DRM Gateway User Guide

Document no	EDGS-168
Version	A12
Created on	2026-04-22

Contents

1 Introduction

1.1 Confidentiality Notice

This document is confidential and may not be reproduced, distributed or used for any purpose other than by the recipient for the assessment, evaluation and use of AgileTV products, unless written permission is given in advance by AgileTV.

1.2 About This Document

This document describes installation, configuration and operation of the eDRM system that handles communication between AgileTV systems (VCP and Software Repackager) and DRM key servers.

1.3 References

EDGS-151 DRM Gateway eDRM Specification

EDGS-167 DRM Gateway Release Notes

1.4 History

Issue	Date	Changes
A1	2015-05-12	First version.
A2	2019-02-20	CCMI url API
A3	2019-09-09	Track keys support
A4	2020-01-17	Updates for multi-threaded DRM GW
A5	2021-12-09	Appendix A added
A6	2022-03-29	KeyRotator update
A7	2023-01-17	Update supported Linux distribution
	2023-04-04	Update new DRM with CPIX
A8	2023-07-10	Update for DRM cache
	2023-07-24	Create DAZN CPIX plugin
A9	2023-10-12	Update processor name in DRM cache
	2024-04-10	Add ssl_version option
	2024-04-15	Drop RHEL7 support
	2024-05-03	Update installation instructions
	2024-05-10	Update Used files and ports
	2024-05-13	Update Installation prerequisites
	2024-06-10	Add removal instructions
A10	2024-09-12	Add support Redis cache
	2025-02-26	Create Verimatrix CPIX plugin
A11	2025-07-31	Support RHEL 9
	2025-11-13	Update Redis cache information
	2025-11-26	Create Verimatrix 5.3 CPIX plugin
	2025-11-26	Update Cenc information
	2025-12-19	Remove outdated Installation steps
	2026-01-19	Add note about OS upgrade
	2026-03-11	Support variant groups for VerimatrixCPIX
A12	2026-04-03	Rename a configuration for VerimatrixCPIX
	2026-04-21	Add StreamViewer Monitoring

2 Installation

2.1 Prerequisites

The DRM Gateway requires RHEL 8.10, or at least 9.7 as official operating system. Other RedHat binary compatible distributions are unofficially supported.

NOTE (in-place OS upgrade, e.g. Red Hat LEAPP)

- Before starting the OS upgrade, back up all configurations and remove all installed StreamBuilder products, including DRM-GW.

- After the OS upgrade is complete, reinstall the products and re-apply the configurations.

Configuration files and certificates are usually renamed to end with `.rpmsave` and must be restored or merged manually.

2.2 Preparation

2.2.1 Firewall

The following ports must be opened in the firewall:

Port	Description
:4443	Optional. DRM Gateway HTTPS API endpoint. Needed if DRM gateway is used by repackagers on other nodes.

Note: The firewall is not part of the DRM Gateway bundle and it is assumed that it is configured and managed by the customer.

2.2.2 SELinux

The SELinux should be in `permissive` or `disabled` mode. In `enforcing` mode, customer has to manage rules themself to let DRM Gateway works.

2.3 Installation

For installing the DRM Gateway

```
[root@esb3009 ~]# chmod +x install-esb3009-X.Y.Z
[root@esb3009 ~]# ./install-esb3009-X.Y.Z
```

2.4 Removal

To uninstall the DRM Gateway do as follows:

```
[root@esb3009 ~]# systemctl stop ew-drm-gw
[root@esb3009 ~]# dnf autoremove esb3009
```

2.5 Upgrade

For upgrading the DRM Gateway follow these steps

```
[root@esb3009 ~]# chmod +x install-esb3009-X.Y.Z
[root@esb3009 ~]# ./install-esb3009-X.Y.Z -u
```

2.6 Downgrade

For downgrading DRM Gateway, it is recommended to uninstall then reinstall the older version of product.

3 Configuration

The service configuration is found in two files. First:

```
    /etc/sysconfig/edgeware/drm-gw
```

which is a regular RedHat system configuration file. Here the location of the service configuration file can be set:

- `CONFIG_PATH`: The path to `config.json` that contains most configuration.

All other configuration is available in:

```
/etc/edgeware/drm-gw/config.json
```

The format of the configuration file is JSON and supported values are:

- `"log_level"` (string "DEBUG"/"INFO"/"WARNING"/"ERROR"/"CRITICAL"): Which level to log to the output.
- `"log_level_libraries"` (string "DEBUG"/"INFO"/"WARNING"/"ERROR"/"CRITICAL"): Which level to log from libraries (`requests` and `suds`).
- `"access_log_file"` (string): Filename of the HTTP access log.
- `"access_log_format"` (string): The formatting string for the log file specified under `"access_log_file"`. See Python Gunicorn documentation for details.
- `"workers"` (integer): The number of worker processes to use. Each worker has its own egress and ingress cache (if supported by the plugin) so using more workers than necessary will create unnecessary load on the key server.
- `"keep_alive"` (integer): The time (in seconds) the server will maintain an open idle connection with the client.
- `"bind_addres"` (string): ipv4/ipv6 address where the eDRM HTTPS server listens. Default value is `0.0.0.0`.
- `"server_port"` (string): TCP port where the eDRM HTTPS server listens. Default value is `4443`.
- `"server_certificate_file"` (string): Path of public certificate file for the HTTPS server.
- `"server_key_file"` (string): Path of private key file for the HTTPS server.
- `"server_shared_secret"` (string): Shared secret string that is used for authentication of incoming requests. Shared secret must be present in the message body as `"shared-secret"` property for CCMI API calls.
- `"ssl_version"` (string "TLS"/"TLSv1"/"TLSv1_1"/"TLSv1_2"): Select SSL version to use. Default value is `TLS`. Note: Due to Gunicorn, only one protocol version can be set at a time, except `TLS`. See Python Gunicorn documentation for details.

SSL Version	Description
TLS	Negotiate highest possible version between client/server.
TLSv1	only TLS 1.0
TLSv1_1	only TLS 1.1
TLSv1_2	only TLS 1.2

- `"drm_processors"` (object): Object that defines mapping eDRM paths to specific DRM processors. Each entry is registered as a DRM processor instance in the eDRM dispatcher and mapped into output profiles. Entry values are objects that must have the key `"type"` with a value corresponding to one of the DRM processors.

For example, configuration:

```
...
"server_shared_secret": "shared-secret"
"drm_processors": {
  "path1": {
    "type": "ProcessorName1"
  },
  "path2": {
    "type": "ProcessorName2"
  }
},
"accounts": [
  {
    "name": "default",
```

```
    "outputProfiles": [
      {
        "name": "drm-profile-1",
        "encryption": "aes-128",
        "processors": [
          "path1"
        ]
      },
      {
        "name": "drm-profile-2",
        "encryption": "aes-128",
        "processors": [
          "path2"
        ]
      }
    ]
  }
]
```

will have the DRM processor "ProcessorName1" and "ProcessorName2" correspondingly available at "/__cl/<location>/__c/<content>/__op/drm-profile-1/__f/<playlist>" and "/__cl/<location>/__c/<content>/__op/drm-profile-2/__f/<playlist>".

- kafka-logger - kafka logger configuration, if this secion is not present in configuration events won't be generated.

```
"kafka-logger":
  {
      "address" : "127.0.0.1:8887",
      "topic" : "events",
      "heartbeat" : {"period": 30}
  }
```

  - address of kafka instance
  - topic - topic used for raporting events from drm-gw
  - heartbeat - heartbeat mechanism configuration - enables publishing simple event every specified in seconds period(>0). Heartbeat events can be used to check if drm-gw is running.

- redis_cache (object): Configuration for enabling Redis-based caching in DRM-GW. If this object is defined and contains at least one parameter, the Redis cache feature will be activated. See Redis cache for details.

```
"redis_cache":
  {
      "cache_wait_timeout_ms": 1000
  }
```

3.1 Output profiles

Output profiles configured in DRM Gateway should match output profiles in repackager, as same CCMI URL is forwarded from repackager to DRM Gateway. Output profiles are account-based.

- "accounts" (list of objects): List of configured accounts for CCMI URL API calls. Account object must contain "name" and "outputProfiles" properties.
- "accounts"."name" (string): Name of the account. Account name will be matched with __a parameter in CCMI URL, or if __a is not present, account with name "default" will be used.
- "accounts"."outputProfiles" (list of objects): List of configured output profiles for account. Output profiles must contain "name", "encryption", "processors" properties.
- "accounts"."outputProfiles"."name" (string): Name of output profile. Output profile name will be matched with __op parameter in CCMI URL.

- `"accounts"."outputProfiles"."encryption"` (string): Encryption configured on output profile. Supported values are `cenc`, `aes-128`, `sample-aes` and `playready`.
- `"accounts"."outputProfiles"."processors"` (list of strings): List of processors that will be applied in a sequence for CCMI URL requests on this output profile.
- `"accounts"."outputProfiles"."use_dci"` (boolean): If set to true, drm content id (`__dci` parameter from CCMI URL) will be used as a resource id. If drm content id is not present in CCMI URL, content id (`__c` parameter) is used as a resource id. If set to false, drm content id from URL is ignored. Defaults to `true`

3.1.1 Output profile cache

Output profile can be configured with a LRU cache that caches complete JSON responses. Cache entries are stored and can be awaited before the final result is calculated. That means that for multiple requests to the same cache key only the first requests perform processing, and the rest are queued to await the first request. Cache key is `(processor_name, resource_id, variants, key_ranges)`.

In the rotation case, the cache only returns when cached keys cover the entire range of requests.

If the cache just matches a part with the request range, it only sends requests for missing parts to the key server, updates key-value to cache after having value and the returned value only includes the requested key range.

Cache does not have explicit API for control, but it is cleared on DRM Gateway service reloads and restarts.

The Output profile cache only works efficiently with a single worker. If you have multiple workers, please consider using the Redis cache. See Redis cache for details.

- `"accounts"."outputProfiles"."cache_size"` (int): If set to positive value responses for CCMI requests will be cached in a LRU cache of specified size. Setting the cache size to `0` disable the caching. Default value is `0`.
- `"accounts"."outputProfiles"."cache_max_age"` (int): Max age in seconds for elements in the cache. If the accessed cache entry is older than this value, it will be removed from the cache, and new value will be calculated. Default value is `86400` (one day).
- `"accounts"."outputProfiles"."cache_result_timeout"` (int): Advanced tuning parameter that specifies time in seconds that requests are willing to wait for an ongoing cache set operation. Default value is `30`.

3.1.2 Example

Example configuration for an account with 3 output profiles. Each output profile handles one output format, DASH, HLS or MSS.

```
...
"drm_processors": {
    "widevine": {
        ...
    },
    "playready": {
        ...
    },
    "clearkey": {
        ...
    },
    "fairplay": {
        ...
    }
},
"accounts": [
    {
        "name": "default",
        "outputProfiles":
        [
            {
                "name": "dash-drm",
```

```
                    "encryption": "cenc",
                    "processors": ["widevine", "playready", "clearkey"]
                },
                {
                    "name": "hls-drm",
                    "encryption": "sample-aes",
                    "processors": ["fairplay"]
                }
                {
                    "name": "mss-drm",
                    "encryption": "playready",
                    "processors": ["playready"]
                }
            ]
        }
    ]
```

3.2 Variant groupers

DRM Gateway provides a number of variant groupers for use in processors. A variant grouper is responsible for grouping the asset's variants, which are provided in an eDRM request message. The result of the variant grouping is used by plugins that supports multiple keys in a single asset.

The following variant groupers are available:

- `"single_key"`: Single group for all video and audio tracks.
- `"video_only"`: Single group for all video tracks. Audio and subtitles are unencrypted.
- `"single_variant"`: Each video and audio variant in its own group.
- `"video_1G_audio_1G"`: One group for all video variants, and an additional group for audio variants.
- `"video_2G_UHD"`: One group for UHD video variants, and another for the rest of the video variants. Audio and subtitles are unencrypted.
- `"video_3G_audio_1G"`: Separate groups for UHD, HD and SD video variants, and an additional group for audio variants. Subtitles are unencrypted.
- `"video_each_audio_1G"`: Each video variant in its own group, and an additional group for audio variants. Subtitles are unencrypted.
- `"none"`: Grouper that disables variant grouping in the processor.

3.3 Key rotation

Key rotation divides UTC time into periods of same duration. Each period is identified by its start time position and contains different DRM information for a content.

Requests that contain only start time position and with unspecified stop time implies that such request is for a live content. In that case DRM Gateway may respond with some keys from the future periods, so that repackager can cache them. Such responses will also contain suggested time to next poll, where repackager should ask for next round of keys.

Times to next poll are centered around middle of the period, with an optional spread factor that randomize the polling offset for each content, i.e. each content will always have same offset, but the offsets for different contents will take a random value in the spread zone around middle of the period.

Changing key rotation period or period offset requires clearing all the downstream cached values (repackager cache, CDN cache, ...) for that output profile.

Default key rotation support is configured in the same way in processors:

- `"key_rotation_period"` (int): Period for key rotation in seconds. The default value of `0` is interpreted as key rotation being disabled.
- `"key_rotation_additional_keys"` (int): Number of additional keys that will be sent in response when clients are making a request for period without end time. Default value is `2`. Any value less than 1 fallback to 1.
- `"max_result_items"` (int): Maximum number of key periods in response. This parameter is used to protect overloading of upstream server from large number of requests. Default value is `100`.

- "key_rotation_period_offset" (int): Number of seconds that offsets each key period. By default key rotation periods are aligned with epoch time. This parameter provides a tuning value that offsets each period by fixed value. For example, if key rotation period is set to 8 hours, period boundaries will be at UTC 0, 8 and 16 hours of each day. By setting period offset to 2 hours period boundaries will be at 2, 10 and 18 hours. Note that setting period offset to 10 hours is the same as setting to 2 hours, because all periods are sought forward 10 or 2 hours results the same periods list. So that in case `key_rotation_period_offset` > `key_rotation_period`, the period offset is set to the value of `key_rotation_period_offset` modulo `key_rotation_period`. Default value is `0`.
- "key_rotation_poll_spread_percentage" (int or null): Percentage of key rotation period where the next poll can occur. Default value `null` does not calculate next poll time spread.

3.4 Redis cache

For sharing cache data among workers, Redis cache should only be used on multiple workers. If only having a single worker, the Output Profile cache should be used to better performance.

The DRM-GW supports caching by using Redis as a external cache storage. Like the Output Profile cache, the Redis cache supports multiple requests to the same cache key, while the first request is processed, others request has to wait until the first request has finished or time out. If the other requests are timed out when waiting for the first request to be completed, they will skip using the Redis cache and send a new request to plugins.

Cache key is `(account_name, output_profile_name, processor_name, resource_id, variants, key_ranges)`.

The key in Redis cache is a chain value resource_id, variants, processor_name with startwith `DRMGW:`

Value includes many fields with different data:

- `ranges`: contains a list of periods from start to end for a key on rotation case. In the none-rotation case, `ranges` contains the last timestamp when this key was used.
- `value`: contains a `Resource Context` object including all positions, and DRM Infos are accumulated on missing request key.

Please check and make sure the `ew-local-cache` service is running to enable the Redis server for DRM-GW

Check service status:

```
systemctl is-active ew-local-cache
```

Start the service if it is not already started:

```
systemctl start ew-local-cache
```

All the DRM-GW cache on the Redis server will be cleaned if starting or restarting the DRM-GW service.

Redis on DRM-GW uses database number 1 to store the cache.

When both the Output Profile cache and Redis cache are active, DRM-GW prioritizes the Redis cache and ignores the Output Profile cache.

If the Redis cache is active but the Redis server (from `ew-local-cache`) is unavailable, DRM-GW automatically falls back to the Output Profile cache if configured. Otherwise, caching is disabled.

Redis cache configuration:

- "cache_wait_timeout_ms" (int): Timeout in milliseconds determines how long for other requests to the same cache key at the same time shall wait before ignoring the Redis cache. The default value of `1000`.
- "cache_result_timeout_s" (int): Timeout in second for retrieve result from Redis cache. The default value of `30`.
- "socket_timeout_s" (int): Timeout in second determines how long the DRM-GW will wait for a response from the Redis server before raising a timeout exception. The default value of `30`.

To enable Redis caching in DRM-GW, define the `redis_cache` configuration with at least one parameter:

3.4.1 Example:

```
"redis_cache": {
  "cache_wait_timeout_ms": 1000
},
```

3.5 Processor types

The available DRM processors that may be used as value for `"type"` can be sorted into three separate groups based on their intent and function: *Core processors*, *DRM system processors* and *Test processors*.

3.5.1 Core processors

These modules form the backbone of the DRM gateway. They aren't associated with a specific DRM implementation, but are used for things like key generation, key storage and chaining of several DRM variants in CENC.

- `Cenc`
- `ClearDrmInfo`
- `Clearkey`
- `EdrmProxy`
- `KeyGenerator`
- `Playready`
- `RedisKeystore`
- `StaticKey`
- `Widevine`

3.5.2 DRM system processors

The DRM processors connect to external DRM systems in order to retrieve encryption parameters.

- `Axinom`
- `Castlabs`
- `Conax`
- `CPKeyGenerator`
- `CPIX`
- `ExpressPlay`
- `Irdeto`
- `IrdetoV2`
- `IrdetoCPIX`
- `Cust1CPIX`
- `DaznCPIX`
- `Kaltura`
- `Nagra`
- `SecureMedia`
- `Ubique`
- `VerimatrixVEI36`
- `VerimatrixVEI38`
- `VerimatrixCEI`
- `VerimatrixCPIX`
- `Verimatrix53CPIX`

3.5.3 Test processors

Test processors aren't meant to be used in production deployments, but rather to verify system functionality.

- `TestAxinomWidevine`
- `TestPlayready`

Each DRM processor has its own configuration specification.

3.6 Cenc

An internal DRM provider that dispatches the same DRM request through a list of providers. At least one of providers needs to set encryption key and key_id. An error is thrown if different providers tries to set different key, key_id or iv.

Configuration parameters are:

- "type" (string "Cenc"): Required.
- "default_providers" (list of strings): Specifies the ordered list of DRM providers that the "Cenc" plugin will use to process each DRM request. When a request is received, the "Cenc" plugin acts as an internal dispatcher, forwarding the same request to each provider in this list. This mechanism allows combining the output of several DRM providers for a single request. The order of providers in the list determines the order in which they are called.

Note: Requests dispatched to DRM providers in this list will bypass the cache, even if caching is enabled.

3.6.1 Example

```
"cenc": {
    "type": "Cenc",
    "default_providers": ["playready", "widevine", "clearkey"]
}
```

The configuration above will result in an eDRM response with 3 different DRM systems specific informations. The playready processor should set key and key_id, while the widevine andclearkey processors should use or re-use that same key and key_id.

3.7 ClearDrmInfo

Utility DRM processor that removes all drm info set by previous processors in chain. Key data (key, key id and iv) are not modified by this plugin. This processor have to be followed by processor that ads drm info.

3.7.1 Example

Intended use case is to make a chain where we first obtain key info with some default drm data. Than we strip additional data with ClearDrmInfo processor. Finally, we add one or more generic drm processor that generates drm info based on key info set in first step.

```
"cleardrminfo": {
    "type": "ClearDrmInfo"
},
"cenc": {
    "type": "Cenc",
    "default_providers": ["3rdpartydrm",
                          "cleardrminfo",
                          "playready",
                          "widevine"]
}
```

3.8 Clearkey

DRM processor for the Clear Key DRM scheme. Does not contact external servers. Processor can either reuse key_id from current DRM context, or it will set key and key_id from configuration.

- "type" (string "Clearkey"): Required.
- "key_id" (string): base64 encoded key ID used for this request.
- "key" (string): base64 encoded key used for this request.

Outputs base64 encoded PSSH version 1 box with single KID.

3.8.1 Example

```
    "clearkey": {
        "type": "Clearkey",
        "key_id": "H3JbV93QV3mPNBKQON2UtQ==",
        "key": "ClKhDPHMtCouEx1vLGsJsA=="
    }
```

3.9 EdrmProxy

DRM processor for proxying requests towards usptream eDRM server. Optional path elements from request will be passed to the upstream. For example, URL at "`https://edrm-server/api/1/eDRM/edrm_proxy/playready / widevine/shared-secret`" with the configuration snippet:

```
        "url": "https://edrm-upstream:4443/api/1/eDRM/cenc"
        "shared_secret": "upstream-secret"
```

will result in calling the upstream server with URL: "`https://edrm-upstream:4443/api/1/eDRM/cenc/ playready/ widevine/upstream-secret`"

- "`type`" (string "`EdrmProxy`"): Required.
- "`url`" (string): URL prefix of the eDRM service end-point. For example: "`https://edrm-server/api/1/eDRM`"
- "`shared_secret`" (string): Shared secret from upstream eDRM server. This value will be appeneded to all constructed URLs towards upstream eDRM server.
- "`certificate`" (string or boolean): How to verify eDRM HTTPS connection: true to use system set of trusted root certificates, false to not verify eDRM certificate, or string value to specify file name with trusted root certificates. Not used if "`url`" is using plain "`http`" protocol.
- "`connection_timeout`" (int): Timeout in seconds for requests towards eDRM key server. Defaults to `10`.

3.9.1 Example

```
    "edrm_proxy": {
        "type": "EdrmProxy",
        "url": "https://testserver:4443/api/1/eDRM",
        "shared_secret": "shared-secret",
        "certificate": false,
        "connection_timeout": 10
    }
```

3.10 KeyGenerator

The internal key generation is not and shall not be regarded as an approved or tested Digital Rights Management ("DRM") system and shall not be used as or substituted for a DRM. AgileTV provides the internal key generation on an "as is basis" only without any conditions and makes no representations or warranties of any kind. Accordingly, to the maximum extent permitted by law AgileTV excludes all representations and warranties, express or implied, including without limitation any warranties for non-infringement, satisfactory quality, merchantability or fitness for any purpose.

Processor with the ability to create AES keys for a content on the fly. Keys can be persisted in the local Redis keystore, or they can be published to the 3rd party server. Key ingestion and revocation is available on separate API endpoints.

The keystore can be configured as an internal Redis backed keystore, or with external REST api. If both are configured, Redis will take the master role and external will take the replicating slave role. The replicating slave means that all write operations (key addition and key removal) will be executed on the master and on the slave keystore, but read operations will only be executed on the master keystore.

Key ingestion is available as a separate eDRM endpoint. The subpath "`/add`" is used to add externally generated keys for a given content into the system and the subpath '/remove' is used to remove keys from the system. For example, if provider is configured on "`keyGenerator`" path, then the URL "`/api/1/eDRM/keyGenerator/add/`" will be the endpoint for adding keys into the keystore and "`/api/1/eDRM/keyGenerator/remove/`" will be the endpoint for removing keys from the keystore.

Key adding and removal operations need to be successful on all nodes in order for eDRM to return success status. If a write operation fails on any of the nodes, an error response code is returned from eDRM, possibly leaving operation successful on some nodes. It is the responsibility of the caller to repair the state, either by re-issuing the command, or by other means, e.g. manual database cleaning.

When the plugin generate the key for content, failure to write the key value on some node will trigger rollback. Rollback is done in best effort mode, e.g. it will not be retried in case of temporary network connection outage.

Keys in the internal keystore are not stored in plain text, but are wrapped with a private AES key. The key for wrapping is based on a secret from the configuration with an HMAC-based Extract-and-Expand Key Derivation Function (HKDF). Key wrapping is based on the RFC 3394 algorithm.

Publishing endpoint should be implemented as a simple REST interface, with support for POST and DELETE methods. If the publishing endpoint will be used as a main keystore, the GET method must also be supported. When the POST method is used, a key will be transfered in plain-text as the body of the request. GET method requests expect that key will be in plain-text as the body of response.

Configuration parameters are:

- "`redis_endpoints`" (list of objects): Optional list of Redis endpoints. Each endpoint is an object consisting of required string "`hostname`" and optional integer "`port`". The default port value is `6379`.
- "`redis_timeout`" (int): Optional timeout in seconds for Redis operations. Default value is `5`.
- "`redis_secret`" (string): Optional value for seeding encryption key for protecting keystore. If Redis endpoints are configured, this is a required parameter.
- "`redis_db`" (int): Optional Redis database id. Default value is `0`.
- "`publish_endpoint`" (string): Optional URL for a 3rd party key-publishing endpoint.
- "`publish_timeout`" (int): Optional timeout, in seconds, for key-publish operation. Default value is `10`.
- "`publish_certificate`" (string or boolean): How to verify the publish HTTPS connection: `true` to use the system's set of trusted root certificates, `false` to not verify connection certificate, or a string value to specify a file name with a trusted root certificates. Not used if "`publish_endpoint`" is not configured or configured with plain "`http`" URL.
- "`key_url_prefix`" (string): Key URL prefix that will be used in outgoing eDRM messages. Content ID will be appended to this URL prefix and the resulting string will be used in the URI attribute of the HLS manifest EXT-X-KEY tag. The URL should point to a location outside of the eDRM.

For example, "`key_url_prefix`" value of "`http://keyserver.com/keys?content=`" and a stream with content ID "`channel1`" will cause an HLS player to request a decryption key from "`http://keyserver.com/keys?content=channel1`".

- "`key_generation`" (boolean): Optional flag that defines if a new key will be generated and persisted when there is no key in keystore. Default value is `true`.

The following examples are different eDRM integration setups with explained roles. In all setups key retrival by clients is done outside of the eDRM. Such an implementation can be as simple as proxying requests towards the eDRM.

3.10.1 KeyGenerator with internal key store

In this setup a Redis store needs to be configured and the publishing endpoint is left unconfigured. Keys for content are automatically generated and persisted in the keystore on the first client access to the content. Key URL should point to the 3rd party location. Such a location can use the eDRM interface to get keys from eDRM for the requested content.

3.10.2 KeyGenerator with internal key store and key publishing

In this setup a Redis store and a publishing endpoint are configured. Keys for content are automatically generated and persisted in the keystore on the first client access to the content, and keys are also pushed to the

3rd party server defined as publishing endpoint. Published keys are expected to be available by the 3rd party server for client key requests.

3.10.3 KeyGenerator with external key store

In this setup a publishing endpoint is configured and the Redis store is left unconfigured. Keys are still generated on first client access to the content, but key persistance is completely left to the 3rd party implementation.

3.10.4 KeyGenerator with ingested keys

Typically this setup will only have configured a Redis keystore. Also, key generation should be turned off in the configuration. Keys for a content will be ingested by the operator. Key URL points to the 3rd party system that can be fully implemented without eDRM, or it can use eDRM interface to read keys for the requested content.

3.10.5 Example

```
    "hlskey": {
        "type": "KeyGenerator",
        "redis_endpoints": [{"hostname":"localhost", "port": 6379}],
        "redis_timeout": 5,
        "redis_secret": "redisstoresecret",
        "publish_endpoint": "",
        "publish_timeout": 10,
        "publish_certificate": "",
        "key_url_prefix": "http://testserver/keys?content="
    }
```

3.11 Playready

DRM processor for Microsoft PlayReady key server. The processing DRM context must have key and key ID set since the proccesor will use them to create a valid PSSH value.

- `"type"` (string `"Playready"`): Required.
- `"la_url"` (string): Optional license acquisition URL to be included in the PSSH value.
- `"license_options"` (object): An optional object containing key/value pairs to be used as query string values appended to the LA URL, e.g:

```
  ...
  "license_options": {
          "PlayRight": "1",
          "UseSimpleNonPersistentLicense": "1"
  }
```

If encryption format is `"playready"` output will be base64 encoded PlayReady Header Object. Otherwise, output will be base64 encoded PSSH version 0 box, with PlayReady Header Object in payload. The value of the PSSH box can be used directly in a DASH common encryption scenario.

Key and key ID need to be supplied by another module, such as Redis keystore.

3.11.1 Example

```
    "playready": {
        "type": "Playready"
    }
```

3.12 RedisKeystore

The RedisKeystore is not and shall not be regarded as an approved or tested Digital Rights Management ("DRM") system and shall not be used as or substituted for a DRM. AgileTV provides the RedisKeystore on an "as is basis" only without any conditions and makes no representations or warranties of any kind. Accordingly, to the maximum extent permitted by law AgileTV excludes all representations and warranties, express or implied, including without limitation any warranties for non-infringement, satisfactory quality, merchantability or fitness for any purpose.

Processor that can read and write drm info from localy accessible Redis database. Drm info that can be stored and retrieved for given resource id is: key, key id, iv and list of system specific drm data, where each system specific item can contain system id and optional header data.

Drm info ingestion is available as a separate eDRM endpoint. The subpath "`/add`" is used to add drm info for a given content into the system and the subpath '/remove' is used to remove keys from the system. For example, if provider is configured on "`redisKeystore`" path, then the URL "`/api/1/eDRM/redisKeystore/add/`" will be the endpoint for adding drm info into the keystore and "`/api/1/eDRM/redisKeystore/remove/`" will be the endpoint for removing drm info from the keystore.

Key adding and removal operations need to be successful on all nodes in order for eDRM to return success status. If a write operation fails on any of the nodes, an error response code is returned from eDRM, possibly leaving operation successful on some nodes. It is the responsibility of the caller to repair the state, either by re-issuing the command, or by other means, e.g. manual database cleaning.

Values in the keystore are not stored in plain text, but are encrypted with a AES in CBC mode with random IV values. The key for AES is based on a secret from the configuration with an HMAC-based Extract-and-Expand Key Derivation Function (HKDF).

Configuration parameters are:

- "`redis_endpoints`" (list of objects): List of Redis endpoints. Each endpoint is an object consisting of required string "`hostname`" and optional integer "`port`". The default port value is `6379`.
- "`redis_timeout`" (int): Optional timeout in seconds for Redis operations. Default value is `5`.
- "`redis_secret`" (string): Value for seeding encryption key for protecting keystore.
- "`redis_db`" (int): Optional Redis database id. Default value is `0`.

3.12.1 Example

```
"rediskeystore": {
    "type": "RedisKeystore",
    "redis_endpoints": [{"hostname":"localhost", "port": 6379}],
    "redis_timeout": 5,
    "redis_secret": "keystoresecret"
}
```

3.13 StaticKey

DRM processor that sets a static key from the configuration. If this processor only sets encryption information, then it needs to be chained with some other processor that sets DRM system specific information. In such a chaining, static key should be the first element. For example "`/cenc/statickey/widevine/playready`".

- "`type`" (string "`StaticKey`"): Required.
- "`key_id`" (string): Optional base64 encoded key ID used for this request.
- "`key`" (string): Optional base64 encoded key used for this request.
- "`iv`" (string): Optional base64 encoded iv used for this request.
- "`postion`" (string): Optional position value used in this request.
- "`header_data`" (string): Optional header data value that is appended in this request.
- "`system_id`" (string): Optional `system_id` value that is used with `header_data`.
- "`time_to_next_poll`" (int or false): Optional value for overriding `time_to_next_poll` in this request. The special value `false` clears any previously set `time_to_next_poll` value.

- "variants" (string array): Optional list of variants that will be set in key info.
- "plaintext" (boolean): Optional boolean value that marks "variants" in key info as plaintext. Applicable only if "variants" are set. The default value is false.
- "variant_grouper" (string) Optional grouper for variants in asset. Value can be one of the system provided grouper names. See variant groupers Applicable only if "variants" are not set. Each group returned by a grouper will share the same static key info.

3.13.1 Example with static key

```
"statickey": {
    "type": "StaticKey",
    "key_id": "H3JbV93QV3mPNBKQON2UtQ==",
    "key": "ClKhDPHMtCouEx1vLGsJsA=="
}
```

3.13.2 Example with position override

In the case that upstream DRM server is configured with key rotation, but key rotation is not used in repackager, position for channel content should be set to fixed value. We first define a processor with fixed position:

```
"static_position": {
    "type": "StaticKey",
    "position": "0"
}
```

and then in the output profile configuration we specify it as a first item in processors chain:

```
{
    "name": "dash-drm",
    "encryption": "cenc",
    "processors": ["static_position", "widevine", "playready", "clearkey
       ↪ "]
}
```

3.13.3 Example with variants

The StaticKey plugin can be used to serve more then one key for asset if we chain them. The ain constraint with this technique is that all variants must be listed in the response, so this type of output profile is applicable only for assets with same track configuration.

For example, we configure two processors with different keys, and one with plaintext flag:

```
"static_variants_video": {
    "type": "StaticKey",
    "key_id": "AAAAAAAAAAAAAAAAAAAAAA==",
    "key": "BBBBBBBBBBBBBBBBBBBBBB==",
    "header_data": "http://example/keyA",
    "variants": ["video_high", "video_low"]
},
"static_variants_audio": {
    "type": "StaticKey",
    "key_id": "CCCCCCCCCCCCCCCCCCCCCC==",
    "key": "DDDDDDDDDDDDDDDDDDDDDD==",
    "header_data": "http://example/keyC",
    "variants": ["audio"]
},
"static_plaintext": {
    "type": "StaticKey",
    "variants": ["subtitle_eng"],
```

```
        "plaintext": true
    }
```

and then in the output profile configuration we specify all items in a chain:

```
    {
        "name": "two_keys",
        "encryption": "aes-128",
        "processors": ["static_variants_video", "static_variants_audio", "
            ↪ static_plaintext"]
    }
```

3.14 Widevine

DRM processor for generic Widevine. The processing DRM context must have had the key and key ID set by another processor since this processor will use them to create a valid PSSH value.

Configuration parameters are:

- `"type"` (string `"Widevine"`): Required.
- `"provider"` (string): Optional provider name to be included in PSSH value.
- `"track_type"` (string): Optional track type to be included in PSSH value. Valid strings are `"SD"`, `"HD"` and `"AUDIO"`.
- `"pssh"` (string): Optional base64 encoded PSSH value to use instead of letting processor compute its own value based on key id, provider name et c.

Output will be base64 encoded PSSH version 0 box, with Widevine system ID in payload. The value of the PSSH box can be used directly in a DASH common encryption scenario.

Key and key ID need to be supplied by another module, such as Redis keystore.

3.14.1 Example

```
    "widevine": {
        "type": "Widevine",
        "provider": "Edgeware",
        "track_type": "SD"
    }
```

3.15 Axinom

Processor for integration with Axionom multi-DRM service. Meant to be used as a provider with Cenc.

- `"type"` (string `"Axinom"`): Required.
- `"drm"` (string): The specific DRM method to use. Supported methods are `"widevine"`, `"playready"` and `"fairplay"`.
- `"url"` (string): URL of Axinom key server.
- `"certificate"` (string or boolean): How to verify Axinom HTTPS connection: `true` to use system set of trusted root certificates, `false` to not verify Axinom certificate, or string value to specify file name with trusted root certificates. Not used if `"url"` is using plain `"http"` protocol.
- `"signing_name"` (string): Name used to sign request messages to Axinom. This and the other signing configurations are provided by Axinom.
- `"signing_key_hex"` (string): 32 byte AES key as a 64 character long hex string, used for generating request message signatures.
- `"signing_iv_hex"` (string): 16 byte AES IV as a 32 character long hex string, used for generating request message signatures.

- `playready_compatible_pssh_box` (boolean): Generate PSSH box using key,key ID and checksum and use it as `header_data`. Required by some of the old PlayReady players. Default value is `False`
- `playready_compatible_url` (boolean): Use Axinom's `skd_uri` as header data to signal used IV. Required by some of the old PlayReady players. Default value is `False`
- `fairplay_key_id_as_iv` (boolean): Use the Key ID as the IV and signal it in the header data, appended to the Key URL. The `playready_compatible_url` must be set to `True` for this option to take effect, which enforces building the custom Key URL, but creates new one as `skd://<Content_Id>:<KeyId_as_IV>`, instead of using `skd_uri` Note: The name `playready_compatible_url` is misleading, but is kept for backward compatibility. Default value is `False`

3.15.1 Examples

```
"axinom_widevine": {
    "type": "Axinom",
    "drm": "widevine",
    "url": "http://testserver/api/getcontentkey",
    "certificate": false,
    "connection_timeout": 10,
    "signing_name": "axinom",
    "signing_key_hex": "0123456789
        ↪ abcdef0123456789abcdef0123456789abcdef0123456789abcdef",
    "signing_iv_hex": "0123456789abcdef0123456789abcdef"
},
"axinom_playready": {
    "type": "Axinom",
    "drm": "playready",
    "url": "http://testserver/api/getcontentkey",
    "certificate": false,
    "connection_timeout": 10,
    "signing_name": "axinom",
    "signing_key_hex": "0123456789
        ↪ abcdef0123456789abcdef0123456789abcdef0123456789abcdef",
    "signing_iv_hex": "0123456789abcdef0123456789abcdef"
}
```

3.16 Castlabs

Processor for integration with Castlabs REST-API DRM service.

- `"type"` (string `"CastLabs"`): Required.
- `"login_url"` (string, required): URL of Castlabs service for login.
- `"key_url"` (string, required): URL of Castlabs service for ingesting key.
- `"username"` (string, required): The user name of API account.
- `"password"` (string, required): The password of API account.
- `"merchant_id"` (string, required): The identifying a merchant, provided by DRMtoday.
- `"ticket_ttl"` (number, required): Number of hour a ticket is valid. Every time login to Castlabs server, the server return a ticket. This ticket is used for all further requests.
- `"drm_schema"` (string array, required): An array specifying which DRM systems should be supported by the module instance. Valid strings are `"fairplay"` and `"cenc"`.
- `"key_seed_id"` (uuid string, required): Key seed used for generating keys. Must be provisioned in Castlabs system.
- `"iv_seed_id"` (uuid string, required): Key seed used for generating initialization vectors. Must be provisioned in Castlabs system.

3.16.1 Example

```
"castlabs": {
    "type": "CastLabs",
    "login_url": "https://auth.staging.drmtoday.com/cas/v1/tickets",
    "key_url": "https://fe.staging.drmtoday.com/frontend/api/keys/v2/
       ↪ ingest/",
    "username": "user1",
    "password": "pass1",
    "merchant_id": "merchant1",
    "ticket_ttl": 1,
    "drm_schema" : ["cenc", "fairplay"],
    "key_seed_id": "01020304-0506-0708-090a-0b0c0d0e0f00",
    "iv_seed_id": "11121314-1516-1718-191a-1b1c1d1e1f10"
}
```

3.17 Conax

Processor for integration with Conax SOAP-based DRM service.

- "type" (string "Conax"): Required.
- "url" (string): URL of Conax service end-point.
- "username" (string): Username for authentication to Conax service.
- "password" (string): Password for authentication to Conax service.
- "drm" (list of strings): A list of DRM systems to use in CENC encryption. This setting allows several DRM system values to be fetched from the Conax system in a single call, saving substantial overhead. Valid strings are "playready" and "widevine".
- "content_type" (string, optional): A string determining what type of content the Conax module is for, either "VOD" or "LIVE". If the option is omitted, "VOD" is chosen by default.
- "certificate" (string or boolean): How to verify Conax HTTPS connection: true to use system set of trusted root certificates, false to not verify Conax certificate, or string value to specify file name with trusted root certificates. Not used if "url" is using plain "http" protocol.
- "connection_timeout" (int): Timeout in seconds for requests towards key server. Default value is 10.

Supported encryption schemes are "cenc", "playready" and "sample-aes". In the case of "cenc" encryption, the "drm" option must be set and a Widevine and/or a PlayReady PSSH header(s) will be returned, depending on the value of the "drm" setting.

3.17.1 Example

```
"conax": {
    "type": "Conax",
    "username": "Edgeware",
    "password": "3dg3w4r3"
    "url": "https://testserver:44304/ca-server/webservices/key-server/
       ↪ conax",
    "certificate": false,
    "content_type": "LIVE",
    "drm": ["widevine", "playready"]
}
```

3.18 CPIX

Processor for integration with CPIX server for a multi DRM protection. Outgoing documents are signed based on key in configuration. Keys in response are expected to be in encrypted form. Note: Response is implicitly trusted, and not verified by remote signature.

- "type" (string "CPIX"): Required.
- "url" (string): URL of Fairplay service end-point.
- "certificate" (string or boolean): How to verify CPIX HTTPS connection: true to use system set of trusted root certificates, false to not verify CPIX certificate, or string value to specify file name with trusted root certificates. Not used if "url" is using plain "http" protocol.
- "connection_timeout" (int): Timeout in seconds for requests towards key server. Default value is 10.
- "key_file" (string): File name of a private key that will be used to sign CPIX request document. Private key should not be password protected.
- "cert_file" (string): File name of a certificate that will be embedded into CPIX request document.
- "full_signalling" (boolean): Controlls whether the complete content of the HLSSignallingData from the CPIX response is returned (true) or the URL is parsed out and returned (false). Defaults to true.
- "key_rotation_period" (int): Period for key rotation in seconds. The default value of 0 is interpreted as key rotation being disabled. See key rotation.
- "key_rotation_additional_keys" (int): Number of additional keys that will be sent in response when clients are making a request for period without end time. Default value is 2. See key rotation.
- "max_result_items" (int): Maximum number of keys in response. This parameter is used to protect overloading of CPIX server from large requests. Default value is 100. See key rotation.
- "key_rotation_period_offset" (int): Number of seconds that offsets each key period. Default value is 0. See key rotation.
- "key_rotation_poll_spread_percentage" (int or null): Percentage of key rotation period where the next poll can occur. Default value null does not calculate next poll time spread. See key rotation.
- "drms" (list): List of drm systems returned in response. If not set all available (fairplay, widevine, playready) drms will be served.
- "variant_grouper" (string) Optional grouper for variants in assets. The value can be one of the system-provided grouper names. See variant groupers.

Supported encryption schemes are "sample-aes" and cenc.

3.18.1 Example

```
    "cpix_fairplay": {
        "type": "CPIX",
        "url": "https://testserver:44304/cpix",
        "certificate": true,
        "key_file": "drm-gw.key",
        "cert_file": "drm-gw.cer",
        "drms" : ["fairplay", "playready", "widevine"]
    }
```

3.19 CPKeyGenerator

Processor for integration with CP Key Generator DRM service. The service is a key source for the adjacent key-push processors in the chain.

- "type" (string "CPKeyGenerator"): Required.
- "url" (string): URL of CP Key Generator service end-point.
- "certificate" (string or boolean): How to verify CP Key Generator HTTPS connection: true to use system set of trusted root certificates, false to not verify CPIX certificate, or string value to specify file name with trusted root certificates. Not used if "url" is using plain "http" protocol.
- "connection_timeout" (int): Timeout in seconds for requests towards key server. Default value is 10.
- "mode" (string): The distribution mode the plugin works in. Used for the resource context validation. Supported values are live or catch-up.

- "variant" (string): The variant the plugin works in. Used to define the key acquisition method and the DRM protocol signaling. Supported values are dash and mss. The mss variant acts as a standalone processor, the dash variant only acquires the key from the key source and must be chained with the PSSH-box generator(s) that the key is pushed to.

3.19.1 Example

```
"cp_keygenerator": {
    "type": "CPKeyGenerator",
    "url": "https://testserver:44304",
    "certificate": true,
    "mode": "live",
}
```

3.20 ExpressPlay

A module for interfacing with the ExpressPlay DRM system by Intertrust. Uses their key server to generate and store keys, and their PlayReady Token server to get a LA URL string.

PSSH data is generated by the module, or optionally hard coded through the configuration options for Widevine. FairPlay support is untested due to Apple certificate limitations, and the IV value is currently hard coded to "AAAAAAAAAAAAAAAAAAAAAA==".

- "type" (string "ExpressPlay"): Required.
- "customer_authentication" (string): A unique customer authentication string, provided by Intertrust.
- "key_url" (string): URL to the ExpressPlay key server.
- "key_certificate" (string or boolean): How to verify the HTTPS connection to the key server: true to use system set of trusted root certificates, false to not verify the key server certificate, or string value to specify file name with trusted root certificates. Not used if "url" is using plain "http" protocol.
- "key_encryption_key" (string): A 32 character long hex string, used by the key server to generate and store content encryption keys securely.
- "drm" (string array): An array specifying which DRM systems should be supported by the module instance. Valid strings are "widevine", "playready" and "fairplay". Widevine and PlayReady can be combined, but FairPlay must be used on its own.
- "playready_url" (string): The URL to an ExpressPlay PlayReady Token server, used to fetch LA URL to embed in the PlayReady PSSH. Is only used when the "drm" array contains "playready".
- "playready_certificate" (string or boolean): How to verify the HTTPS connection to the PlayReady Token server: true to use system set of trusted root certificates, false to not verify the key server certificate, or a string value to specify file name with trusted root certificates. Not used if "url" is using plain "http" protocol.
- "widevine_pssh" (string): An optional base64 encoded PSSH data string which can be used to override the otherwise dynamically generated PSSH data placed within the PSSH box.

ExpressPlay suggests using "CAESEAABAgMEBQYHCAkKCwwNDg8aCmludGVydHJ1c3QiASo=".

- "connection_timeout" (int): Timeout in seconds for requests towards key server. Default value is 10.

3.20.1 Examples

```
"expressplay_playready": {
    "type": "ExpressPlay",
    "customer_authentication": "Edgeware",
    "key_url": "https://testserver/keystore",
    "key_certificate": false,
    "key_encryption_key": "00112233445566778899aabbccddeeff",
    "drm": ["playready"],
    "playready_certificate": false,
    "playready_url": "https://testserver/hms/pr/",
```

```
        "connection_timeout": 10
    },
    "expressplay_widevine": {
        "type": "ExpressPlay",
        "customer_authentication": "Edgeware",
        "key_url": "https://testserver/keystore",
        "key_certificate": false,
        "key_encryption_key": "00112233445566778899aabbccddeeff",
        "drm": ["widevine"],
        "widevine_pssh": "CAESEAABAgMEBQYHCAkKCwwNDg8aCmludGVydHJ1c3QiASo=",
        "connection_timeout": 10
    },
    "expressplay_both": {
        "type": "ExpressPlay",
        "customer_authentication": "Edgeware",
        "key_url": "https://testserver/keystore",
        "key_certificate": false,
        "key_encryption_key": "00112233445566778899aabbccddeeff",
        "drm": ["widevine", "playready"],
        "widevine_pssh": "CAESEAABAgMEBQYHCAkKCwwNDg8aCmludGVydHJ1c3QiASo=",
        "playready_certificate": false,
        "playready_url": "https://testserver/hms/pr/",
        "connection_timeout": 10
    },
    "expressplay_fairplay": {
        "type": "ExpressPlay",
        "customer_authentication": "201301,a04e17a5741c475eb2844c45f92dc066",
        "key_url": "https://testserver/keystore",
        "key_encryption_key": "00112233445566778899aabbccddeeff",
        "drm": ["fairplay"],
        "connection_timeout": 10
    }
```

3.21 Irdeto

Processor for integration with Irdeto SOAP-based DRM service.

- `"type"` (string `"Irdeto"`): Required.
- `"url"` (string): URL of Irdeto service end-point.
- `"username"` (string): Username for authentication to Irdeto service.
- `"password"` (string): Password for authentication to Irdeto service.
- `"account_id"` (string): Account ID provided by Irdeto.
- `"kms_username"` (string): Key management system username, provided by Irdeto.
- `"kms_password"` (string): Key management system password, provided by Irdeto.
- `"drm"` (string array): List of DRM systems to use. Possible values are `"IrdetoProtection"`, `"PlayReady"` and `"Widevine"`.
- `"test_content_id"` (string): Optional test content ID that overrides the actual content ID for test purposes.
- `"content_prefix"` (string): Prefix prepended to the content id.
- `"content_postfix"` (string): Postfix appended to the content id.
- `"certificate"` (string or boolean): How to verify Irdeto HTTPS connection: `true` to use system set of trusted root certificates, `false` to not verify Irdeto certificate, or a string value to specify file name with trusted root certificates. Not used if `"url"` is using plain `"http"` protocol.
- `"connection_timeout"` (int): Timeout in seconds for requests towards key server. Default value is `10`.

- "key_validity_period" (int): Optional setting of the key validity period in seconds. The acquired key and the corresponding PSSH data are considered valid within that period and need to be refreshed when the time is up. The default value of 0 is interpreted as setting no expiration time.
- "crypto_period" (int): This value is used internally by Irdeto. Please refer to Irdeto document "Rights & Rules Manager API Reference" for more information. The default value is 1, which is also the lowest possible value.

The only supported encryption scheme is "cenc". The "drm" option may contain any and all of the three supported DRMs.

3.21.1 Example

```
"irdeto": {
    "type": "Irdeto",
    "url": "https://testserver/LiveDRMService/LiveDRMService.asmx",
    "certificate": false,
    "username": "Edgeware",
    "password": "3dg3w4r3"
    "account_id": "Edgeware",
    "kms_username": "Edgeware",
    "kms_password": "3dg3w4r3"
    "drm": ["IrdetoProtection", "PlayReady", "Widevine"],
    "content_prefix": "",
    "content_postfix": "CT"
}
```

3.22 IrdetoV2

A second version of the processor for integration with Irdeto SOAP-based DRM service (UEP API). It acts a key sink so it expects a key to be set before it starts processing it and pushes it to Irdeto DRM service.

- "type" (string "IrdetoV2"): Required.
- "url" (string): URL of Irdeto service end-point.
- "account_id" (string): Account ID provided by Irdeto.
- "kms_username" (string): Key management system username, provided by Irdeto.
- "kms_password" (string): Key management system password, provided by Irdeto.
- "drm" (string array): List of DRM systems to use. Possible values are "IrdetoProtection", "PlayReady" and "Widevine".
- "content_prefix" (string): Prefix prepended to the content id.
- "content_postfix" (string): Postfix appended to the content id.
- "key_validity_period" (int): Optional setting of the key validity period in seconds. The acquired key and the corresponding PSSH data are considered valid within that period and need to be refreshed when the time is up. The default value of 0 is interpreted as setting no expiration time.

The only supported encryption scheme is "cenc". The "drm" option may contain any and all of the three supported DRMs.

3.22.1 Example

```
"irdetov2": {
    "type": "IrdetoV2",
    "url": "https://testserver/LiveDRMService/LiveDRMService.asmx",
    "account_id": "Edgeware",
    "kms_username": "Edgeware",
    "kms_password": "3dg3w4r3"
    "drm": ["IrdetoProtection", "PlayReady", "Widevine"],
    "content_prefix": "",
```

```
        "content_postfix": "CT"
    }
```

3.23 IrdetoCPIX

The processor for integration with the Irdeto CPIX-based DRM service (CPIX API v2). The plugin acts as a key source and retrieves the keys from Irdeto's The Key Maker (TKM) using the CPIX protocol underneath, for various drm systems. Please refer to Irdeto documentation

- "type" (string "IrdetoCPIX"): Required.
- "ic_host" (string): Host of Irdeto service end-point.
- "account_id" (string): Tenant account ID provided by Irdeto.
- "tkm_username" (string): The Key Maker username, provided by Irdeto.
- "tkm_password" (string): The Key Maker password, provided by Irdeto
- "drm" (string array): List of DRM systems to use.
- "content_prefix" (string): Prefix prepended to the content id.
- "content_postfix" (string): Postfix appended to the content id.

The only supported *common* encryption scheme is "cenc",but the "drm" option must contain DRMs which support the drm signaling, based on *requested* enctiption, i.e. if *requested* encryption is "cenc" then selected DRMs must support PSSH protocol, but if "aes-128" is requested then supported protocol is HLS Signaling

3.23.1 Example

```
    "irdetocpix": {
        "type": "IrdetoCPIX",
        "ic_host": "1.2.3.4",
        "account_id": "Edgeware",
        "tkm_username": "Edgeware",
        "tkm_password": "3dg3w4r3"
        "drm": ["IrdetoProtection", "PlayReady", "Widevine"],
    }
```

3.24 Cust1CPIX

The processor for integration with the CPIX-based DRM service (CPIX API v2).

- "type" (string "Cust1CPIX"): Required.
- "host" (string): Host of service end-point.

Drm service will generate a GET request to the following path:

```
    {url}/cpix/anevia/instances/{instance_id}/type/{playback_type}/channel/{
        ↪ channel_name}
```

Where:

- {url}: The server domain
- {instance_id}: It will be the id of the instance, that can be used to track the requests from the different headends
- {playback_type}: The type of playback that will allow two possible values (live, npvr)
- {channel_name}: Name of channel using DRM

Supported encryption schemes are "sample-aes" and cenc.

3.24.1 Example

```
"cust1cpix": {
    "type": "Cust1CPIX",
    "host": "<addr>:<port>/cpix/anevia/instances/28/type/live/channel/"
}
```

3.25 DaznCPIX

The processor for integration with the CPIX-based DRM service (CPIX API v2). The plugin acts as a key source and retrieves the keys from DAZN server end-point using the CPIX protocol underneath, for various DRM systems.

The request/response is authenticated via HMAC Authentication. HMAC Verification ensures that both the client and server have access to a single shared secret key. It does this by signing the HTTP Payload of both the request and the response. This helps the server verify that the client is authorised, and helps the client verify that the server hasn't been compromised.

- `"type"` (string `"DaznCPIX"`): Required.
- `"host"` (string): Host of DAZN service end-point.
- `"path"` (string): Normalised request path.
- `"drm"` (string array): List of DRM systems to use. Possible values are `"FairPlay"`, `"PlayReady"` and `"Widevine"`. Default value is `"PlayReady"` and `"Widevine"`.
- `"verify_response"` (boolean): Optionally verify response if it was properly signed throws exception if invalid response was returned. Default value is `true`.
- `"auth_Key"` (string): Identifying secret used to sign request.
- `"secret_content"` (string): Secrets used to sign request.

The `"cenc"` encryption for single DRM system `"PlayReady"` and `"Widevine"` or multi-DRM, `"cenc"` encrypyion supports `PSSH` protocol. The `"sample-aes"` encryption supports `"FairPlay"` DRM system and supports `HLS Signaling` protocol. The `"playready"` encryption supports `"PlayReady"` DRM system and supports `SmoothStreaming` protocol.

3.25.1 Example

```
"dazncpix": {
    "type": "DaznCPIX",
    "host": "drm-registration.playback.dazn-stage.com",
    "path": "/v1/cpix",
    "drm": ["widevine","playready"],
    "verify_response": true
    "auth_Key": "auth_Key/auth_Key",
    "secret_content": "
        ↪ aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
}
```

3.26 Kaltura

DRM processor for integration with Kaltura's uDRM key server. Configuration options are:

- `"type"` (string `"Kaltura"`): Required.
- `"url"` (string): URL prefix of Kaltura service end-point.
- `"default_format"` (string): Default encryption format that will be appended to Kaltura URL, if none is provided in input URL. If the value is empty string, no additional path elements will be appended to Kaltura URL.
- `"account_id"` (string): Kaltura account ID. Account ID is provided as part of the account onboarding, see Kaltura documentation for more information.

- "key" (string): Private key for Kaltura message signing. See Kaltura documentation for how to obtain one. Format is plain string.
- "certificate" (string or boolean): How to verify Kaltura HTTPS connection: true to use system set of trusted root certificates, false to not verify Kaltura certificate, or string value to specify file name with trusted root certificates. Not used if "url" is using plain "http" protocol.
- "connection_timeout" (int): Timeout in seconds for requests towards Kaltura. Default value is 10.

Configuration examples for common cases, with eDRM clients requesting keys for "playready" encryption:

- For Kaltura URL "http://kaltura-server/playready/encryption", configuration excerpt is:

```
      "url": "http://kaltura-server/"
      "default_format": ""
```

- For Kaltura URL "http://kaltura-server/playready/prtocencwv/encryption", configuration excerpt is:

```
      "url": "http://kaltura_server/"
      "default_format": "prtocencwv"
```

3.26.1 Example

```
    "kaltura": {
        "type": "Kaltura",
        "url": "http://testserver:12345/playready/",
        "default_format": "",
        "account_id": "1",
        "key": "kiQP23HCpvF1HNmI8UBoxzk1bo884wTqshmm8MPkuso=",
        "certificate": "",
        "connection_timeout": 10
    }
```

3.27 Nagra

Processor for integration with Nagra SOAP-based DRM service.

- "type" (string "Nagra"): Required.
- "url" (string): URL of Nagra service end-point.
- "username" (string): Username for authentication with Nagra service.
- "password" (string): Password for authentication with Nagra service.
- "content_prefix" (string): Prefix appended to content ID before sending to Nagra API.
- "disable_suffix" (boolean): Remove the suffix in content ID. Default value is false.
- "emi" (string): Optional emi parameter, provided by Nagra. No DRM list is required if emi is set.
- "distribution_mode" (string): Nagra distribution mode parameter. Possible values are "LIVE" or "VOD"
- "drm" (string array): Optional list of DRM systems to use. Possible values are "NagraMediaPRM", "PlayReady" and "Marlin".
- "certificate" (string or boolean): How to verify Nagra HTTPS connection: true to use system set of trusted root certificates, false to not verify Nagra certificate, or a string value to specify file name with trusted root certificates. Not used if url is using plain http protocol.
- "connection_timeout" (int): Timeout in seconds for requests towards key server. Default value is 10.

3.27.1 Examples

```
    "nagra": {
        "type": "Nagra",
        "url": "http://testserver/ind1/cks-ws-keyAndSignalization/key?wsdl",
        "username": "Edgeware",
        "password": "3dg3w4r3",
        "content_prefix": "EW_",
        "emi": "12345",
        "distribution_mode": "LIVE",
        "certificate": false
    },
    "nagra_all": {
        "type": "Nagra",
        "url": "http://testserver/ind1/cks-ws-keyAndSignalization/key?wsdl",
        "username": null,
        "password": null,
        "content_prefix": "EW_",
        "distribution_mode": "LIVE",
        "drm": ["NagraMediaPRM", "PlayReady", "Marlin"],
        "certificate": false
    },
    "nagra_playready": {
        "type": "Nagra",
        "url": "http://testserver/ind1/cks-ws-keyAndSignalization/key?wsdl",
        "content_prefix": "EW_",
        "distribution_mode": "VOD",
        "drm": ["PlayReady"],
        "certificate": false
    }
```

Supported encryption schemes are `"cenc"`, `"playready"`, `"sample-aes"` and `"aes-128"`. In the case of `"sample-aes"` and `"aes-128"` encryption, The `"fairplay"` DRM will be chosen if having multiple DRMs.

3.28 Nagra V2

A second version of the processor for integration with Nagra SOAP-based DRM service. It acts a key sink so it expects a key to be set before it starts processing it and pushes it to Nagra DRM service..

- `"type"` (string `"NagraV2"`): Required.
- `"url"` (string): URL of Nagra service end-point.
- `"prefix"` (string): Prefix appended to content ID before sending to Nagra API.
- `"emi"` (string): Optional emi parameter, provided by Nagra. No DRM list is required if emi is set.
- `"distribution_mode"` (string): Nagra distribution mode parameter. Possible values are `"LIVE"` or `"VOD"`
- `"drms"` (string array): Optional list of DRM systems to use. Possible values are `"NagraMediaPRM"`, `"PlayReady"` and `"Marlin"`.
- `"certificate"` (string or boolean): How to verify Nagra HTTPS connection: `true` to use system set of trusted root certificates, `false` to not verify Nagra certificate, or a string value to specify file name with trusted root certificates. Not used if `url` is using plain `http` protocol.
- `"connection_timeout"` (int): Timeout in seconds for requests towards key server. Default value is `10`.

3.28.1 Examples

```
    "nagrav2": {
        "type": "NagraV2",
        "url": "http://testserver/ind1/cks-ws-keyAndSignalization/key?wsdl",
        "cprefix": "EW_",
        "emi": "12345",
```

```
        "distribution_mode": "VOD",
        "certificate": false
    },
```

3.29 SecureMedia

Processor for SecureMedia KVHLS server integration. Supported encryption is "aes-128" only. Configuration options are:

- "type" (string "SecureMedia"): Required.
- "url" (string): URL prefix of SecureMedia KVHLS service end-point. For example: "http://kvhls.test
  ↪ .server.com:12684/"
- "stream_type" (string): SecureMedia supports using different keys for VOD streams and live streams. Select between them by setting "stream_type" to "VOD" and "DTV", respectively.
- "certificate" (string or boolean): How to verify SecureMedia HTTPS connection: true to use system set of trusted root certificates, false to not verify SecureMedia certificate, or string value to specify file name with trusted root certificates. Not used if "url" is using plain "http" protocol.
- "connection_timeout" (int): Timeout in seconds for requests towards SecureMedia key server. Default value is 10.
- "key_rotation_period" (int): Period for key rotation in seconds. The default value of 0 is interpreted as key rotation being disabled. See key rotation.
- "key_rotation_additional_keys" (int): Number of additional keys that will be sent in response when clients are making a request for period without end time. Default value is 2. See key rotation.
- "max_result_items" (int): Maximum number of keys in response. This parameter is used to protect overloading of SecureMedia server from large number of requests. Default value is 100. See key rotation.
- "key_rotation_period_offset" (int): Number of seconds that offsets each key period. Default value is 0. See key rotation.
- "key_rotation_poll_spread_percentage" (int or null): Percentage of key rotation period where the next poll can occur. Default value null does not calculate next poll time spread. See key rotation.

Supported encryption is "aes-128" only

3.29.1 Examples

```
    "securemedia" : {
        "type": "SecureMedia",
        "url": "http://kvhls-server",
        "certificate": "edrm_plugin/test/cfg/server.crt",
        "connection_timeout": 10,
        "stream_type": "DTV",
        "key_rotation_period": 10
    }
```

3.30 Ubique

Processor for integration Ubique server with Latens DRM. Configuration options are:

- "type" (string "Ubique"): Required.
- "url" (string): URL of the Ubique service end-point. For example: ‘"http://188.121.2.242:18080/ubique/mw"
- "username" (string): Username for authentication with the Ubique service.
- "password" (string): Password for authentication with the Ubique service.
- "certificate" (string or boolean): How to verify Ubique HTTPS connection: true to use system set of trusted root certificates, false to not verify the Ubique certificate, or a string value to specify file name with trusted root certificates. Not used if url is using plain http protocol.

- "connection_timeout" (int): Timeout in seconds for requests towards Ubique key server. Default value is 10.
- "swap_keyid_bytes" (boolean) Swap endianness in received key id data. Default value is true

3.30.1 Example:

```
"ubique": {
    "type": "Ubique",
    "url": "http://188.121.2.242:18080/ubique/mw",
    "username":"admin",
    "password":"admin",
    "certificate": false,
    "connection_timeout": 10
}
```

3.31 VerimatrixVEI36

DRM processor for integration with Verimatrix's VCAS key server version 3.6. Configuration options are:

- "type" (string "VerimatrixVEI36"): Required.
- "url" (string): URL prefix of Verimatrix service end-point. For example: "http://public-ott.verimatrix.com:12684/"
- "stream_type" (string): Verimatrix supports using different keys for VOD streams and live streams. Select between them by setting "stream_type" to "VOD" and "DTV", respectively.
- "company_name" (string): Verimatrix account identifier. Prefix is appended to resource ID in communication with Verimatrix key server. Can be empty string.
- "certificate" (string or boolean): How to verify Verimatrix HTTPS connection: true to use system set of trusted root certificates, false to not verify Verimatrix certificate, or string value to specify file name with trusted root certificates. Not used if "url" is using plain "http" protocol.
- "connection_timeout" (int): Timeout in seconds for requests towards Verimatrix key server. Default value is 10.
- "key_rotation_period" (int): Period for key rotation in seconds. The default value of 0 is interpreted as key rotation being disabled. See key rotation.
- "key_rotation_additional_keys" (int): Number of additional keys that will be sent in response when clients are making a request for period without end time. Default value is 2. See key rotation.
- "max_result_items" (int): Maximum number of keys in response. This parameter is used to protect overloading of Verimatrix server from large number of requests. Default value is 100. See key rotation.
- "key_rotation_period_offset" (int): Number of seconds that offsets each key period. Default value is 0. See key rotation.
- "key_rotation_poll_spread_percentage" (int or null): Percentage of key rotation period where the next poll can occur. Default value null does not calculate next poll time spread. See key rotation.
- "variant_grouper" (string) Optional grouper for variants in assets. The value can be one of the system-provided grouper names. See variant groupers.

Supported encryption is "aes-128" and output from processor is key server URL.

3.31.1 Example

```
"verimatrix": {
    "type": "VerimatrixVEI36",
    "url": "http://testserver:12345/verimatrix/",
    "stream_type": "VOD",
    "company_name": "Edgeware_",
    "certificate": "",
    "connection_timeout": 10
```

```
    },
    "verimatrix_key_rotation": {
        "type": "VerimatrixVEI36",
        "url": "http://testerver:12345/verimatrix/",
        "stream_type": "DTV",
        "company_name": "Edgeware_",
        "certificate": "",
        "connection_timeout": 10,
        "key_rotation_period": 60,
        "key_rotation_additional_keys": 3
    }
```

3.32 VerimatrixVEI38

DRM processor for integration with Verimatrix's VCAS key server version 3.8.

Responses for `verimatrix` drm type can be cached in the plugin. Same type of cache that is used for output profiles is used in the plugin. See output profiles cache for a details on cache functionality. Cache key is `(resource_id, position)` pair.

Configuration options are:

- `"type"` (string `"VerimatrixVEI38"`): Required.
- `"url"` (string): URL prefix of Verimatrix service end-point. For example: `"http://public-ott.verimatrix.com:12684/"`.
- `"drm"` (string): Which DRM type to get from the Verimatrix server. Valid options are `"verimatrix"`, `"widevine"`, and `"playready"`. Note that different DRM systems will likely use different server URLs.
- `"stream_type"` (string): Verimatrix supports using different keys for VOD streams and live streams. Select between them by setting `"stream_type"` to `"VOD"` and `"DTV"`, respectively.
- `"use_content_id"` (boolean): If set to true, uuid version of resource_id will be used in communication with Verimatrix backend for Widevine requests. When set to false, resource_id is used. Default value: `true`.
- `"binary_content_id"` (boolean): Used only when `use_content_id` is `true`. If set to true, uuid version of resource_id is used in binary format. When set to false, uuid version is used in hexadecimal string format. Default value: `true`.
- `"company_name"` (string): Verimatrix account identifier. Prefix is appended to resource ID in communication with Verimatrix key server. Can be empty string.
- `"certificate"` (string or boolean): How to verify Verimatrix HTTPS connection: `true` to use system set of trusted root certificates, `false` to not verify Verimatrix certificate, or string value to specify file name with trusted root certificates. Not used if `"url"` is using plain `"http"` protocol.
- `"connection_timeout"` (int): Timeout in seconds for requests towards Verimatrix key server. Default value is `10`.
- `"cache_size"` (int): If set to value positive value responses for CCMI requests will be cached in a LRU cache of specified size. Setting the cache size to `0` disable the caching. Default value is `0`.
- `"cache_max_age"` (int): Max age in seconds for elements in the cache. If the accessed cache entry is older than this value, it will be removed from the cache, and new value will be calculated. Default value is `86400` (one day).
- `"cache_result_timeout"` (int): Advanced tuning parameter that specifies time in seconds that requests are willing to wait for an ongoing cache set operation. Default value is `30`.

Depends on the value of option `"drm"`, there are additional configuration options:

- If the value is `"verimatrix"`: key rotation is supported
  - `"key_rotation_period"` (int): Period for key rotation in seconds. The default value of `0` is interpreted as key rotation being disabled.
  - `"key_rotation_additional_keys"` (int): Number of additional keys that will be sent in response when clients are making a request for period without end time. Default value is `2`.

  - "max_result_items" (int): Maximum number of keys in response. This parameter is used to protect overloading of Verimatrix server from large number of requests. Default value is 100.
  - "key_rotation_period_offset" (int): Number of seconds that offsets each key period. Default value is 0.
  - "key_rotation_poll_spread_percentage" (int or null): Percentage of key rotation period where the next poll can occur. Default value null does not calculate next poll time spread.

- If the value is "widevine": signing information has to be provided
  - "signing_name" (string)
  - "signing_key_hex" (string)
  - "signing_iv_hex" (string)

Supported encryption is "playready" if "drm" is "playready", "cenc"—if "drm" is either "widevine" or "playready"—or "aes-128" for any "drm" value. Output for "verimatrix" is the key server URL, and a PSSH box for the other types.

When using common encryption "cenc" with both Widevine and PlayReady from this module, place the Widevine processor before the PlayReady processor in outputProfiles to ensure that the key is shared properly between the two systems and to avoid multiple requests to the Verimatrix server. See example below.

3.32.1 Example

```
"drm_processors": {
  "verimatrix38_rotation": {
    "type": "VerimatrixVEI38",
    "url": "http://testserver:12684/",
    "drm": "verimatrix",
    "stream_type": "DTV",
    "company_name": "Edgeware_",
    "certificate": "",
    "connection_timeout": 10
    "key_rotation_period": 60,
    "key_rotation_additional_keys": 3
  },
  "verimatrix38_wv": {
    "type": "VerimatrixVEI38",
    "url": "http://testserver:8055/",
    "drm": "widevine",
    "stream_type": "DTV",
    "company_name": "Edgeware_",
    "certificate": "",
    "connection_timeout": 10,
    "signing_name": "widevine_test",
    "signing_key_hex": "1
       ↪ ae8ccd0e7985cc0b6203a55855a1034afc252980e970ca90e5202689f947ab9",
    "signing_iv_hex": "d58ce954203b7c9a9a9d467f59839249"
  },
  "verimatrix38_pr": {
    "type": "VerimatrixVEI38",
    "url": "http://testserver:12681/",
    "drm": "playready",
    "stream_type": "DTV",
    "company_name": "Edgeware_",
    "certificate": "",
    "connection_timeout": 10
  }
},
"accounts": [
  {
    "name": "default",
    "outputProfiles": [
      {
```

```
        "name": "drm-profile",
        "encryption": "cenc",
        "processors": [
          "verimatrix38_wv", "verimatrix38_pr"
        ]
      },
      {
        "name": "drm-rotate",
        "encryption": "cenc",
        "processors": [
          "verimatrix38_rotation"
        ]
      }
    ]
  }
]
```

3.33 VerimatrixCEI

DRM processor for integration with Verimatrix 4.1/4.2 VCAS key server. Configuration options are:

- `"type"` (string `"VerimatrixCEI"`): Required.
- `api_version` Valid values are: `41` and `42` for VCAS 4.1 and VCAS 4.2 respectively.
- `"drm"` (string array) one or more of the following: `"VERIMATRIX_ITV"`, `"PLAYREADY"`, `"WIDEVINE"`, `"FAIRPLAY_STREAMING"`. Apple Fairplay can only be used with `"api_version": 42` (or later). Only a single value is supported for `"aes-128"`, `"sample-aes"` and `"playready"` encryption, but `"cenc"` accepts any combination of `"VERIMATRIX_ITV"`, `"PLAYREADY"` and `"WIDEVINE"` at the same time. For example: `["PLAYREADY", "VERIMATRIX_ITV"]`.
- `"url"` (string): URL prefix of Verimatrix service end-point. For example:`"http://public-ott.verimatrix.com:12684/"`.
- `"company_name"` (string): Verimatrix account identifier. Prefix is appended to resource ID in communication with Verimatrix key server. Can be empty string.
- `"certificate"` (string or boolean): How to verify Verimatrix HTTPS connection: `true` to use system set of trusted root certificates, `false` to not verify Verimatrix certificate, or string value to specify file name with trusted root certificates. Not used if `"url"` is using plain `"http"` protocol.
- `"connection_timeout"` (int): Timeout in seconds for requests towards Verimatrix key server. Default value is `10`.
- `"site_id"` (int): Can be any integer. In the case of multiple Verimatrix deployments, the site id together with the content id is a unique identifier across the Verimatrix deployments. Default value: `1`.
- `"use_content_id"` (boolean): If set to true, uuid version of resource_id will be used in communication with Verimatrix backend. This can help in situations where resource_id contains `/` characters, for example VOD managed content. Default value: `false`.
- `"key_rotation_period"` (int): Period for key rotation in seconds. The default value of `0` is interpreted as key rotation being disabled. See key rotation.
- `"key_rotation_additional_keys"` (int): Number of additional keys that will be sent in response when clients are making a request for period without end time. Default value is `2`. See key rotation.
- `"max_result_items"` (int): Maximum number of keys in response. This parameter is used to protect overloading of Verimatrix server from large number of requests. Default value is `100`. See key rotation.
- `"key_rotation_period_offset"` (int): Number of seconds that offsets each key period. Default value is `0`. See key rotation.
- `"key_rotation_poll_spread_percentage"` (int or null): Percentage of key rotation period where the next poll can occur. Default value `null` does not calculate next poll time spread. See key rotation.

Supported encryptions are `playready`, `aes-128`, `sample-aes` and `cenc`.

3.33.1 Example

```
"verimatrix41": {
    "type": "VerimatrixCEI",
    "drm": ["VERIMATRIX_ITV", "WIDEVINE", "PLAYREADY"],
    "url": "http://testserver:8058/",
    "company_name": "Edgeware_",
    "certificate": "",
    "connection_timeout": 10,
    "site_id": "1234",
    "api_version": 41
},
"verimatrix41_rotation": {
    "type": "VerimatrixCEI",
    "drm": ["VERIMATRIX_ITV", "WIDEVINE", "PLAYREADY"],
    "url": "http://testserver:8058/",
    "company_name": "Edgeware_",
    "certificate": "",
    "connection_timeout": 10,
    "site_id": "1234",
    "api_version": 41,
    "key_rotation_period": 60,
    "key_rotation_additional_keys": 3
}
```

3.34 VerimatrixCPIX

The processor for integration with the CPIX-based DRM service (CPIX API v2). The plugin acts as a key source, retrieving keys from the server endpoint using the CPIX protocol for various DRM systems.

Token-based authentication is employed to verify requests. Each CPIX request must include an authentication token which is added to the HTTP headers.

Configuration options are:

- `"type"` (string `"VerimatrixCPIX"`): Required.
- `"url"` (string): Required. The URL prefix of Verimatrix service end-point. For example: `"https://multidrm.core.verimatrixcloud.net/cpix/v2.0"`.
- `"issuer"` (string): Required. The `"issuer"` claim identifies the issuer of the token. The issuer claim value is defined by Verimatrix.
- `"key_id_signer"` (string): Required. The Key Id claim identifies the key used to sign the token with. KeyIDs are assigned by Verimatrix.
- `"private_key_path"` (string): Required. The path to the private key file. Verimatrix requires that tokens being received are signed using a private key.
- `"expiration_time"` (int): It is an integer value that expresses the epoch time when the license is to expire. Default is 300.
- `"drms"` (string array): List of DRM systems to use. Possible values are `"FairPlay"`, `"PlayReady"` and `"Widevine"`. Default value is `"PlayReady"` and `"Widevine"`. Note: `"FairPlay"` cannot be used at the same time with the other 2 DRM systems.
- `"key_rotation_period"` (int): Period for key rotation in seconds. The default value of `0` is interpreted as key rotation being disabled. See key rotation.
- `"key_rotation_additional_keys"` (int): Number of additional keys that will be sent in response when clients are making a request for period without end time. Default value is `2`. See key rotation.
- `"max_result_items"` (int): Maximum number of keys in response. This parameter is used to protect overloading of Verimatrix server from large number of requests. Default value is `100`. See key rotation.
- `"key_rotation_period_offset"` (int): Number of seconds that offsets each key period. Default value is `0`. See key rotation.

- "key_rotation_poll_spread_percentage" (int or null): Percentage of key rotation period where the next poll can occur. Default value null does not calculate next poll time spread. See key rotation.
- "variant_grouper" (string) Optional grouper for variants in assets. The value can be one of the system-provided grouper names. See variant groupers.

Supported encryptions are sample-aes and cenc.

The "cenc" encryption for single DRM system "PlayReady" and "Widevine" or multi-DRM, "cenc" encrypyion supports PSSH protocol. The "sample-aes" encryption supports "FairPlay" DRM system and supports HLS Signaling protocol.

3.34.1 Example

```
"verimatrixcpix-rotation": {
    "type": "VerimatrixCPIX",
    "url": "https://multidrm.core.verimatrixcloud.net/cpix/v2.0",
    "issuer": "agile_packager",
    "key_id_signer": "aaaaaaa-1111-bbbb-2222-cccccccccccc",
    "private_key_path": "/etc/edgeware/drm-gw/key-cpix-private.pem",
    "expiration_time": 300,
    "drms": ["widevine", "playready"],
    "key_rotation_period": 30,
    "key_rotation_additional_keys": 2,
    "variant_grouper": "video_3G_audio_1G"
},
```

3.35 Verimatrix53CPIX

The DRM processor for integration with Verimatrix 5.3 VCAS key server and CPIX API.

Configuration options are:

- "type" (string "Verimatrix53CPIX"): Required.
- "drm" (string array) one or more of the following: "Widevine", "Fairplay".
- "url" (string): URL prefix of Verimatrix service end-point. For example: "http://verimatrix-server.com:8058/cpix/v1.1/".
- "prefix" (string): Prefix appended to content ID before sending to server.
- "key_rotation_period" (int): Period for key rotation in seconds. The default value of 0 is interpreted as key rotation being disabled. See key rotation.
- "key_rotation_additional_keys" (int): Number of additional keys that will be sent in response when clients are making a request for period without end time. Default value is 2. See key rotation.
- "max_result_items" (int): Maximum number of keys in response. This parameter is used to protect overloading of Verimatrix server from large number of requests. Default value is 100. See key rotation.
- "key_rotation_period_offset" (int): Number of seconds that offsets each key period. Default value is 0. See key rotation.
- "key_rotation_poll_spread_percentage" (int or null): Percentage of key rotation period where the next poll can occur. Default value null does not calculate next poll time spread. See key rotation.

Supported encryptions are sample-aes and cenc.

The "cenc" encryption scheme is used with the "Widevine" DRM system and supports the PSSH format. The "sample-aes" encryption scheme is used with the "FairPlay" DRM system and supports HLS Signalingformat.

3.35.1 Example

```
    "verimatrix53": {
        "type": "Verimatrix53CPIX",
        "url": "http://verimatrix-server.com:8058/cpix/v1.1/",
        "drm": "Widevine",
```

```
        "prefix": "dash",
        "key_rotation_period": 60,
        "key_rotation_additional_keys": 3
    }
```

3.36 TestAxinomWidevine

DRM processor for public Axinom widevine testing server.

Configuration parameters are:

- "type" (string "TestAxinomWidevine"): Required.

Uses hardcoded values:

- key_id: "nrQFDeRLSAKTLifXUIPiZg=="
- key: "FmY0xnWCPCNaSpRG+tUuTQ=="
- pssh: "AAAANHBzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAABQIARIQnrQFDeRLSAKTLifXUIPiZg=="

Licensing URL is "http://drm-widevine-licensing.axtest.net/AcquireLicense" and licensing request needs to have a custom HTTP header "X-AxDRM-Message" with the value:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ2ZXJzaW9uIjoxLCJjb21f
a2V5X2lkIjoiYjMzNjRlYjUtNTFmNi00YWUzLThjOTgtMzNjZWQ1ZTMxYzc4I
iwibWVzc2FnZSI6eyJ0eXBlIjoiZW50aXRsZW1lbnRfbWVzc2FnZSIsImtleX
MiOlt7ImlkIjoiOWViNDA1MGQtZTQ0Yi00ODAyLTkzMmUtMjdkNzUwODNlMjY
2IiwiZW5jcnlwdGVkX2tleSI6ImxLM09qSExZVzI0Y3Iya3RSNzRmbnc9PSJ9
XX19.4lWwW46k-oWcah8oN18LPj5OLS5ZU-_AQv7fe0JhNjA
```

Since encryption info is hardcoded, it has to be the first provider in the chain when used with other CENC providers.

3.36.1 Example

```
"test_widevine": {
    "type": "TestAxinomWidevine"
}
```

3.37 TestPlayready

DRM processor for public Microsoft PlayReady key server. If the processing DRM context has key and key_id set, the processor will use them and the licence URL will contain the wanted key. If key and key_id are not set, the processor will set and use values from the configuration. In this case the licensing URL will not contain any key. Key and checksum can be calculated from key_id with default seed based on Microsoft's alogrithm.

- "type" (string "TestPlayready"): Required.
- "key_id" (string): Optional key ID used for this request. Defaults to "EAAAABAAEAAQABAAAAAAAQ==".
- "key" (string): Optional key used for this request. Defaults to "OiobaN0r2bLusl6ExHdmaA==".
- "checksum" (string): Optional license checksum used for this request. Defaults to "5TzIYQ2hrOY=".

If encryption format is "playready" output will be base64 encoded PlayReady Header Object. Otherwise, output will be base64 encoded PSSH version 0 box, with PlayReady Header Object in payload. The value of the PSSH box can be used directly in DASH common encryption scenario.

3.37.1 Example

```
"test_playready": {
    "type": "TestPlayready"
}
```

3.38 Event reporting and latency monitoring

DRM-GW supports reporting events to Convoy or standalone Kafka instance. In order to enable it include a `kafka_logger` configuration element. In the configuration snippet below the configuration options are shown. The `address` parameter needs to point to a functional Kafka instance, the `topic` parameter sets the Kafka topic to which the events will be sent to.

```
    "kafka_logger": {
        "address": "127.0.0.1:8887",
        "topic": "events"
    },
    "latency_thresholds": {
        "minor": {
            "50%": 50,
            "90%": 100,
            "99%": 250
        },
        "major": {
            "50%": 250,
            "90%": 500,
            "99%": 1000
        }
    }
```

Additionally the application can report latency statistics. The `latency_thresholds` element configures request handling percentile thresholds (in milliseconds). In the example shown above the application will send a notification event with minor severity when the median (50th percentile) of the handling time is larger than 50ms. Similarly when the median processing time is above 250ms an event with major severity will be sent.

4 Operation

After installation and configuration, the service is operated as a systemd service named `ew-drm-gw`. For example, to start service:

```
systemctl start ew-drm-gw
```

5 Logging

5.1 Service log

DRM Gateway service log is by default stored in systemd's journal.

Log entris can be viewed with `journalctl` command, for example:

```
# journalctl -u ew-drm-gw
```

Below is a list of log levels and what's logged to them (default INFO):

- ERROR:
  - when an exception is raised when handling a request from DRM server.
  - if something goes wrong in other parts of the system.
- WARNING:
  - used by requests library for warnings.
- INFO:
  - when a request is received and is about to return a response.
  - when a request is generated and is about to get a response.
  - application starting and shutting down.
- DEBUG:
  - displays all content coming in and out of the server.
  - more details regarding request processing.

The logs is of the following structure:

```
<log level> - [<logger name>] - <message>
```

For example:

```
INFO    [root] eDRM starting
```

5.2 HTTP access log

HTTP access log is stored by default in `/var/log/edgeware/drm-gw/drm-gw-access.log` file.

HTTP access log has default format:

```
<remote address> - - [<timestamp>] "<request line>" <status> <response length
    ↪ > <request time>
```

For example:

```
127.0.0.1 - - [11/Dec/2025:13:39:52 +0700] "POST /__cl/s:esf/__c/content_path
    ↪ /__op/drm-profile/__f/manifest.mpd HTTP/1.1" 200 2025 0.002146
```

6 StreamViewer Monitoring

6.1 Introduction

The SW Repackager (ESB3002), SW LiveIngest (ESB3003) and DRM Gateway (ESB3009) are shipped with the StreamViewer package. The default configuration offers graphical, web-based views of system metrics, such as CPU, memory and disk load. It also monitors service-specific parameters such as HTTP status codes and channel status. This chapter describes the usage and configuration of monitoring features for both products.

6.1.1 Monitoring agent and server

Each node running any combination of ESB3002, ESB3003 and ESB3009 needs a monitoring agent. Typically all agents push data to one central monitoring server.

6.1.2 Activating the StreamViewer server

The server RPM `ew-streamviewer-server` is distributed with each product installer. It is recommended to install the server on a separate node. The server package correspondingly provides handy configurations for the VictoriaMetrics database and the Grafana dashboards.

To install and start streamviewer server:

```
$ dnf install ew-streamviewer-server-<version>.rpm
$ systemctl start ew-streamviewer-server
```

To uninstall the service:

```
$ systemctl stop ew-streamviewer-server
$ dnf autoremove ew-streamviewer-server
```

The default user/password for the web server is `admin/admin`. For user/password management use the web interface: "Administration" => "Users and access".

The default port for the web server is `:3000`. The network ports of the services can be changed in `/etc/containers ↪ /systemd/*.container`. Afterwards, the server configuration at `/etc/edgeware/ew-streamviewer-server ↪ /grafana/provisioning/datasources/victoriametrics.yaml` and agent configurations (section below) have to be updated before restarting service.

```
$ systemctl daemon-reload
$ systemctl restart ew-streamviewer-server
```

Note:

- For offline system, e.g. no public internet access, the dependency `podman` has to be installed manually before installing server RPM. It can be found from OS **full** installation medium.
- For running without containers, the services must be installed and setup manually with the provided configurations at `/etc/edgeware/ew-streamviewer-server` and `/var/lib/edgeware/ew-streamviewer-server`.

6.1.3 Activating the StreamViewer agent

The agent is installed with every ESB3002, ESB3003 and ESB3009. It contains Telegraf configurations and scripts to collect compatible data.

At SW Repackager, make sure to enable HTTP counters and performance log with debug level

```
$ confcli integration.counters.http.enable true
$ confcli services.repackaging.logging.performance.enable true
$ confcli services.repackaging.logging.performance.level debug
```

Edit `/etc/edgeware/ew-streamviewer-agent/outputs/outputs.influxdb.conf` and change the `urls` field to point to the StreamViewer server address.

Start service and enable it on boot:

```
$ systemctl start ew-streamviewer-agent.target
$ systemctl enable ew-streamviewer-agent.target
```

This also start some necessary services if they are not running, such as `monitord` of SW Live Ingest, or `ew-repackager-monitord` and `ew-repackager-health` of SW Repackager.

7 Appendix A: Used files and ports

This is an overview of resources used by esb3009. This information can be useful for security audits.

7.1 Files

7.1.1 Written at install/upgrade time

These files do not change at runtime

Files/directories	Purpose
`/etc/sysconfig/edgeware/drm-gw`	Configuration
`/etc/edgeware/drm-gw/config.json` (*)	Configuration
`/etc/edgeware/drm-gw/server.crt` (*)	Configuration
`/etc/edgeware/drm-gw/server.key` (*)	Configuration
`/etc/edgeware/ew-streamviewer-agent/*`	StreamViewer Configuration
`/usr/lib/python3.11/site-packages/edrm*`	Libraries
`/usr/lib64/python3.11/site-packages/ew*`	Libraries
`/usr/lib/systemd/system/ew-drm-gw.service`	Configuration
`/usr/bin/ew-drm-gw`	Executable
`/usr/libexec/ew-streamviewer-esb3009/*`	Executables for StreamViewer monitoring

The `/etc/sysconfig/edgeware/drm-gw` is the regular system configuration file, it points both to the DRM-GW specific configuration ("DRM-GW configuration") and the binary directory.

Additionally, the DRM configuration contains a multiple DRM processors' definitions that may specify a certificate files of their own (one per the processor). Refer to the Configuration|Processors documentation for more information.

```
(*) The DRM-GW configuration, the public certificate file and the private key
    ↪  file
can be redefined by the user.

Default values:
- DRM-GW configuration is set to `/etc/edgeware/drm-gw/config.json`
  and it is defined in the `/etc/sysconfig/edgeware/drm-gw` as `CONFIG_PATH`
     ↪ option
- The public certificate file is set to `/etc/edgeware/drm-gw/server.crt`
  and it is defined in the DRM-GW configuration as `server_certificate_file`
     ↪ option
- The private key file is set to `/etc/edgeware/drm-gw/server.key`
  and it is defined in the DRM-GW configuration as `server_key_file` option
```

7.1.2 Changed at runtime when you reconfigure

This log file changes at runtime.

Files/directories	Purpose
`/etc/logrotate.d/ew-drm-gw`	Log rotation configuration
`/var/log/edgeware/drm-gw/drm-gw-access.log` (*)	Access log
`/var/lib/edgeware/drm-gw/*`	Configuration

The DRM-GW access log file ("access log") contains an information about the HTTP requests to the DRM-GW and their status.

```
(*) The access log file location can be redefined by the user.

Default values:
- The access log path is set to `/var/log/edgeware/drm-gw/drm-gw-access.log`
  and it is defined in the DRM-GW configuration as `access_log_file` option
```

7.2 Ports

7.2.1 Server:

Port	Service
4443 (*)	ew-drm-gw

The DRM-GW binds and listens for the incoming connections on the HTTP/TCP port.

```
(*) The server port number can be redefined by the user.

Default value:
- Server port is set to 4443 and it is defined in the DRM-GW configuration
  as `server_port` option
```

For more information, please refer to the Configuration section.