ATSC Recommended Practice: A/362:2023-03 Amendment No. 1, “DRM Operation for MMT”

Doc. A362-2023-03 Amend. No. 1
14 February 2024

Advanced Television Systems Committee
1300 I Street, N.W., Suite 400E
Washington, D.C. 20005
202-872-9160

Revision History

Version	Date
Amendment approved	14 February 2024

ATSC Recommended Practice: A/362:2023-03 Amendment No. 1, “DRM Operation for MMT”

1. OVERVIEW

1.1 Definition

An Amendment is generated to document an enhancement, an addition or a deletion of functionality to previously agreed technical provisions in an existing ATSC document. Amendments shall be published as attachments to the original ATSC document. Distribution by ATSC of existing documents shall include any approved Amendments.

1.2 Scope

This document describes changes to A/362:2023-03 that describe recommended practices for Digital Rights Management (DRM) for use with content delivery based on the MMT protocol. This amendment is in response to New Project Proposal N-054r0, “Modify A/362 to clarify DRM (Digital Rights Management) operation for MMT.”

1.3 Rationale for Changes

The changes described in this document are being proposed in order to describe recommended practices for Digital Rights Management (DRM) for use with content delivery based on the MMT protocol. ATSC A/362:2023-03 considered DRM for use with content delivery based on the ROUTE/DASH protocol, but not MMT.

1.4 Compatibility Considerations

By nature, updates to Recommended Practices are backward-compatible, given that they cannot include normative requirements. Normative requirements for DRM for use with content delivery based on the MMT protocol are specified in ATSC A/331 and A/360.

2. LIST OF CHANGES

Change instructions are given below in *italics*. Unless otherwise noted, inserted text, tables, and drawings are shown in blue; deletions of existing text are shown in ~~red strikeout~~. The text “[ref]” indicates that a cross reference to a cited referenced document should be inserted.

2.1 Informative References

Update the following informative references.

[5] ATSC: “ATSC Standard: Signaling, Delivery, Synchronization and Error Protection,” Doc. A/331:2023-10~~03~~, Advanced Television System Committee, Washington, DC, ~~28 February~~4 October 2023.

[8] ATSC: “ATSC 3.0 Security and Service Protection,” Doc. A/360:2023-08~~03~~, Advanced Television System Committee, Washington, DC, ~~28 March~~15 August 2023.

[9] ATSC: “ATSC 3.0 Interactive Content,” Doc. A/344:2023-05~~03~~, Advanced Television System Committee, Washington, DC, ~~28 March~~19 May 2023.

Add the following informative reference.

[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.

2.2 Acronyms and Abbreviations

No changes.

2.3 Terms

No changes.

2.4 Change Instructions

Apply the changes as instructed below.

Modify Section 1.2 as follows:

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 – ~~ROUTE/DASH c~~Client processing
- Section 6 – Use of A/344 APIs
- Annex A – Example message flows
- Annex B – Example SLT

Modify the top level of Section 4 as follows:

4 SYSTEM OVERVIEW

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

ATSC constrains DRM technology through ~~the~~ 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]. ~~References are generally made to the most constrained document (DASH-IF). However, readers will need all the above documents to understand this RP.~~

Modify Section 4.1.1 as follows:

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").

Modify Section 4.1.2 as follows:

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 5.2~~3~~ or Section 5.3, respectively.):

…

- as content protection elements in the MPD (ROUTE/DASH) or the security_properties_descriptor (MMT), or

Modify the title of Section 5 as follows:

5 ~~ROUTE/DASH~~ CLIENT PROCESSING FOR CENC AND DRM OPERATION ~~API~~

Modify Section 5.1 as follows:

5.1 ~~Introduction~~ General Client Processing for CENC and DRM Operation

~~This section describes the operation of a ROUTE/DASH receiver when accessing CENC-protected media. ROUTE/DASH supports the Common Encryption (CENC) framework for multiple DRM systems to protect DASH ISO BMFF-formatted content. ROUTE/DASH includes protection system specific and proprietary signaling information delivered in two ways:~~

- ~~in the MPD, or~~
- ~~carried in-band in designated metadata boxes of the ISO BMFF Segments.~~

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

Move the previous Section 5.1 to Section 5.1.1 with the following modifications:

5.1.1 Introduction

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

- in the ~~MPD~~signaling (MPD for ROUTE/DASH or security_properties_descriptor for MMT), or
- carried in-band in a protection system specific header box (‘pssh’)~~designated metadata boxes of the ISO BMFF Segments~~.

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

Move the original Section 5.2 to Section 5.1.2 as follows:

5.2.1 Basic CENC Operation in ROUTE/DASH and MMT

This section describes the basic mechanisms of how CENC-protected ~~DASH-formatted~~ 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 the A/344 [9] 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 method (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.

Add a new Section 5.1.3 as follows. It is modified from the original Section 5.4 to make it general for ROUTE-DASH and for MMT.

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~~ carried by ROUTE/DASH~~:

- Common encryption of NAL structure video and other media with ~~AES-128 CTR mode~~ 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 ~~Extension of the ContentProtection descriptor to enable the signaling~~ of `default_KID` and ‘`pssh`’ parameters in the MPD or security_properties_descriptor

The primary DRM related signaling components and tools ~~available for use in ROUTE/DASH~~ are as follows:

1) For ROUTE/DASH, ~~T~~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 ~~**`cenc:`**~~`pssh` information ~~in the MPD~~ increases the ~~MPD~~ payload size, it may allow faster parsing, earlier access, and addition of DRM systems without content modification.

5) 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.)

Move the original Section 5.3, including all of its content, to Section 5.2.

Add a new Section 5.3 as follows. Note that the text largely follows Section 5.2, except it is for MMT, rather than ROUTE/DASH.

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.

Add Section 6 as follows. Note that the text is from Section 5.5 with changes to the original content shown in blue and ~~red strikeout~~:

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~~5.5.6~~.

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 5.1 and the syntax for this message is in the accompanying schema file org.atsc.notify-DRM.json. Additional semantic recommendations follow the table.

Table 5.1 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:~~uid~~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`** ~~set~~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 5.2 and the syntax for this message is in the accompanying schema file org.atsc.notify-DRM.json. Additional semantic recommendations follow the table.

Table 5.2 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:~~uud~~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 5.3 and the syntax defined in the accompanying schema file org.atsc.notify-DRM.json.

Table 5.3 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~~5.5.6~~

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~~5~~-encoded-licenseMessage>"
        }]
    },
    "id": 100
}

<-- Successful Response

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

<-- unsuccessful Response

{
    "jsonrpc": "2.0",
    "error": {
        "code":~~ -TBD~~-27,
        "message": "~~license~~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 5.4 and the syntax for this message is in the accompanying schema file org.atsc.notify-DRM.json. Additional semantic recommendations follow the table.

Table 5.4 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:~~uud~~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 5.5 and the syntax defined in the accompanying schema file org.atsc.notify-DRM.json.

Table 5.5 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~~5.5.6~~

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 5.6 and the syntax for this message is in the accompanying schema file org.atsc.notify-DRM.json. Additional semantic recommendations follow the table.

Table 5.6 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:~~uud~~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 5.7 and the syntax defined in the accompanying schema file org.atsc.drmOperation-simple-response.json.

Table 5.7 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~~5.5.6~~

```
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 5.8 and the syntax defined in the accompanying schema file org.atsc.drmOperation-request.json.

Table 5.8 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:~~uud~~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 5.9 and the syntax defined in the accompanying schema file org.atsc.notify-DRM.json.

Table 5.9 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:~~uud~~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

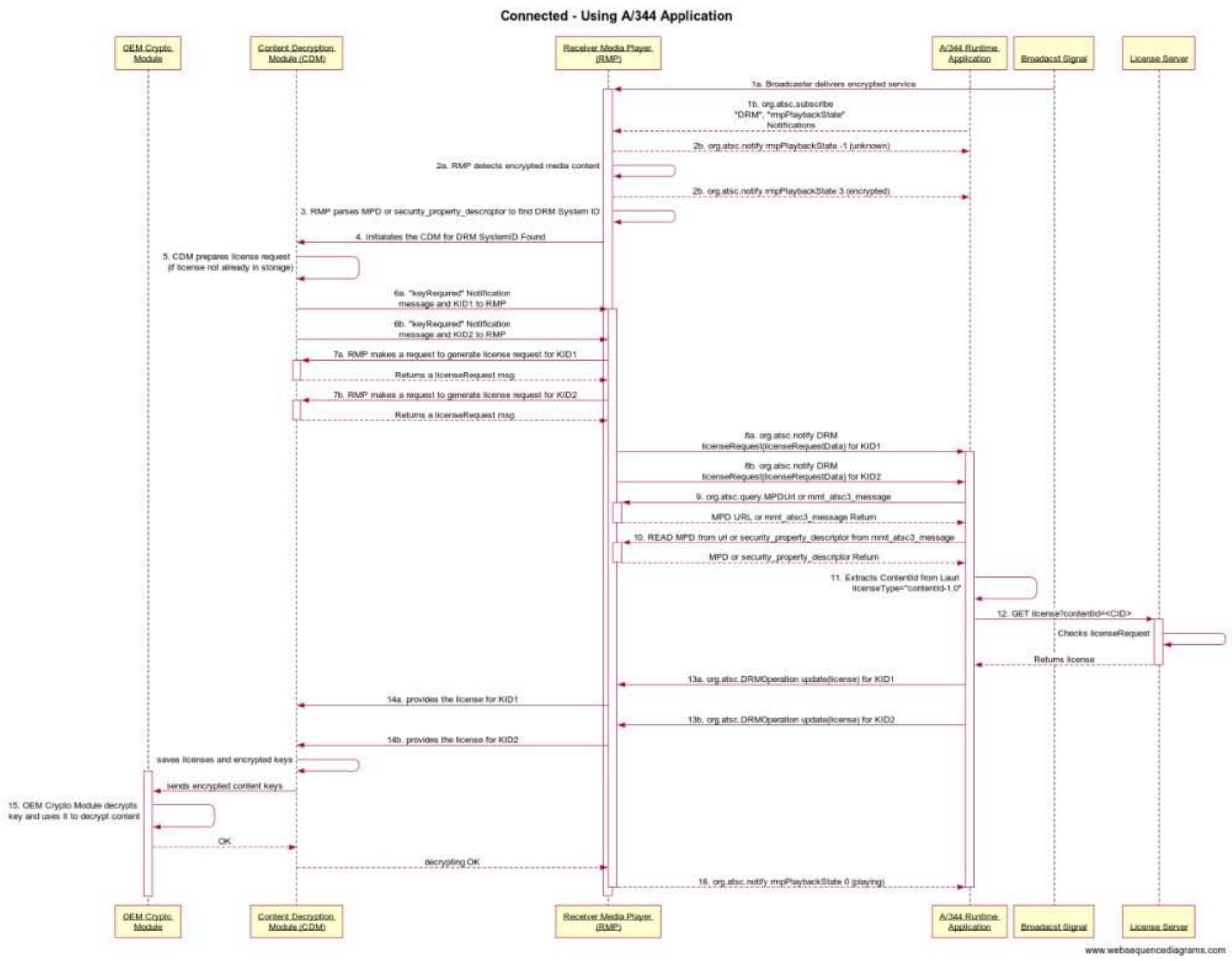
See A/344 Section 8.3.3, A/344 Table 8.2 JSON-RPC ATSC Error Codes [9].

Update the first paragraph of A.1 as follows:

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.

Add page breaks to place Figures A-1 through A-4 within their own pages, possibly in landscape mode, depending on which orientation is the most efficient for each figure.

Replace Figure A-1 with the following image:


Connected - Using A/344 Application
OEM Crypto Module
Content Decryption Module (CDM)
Receiver Media Player (RMP)
A/344 Runtime Application
Broadcast Signal
License Server
1a. Broadcaster delivers encrypted service
1b. org.atsc.subscribe "DRM", "rmpPlaybackState" Notifications
2b. org.atsc.notify rmpPlaybackState -1 (unknown)
2a. RMP detects encrypted media content
2b. org.atsc.notify rmpPlaybackState 3 (encrypted)
3. RMP parses MPD or security_property_descriptor to find DRM System ID
4. Initializes the CDM for DRM SystemID Found
5. CDM prepares license request (if license not already in storage)
6a. "keyRequired" Notification message and KID1 to RMP
6b. "keyRequired" Notification message and KID2 to RMP
7a. RMP makes a request to generate license request for KID1
Returns a licenseRequest msg
7b. RMP makes a request to generate license request for KID2
Returns a licenseRequest msg
8a. org.atsc.notify DRM licenseRequest(licenseRequestData) for KID1
8b. org.atsc.notify DRM licenseRequest(licenseRequestData) for KID2
9. org.atsc.query.MPDUrl or mmt_atsc3_message
MPD URL or mmt_atsc3_message Return
10. READ MPD from url or security_property_descriptor from mmt_atsc3_message
MPD or security_property_descriptor Return
11. Extracts ContentId from Laurl licenseType="contentid-1.0"
12. GET license?contentId=<CID>
Checks licenseRequest
Returns license
13a. org.atsc.DRMOperation update(license) for KID1
14a. provides the license for KID1
13b. org.atsc.DRMOperation update(license) for KID2
14b. provides the license for KID2
saves licenses and encrypted keys
sends encrypted content keys
15. OEM Crypto Module decrypts key and uses it to decrypt content
OK
decrypting OK
16. org.atsc.notify rmpPlaybackState 0 (playing)
www.websequencediagrams.com

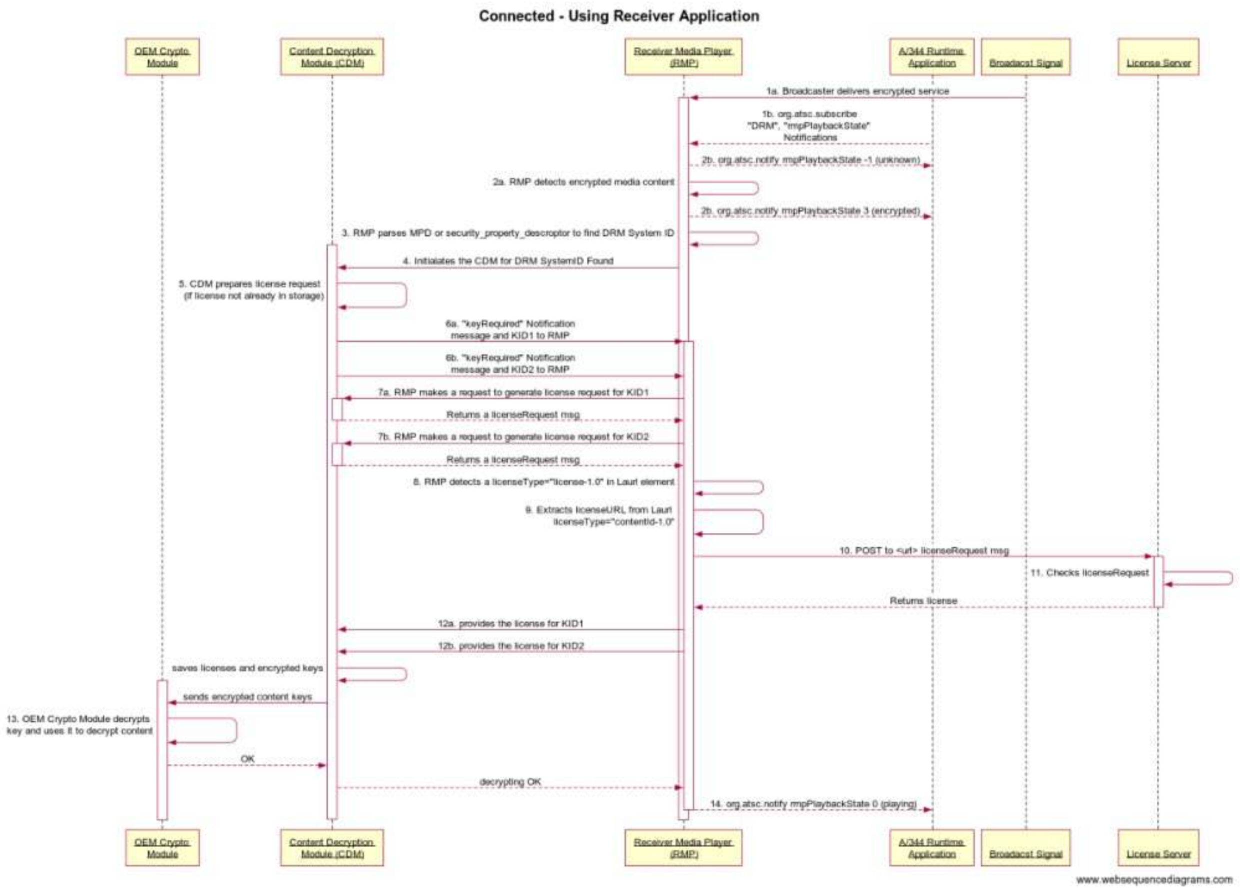
Update A.1, item 3 as follows:

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 it lists a DRM System ID that the RMP supports. If one is found, then processing continues.

Update A.1, items 9 and 10 as follows:

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.

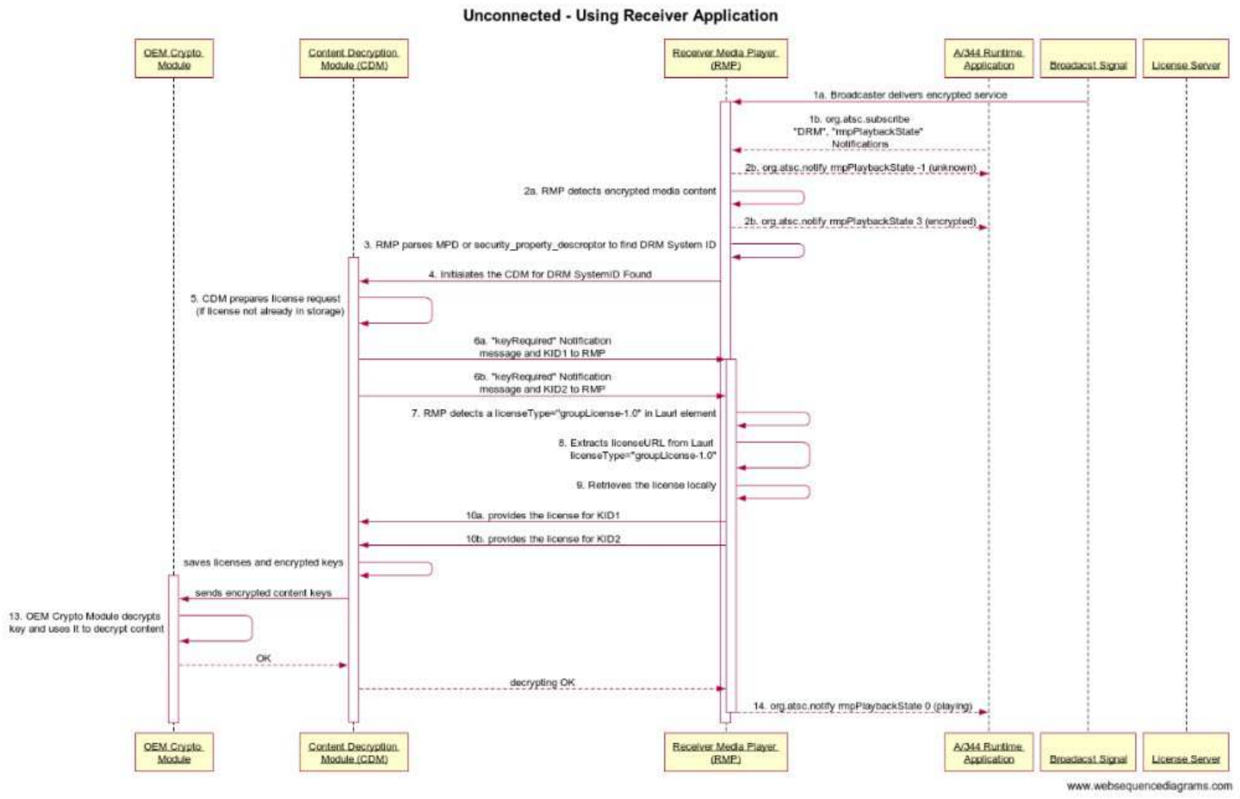
Replace Figure A-2 with the following image:


Connected - Using Receiver Application
OEM Crypto Module
Content Decryption Module (CDM)
Receiver Media Player (RMP)
A/344 Runtime Application
Broadcast Signal
License Server
1a. Broadcaster delivers encrypted service
1b. org.atsc.subscribe "DRM", "rmpPlaybackState" Notifications
2b. org.atsc.notify rmpPlaybackState -1 (unknown)
2a. RMP detects encrypted media content
2b. org.atsc.notify rmpPlaybackState 3 (encrypted)
3. RMP parses MPD or security_property_descriptor to find DRM System ID
4. Initializes the CDM for DRM SystemID Found
5. CDM prepares license request (if license not already in storage)
6a. "keyRequired" Notification message and KID1 to RMP
6b. "keyRequired" Notification message and KID2 to RMP
7a. RMP makes a request to generate license request for KID1
Returns a licenseRequest msg
7b. RMP makes a request to generate license request for KID2
Returns a licenseRequest msg
8. RMP detects a licenseType="license-1.0" in LaurI element
9. Extracts licenseURL from LaurI licenseType="contentid-1.0"
10. POST to <url> licenseRequest msg
11. Checks licenseRequest
Returns license
12a. provides the license for KID1
12b. provides the license for KID2
saves licenses and encrypted keys
sends encrypted content keys
13. OEM Crypto Module decrypts key and uses it to decrypt content
OK
decrypting OK
14. org.atsc.notify rmpPlaybackState 0 (playing)
www.websequencediagrams.com

Update A.2, bullet 3 as follows:

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.

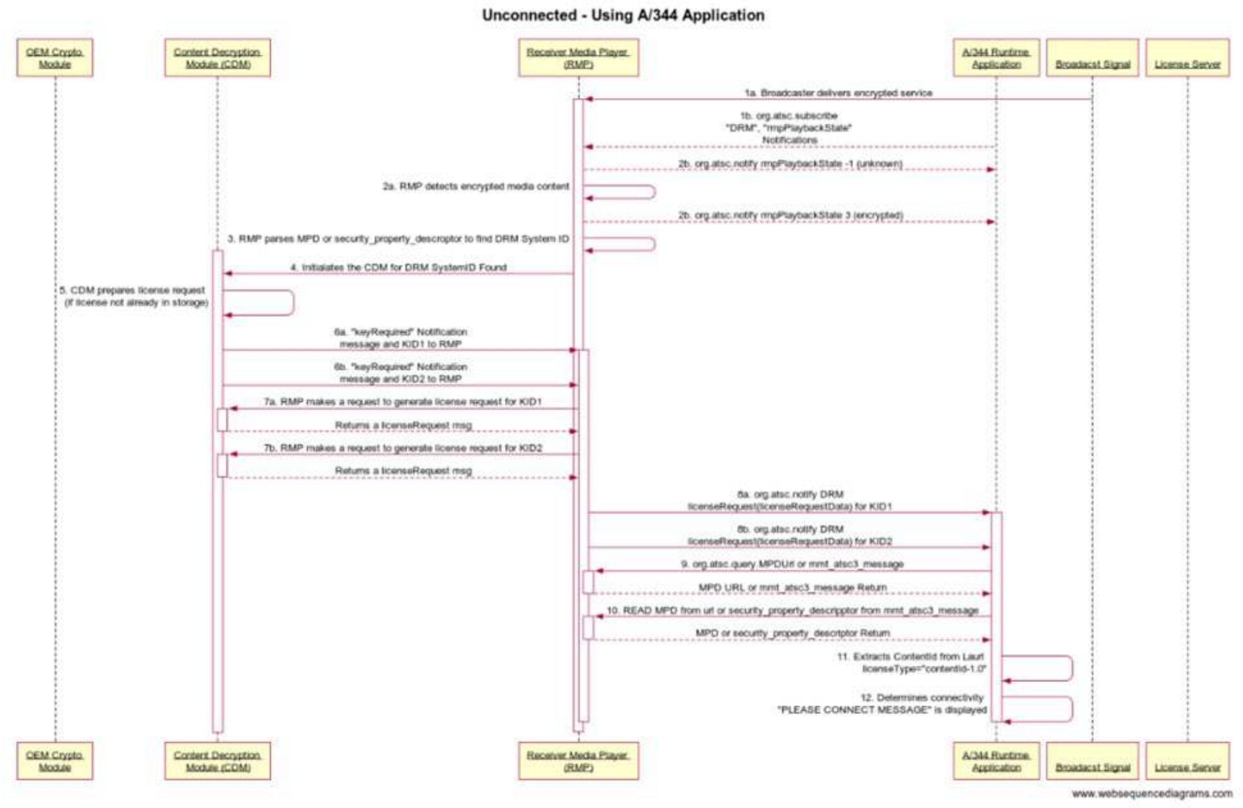
Replace Figure A-3 with the following image:


Unconnected - Using Receiver Application
OEM Crypto Module
Content Decryption Module (CDM)
Receiver Media Player (RMP)
A/344 Runtime Application
Broadcast Signal
License Server
1a. Broadcaster delivers encrypted service
1b. org.atsc.subscribe "DRM", "rmpPlaybackState" Notifications
2b. org.atsc.notify rmpPlaybackState -1 (unknown)
2a. RMP detects encrypted media content
2b. org.atsc.notify rmpPlaybackState 3 (encrypted)
3. RMP parses MPD or security_property_descroptor to find DRM System ID
4. Initialates the CDM for DRM SystemID Found
5. CDM prepares license request (if license not already in storage)
6a. "keyRequired" Notification message and KID1 to RMP
6b. "keyRequired" Notification message and KID2 to RMP
7. RMP detects a licenseType="groupLicense-1.0" in Laurl element
8. Extracts licenseURL from Laurl licenseType="groupLicense-1.0"
9. Retrieves the license locally
10a. provides the license for KID1
10b. provides the license for KID2
saves licenses and encrypted keys
sends encrypted content keys
13. OEM Crypto Module decrypts key and uses it to decrypt content
OK
decrypting OK
14. org.atsc.notify rmpPlaybackState 0 (playing)
www.websequencediagrams.com

Update A.3, item 3 as follows:

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.

Replace Figure A-4 with the following image:


Unconnected - Using A/344 Application
OEM Crypto Module
Content Decryption Module (CDM)
Receiver Media Player (RMP)
A/344 Runtime Application
Broadcast Signal
License Server
1a. Broadcaster delivers encrypted service
1b. org.atsc.subscribe "DRM", "rmpPlaybackState" Notifications
2b. org.atsc.notify rmpPlaybackState -1 (unknown)
2a. RMP detects encrypted media content
2b. org.atsc.notify rmpPlaybackState 3 (encrypted)
3. RMP parses MPD or security_property_descroptor to find DRM System ID
4. Initializes the CDM for DRM SystemID Found
5. CDM prepares license request (if license not already in storage)
6a. "keyRequired" Notification message and KID1 to RMP
6b. "keyRequired" Notification message and KID2 to RMP
7a. RMP makes a request to generate license request for KID1
Returns a licenseRequest msg
7b. RMP makes a request to generate license request for KID2
Returns a licenseRequest msg
8a. org.atsc.notify DRM licenseRequest(licenseRequestData) for KID1
8b. org.atsc.notify DRM licenseRequest(licenseRequestData) for KID2
9. org.atsc.query.MPDUrl or mmt_atsc3_message
MPD URL or mmt_atsc3_message Return
10. READ MPD from url or security_property_descripptor from mmt_atsc3_message
MPD or security_property_descriptor Return
11. Extracts ContentId from Laurl licenseType="contentid-1.0"
12. Determines connectivity "PLEASE CONNECT MESSAGE" is displayed
www.websequencediagrams.com

Update A.4, item 3 as follows:

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.

Update A.4, items 9 and 10 as follows:

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.

– End of Document –