) The OEM Crypto Module decrypts the enhancement layer and the user enjoys watching the service. The RMP player notifies the Broadcast Application with rmpPlaybackStateChange set to **0**.

A.4 LICENSE ACQUISITION – BROADCAST APPLICATION, UNCONNECTED

Figure A-4 is an example message flow illustrating the Broadcast Application attempting to retrieve the license from the license server directly using the details contained, however the receiver is not connected to the internet.

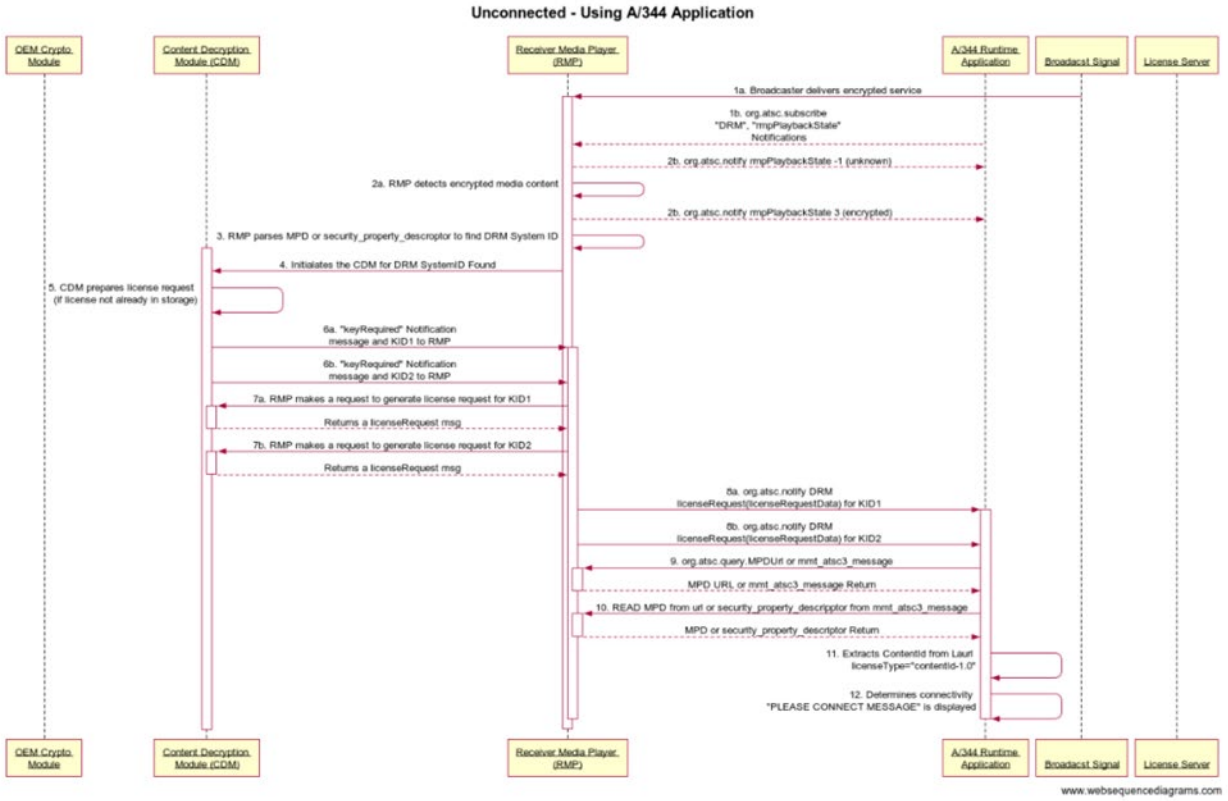

Figure A-4 License acquisition – Broadcast Application, unconnected.

1) A user selects a Service with encrypted audio and video. A Broadcast Application is also present in the definition of the Service, and the Receiver downloads and executes it when the package containing it has been retrieved and validated.
2) The RMP sets rmpPlaybackStateChange to **-1**. The RMP discovers the service is encrypted and sets rmpPlaybackStateChange to **3**.
3) The RMP parses the MPD in the Service Layer Signaling (SLS), specifically the `ContentProtection` element, or the security_property_descriptor in MA3 messages to discover whether or not it lists a DRM System ID that the RMP supports. If one is found, then processing continues.
4) The RMP initiates the relevant CDM for the given DRM System ID.
5) The Content Decryption Module (CDM) in the Receiver associated with the DRM System ID checks to see whether the license key required to decrypt the video and audio is already in storage. In this example, the CDM determines that no pre-existing license for this content is available and prepares a license request for the key.
6) The CDM notifies the RMP that a key is required for the KID1 and KID2.

7) The RMP player needs a valid license request message from the CDM for both KID1 and KID2 to forward onto the Broadcast Application.
8) The RMP notifies the Broadcast Application using the DRM Notification API, passing the DRM system ID and a message understood by Broadcast Applications which support that system ID. The message in the notification would typically include the Key ID associated with this content, and a request for a license.
9) The Broadcast Application requires the content identifier associated with the KID1 and KID2 contained in the MPD or security_property_descriptor. The Broadcast Application uses the org.atsc.query.MPDUrl to retrieve the location of the MPD or the org.atsc.query.mmt_atsc3_message to retrieve the security_property_descriptor.
10) The Broadcast Application reads the MPD from the url provided by the receiver or the security_property_descriptor from MA3 messages provided by the receiver.
11) The Broadcast Application locates the elements containing the corresponding KID1 and reads the licenseType=contentId-1.0 value from the Laurl element.

```
<dashif:Laurl licenseType=”contentId-1.0”>
file://contentId/superball2019
</dashif:Laurl>
```

12) The Broadcast Application determines it is not connected to the Internet. The Broadcast Application provides an additional message on why to connect to the internet in order to receive the provided service. The rmpPlaybackState remains with the value of **3**.

Annex B: Signaling Examples

B.1 SERVICE IS "PROTECTED"

The SLT table of a given service shall have the protection attribute set to TRUE (see green highlight below). This provides an indication on whether a service has components which are encrypted either all or part of the time.

```
<?xml version="1.0" encoding="UTF-8"?>
<SLT
xmlns="tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/SLT/1.0/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/SLT/1.0/ SLT-1.0-
20180228.xsd"
bsid="369"> <!-- 0x0171 -->
   <Service protected="true" majorChannelNo="14" minorChannelNo="5" serviceId="5"
globalServiceID="https://doi.org/10.5239/8A23-2B0B"              sltSvcSeqNum="0"
serviceCategory="1" shortServiceName="KPBS-DT">
        <BroadcastSvcSignaling                        slsSourceIpAddress="10.1.1.1"
slsDestinationUdpPort="8100"                                    slsProtocol="1"
slsDestinationIpAddress="239.255.14.5"/>
   </Service>
</SLT>
```

– End of Document –