ESB3002 Software Repackager User Guide

Document no	EDGS-176
Version	A16
Created on	2025-11-20

Contents

1 Confidentiality notice

2 About this document

This document covers an overview of the SW Repackager including chapters describing the installation, configuration, and running of the product. No prior knowledge about the system is assumed. Intended audiences include system administrators who install the system, operators who use it, and anyone who is interested in learning more about the product.

3 How to use this document

This document aims to be used as a foundation to build an understanding around the product. The document is to be used a guideline when working with the product. This document does not define any specific information on feature development in specific releases, please refer to “Release Notes”. Neither does this document define any applications relying on the specific product, please refer to “Application Notes”.

4 References

[1] EDGS-069 Convoy Management Software - User Guide

[2] EDGS-122 Convoy Management Software - CCMI Specification

[3] EDGS-138 VCP Origin User Guide

[4] Nginx documentation - http://nginx.org/en/docs/

[5] Timed Text Markup Language 1 (TTML1) - https://www.w3.org/TR/ttaf1-dfxp/

[6] DASH Industry Forum Profiles -http://dashif.org/identifiers/profiles/

[7] EDGS-156 TV content capture - CCMI CLI

[8] RFC6381 - The 'Codecs' and 'Profiles' Parameters for "Bucket" Media Types

[9] HLS Authoring Specification for Apple Devices - https://developer.apple.com/library/content/documentation/General/Reference/HLSAuthoringSpec/Requirements.html

[10] EDGS-107 - HTTP Configuration Interface

[11] EDGS-168 - DRM Gateway User Guide

[12] Codes for the Representation of Names of Languages - https://www.loc.gov/standards/iso639-2/php/code_list.php

[13] License generation in Xops and ScientaMedia for SW Licenses

[14] VoD Ingest user guide - https://docs.agilecontent.com/docs/acp/esb3021

[15] EDGS-216 - Software Origin Use Cases

4.1 Third party acknowledgement

The Software Repackager uses the Bento4 C++ library to read MP4 files in some workflows.

5 History

Version	Date	Changes
A1	2017-06-12	First version
A2	2017-01-15	Add CacheFS section

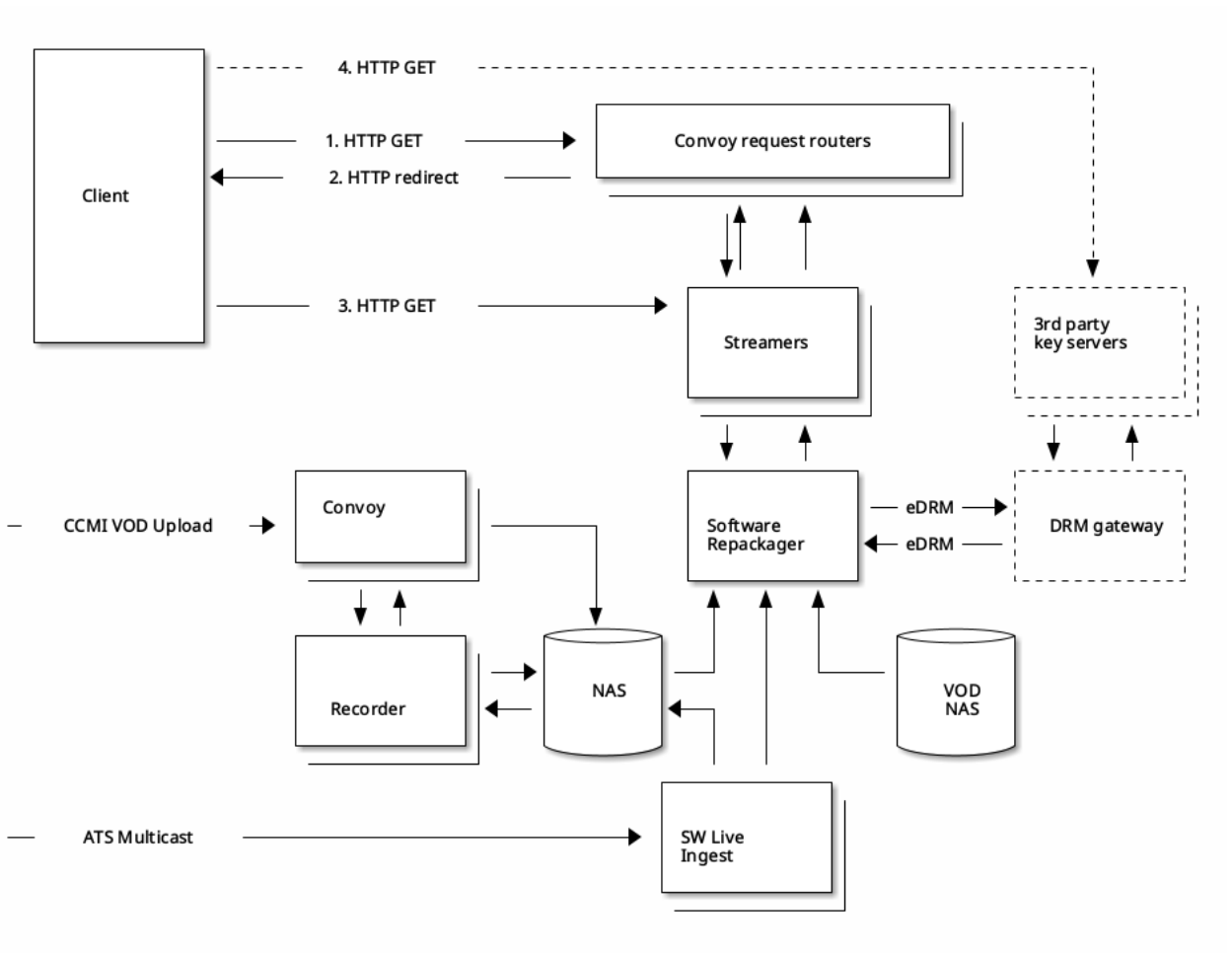
Version	Date	Changes
A3	2017-02-22	ESB3002-1.4.0 features
A5	2018-08-28	Update DASH configuration
	2018-12-04	Update DRM configuration
	2018-12-08	Add Alarms
A6	2019-06-12	Add DRM key rotation
	2019-06-13	Update HLS configuration
A7	2019-08-16	Add new HLS ad marker type
	2019-09-06	Add health and monitoring
	2019-11-28	Add S3 storage support
A8	2020-01-24	Add HLS fMP4 support
	2021-06-04	Add subtitle styling support
A9	2022-02-09	Add WebVTT support to HLS fMP4
	2022-02-22	Drop unmanaged VOD support
	2022-04-12	Update HLS configuration
	2022-05-11	Update storage configuration
A10	2022-07-18	Update HLS configuration
	2022-07-20	Update upgrade instructions
	2022-08-29	Drop Active Edge
A11	2022-11-28	Add segment cache
	2023-01-17	Add RHEL8 support
A12	2023-03-07	Update cache flushing
	2023-05-08	Add channel timeshift
A13	2023-06-22	Update cache configuration
	2023-09-26	Add DASH Timeline with `$Number$` support
	2023-09-15	Add DASH WebVTT subtitle support
	2023-10-09	Add HLS muxed TS support
	2023-11-23	Add colorspace filter
A14	2024-03-14	Update DASH configuration
	2024-04-19	Drop RHEL7 support
	2024-04-26	Update installation instructions
	2024-05-29	Update Used files and ports
	2024-08-14	Update segment cache
	2024-10-02	Add DASH thumbnails support
	2024-10-18	Add DASH Low Latency configuration
A15	2025-05-20	Add DASH and HLS thumbnail configuration
A16	2025-07-01	Update HLS fMP4 recommendations
	2025-07-02	Update new DRM configuration
	2025-07-31	Support RHEL 9
	2025-11-04	Remove old migration guide

6 SW Repackager overview

6.1 Description

The SW Repackager provides just-in-time packaging (JITP) of adaptive bitrate content into the commonly used formats DASH, HLS, and MSS. Among other features the SW Repackager supports encryption on egress, subtitling and multiple audio tracks.

The image below shows how the SW Repackager fits into AgileTV's ecosystem of products and services:



6.2 Content source format

For video-on-demand the repackager supports only content that has been ingested in the internal ESF format. All media files must be accessible in the file system on the SW Repackager, e.g. by using an NFS-mounted resource, alternatively as S3/Azure object storage.

The SW Repackager can also serve live content that has been generated by the SW Live Ingest.

For all types of content, there is a possibility to add a whitelist of allowed output profiles. A typical use case is that some live channels should be provided in the clear, while other should only be available encrypted with DRM. One should then make two channel groups and include the output profile for clear content in one, and an output profile with encryption in the whitelist for the other.

The configuration is done via *services.repackaging.content.X.name.outputProfiles* where X is the type of content, and name is the name of the content group. For example: The whitelist parameter for live channels in channel group "live" is *services.repackaging.content.channelGroups.live.outputProfiles*.

If the outputProfiles list is empty, all output profiles are allowed for that content group, while if the list have some entries, that content group is only accessible via the ones listed.

The default configuration for the outputProfiles whitelist is `["default"]` which means that the content is only accessible with the "default" output profile. By emptying the list, all output profiles will be accessible. For instance, to empty the output profile list for the channel group named live do:

```
confcli services.repackaging.content.channelGroups.live.outputProfiles -d
```

The matching against the whitelist is performed only on the output profile **name**, regardless of the account.

6.2.1 Live Content

In order to stream a live channel from SW Live Ingest, the SW Repackager needs to be configured with a source of type **channel** and provide a location which points to the SW Live Ingest host using HTTP over port 8090.

For example, `http://sw-liveingest:8090`. There is also a possibility to add a backup location pointing to secondary SW Live Ingest host. See Content Sources for more details how to configure a live channel source.

6.2.2 Video-on-demand Content

The common storage format that is used by the Software Repackaging System consists of MPEG CMAF (Common Media Application Format) files for audio, video, and text data, in combination with a number of additional metadata files. This complete set of files is referred to as a **managed asset** or an ESF (Edgeware Storage Format) asset. This format is what the WebTV Recorder produces when it makes a recording (content record) from a live channel or when it converts (content upload) an existing VOD asset.

In a workflow where output from the Recorder is repackaged and streamed, a source of the type **managed** should be added to the SW Repackager. See Content Sources for more details how to configure a managed source.

6.3 Encryption and Digital Rights Management (DRM)

The SW Repackager supports integration with different DRM solutions through the DRM Gateway [11] which provides a single integration point for third party DRM Systems. Once the SW Repackager has obtained the DRM information it can protect the content by encryption before streaming it to the client. The following modes of encryption is supported by the SW Repackager:

Streaming Format	Encryption Mode
HLS	AES-CBC (AES-128 and SAMPLE-AES)
MSS	AES-CTR (Full sample)
DASH	AES-CTR (Full sample)

DRM is configured by adding DRM-enabled output profiles. See the section Output profiles for further details.

DRM key rotation is supported for live content, with output formats DASH and HLS. For DASH, multiple periods will be used when a manifest spans across a key period boundary. Key rotation is set up in the DRM Gateway [11].

See [2], [7], and [11] for details on how to control the content DRM ID.

Note: SW Repackager from version 1.14.2 and onwards use newer version of eDRM API and needs DRM Gateway version 2.4.1 or higher.

6.4 Presentations

The set of variants that is presented to a client when accessing multi-bitrate content can be controlled by using presentation filters. Each filter is configured to accept variants based on media properties such as bitrate or the number of audio channels. Variants that do not match the filtering rules will be excluded from the content's manifest.

6.4.1 Accessing presentations

The SW Repackager uses the manifest name - or the name of the master playlist for HLS - to determine what presentation filters to use when rendering a response. More specifically, the SW Repackager will remove any extension from the requested manifest filename and use the remaining name to search among the defined presentations. The example below shows what manifest names to use to select a user-defined presentation called `mobile`:

Streaming Format	Manifest name	Selected presentation
HLS	mobile.m3u8	mobile
MSS	mobile.manifest	mobile
DASH	mobile.mpd	mobile

In order to support standard manifest names for HLS, MSS and DASH, the Software Repackager is shipped with the following presentations installed:

Streaming Format	Manifest name	Presentation name
HLS	index.m3u8	index
MSS	Manifest	Manifest
DASH	manifest.mpd	manifest

6.4.2 Filter chain logic

When the current presentation has been selected based on the requested manifest name, the SW Repackager filters the variants in the content through the filters in the presentation. The variants are grouped and filtered based on their media type, i.e. video variants are sent through the `videoFilters` list and so on.

Within each media type, the filters are applied in the order that they appear in the filter list. As the set of variants pass through the list of filters the subset of variants that matches the current filter is moved to the output and the rest of the variants are passed on to the next filter in the list.

The pre-installed `index` presentation for HLS illustrates how the filter chain can be used to promote variants in the resulting manifest. In their HLS Authoring Specification [9], Apple specifies that the 2000 kb/s video variant should be listed first in the master playlist, so the first video filter, `2Mbps_selection`, picks variants with a bitrate between 1800 kb/s and 2200 kb/s and adds to the output. The remaining video variants will be passed on to the next filter, `remaining_video`, that sorts the remaining variants before adding them to the output.

```
{
    "index": {
        "audioFilters": [
            {
                "codec": "*",
                "languages": [
                    "*"
                ],
                "maxBitrate": 4294967295,
                "minBitrate": 0,
                "name": "all_audio",
                "numChannels": 0,
                "order": "none"
            }
        ],
        "name": "index",
        "subtitleFilters": [
            {
                "languages": [
                    "*"
                ],
                "name": "all_subtitle"
            }
        ],
        "videoFilters": [
            {
                "codec": "*",
                "colorspaces": [
                    "*"
                ],
                "maxBitrate": 2200000,
                "minBitrate": 1800000,
                "name": "2Mbps_selection",
                "order": "descending"
            },
            {
                "codec": "*",
                "colorspaces": [
                    "*"
                ],
                "maxBitrate": 4294967295,
```

```
                "minBitrate": 0,
                "name": "remaining_video",
                "order": "descending"
            }
        ]
    }
}
```

6.4.3 Trick Play Support

SW Repackager can generate trick play tracks as defined in the HLS and MPEG-DASH specifications. These can be used to implement trick play modes, but possibly also scrub bar with thumbnails, in the client applications. Whether such extra variants or representations are included in the master playlist or MPD is controlled by parameters in the presentations.

6.4.3.1 For HLS

- I-frame Playlists are used as defined in the HLS specification.
- There is one I-frame Playlist per video track.
- Each I-frame segment is generated on the fly and contains an I-frame from the corresponding video segment.
- Works for both TS segments and fMP4 segments.
- The I-frame Playlist bitrate, as reported by the `BANDWIDTH` attribute, is calculated by multiplying the `BANDWIDTH` of the corresponding video track by `(number_of_i_frames_per_video_segment / (2 * video_segment_length_in_seconds))`.
- This feature is enabled via the `iframePlaylist` parameter in the presentation's HLS configuration.

6.4.3.2 For DASH

- Following the DASH-IF IOP specification, there is an extra adaptation set with EssentialProperty descriptor with URL `http://dashif.org/guidelines/trickmode` that provides trick mode representations corresponding to all the normal video representations
- Each trick play segment is generated on the fly and contains the first I-frame of the corresponding video segment
- Each trick mode segment carries one I-frame corresponding to the first I-frame in the normal segment
- This feature is enabled via the `trickMode` parameter in the presentation's DASH configuration.
- Following the DASH-IF IOP specification, Representations in two or more Adaptation Sets may provide the same content. In addition, the content may be time-aligned and may be offered such that seamless switching across Representations in different Adaptation Sets is possible.
- A content author may signal such seamless switching property across Adaptation Sets by providing a Supplemental Descriptor along with an Adaptation Set with `@schemeIdUri` set to `urn:mpeg:dash:adaptation-set-switching:2016` and the `@value` is a comma separated list of Adaptation Set IDs that may be seamlessly switched to from this Adaptation Set.
- Different players will have different configurations, some players don't support `adaptation-set-switching` and some players already support switching across Adaptation Sets without any further editing in the manifest.

Tip: Use `ew-repackager-tool` for quickly testing the presentation configuration, see ew-repackager-tool.

6.4.4 Thumbnail support

Many client applications do not support scrub bar with thumbnails using Trick Play tracks. One of the alternatives, "image-based thumbnail track", can be included to HLS master playlist or DASH MPD by SW Repackager if:

- The upstream supports thumbnail generation, either on-the-fly in ESB3003 or at ingested time in ESB3021.
- Corresponding parameter under `presentations` is enabled.

NOTE: It is recommended to enable `segmentCache` in SW Repackager and CBM Accelerator cache in SW Live Ingest to reduce performance impact of thumbnail generation.

6.4.4.1 For HLS

- Image media playlist specification is not developed by Apple, but 3rd-party.
- This feature is enabled via `thumbnail` parameter in the presentation's HLS configuration.

NOTE: It is recommended to check and test player compatibility before using this feature since no Apple player and only a few non-Apple players support this.

6.4.4.2 For DASH

- Follow DASH IOP specification, the adaptation set with EssentialProperty descriptor with URL `http://dashif.org/guidelines/thumbnail_tile` provides thumbnail representations.
- This feature is enabled via `thumbnail` parameter in the presentation's DASH configuration

6.5 Scripting

ESB3002 has scripting capabilities provided by Lua scripts https://www.lua.org/. This is made possible by using the Lua Nginx module https://github.com/openresty/lua-nginx-module.

Some use cases where scripting can be useful in repackager are:

- Manipulating repackager manifests/playlists
- Manipulating text based segments e.g. subtitles
- Add or change HTTP headers

Lua scripts can be pretty powerful and documentation on how to use it together with Nginx event model can be found on the module's Github page (see link above).

6.5.1 Scripting in Repackager

The Lua code can be written either in separate files or in nginx config blocks.

When using Lua scrips in repackager they should always be implemented in, or called from `/etc/ew-repackager/ew-repackager-custom.conf` in order to not be overwritten on an upgrade.

NOTE:

- According to doc and due to the fact that Repackager might issue multiple subrequests to many upstreams (i.e. drm-gw, live ingest), lua scripts can be executed more than once each client requests. Be careful on conditioning to process the right target. For illustration, the uri of client request is a substring of uri to drm-gw, only checking extension like below example isn't enough.
- Locations inside `ew-repackager-custom.conf` that shallows locations in `ew-repackager-server.conf` MUST including to send corresponding customer headers (if any), since only the first match is executed (see nginx doc).

6.5.1.1 Example

This is an example on how to change the language attributes in a DASH manifest from 'dan' to 'Dansk' using lua script implemented in nginx config blocks.

Content of `/etc/ew-repackager/ew-repackager-custom.conf`:

```
# When changing the length of the response body,
# clear content-length to enforce streaming output
header_filter_by_lua_block {
    if ngx.status == 200 then
        -- Only act on 200 responses
        local uri = ngx.var.uri
        if uri:sub(-4) == ".mpd" then
            -- It is a DASH manifest request -
            -- clear content length
            ngx.header.content_length = nil
        end
    end
}

body_filter_by_lua_block {
```

```
    if ngx.status == 200 then
        -- Only act on 200 responses
        local uri = ngx.var.uri
        if uri:sub(-4) == ".mpd" then
            ngx.log(ngx.DEBUG, "edgeware - DASH Manifest request: "..uri)
            -- It is a DASH manifest request - read the response body
            local chunk = ngx.arg[1]
            -- Substitute language code and update response body
            ngx.arg[1] = string.gsub(chunk, "lang=\"dan", "lang=\"Dansk")
        end
    end
}
```

Since the new text is longer the resulting response body will be larger and hence the HTTP header `Content-Length` will be incorrect. This is fixed by clearing the Content-Length header which will force use of chunked transfer encoding (this is being done in `header_filter_by_lua_block` in the example). If you know how much content length will change you can just update it with, in this case 2 bytes

```
header_filter_by_lua_block {
    ngx.header.content_length = ngx.header.content_length + 2
}
```

The example also uses Lua comments `--` and is logging to the repackager error log (on debug log level).

7 Output Formats

7.1 Description

The SW Repackager outputs HTTP streaming media in three formats: MPEG-DASH (DASH), Apple HTTP Live Streaming (HLS), and Microsoft Smooth Streaming (MSS) from a common internal format based on CMAF. The output consists of manifests (or playlists) and media segments for video, audio and subtitles. Video and audio can be encrypted.

The output formats are configured in two places. Output profiles define essential properties that influence the media segments and manifests such as encryption, while presentations only influence the manifests, but not the segments.

The output is also depending on the ingest process. In particular, the video sample entry format (e.g. avc1 or avc3 for H.264) is transparently used. When inband parameter sets are used, like in avc3, the stream properties may change over time as the original source parameter sets changes. A typical change would be picture aspect ratio. For audio, properties that may change are the number of channels or language. This type of changes may lead to minor discrepancies between the description of the media in manifests or playlists compared to what is currently streamed.

7.1.1 Language codes

For source material using three-letter ISO 639-2 language codes, being either live channels or ingested VOD content, two-letter ISO 639-1 codes are used in the output whenever there is a direct translation to this format. ISO 639-2/B and ISO 639-2/T codes having ISO 639-1 counterparts are supported. See e.g. [12] for a listing of these codes.

7.1.2 HLS Output

The HLS output format is MPEG-2 TS segment for audio and video and WebVTT segments for subtitles. Individual files are used for all segments and these are addressed as numbers.

Video and audio are always delivered as separate streams (not multiplexed). The HLS version is automatically chosen depending on what features are used. The minimal version is 4, but using Sample-AES means that the version is at least 5.

7.1.2.1 Subtitle Styling and Layout

For VOD, ingested from HLS/WebVTT, the following elements are output as ingested: inline styling tags, the file header (with e.g. stylesheets) and layout control (from WEBVTT settings). For a precise description of exactly what is ingested please refer to the ESB-3005 user guide.

7.1.3 DASH Output

DASH can be served either using the live profile, `urn:mpeg:dash:profile:isoff-live:2011`, or the on-demand profile, `urn:mpeg:dash:profile:isoff-on-demand:2011`.

For live and catchup content, regular segmentation is used throughout the system. Segment URLs are described in the manifest using `SegmentTemplate` with either `$Number$` or `SegmentTimeline` depending on output profile settings. Please see Output profiles for details.

For VoD, the `SegmentTemplate` is used together with `SegmentTimeline` to provide a more flexible solution that can handle irregular segmentation.

The on-demand profile requires that the client uses HTTP byte-range requests to download parts of the media content. This profile can be used for both streaming and for dowloading assets for offline playback.

7.1.3.1 DASH thumbnails

For `VOD` and `Recordings`, thumbsnails tracks can be served along with DASH manifest if there are tracks with media type `thumbnails` in `content_info.json`. This can be generated by enable thumbnails option when recording or uploading by `ew-vodingest` (ESB3021). For assets lacking thumbnails tracks, `ew-generate-thumbnails` can be used to generate that kind of track. See [14].

Image-based thumbnails are not part of the DASH standard, but specified by DASH-IF. According to the DASH-IF IOP specification, imaged-based thumbnails should be provided as separate AdaptationSets. An additional constraint is that they must use SegmentTemplate with `$Number$` independent of the SegmentTemplate of other AdaptationSets. Below is an example:

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" profiles="urn:mpeg:dash:profile:
   ↪ isoff-on-demand:2011" type="static" mediaPresentationDuration="
   ↪ PT9H57M21S" minBufferTime="PT3S">
  <Period>
    <AdaptationSet contentType="video" par="16:9" segmentAlignment="true"
       ↪ mimeType="video/mp4" startWithSAP="1" scanType="progressive">
      <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"></Role>
      <Representation id="video_h264_150kbps" bandwidth="150000" width="640"
         ↪ height="360" sar="1:1" frameRate="50/1" codecs="avc1.64001F">
        <BaseURL>video_h264_150kbps.cmfv</BaseURL>
        <SegmentBase indexRange="715-75422">
          <Initialization range="0-714"></Initialization>
        </SegmentBase>
      </Representation>
    </AdaptationSet>
    ...
    <AdaptationSet contentType="image" mimeType="image/jpeg">
      <SegmentTemplate media="$RepresentationID$/$Number$.jpeg" duration
         ↪ ="35841" startNumber="1"></SegmentTemplate>
      <Representation id="thumbnails-240x135" bandwidth="1000" width="720"
         ↪ height="22545" codecs="jpeg">
        <EssentialProperty schemeIdUri="http://dashif.org/thumbnail_tile"
           ↪ value="3x167"></EssentialProperty>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

It depends on player to support mixed SegmentTemplates and tiling of thumnails.

NOTE: A limitation of the $Number$-based SegmentTemplate is that the time of the thumbnails cannot be fine tuned. The thumbnail process in VoD Ingest generates thumbnails from the first image of every segment.

7.1.3.2 Clock Synchronization

For live DASH with `$Number$` template, it is very important that the server and client clocks are synchronized. To guide the client, DASH has standardized a number of different `UTCTiming` methods and they can be listed in decreasing priority order in the MPD.

The default output from the SW Repackager is a single entry:

```
<UTCTiming
    "schemeIdUri" : "urn:mpeg:dash:utc:http-iso:2014"
    "value": "https://time.akamai.com/?iso"
/>
```

but a list of UTCTiming values can be specified as `dash` parameters in a presentation.

7.1.3.3 Subtitle Styling and Layout

The subtitle style and layout is controlled from the subtitle styling template. This style is used for **all** DASH subtitle output.

For DASH, the output TTML should follow the IMSC-1 specification and the template path is: `/etc/ew-repackager/ttml-styling-DASH-custom.template`. If the custom template is missing then the default template is used: `/etc/ew-repackager/ttml-styling-DASH-default.template`. The default template may be modified when upgrading to new versions. Upgrading will not affect the custom template.

The styling and layout sections of the DASH template can look like this:

```
    <styling>
      <style xml:id="s0"  tts:fontFamily="sansSerif"
                          tts:fontSize="100%"
                          tts:lineHeight="normal"
                          tts:color="#FFFFFF"
                          tts:wrapOption="noWrap"
                          tts:textAlign="center"/>
      <style xml:id="s1"  tts:color="#ffffff"
                          tts:backgroundColor="#000000"
                          ebutts:linePadding="0.5c"/>
    </styling>
    <layout>
      <region xml:id="r0" tts:origin="15% 80%"
                          tts:extent="70% 20%"
                          tts:overflow="visible"
                          tts:displayAlign="after"/>
    </layout>
```

E.g: to get a black background with 75% transparency change `tts:backgroundColor="#000000"` [opaque] to `tts:backgroundColor="#00000040"` For changes in the styling template to take effect, the repackager service must be restarted. The template variables such as `{{ lang }}` and `{% block cues %}`, must not be touched. Edit the template with great care, no verification is made of its correctness.

7.1.3.4 Ingested Styling Tags

For VOD, ingested from HLS/WEBVTT, inline styling is translated as follows: Style tags <b>, <i> and <u> are translated into the corresponding TTML span attributes. Color tags (e.g: <c.yellow>) are translated only for the foreground colors, and only for the standard colors. Background colors are ignored. (The standard colors are: "white", "lime", "cyan", "red", "yellow", "magenta", "blue", "black", "green") Ingested layout control (from WEBVTT settings) and stylesheet styling is currently not translated for DASH and MSS.

7.1.4 MSS Output

The MSS follows the Microsoft Smooth Streaming format 2.0 by default.

According to the MSS specifications, the manifest URLs must have the format `..../something.ism(l)/Manifest`. The repackager will accept such URL but ignores the `something.ism(l)/`.

The timescale used in MSS output is 10MHz. This can lead to very large timestamps for live content, since by default all timestamps are in UNIX Epoch time. Javascript cannot handle integers needing more than 53 bits, and therefore some JavaScript players may have issues with the timestamps. To make the timestamps smaller, the `referenceTimeVsEpochS` setting in ESB3003 can be changed. Note that this will change the time reference for DASH as well.

In order to use the `r=` tag feature, to make manifests shorter when having consecutive segments with identical duration, change MSS version to 2.2 using the presentation configuration parameter 'mss.version'.

Example below:

```
services.repackaging.streaming.accounts.default.presentations.Manifest.
    ↪ parameters.mss.version 2.2
```

7.1.4.1 Subtitle Styling and Layout

Templatized subtitle styling and layout is described under **DASH Output -> Subtitle Styling** above.

For MSS the template path is: `/etc/ew-repackager/ttml-styling-MSS-custom.template`. If the custom template is missing then the default template is used: `/etc/ew-repackager/ttml-styling-MSS-default.template`. The default template may by modified when upgrading to new versions. Upgrading will not affect the custom template.

7.1.4.2 Ingested Styling Tags

Ingested styling is handled the same way for MSS and DASH. Please see **DASH Output -> Ingested Styling Tags**

7.2 Live and Catchup Manifests

Live and catchup have many similarities. Live manifests provide a sliding window of fixed duration. Its duration is given by the configuration parameter `liveWindow`, which is set in the corresponding presentations. A catchup request is a request for a live channel with extra time-marker parameters in the query string.

7.2.1 Time markers to extract catch-up content

SW Repackager accepts two query parameters, **startTime** and **stopTime** in the request URL. These can be used to specify an open-ended (startTime only) or double-ended (startTime and stopTime) interval to limit the content. The resulting manifest will either be of live or VOD type, depending on whether stopTime is in the past, or in the future. The details are described below, and for DASH, there is a table describing how all the relevant Manifest parameters change.

The times can be specified as IETF timestamps in UTC time such as `20170101T140000Z` or as epoch time (seconds since 1970-01-01T00:00:00Z) such as `1483279200`.

7.2.1.1 Open-ended interval

This is the case where only startTime is specified. The result is a live manifest that will start at startTime and grow until it reaches `maximumManifestDuration` at which time it becomes a sliding window. For DASH, which does not have an explicit list of segments, this behavior is achieved by setting `availabilityStartTime = startTime` and `timeShiftBufferDepth = maximumManifestDuration`.

Note that this manifest is always a live manifest. The typical player behavior is therefore to start playing from the live-point and not from the start of the interval.

The manifest is limited in the following way:

7.2.1.1.1 startTime > now

In this case, there is no valid content, so the HTTP response is "HTTP 412 Precondition Failed".

7.2.1.1.2 startTime < now and (now - startTime) < maximumManifestDuration

For HLS and MSS, a live manifest starting at startTime and ending at now is returned.

7.2.1.1.3 startTime < now and (now - startTime) > maximumManifestDuration

In this case, the manifest is limited by the `maximumManifestDuration` and transitions into becoming a sliding window covering the the interval `[now - maximumManifestDuration, now]`. For DASH, this is achieved automatically without updating the manifest, since the timeShiftBufferDepth value governs what is available.

7.2.1.2 Double-ended interval

In this case, both startTime and stopTime are specified.

- If stopTime is in the past, the manifest returned is a VoD manifest.
- If stopTime is in the future, a live manifest with a fixed first segment is returned. This corresponds to #EXT-X-PLAYLIST-TYPE:EVENT in HLS, and this is also signaled. For DASH, and the manifest is of type `dynamic` before stopTime, and of type `static` after.

7.2.1.2.1 startTime > stopTime

`startTime` must always be before `stopTime`, so in this case, "HTTP 400 Bad Request" is returned.

7.2.1.2.2 startTime > now

Content not available, and the HTTP response is "HTTP 412 Precondition Failed".

7.2.1.2.3 startTime < now < stopTime

Content is available as a growing live interval. The HLS media playlists signal `#EXT-X-PLAYLIST-TYPE:EVENT`. DASH manifest is of type `dynamic` with `AST= startTime`, `timeShiftBufferDepth = stopTime - startTime`. Manifest is limited by `maximumManifestDuration` just like open-ended case.

7.2.1.2.4 stopTime < now and startTime > (now - circularBuffer)

Content is available as a fixed VoD asset limited by `maximumManifestDuration`, e.g, covers interval `[max(` ↪ `startTime, stopTime - maximumManifestDuration), stopTime]`. The HLS media playlists signal #EXT-X-PLAYLIST-TYPE:VOD. For DASH, the manifest type is now `static`.

7.2.1.2.5 startTime < (now - circularBuffer)

`startTime` is outside the circularBuffer interval, so the content range is not available, "HTTP 410 Gone" is returned.

7.2.1.3 DASH manifest details

DASH Manifests have a number of parameters that will have different values in the different live and catchup cases. In the table below, these values are listed:

Parameter	Live	startTime only	stopTime in future	stopTime in past
MPD@type	dynamic	dynamic	dynamic	static
MPD@ availabilityStartTime	reference time	startTime	startTime	NA
SegmentTemplate@ startNumber	Number at (now - liveWindow)	Number at startTime	Number at startTime	Number at startTime
MPD@ minimumUpdatePeriod	DASH MUP	DASH MUP	DASH MUP	NA
MPD@ timeShiftBufferDepth	liveWindow	Max manifest duration	stopTime - startTime	NA
MPD@ mediaPresentationDuration	NA	NA	NA	stopTime - startTime

Parameter	Live	startTime only	stopTime in future	stopTime in past
AdaptationSet@ presentationTimeOffset	now - liveWindow	startTime	startTime	startTime
MPD@publishTime (Number Template)	AST	startTime	startTime	Last segment end time
MPD@publishTime (Segment Timeline and Segment Timeline with $Number$)	Latest end time	Latest end time	Latest end time	Last segment end time
Event@presentationTime	0	variable	variable	variable
Event@duration Latency@target	original value >= `targetLatency`	variable >= `targetLatency`	variable >= `targetLatency`	variable NA

Notes:

- The `availabilityStartTime` for live is configured with the `referenceTimeVsEpochS` parameter.
- The `minimumUpdatePeriod`, DASH MUP, is configured as as a dash parameter in a presentation
- The `liveWindow` and `maximumManifestDuration` are general parameters in a presentation
- The `mediaPresentationDuration` and `timeShiftBufferDepth` values will not be exactly stopTime-startTime, but be clamped to the closest segment start and end times.
- An Event's `presentationTime (EPT)` and `duration (ED)` parameters are adjusted based on the first segment's presentation time (`SPT`), which is also the Period start time.

  If the `EPT` is before `SPT`, then:

  - the `EPT` value is set to the relative beginning of the `SPT`, which is 0,
  - the `ED` value is shortened by the difference between `SPT` and `EPT`

  If the `EPT` is after `SPT`, then:

  - the `EPT` value is set to the relative beginning of the `SPT`, which is the difference between `EPT` and `SPT`
  - the `ED` value remains unchanged
- The Latency's `target` won't exceed the configured `targetLatency` by an amount of average segment duration.

8 Installation

8.1 Prerequisites

The SW Repackager requires RHEL 8.10, or at least 9.3 as operating system. Other RedHat binary compatible distributions are unofficially supported.

8.1.1 Hardware

Contact AgileTV for information regarding the recommended hardware setup.

8.2 Preparation

8.2.1 Firewall

The following ports must be opened in the firewall:

Port	Description
:80	Default HTTP streaming port
:5000	Optional. Control port for the configuration service. Needed if Convoy or ESB3011 GUI is used for propagating the configuration.
:12345	Optional. Control port for the monitor service. Needed if ESB3011 GUI is used for monitoring the SW Repackager node.
:4999	Optional. Port to get error log and version of SW Repackager. Needed if ESB3011 GUI is used for monitoring the SW Repackager node.
:2000	Optional. Port for health check service. Needed if ESB3011 GUI is used for monitoring the SW Repackager node.

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

8.2.2 SELinux

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

8.2.3 System limit

To avoid service runs out of file descriptors when in high load, the system-wide and process limit of opened files must be at least 1048576.

First, check the current system value

```
# sysctl fs.nr_open
# sysctl fs.file-max
```

If any of them is smaller than 1048576, then add corresponding entry to `/etc/sysctl.conf`, `nr_open` must not exceeds `file-max`. For example:

```
fs.nr_open = 1048576
fs.file-max = 6537812
```

To apply the changes without rebooting:

```
# sysctl -p
```

You can verify the change by running `sysctl` without `-p` again.

8.3 Installation

For installing the SW Repackager

```
[root@esb3002 ~]# chmod +x install-esb3002-X.Y.Z
[root@esb3002 ~]# ./install-esb3002-X.Y.Z
```

8.4 Starting, stopping and restarting

To perform any action on SW Repackager:

```
[root@esb3002 ~]# systemctl <action> ew-repackager
```

where <action> is one of below:

<action>	What it does
start	Start service
status	Verify service status (running, stopped, failure, etc.)
restart	Restart service and flush the cache
reload	Restart service without flushing the cache, no downtime
enable	Auto-start service on system boot

8.5 Removal

To uninstall the SW Repackager do as follows:

```
[root@esb3002 ~]# systemctl stop ew-repackager ew-repackager-confd confd
[root@esb3002 ~]# dnf autoremove esb3002
```

9 Upgrade

Follow these steps to upgrade the SW Repackager to a newer version.

Note: Always review the release note for any particular instructions or compatibility issues in addition to the general procedure explained here in the user guide.

Run the installer with option -u:

```
[root@esb3002 ~]# chmod +x install-esb3002-X.Y.Z
[root@esb3002 ~]# ./install-esb3002-X.Y.Z -u
```

Then restart confd to load any configuration updates in the new version. Finally start the repackager service which in turn should start ew-repackager-confd service:

```
[root@esb3002 ~]# systemctl restart confd ew-repackager-confd
[root@esb3002 ~]# systemctl restart ew-repackager
```

9.1 Complete reinstallation

1. Remove the repackager according to above
2. Install the new Repackager according to above
3. Restore configuration If you have made manual changes to configuration files such files will be saved in textfiles named <filename>.rpmsave, e.g: nginx.conf.rpmsave for nginx.conf. Inspect the following directories for such saved files, and manually merge your changes from the save files:

- /etc/ew-repackager/*
- /etc/logrotate.d/*

1. Restart the service according to above

10 Configuration

10.1 Configuring using confd

The SW Repackager is configured through the confd service, that is included in the SW Repackager installer. The SW Repackager can also be configured via an HTTP REST API. For more information about this, see [10].

In this document, all configuration will be performed via the confd command line interface confcli.

To work with the configuration, a basic understanding of the JSON format is needed. Please see http://en.wikipedia.org/wiki/JSON for more information.

The configuration is modeled as a tree, with different paths for the various groups of configuration. The base path for all repackaging configuration is **services.repackaging**.

To list the complete configuration for the SW Repackager, enter the following on the command line on the SW Repackager node:

```
confcli services.repackaging
```

The command line switch `-h` can be used to make `confcli` display a short description of the current configuration element if such a description is available. Below, the help for the SW Repackagers logging settings is shown:

```
[root@esb3002 ~]# confcli services.repackaging.logging.repackaging. -h
repackaging:

Resource Commands:
  enable  Turn logging on or off (default: True)
  level   The repackaging log level (default: info)
  path    The path to the repackaging log (default: /var/log/edgeware/ew-
          repackager/repackaging.log)
```

10.2 Configuration elements

10.2.1 Cache

The cache configuration is located in:

- **services.repackaging.metadataCache**: store upstream server metadata objects, divided into 3 groups: **drm** for DRM objects, **esf** for managed content, and **channel** for live content. For performance reason, they are cached in each worker memory.
- **services.repackaging.segmentCache**: store jitp output segments, and HTTP upstream server objects. For same reason, but due to nature of objects (mostly big binary files), they are stored on RAM disk.

NOTE: Caches from this level and downstream must be purged after any changes in configuration that affects content generation, for example, change `dashContent` of output profile, in order to get new fresh content. However, that action will result in temporarily increased load on SW Origin as the caches are refilled. To clear all caches, run the following command:

```
sudo ew-repackager-tool clear-cache all
```

10.2.1.1 Segments cache

The generated files and input files from ingest servers, e.g. metadata, media segments, etc., are cached on `/dev/shm/cache-repack`, which basically a RAM disk.

enable - Toggle on/off the cache. The default value is `True`.

numObjects - The maximum number of files to be cached. The default value is 20000.

capacity - The capacity of cache. The default value is 10GB.

upstreamLive - Enable caching for HTTP Live channel upstream responses. The default value is `False`.

upstreamVod - Enable caching for HTTP VOD upstream responses. The default value is `False`.

The cache-missed requests are logged into `/var/log/edgeware/ew-repackager/access-forward.log`. The subrequests to ingest servers are logged into `/var/log/edgeware/ew-repackager/access-upstream.log`.

NOTE:

- The cache efficiency depends on the uniqueness of requests and upstream latency. Check the system setup and properties of incoming traffic before using. In some conditions, with `upstreamLive` and/or `upstreamVod` enabled, response time is not improved or even worse, such as:
  - The requests are mostly live toward one dominant pair of output profile and presentation.
  - SW Repackager and SW Live Ingest (or HTTP VOD service) are on same host or latency between them is better than RAM disk.

- The cache `capacity` must not exceed the size of `/dev/shm`. An excessive cache would cause serious system malfunction. For this reason, such values are not applied and the cache is turned off with a log warning to `/var/log/edgeware/ew-repackager-confd/ew-repackager-confd.log`. As a result, the cache is always disabled when RAM amount on system is 2GB or less.
- The ratio `capacity/numObjects` should be slightly below the average segment size. A too high ratio makes cleaning less efficient and potentially fills up cache partition, while a too low one drops the cache hit ratio.

10.2.1.2 Content cache

The metadata, such as the codec and the bitrate, of media assets that are repackaged are cached in memory. This cache is cleared upon `reload` or `restart` service.

See Content Sources for details about the different content types.

For `esf` group:

- **movieObjects** - The maximum number of movie objects that are cached for ESF content. The default value is 500. Set to 0 to disable caching.
- **trackObjects** - The maximum number of track objects that are cached for ESF content. The default value is 2000. Set to 0 to disable caching.

For `channel` group:

- **numObjects** - The maximum number of channel objects that are cached for live content. The default value is 500. Set to 0 to disable caching.

10.2.1.3 DRM cache

The cache size for DRM objects is set in **services.repackaging.metadataCache.drm**.

redisCache: when set to `True` (default) a shared cache in a Redis database is used which reduces the number of DRM requests towards DRM-GW. When set to `False` an internal per worker cache will be used.

numObjects: defines the maximum number of DRM objects to keep in the cache. Only effective when **redisCache** is False. The default value is 500. Set to 0 to disable per worker cache.

10.2.2 Repackaging log

Information about each repackaging request is logged to file. The repackaging log is controlled in **services.repackaging.logging.repackaging** in the confd configuration and has the following parameters:

enable - Turn repackaging log on or off. Valid values are `True` and `False`. The default value is `True`, meaning that repackaging log is on.

level - Sets the repackaging log level. Messages with a log level below the provided value will be suppressed. Valid values are **trace**, **debug**, **info**, **warning**, **error**, and **fatal**. The default value is **info**.

path - The path to the repackaging log in the file system. The default log location is **/var/log/edgeware/ew-repackager/repackaging.log**.

10.2.3 Performance log

The logging of information regarding the SW Repackager's performance can be configured in **services.repackaging.logging.performance**. The performance log has the following configuration parameters:

enable - Turn performance log on or off. Valid values are **True** and **False**. The default value is **False**, meaning that performance log is off.

level - Sets the performance log level. Messages with a log level below the provided value will be suppressed. Valid values are **trace**, **debug**, **info**, **warning**, **error**, and **fatal**. The default value is **info**.

path - The path to the performance log in the file system. The default log location is **/var/log/edgeware/ew-repackager/repackaging-performance.log**.

10.2.4 Request details log

This log is used to show detailed information for certain requests without having to use logging with high verbosity for all requests. The log is configured in **services.repackaging.logging.requestDetails**. The **path** parameter sets the location of the request details log in the file system, which defaults to `/var/log/edgeware/ew-repackager/request-details.log`.

10.2.4.1 Details for slow requests

To log information for requests that have a high internal processing time in the repackager, use the configuration in **requestDetails.slow**:

requestDetails.slow.enable - Turn the slow request details log on or off. Valid values are `True` and `False`. The default value is `False`, meaning that this log is turned off.

requestDetails.slow.processingThreshold - Requests having an internal processing time that exceeds this value will be logged in detail regardless of the repackaging log level. The unit of this parameter is milliseconds and the default value is 2000.

The processing time includes the time spent downloading the media data and the DRM information needed to complete the request. It also includes the repackaging of the content to the output format, including the encryption.

The processing time does not include the time spent sending the response to the client. Having slow responses, in e.g. the access log, without seeing any entries in the request details log means that the client, or the network to the client, might be the limiting factor.

10.2.5 Upstream TrackingID

When handling requests which trigger upstream live-ingest requests, a tracking ID can be included in the upstream requests in the form of a "trackingID" query parameter. The value of this parameter will be the same as the internal repackager request ID, as seen in e.g. **repackaging.log**. It's enabled via **services.repackaging.logging.sendUpstreamTrackingIDs** and defaults to `False`.

NOTE: It might break upstream caching.

10.2.6 Content Sources

The repackager can handle two kinds of content:

- Managed - Content that has been converted to the intermediate ESF format by the Recorder.
- Channel - Live channel content, including catch-up and start-over that still resides in the SW Live Ingest circular buffer. Channels are organized into *channel groups*.

In summary, the **content** section of the configuration is broken down into the above types of content. From there, references are made to the **locations** section which is found parallel to the **content** section in the configuration hierarchy. For an incoming request for a content a lookup will be performed to find the location or locations from which to fetch the content.

In a bit more detail, the **locations** section contains the subsections:

- **azureContainers** - A list with the configuration fields necessary to reach an Azure Blob storage container with a name for each entry.
- **fileMounts** - A list of paths along with a name for each path.
- **ingestServers** - A list of HTTP URLs to SW Live Ingest servers along with a name for each entry.
- **s3Buckets** - A list with the configuration fields necessary to reach an S3 object storage bucket together with a name for each location entry.
- **httpServers** - A list of HTTP servers providing managed content.

The **content** section contains the subsections:

- **channelGroups** - A list of named objects pointing to locations specified in the **ingestServers** subsection of **locations**.
- **managed** - A list of named objects, each pointing to locations specified either in **fileMounts**, **s3Buckets**, **azureContainers** or **httpServers**.

Where multiple locations are allowed in a content object, the list of locations will represent a failover chain and each location will be tried in the listed order.

The **name** field in a content object corresponds to the `__cl` part in the CCMI URL format. When accessing managed content, the name is prefixed with `s:` (for storage) in the resource URL. Live channels, catch-up and start-over are prefixed with `cg:` which is short for channel group.

10.2.6.1 Azure blob storage container locations

Example: Adding one `managed` source with an azure blob storage container using confcli's wizard mode. Note that for Azure blob storage containers it is recommended that `endpoint` is left blank, unless you need to use a custom azure endpoint.

```
[root@esb3002 ~]$ confcli services.repackaging.locations.azureContainers -w
Running wizard for resource 'Azure Containers'
<List of Azure Containers used for managed content>

Hint: Hitting return will set a value to its default.
Enter '?' to receive the help string

Azure Containers <List of Azure Containers used for managed content>: [
  azureContainer : {
    Name (default: ): vod-container
    Container (default: ): some-container
    Path (default: ):
    Azure endpoint (default: ):
    Azure account (default: ): azure-storage-account
    Authentication method (default: managedIdentity):
    Azure storage account configuration file (default: ):
    Connection Timeout (default: 300):
    Read Timeout (default: 2000):
    Send Timeout (default: 2000):
    Timeout read retries (default: 3):
  Add another 'azureContainer' element to array 'azureContainers'? [y/N]:
Generated config:
{
  "azureContainers": [
    {
      "name": "vod-container",
      "container": "some-container",
      "path": "",
      "endpoint": "",
      "account": "azure-storage-account",
      "authType": "managedIdentity",
      "authConfigLocation": "",
      "connectTimeoutMs": 300,
      "readTimeoutMs": 2000,
      "sendTimeoutMs": 2000,
      "timeoutRetries": 3
    }
  ]
}
Merge and apply the config? [y/n]: y
```

followed by:

```
[root@esb3002 ~]$ confcli services.repackaging.content.managed -w
Running wizard for resource 'Managed'
<Configuration of source locations for ingested VoD and recordings>

Hint: Hitting return will set a value to its default.
Enter '?' to receive the help string

Managed <Configuration of source locations for ingested VoD and recordings>:
    ↪ [
```

```
  managed : {
    Name (default: ): azure-vod
    Locations <List of locations>: [
      location (default: ): vod-container
      Add another 'location' element to array 'locations'? [y/N]: n
    Output profiles <List of allowed output profiles>: [
      outputProfile (default: ):
      Add another 'outputProfile' element to array 'outputProfiles'? [y/N]: n
  Add another 'managed' element to array 'managed'? [y/N]: n
Generated config:
{
  "managed": [
    {
      "name": "azure-vod",
      "locations": [
        "vod-container"
      ],
      "outputProfiles": [
        "default"
      ]
    }
  ]
}
Merge and apply the config? [y/n]: y
```

The delivery URL for a content with the above defined source would look something like:

```
    /__cl/s:azure-vod/__c/casablanca/__op/default/__f/index.m3u8
```

Please see Content source format for how to limit or expand the list of allowed output profiles per content.

10.2.6.1.1 Authentication

The repackager supports authentication to Azure blob storage containers by

- **managedIdentity** - This is the default method for authentication and requires no further configuration. However, this method works only for repackager running on an Azure VM. The repackager will fetch authorization tokens from Azure Instance Metadata Service. See Azure documentation for further information regarding Instance Metadata Service.
- **sharedKey** - The storage account key is read from a file. The authentication config file and its directory needs to be fully readable by the `edgeware` user; both read and execute permissions are needed for the directory and read permission is needed for the file. Example:

```
[root@esb3002 ~]$ confcli services.repackaging.locations.azureContainers.vod-
   ↪ container.authType sharedKey
[root@esb3002 ~]$ confcli services.repackaging.locations.azureContainers.vod-
   ↪ container.authConfigLocation /etc/ew-repackager/vod-container.conf
```

The azure account key must be specified in the authentication config file.

```
accountKey <storage account key>
```

Note that if the authentication config file is modified while the repackager is running, the repackager needs to be reloaded for the changes to be active. Example:

```
[root@esb3002 ~]$ systemctl reload ew-repackager.service
```

10.2.6.1.2 Latency considerations

In order to achieve low latency for Azure storage requests, you may need to tweak the connections parameters. This can be done using the `connectionTimeoutMs`, `readTimeoutMs` and `timeoutRetries` parameters. Example:

```
[root@esb3002 ~]$ confcli services.repackaging.locations.azureContainers.vod-
    ↪ container.connectTimeoutMs 300
[root@esb3002 ~]$ confcli services.repackaging.locations.azureContainers.vod-
    ↪ container.readTimeoutMs 1000
[root@esb3002 ~]$ confcli services.repackaging.locations.azureContainers.vod-
    ↪ container.timeoutRetries 3
```

If different timeout settings per retry attempt is desired, multiple content locations for the same azure storage container can be specified with a `timeoutRetries` count of 0. The locations will be tried in sequential order, where the timeout settings could be individually configured. Example:

```
"managed": [
  {
    "name": "vod",
    "locations": [
      "vod-container",
      "vod-container-retry1",
      "vod-container-retry2",
    ]
  }
]
```

10.2.6.2 File mount content locations

File mounts used for managed content, such as ingested VODs or recordings, are specified in `services.repackaging.locations.fileMounts`.

name - Each file mount has a `name` by which it can be referred to from the `content` section of the configuration.

path - The `path` of the file mount is a directory that is available in the file system on the repackager node, for instance an NFS mount point.

useAIO - For each file mount it can be specified whether or not asynchronous I/O (AIO) should be used. The default is to use AIO.

Note: The Repackager's nginx configuration in `/etc/ew-repackager/nginx.conf` specifies `aio on`. This is needed for asynchronous handling of HTTP upstreams such as live channels and DRM servers. `useAIO` overrides the `aio` setting specifically for file mounts, so `aio` in nginx.conf should not be turned off on order to use synchronous file mounts.

Example: Adding one `managed` source with a locally mounted file system using confcli's wizard mode.

```
[root@esb3002 ~]# confcli services.repackaging.locations.fileMounts -w
Running wizard for resource 'File Mounts'
<File mount points in the local file system that can be used to fetch managed
    ↪  content>

Hint: Hitting return will set a value to its default.
Enter '?' to receive the help string

File Mounts <File mount points in the local file system that can be used to
    ↪ fetch managed content>: [
  fileMount : {
    Name (default: ): vod-nas
    Path (default: ): /mnt/media/vod
    Asynchronous (default: True): true
  Add another 'fileMount' element to array 'fileMounts'? [y/N]: n
Generated config:
{
```

```
  "fileMounts": [
    {
      "name": "vod-nas",
      "path": "/mnt/media/vod",
      "useAIO": true
    }
  ]
}
Merge and apply the config? [y/n]: y
```

followed by:

```
[root@esb3002 ~]# confcli services.repackaging.content.managed -w
Running wizard for resource 'Managed'
<Configuration of source locations for ingested VoD and recordings>

Hint: Hitting return will set a value to its default.
Enter '?' to receive the help string

Managed <Configuration of source locations for ingested VoD and recordings>:
   ↪ [
  managed : {
    Name (default: ): vod
    Locations <List of locations>: [
      location (default: ): vod-nas
      Add another 'location' element to array 'locations'? [y/N]: n
    Output profiles <List of allowed output profiles>: [
      outputProfile (default: ): hls-drm
      Add another 'outputProfile' element to array 'outputProfiles'? [y/N]: n
  Add another 'managed' element to array 'managed'? [y/N]: n
Generated config:
{
  "managed": [
    {
      "name": "vod",
      "locations": [
        "vod-nas"
      ],
      "outputProfiles": [
        "hls-drm"
      ]
    }
  ]
}
Merge and apply the config? [y/n]: y
```

The delivery URL for a content with the above defined source would look something like:

```
    /__cl/s:vod/__c/casablanca/__op/hls-drm/__f/index.m3u8
```

Please see Content source format for how to limit or expand the list of allowed output profiles per content.

10.2.6.3 Ingest server content locations

Example: Adding one `channel` source for live content with a primary and secondary location using confcli's wizard mode.

```
[root@esb3002 ~]# confcli services.repackaging.locations.ingestServers -w
Running wizard for resource 'Ingest Servers'
<Live Ingest servers from which to serve live, catchup and start-over content
   ↪ >
```

```
Hint: Hitting return will set a value to its default.
Enter '?' to receive the help string

Ingest Servers <Live Ingest servers from which to serve live, catchup and
    ↪ start-over content>: [
  ingestServer : {
    Name (default: ): live-primary
    URL (default: ): http://sw-liveingest-1:8090
    Connection Timeout (default: 500): 400
    Read Timeout (default: 1000): 800
    Send Timeout (default: 2000): 1000
  Add another 'ingestServer' element to array 'ingestServers'? [y/N]: y
  ingestServer : {
    Name (default: ): live-secondary
    URL (default: ): http://sw-liveingest-2:8090
    Connection Timeout (default: 500): 400
    Read Timeout (default: 1000): 800
    Send Timeout (default: 2000): 1000
  Add another 'ingestServer' element to array 'ingestServers'? [y/N]: n
Generated config:
{
  "ingestServers": [
    {
      "name": "live-primary",
      "url": "http://sw-liveingest-1:8090",
      "connectTimeoutMs": 400,
      "readTimeoutMs": 800,
      "sendTimeoutMs": 1000
    },
    {
      "name": "live-secondary",
      "url": "http://sw-liveingest-2:8090",
      "connectTimeoutMs": 400,
      "readTimeoutMs": 800,
      "sendTimeoutMs": 1000
    }
  ]
}
Merge and apply the config? [y/n]: y
```

followed by:

```
[root@esb3002 ~]$ confcli services.repackaging.content.channelGroups -w
Running wizard for resource 'Channel Groups'
<Configuration of source locations for live, catchup and start-over served by
    ↪  circular buffer>

Hint: Hitting return will set a value to its default.
Enter '?' to receive the help string

Channel Groups <Configuration of source locations for live, catchup and start
    ↪ -over served by circular buffer>: [
  channelGroup : {
    Name (default: ): live
    Locations <List of Ingest servers>: [
      location (default: ): live-primary
      Add another 'location' element to array 'locations'? [y/N]: y
      location (default: ): live-secondary
      Add another 'location' element to array 'locations'? [y/N]: n
    Output profiles <List of allowed output profiles>: [
      outputProfile (default: ):
```

```
      Add another 'outputProfile' element to array 'outputProfiles'? [y/N]: n
  Add another 'channelGroup' element to array 'channelGroups'? [y/N]: n
Generated config:
{
  "channelGroups": [
    {
      "name": "live",
      "locations": [
        "live-primary",
        "live-secondary"
      ],
      "outputProfiles": [
        "default"
      ]
    }
  ]
}
Merge and apply the config? [y/n]: y
```

The delivery URI for the example above would look something like this:

```
    /__cl/cg:live/__c/channel1/__op/default/__f/index.m3u8
```

10.2.6.3.1 Ingest server timeout configuration

As seen in the example above, timeout values can be specified for Live Ingest locations when they are created. When communicating with the server, `connectTimeoutMs`, `readTimeoutMs`, and `sendTimeoutMs` determine the maximum time for establishing the connection, and the maximum time for each read and write operation towards the server. The unit for these parameters is milliseconds. If a timeout value is exceeded, the upstream request to the current ingest server is abandoned and the request fails over to any remaining ingest server in the channel group.

The timeout values for an ingest server named ingest01 can be set in the following way using confcli:

```
[root@esb3002 ~]$ confcli services.repackaging.locations.ingestServers.
   ↪ ingest01.connectTimeoutMs 300
[root@esb3002 ~]$ confcli services.repackaging.locations.ingestServers.
   ↪ ingest01.readTimeoutMs 1000
[root@esb3002 ~]$ confcli services.repackaging.locations.ingestServers.
   ↪ ingest01.sendTimeoutMs 1000
```

10.2.6.4 S3 bucket content locations

Example: Adding one `managed` source with an S3 bucket using confcli's wizard mode. Note that for AWS S3 buckets it is recommended that `endpoint` is left blank, unless you need to use a custom AWS endpoint.

```
[root@esb3002 ~]$ confcli services.repackaging.locations.s3Buckets -w
Running wizard for resource 'S3 Buckets'
<Definition of S3 Buckets that can be used for managed content>

Hint: Hitting return will set a value to its default.
Enter '?' to receive the help string

S3 Buckets <Definition of S3 Buckets that can be used for managed content>: [
  s3Bucket : {
    Name (default: ): vod-bucket
    Bucket (default: ): some-bucket
    Path (default: ):
    Region (default: ): eu-central-1
    S3 Endpoint (default: ):
```

```
    AWS credentials profile (default: ):
    AWS credentials location (default: http://169.254.169.254/latest/meta-
       ↪ data/iam/security-credentials/ew-repackager):
    Connection Timeout (default: 100):
    Read Timeout (default: 2000):
    Send Timeout (default: 2000):
    Timeout read retries (default: 3):
  Add another 's3Bucket' element to array 's3Buckets'? [y/N]: n
Generated config:
{
  "s3Buckets": [
    {
      "name": "vod-bucket",
      "bucket": "some-bucket",
      "path": "",
      "region": "eu-central-1",
      "endpoint": "",
      "credentialsProfile": "",
      "credentialsLocation": "http://169.254.169.254/latest/meta-data/iam/
         ↪ security-credentials/ew-repackager",
      "connectTimeoutMs": 100,
      "readTimeoutMs": 2000,
      "sendTimeoutMs": 2000,
      "timeoutRetries": 3
    }
  ]
}
Merge and apply the config? [y/n]: y
```

followed by:

```
[root@esb3002 ~]$ confcli services.repackaging.content.managed -w
Running wizard for resource 'Managed'
<Configuration of source locations for ingested VoD and recordings>

Hint: Hitting return will set a value to its default.
Enter '?' to receive the help string

Managed <Configuration of source locations for ingested VoD and recordings>:
   ↪ [
  managed : {
    Name (default: ): s3-vod
    Locations <List of locations>: [
      location (default: ): vod-bucket
      Add another 'location' element to array 'locations'? [y/N]: n
    Output profiles <List of allowed output profiles>: [
      outputProfile (default: ):
      Add another 'outputProfile' element to array 'outputProfiles'? [y/N]: n
  Add another 'managed' element to array 'managed'? [y/N]: n
Generated config:
{
  "managed": [
    {
      "name": "s3-vod",
      "locations": [
        "vod-bucket"
      ],
      "outputProfiles": [
        "default"
      ]
    }
  ]
```

```
}
Merge and apply the config? [y/n]: y
```

The delivery URL for a content with the above defined source would look something like:

```
    /__cl/s:s3-vod/__c/casablanca/__op/default/__f/index.m3u8
```

Please see Content source format for how to limit or expand the list of allowed output profiles per content.

10.2.6.4.1 Authentication

By default S3 bucket content sources are setup using an AWS IAM role `ew-repackager` for AWS authentication credentials. Retrieved from the instance IAM metadata, using the AWS URL: `http://169.254.169.254/latest/meta-data/iam/security-credentials/<role>`. See AWS documentation for further information regarding IAM roles.

Authentication credentials could also be specified in a file. The credentials file and its directory needs to be fully readable by the `edgeware` user; both read and execute permissions are needed for the directory and read permission is needed for the file. Example:

```
[root@esb3002 ~]# confcli services.repackaging.locations.s3Buckets.vod-bucket
   ↪ .credentialsLocation /etc/ew-repackager/aws-credentials
```

The file should contain the credentials for one or more profiles, where `default` will be used unless a specific `credentialProfile` is configured.

```
[<profile name>]
aws_access_key_id=<access key id>
aws_secret_access_key=<secret access key>
```

Note that if the credential file is modified while the repackager is running, the repackager needs to be reloaded for the changes to be active. Example:

```
[root@esb3002 ~]$ systemctl reload ew-repackager.service
```

10.2.6.4.2 Latency considerations

In order to achieve low latency for S3 requests, AWS recommends performing requests with short timeouts and several retries. That is, if a single S3 request is slow, it should be aborted quickly and retried. The successive retry for the same end point is likely to be served from a different location and provide a faster response. To achieve this in the SW repackager, set the `timeoutRetries` and timeout parameters for the bucket. Example:

```
[root@esb3002 ~]# confcli services.repackaging.locations.s3Buckets.vod-bucket
   ↪ .connectTimeoutMs 20
[root@esb3002 ~]# confcli services.repackaging.locations.s3Buckets.vod-bucket
   ↪ .readTimeoutMs 100
[root@esb3002 ~]# confcli services.repackaging.locations.s3Buckets.vod-bucket
   ↪ .timeoutRetries 3
```

If different timeout settings per retry attempt is desired, multiple content locations for the same S3 bucket can be specified with a `timeoutRetries` count of 0. The locations will be tried in sequential order, where the timeout settings could be individually configured. Example:

```
"managed": [
  {
    "name": "s3-vod",
    "locations": [
      "vod-bucket",
      "vod-bucket-retry1",
```

```
      "vod-bucket-retry2",
    ]
  }
]
```

10.2.6.5 HTTP server content locations

Managed VOD assets that are located on another HTTP server can be accessed through the repackager by specifying an HTTP server location in `httpServers`. In essence an HTTP location works like an S3 or Azure Blob storage container without any authorization.

Example: Adding an HTTP server upstream for managed content using confcli's wizard mode.

```
[root@esb3002 ~]$ confcli services.repackaging.locations.httpServers -w
Running wizard for resource 'HTTP servers'
<HTTP servers that can be used for managed content>

Hint: Hitting return will set a value to its default.
Enter '?' to receive the help string

HTTP servers <HTTP servers that can be used for managed content>: [
  httpServer : {
    Name (default: ): HttpVodServer
    URL (default: ): http://myserver/vods/
    Connection Timeout (default: 100):
    Read Timeout (default: 2000):
    Send Timeout (default: 2000):
    Timeout read retries (default: 3):
  Add another 'httpServer' element to array 'httpServers'? [y/N]: n
Generated config:
{
  "httpServers": [
    {
      "name": "HttpVodServer",
      "url": "http://myserver/vods/",
      "connectTimeoutMs": 100,
      "readTimeoutMs": 2000,
      "sendTimeoutMs": 2000,
      "timeoutRetries": 3
    }
  ]
}
Merge and apply the config? [y/n]: y
```

followed by:

```
[root@esb3002 ~]$ confcli services.repackaging.content.managed -w
Running wizard for resource 'Managed'
<Configuration of source locations for ingested VoD and recordings>

Hint: Hitting return will set a value to its default.
Enter '?' to receive the help string

Managed <Configuration of source locations for ingested VoD and recordings>:
    ↪ [
  managed : {
    Name (default: ): vods
    Locations <List of locations>: [
      location (default: ): HttpVodServer
      Add another 'location' element to array 'locations'? [y/N]: n
    Output profiles <List of allowed output profiles>: [
```

```
      outputProfile (default: default):
      Add another 'outputProfile' element to array 'outputProfiles'? [y/N]: n
  Add another 'managed' element to array 'managed'? [y/N]: n
Generated config:
{
  "managed": [
    {
      "name": "vods",
      "locations": [
        "HttpVodServer"
      ],
      "outputProfiles": [
        "default"
      ]
    }
  ]
}
Merge and apply the config? [y/n]: y
```

The delivery URL for a content with the above defined source would look something like:

```
    /__cl/s:vods/__c/casablanca/__op/default/__f/index.m3u8
```

Please see Content source format for how to limit or expand the list of allowed output profiles per content.

10.2.7 Streaming accounts

To facilitate creation of multi-tenant video networks, where parts of the configuration can be customized for each customer, the following properties are defined per account:

- Output profiles
- Presentations
- Custom headers
- Expire time

Each of these configuration elements are described in detail in later sections of this document.

The accounts are available in a list, located at `services.repackaging.streaming.accounts` in the configuration structure. For example, to see the custom response headers that are configured in the account named `CustomerX`, enter the following on the command line:

```
[root@esb3002 ~]# confcli services.repackaging.streaming.accounts.CustomerX.
    ↪ customHeaders
```

The Software Repackager is shipped with an account named `default`. This account contains the output profiles and presentations needed to start working with the product.

To stream content from the Software Repackager using a specific account, the name of the account is included in the CCMI URI using the token `__a` followed by the account name. The account token should be located after the content location, `__cl`, in the URI. For example:

```
http://host:port/__cl/cg:live/__a/CustomerX/__c/channel2/__op/...
```

If no account token is present in the content URI, the default account will be used when serving the request.

10.2.8 Custom headers

A list of custom HTTP headers, that will be included in all responses, can be defined in **services.repackaging.streaming.accounts..customHeaders**. Each object in the list has a **name** and a **value** that together will constitute one header field in the HTTP response.

Example: Configure a custom header field with the name `Access-Control-Allow-Origin` and the value * in the default account using confcli's wizard mode:

```
[root@esb3002 ~]# confcli services.repackaging.streaming.accounts.default.
   ↪ customHeaders. -w
Running wizard for resource 'customHeaders'

Hint: Hitting return will set a value to its default.
Enter '?' to receive the help string

customHeaders : [
  customHeader <Adds a custom HTTP header in the HTTP response.>: {
    Field name (default: ): Access-Control-Allow-Origin
    Field value (default: ): *
  Add another 'customHeader' element to array 'customHeaders'? [y/N]: N
Generated config:
{
  "customHeaders": [
    {
      "name": "Access-Control-Allow-Origin",
      "value": "*"
    }
  ]
}
Merge and apply the config? [y/n]: n
Exiting...
```

10.2.9 Expire time

10.2.9.1 Vod

The expiration time to include in VOD asset responses, i.e. not a channel source, is specified in **expireTime.defaultMaxAgeVod** located in **services.repackaging.streaming.accounts.<account>.defaultMaxAgeVod** is defined relative to the point in time when the client fetches the media asset.

The table below shows the suffixes that can be used when specifying the **defaultMaxAge**, the default value is `12h`, meaning twelve hours.

Unit	Suffix	Example
Millisecond	ms	1ms
Second	s	1s
Minute	m	1m
Hour	h	1h
Day	d	1d
Week	w	1w
Month	M	1M
Year	Y	1Y

Example: Set max-age in Cache-Control HTTP header to 10 days in the default account:

```
[root@esb3002 ~]# confcli services.repackaging.streaming.accounts.default.
   ↪ expireTime.defaultMaxAgeVod 10d
services.repackaging.streaming.accounts.default.expireTime.defaultMaxAgeVod =
   ↪  '10d'
```

10.2.9.2 Live channels

For live channel content the expiration time is dependent on the content type. Manifests have an expire time of half a segment duration. The expire time for a media segment is by default based on the time it falls outside the catchup buffer. In other words, the CDN will be told to cache the segment as long it exists inside the catchup buffer. If a media segment is requested using an output profile with a catchup limitation (set with **maxAllowedCatchupDuration**) the expire time will be based on that limit instead.

As a final step, it is also possible to further limit the expire time using the **expireTime.defaultMaxAgeLive** configuration parameter.

Example: A configuration with one output profile with a limited catchup duration of 1 hour and one output profile with a catchup duration of 7 days, also a maximum expiretime of 24 hours for any segment from a channel source.

```
[root@esb3002 ~]# confcli services.repackaging.streaming.accounts.default.
    ↪ outputProfile.short.maxAllowedCatchupDuration. 3600
services.repackaging.streaming.accounts.default.outputProfiles.short.
    ↪ maxAllowedCatchupDuration = 3600

[root@esb3002 ~]# confcli services.repackaging.streaming.accounts.default.
    ↪ outputProfile.long.maxAllowedCatchupDuration. 604800
services.repackaging.streaming.accounts.default.outputProfiles.long.
    ↪ maxAllowedCatchupDuration = 604800

[root@esb3002 ~]# confcli services.repackaging.streaming.accounts.default.
    ↪ expireTime.defaultMaxAgeLive. 24h
services.repackaging.streaming.accounts.default.expireTime.defaultMaxAgeLive
    ↪ = '24h'
```

Manifests on any of these profiles will have an expire time of half a segment.

Requesting the latest created live segment from the **`short`** profile will have an expiration time of around 3600 seconds. Note that requesting a segment that is 3500 seconds old will expire 100 seconds later.

Segments requested from the **`long`** profile will have an expire time of 24 hours. However, if the segment is older than 6 days, the expire time will be set to the time when the segment will be unavailable (shifted out of the catchup buffer).

10.2.10 Output profiles

Egress encryption is controlled by adding DRM-enabled output profiles. The SW Repackager communicates with the DRM Gateway to get access to the DRM information that is needed for the encryption. A secondary DRM Gateway can be configured for redundancy. Output profiles that have enabled encryption in SW repackager needs to be configured in DRM Gateway also. Please see [11] for instructions on how to set up the DRM Gateway.

Output profiles are set in **services.repackaging.streaming.accounts..outputProfiles**. Each **outputProfile** has the following parameters:

- **name** - The name of this output profile. The value corresponds to the `__op` (output profile) part of the CCMI URL format.
- **drm.enable** - Enable encryption. The default value is `false`.
- **drm.certPath** - Path to a trusted certificate file for communication with the DRM Gateway. The default value is `/etc/edgeware/drm-gw/server.crt`.
- **drm.encryptAudio** - Turn audio encryption on or off. The default value is `true`. N.B. if the DRM Gateway returns a "per variant" DRM configuration or in HLS presentation the `muxedTS` segment format is used, this parameter has no effect.
- **drm.disableIFrameTrackEncryption** - If `true`, disables encryption of the I-frame track (I-frame playlist for HLS, TrickMode AdaptationSet for DASH). Default `false`.
- **drm.keyServer** - The address to the DRM Gateway. See [11] for details.
- **drm.keyServerBackup** - The address to the secondary DRM Gateway.
- **drm.sharedSecret** - Shared secret used to authenticate messages toward DRM Gateway,
- **drm.validateCert** - Whether or not to validate the certificate in case of HTTPS. The default value is `true`.
- **drm.connectTimeoutMs** - Timeout in milliseconds for establishing connections to the DRM server.
- **drm.readTimeoutMs** - Timeout in milliseconds between successive read operations towards the DRM server.

- **drm.sendTimeoutMs** - Timeout in milliseconds between successive write operations towards the DRM server.
- **drm.psshInMedia** – Adds `pssh` box that contains the header to DASH media segments if key rotation is supported. The default value is `false`.
- **liveChannelDelay** - Delay in seconds for a live channel. Also called *channel time-shift*. **Note:** Should be an integer multiple of the segment length. A typical use case for delaying a channel is to create regional channel versions, in different time-zones.
- **expectedSegmentDurationMs** - The expected duration of a segment in milliseconds. It is recommended to be at least double the average segment length of ESF, else it has no effect. **Note:** Only applies to HLS and DASH output of VOD and recordings.
- **maxAllowedCatchupDuration** - Limit the available catchup duration to this value expressed in seconds.
- **dashContent** - Type of DASH repackaging. The available types are: `SegmentTimeline`, `SegmentNumber` ↪ , `SegmentTimelineNumber`, `OnDemand` and `None`. `SegmentTimeline` uses `$Time$` addressing while `SegmentTimelineNumber` uses `$Number$` addressing. The default value is `SegmentTimeline`. A number-based addressing is required in order to be compatible with both DASH and HLS at the same time. `SegmentTimeline` and `SegmentTimelineNumber` can be used for both Live and VOD. `OnDemand` is VOD exclusively while `SegmentNumber` is for Live only. `None` is used for HLS/MSS profiles. To have identical segments for both DASH and HLS format, in same output profile, the DASH packaging formats using `$Number$` addressing (`SegmentNumber` and `SegmentTimelineNumber`) and the `hls.segmentFormat` set to `fMP4` are required.
- **avcSampleDescriptionFormat** - Change the AVC version announced for H264 content in DASH manifests and CMAF/fMP4 init segments. The default is `keepInput` and this will leave the content untouched. Other options are `avc1` and `avc3`. These will overwrite the codec version in an H264 video adaptation set in the manifest, as well as in the STSD box in the video init segment.
- **hevcSampleDescriptionFormat** - Change the HEVC version announced for HEVC content in DASH manifests and CMAF/fMP4 init segments. The default is `keepInput` and this will leave the content untouched. Other options are `hev1` and `hvc1`. These will overwrite the codec version in an HEVC video adaptation set in the manifest, as well as in the STSD box in the video init segment.
- **avoidIdenticalWebvttCues** - Some HLS players have problems with cues spanning multiple segments since this leads to identical adjacent subtitle cues in the output. This setting breaks the sequence of identical cues by adding an invisible unicode space in every other WebVTT segment. Not all clients have full unicode support, so use this setting with care.

NOTE: Unencrypted output profile (including the `default`) can be used to steal content if their names and urls to encrypted content are both known. It is recommended to remove or regularly rename those profiles.

Example: Setup a Sample AES encryption output profile with a 24 hour catchup limitation in the default account using confcli's wizard mode:

```
[root@esb3002 ~]$ confcli -w services.repackaging.streaming.accounts.
   ↪ default.outputProfiles
Running wizard for resource 'Output profiles'
<List of output profiles>

Hint: Hitting return will set a value to its default.
Enter '?' to receive the help string

Output profiles <List of output profiles>: [
  Output profile <Configuration for an output profile>: {
    Name (default: ): hls-sample-aes
    DASH content type (default: SegmentTimeline): None
    DASH output AVC version (default: keepInput):
    Maximal allowed catchup duration (default: 8640000): 86400
    Delay for live channel (default: 0): 0
    Avoid identical adjacent WebVTT cues (default: False):
    DRM <Configuration for DRM>: {
      Enable audio encryption (default: True): True
      Enable (default: False): True
      Disable i-frame-track encryption (default: False): False
```

```
        Key Server host:port (default: ): edgeware-drm-gw:4443
        Secondary Key Server host:port (default: ): ew-drm-gw-2:4443
        Key Server shared secret (default: ): shared-secret
        Validate Certificate (default: True): True
        Certificate path (default: /etc/edgeware/drm-gw/server.crt):
        Connection Timeout (default: 2000): 1000
        Read Timeout (default: 4000): 2000
        Send Timeout (default: 4000): 2000
    Add another 'outputProfile' element to array 'outputProfiles'? [y/N]: n
  Generated config:
  {
    "outputProfiles": [
      {
        "name": "hls-sample-aes",
        "dashContent": "None",
        "avcSampleDescriptionFormat": "keepInput",
        "hevcSampleDescriptionFormat": "keepInput",
        "maxAllowedCatchupDuration": 86400,
        "avoidIdenticalWebvttCues": false,
        "drm": {
          "encryptAudio": true,
          "enable": true,
          "disableIFrameTrackEncryption": false,
          "keyServer": "edgeware-drm-gw:4443",
          "keyServerBackup": "ew-drm-gw-2:4443",
          "sharedSecret": "shared-secret",
          "validateCert": true,
          "certPath": "/etc/edgeware/drm-gw/server.crt",
          "connectTimeoutMs": 1000,
          "readTimeoutMs": 2000,
          "sendTimeoutMs": 2000
        }
      }
    ]
  }
  Merge and apply the config? [y/n]: y
```

A delivery URI for the example above would look something like this:

```
/__cl/s:vod/__c/drama/__op/hls-sample-aes/__f/index.m3u8
```

Please see Content source format for how to limit or expand the list of allowed output profiles per content.

10.2.10.1 Single node DRM Gateway configuration

When The DRM Gateway is configured with the default configuration on the same node as the SW Repackager, `edrm-server` hostname should be resolved as the loopback ip address. To achieve that, append the following line to `/etc/hosts` on the SW Repackager node:

```
127.0.0.1   edrm-server
```

10.2.11 Presentations

10.2.11.1 Filter parameters overview

Below follows a summary of the configuration parameters that are available for presentation filters. If a parameter is omitted in the definition of a filter, the default value for that parameter will be used.

Video filter parameter	Description
codec	The accepted codec pattern for this filter
maxBitrate	The maximum bitrate to pass through the filter
minBitrate	The minimum bitrate to pass through the filter
name	The name of this filter
order	Sort order based on variant bitrate

Audio filter parameter	Description
codec	The accepted codec pattern for this filter
maxBitrate	The maximum bitrate to pass through the filter
minBitrate	The minimum bitrate to pass through the filter
name	The name of this filter
order	Sort order based on variant bitrate
numChannels	The number of channels to accept in the filter
languages	A list of languages to accept in the filter

Subtitle filter parameter	Description
name	The name of this filter
languages	A list of languages to accept in the filter

10.2.11.2 Filter parameters detailed description

codec - The `codec` parameter controls what media formats a filter lets through. The format identifier is defined in [8] and looks something like `avc1.4d401e` or `mp4a.40.2`. The codec pattern can either be a fixed string containing the exact format identifier, or end with `*` to match multiple codecs. For instance, specifying `avc1*` as `codec` in a video filter will let all H264 video variants pass through the filter regardless of their profile.

languages - The `languages` parameter specifies the languages that a filter will pass through in the form of a list of ISO 639 codes. The order of the languages in the list will be used when generating the manifest. A final wildcard element * can be added to the list to allow all languages while keeping the desired language order. For instance, specifying `"languages": ["eng", "*"]` will pass through all languages while making sure English ends up first, whereas `"languages": ["eng"]` will pass through English as the only language.

When not explicitly sorted, the order of the languages in the input will be maintained in the output.

maxBitrate and minBitrate - The bitrate filter parameters accepts or rejects variants based on their bitrate. The bitrate is specified in bits per second. The value of maxBitrate should be greater than or equal to the value of minBitrate.

10.2.11.3 Manifest parameters

Manifest parameters applicable for all output formats are grouped under **parameters.general**. Parameters specific to just one output format are grouped under the appropriate group in **parameters**, e.g. **parameters.dash**.

parameters.general.liveWindow - The duration of the manifest's live window in seconds.

parameters.general.maximumManifestDuration - The maximum duration of the manifest in seconds.

parameters.dash.minimumUpdatePeriod - The description to the client on a minimum validity duration of a live DASH manifest. It is automatically calculate to be segmentation duration, but can be overriden by setting another value that greater than default `-1`. See DASH manifest details for how this parameter is used.

It is highly recommended that a non-default value is only used for tuning the behavior of streaming clients when playback issues have been discovered.

parameters.dash.suggestedPresentationDelay - The fixed delay offset from the presentation time in seconds. Sets the value for the `suggestedPresentationDelay` attribute in the DASH manifest. When the value is 0, the attribute is not written to the manifest.

parameters.dash.trickMode - The `trickMode` configuration parameter determines whether or not to include DASH trick mode AdaptationSets in the manifest. The default value is `true`.

parameters.dash.thumbnail - Toggle on/off whether or not to include DASH thumbnail AdaptationSet to the MPD. The default value is `false`. Only effective if upstream supports thumbnail generation.

parameters.dash.UTCTiming - List of **schemeIdUri** and **value** pairs that are set in a <`UTCTiming`/> element in the DASH manifest. These values are used to signal to clients how to synchronize time.

parameters.dash.separateRepresentation - Toggle on/off to whether or not to allow each Representation in a separated AdaptationSet. The default value is `false`.

parameters.dash.subtitleFormat - Dash subtitle format, either `wvtt` or `stpp`. The default value is `stpp`.

parameters.dash.singleSubtitleSegmentVod - Applies only to VOD assets. Serve the complete subtitle track as a single segment, like what always happens on DASH OnDemand. The default value is `false`.

parameters.dash.compact - Toggle on/off inserting indentation to output MPD. The indentation makes output MPD more human-friendly while, without it, MPD has significantly smaller payload. The default value is `false`.

parameters.dash.targetLatency - involved in specifying `Latency` in MPD (see DASH manifest details and [15]). Only effective if low latency is enabled in upstream. The default value is `3.5` seconds. It is also recommended to set **UTCTiming**, according to section 9.X.4.2, Part 4 of DASH IOP v5.

10.2.11.3.1 Ad marker parameters

Ad markers, carried as SCTE-35 messages in the input signal of the channel source, can be propagated to the manifests produced by the SW Repackager by configuring the following presentation parameters.

parameters.dash.adMarkers - In a DASH media presentation, ad markers are inserted either as multiple events in one single period or as separate periods for each event.

Set `adMarkers` to `SinglePeriod` to get one period with multiple events, or to `MultiPeriod` to get a separate period for each event.

If set to `Disabled`, no ad markers will be added to the DASH media presentation. This is the default behavior.

The value of `dash.adMarkers` in combination with the output profile's `dashContent` setting determines when a new Period will be created in a DASH manifest. New Periods can be created either at ad period boundaries or when there are gaps in the segment sequence. Gaps may be caused by interrupted network traffic, or by configuration changes, on the ingest side. The table below illustrates when new periods are created based on the configured DASH manifest type and ad marker setting:

	Single Period	Disabled	MultiPeriod
SegmentTimeline	Never	Never	At ad markers
SegmentNumber	Never	After gaps	At ads and gaps
SegmentTimelineNumber	Never	After gaps	At ads and gaps

In addition to this, new periods will also be created at DRM key boundaries if a key rotation enabled output profile is used. See Encryption and Digital Rights Management (DRM) for details.

parameters.hls.adMarkers - In HLS media playlists ad markers can be inserted as `EXT-X-SCTE35` tags, as specified in SCTE-35, as `EXT-OATCLS-SCTE35` tags, as specified in SCTE-35 Enhanced, or be embedded in `EXT-X-DATERANGE` tags, as suggested in Apple's HTTP Live Streaming specification. Note that `EXT-X-SCTE35` and `EXT-OATCLS-SCTE35` are not recognized by Apple parsers.

Set `adMarkers` to `SCTE35` to use `EXT-X-SCTE35` tags, to `OATCLS` to use `EXT-OATCLS-SCTE35`, or to `HLS` to use the `EXT-X-DATERANGE` format.

If set to `Disabled`, no ad markers will be added to the HLS playlists. This is the default behavior. It is also the only mode works with `muxedTS` segment format due to incompatiblity.

parameters.hls.adMarkerDaterangeID - Format of ID attribute in DATERANGE tag. This configuration is applied when `parameters.hls.adMarkers = HLS`. The options are `extended`, `short` and work as follows:

- `extended` - The ID attribute includes the event id and the timestamp (default option).

- `short` - The ID attribute does only include the event id.

In some cases, a recording contains a partial ad event, the ad marker timestamp is updated to fit inside the recording. Ad event's ID should not include timestamp. This configuration should set to `short`.

10.2.11.3.2 Additional HLS playlist parameters

parameters.hls.audioGroupingRelativeBandwidth - Audio variants are grouped by codec and bitrate. The `audioGroupingRelativeBandwidth` specifies how different bitrates are grouped together. The value 0 means that all audio variants in a group will have exactly the same bitrate, the value `10` means that the lowest bitrate will not differ by more than 10% from the highest bitrate in a group. The value 100 means that all bitrates are grouped together.

parameters.hls.singleWebVTTSegmentVod - Applies only to VOD assets. Serve the complete subtitle track as a single WebVTT segment, for each variant respectively, of all `segmentFormat`, except the `muxedTS` does not support subtitles. The default behaviour is to use subtitle segments with identical intervals as the video segments, and to split subtitles that cross segment boundaries. Some HLS clients, notably from Apple, have issues with such split cues. This option merges the cues, but also has the benefit of a single segment download instead of many. The default value is `false`.

parameters.hls.iframePlaylist - The `iframePlaylist` configuration parameter determines whether or not to include links to I-frame playlists in HLS master playlists. The default value is `true`, but it must be set to `false` when `segmentFormat` is `muxedTS` due to incompatibility.

parameters.hls.iframeTsMapTag - The `iframeTsMapTag` parameter decides if and how `EXT-X-MAP` tags are inserted in I-frame playlists when TS segments are used. The options are `Disabled`, `ByteRange` and `FullIFrame` and work as follows:

- `Disabled` - No map tags are added to the playlist
- `ByteRange` - Map tags with byte ranges inside full I-frame segments
- `FullIFrame` - Map tags point to full I-frame segments

The default value is `Disabled`.

parameters.hls.denserIframes - The `denserIframes` configuration parameter controls whether or not the generated I-frame playlists can contain more than 1 I-frame per "normal" video segment. If set to `true`, the I-frame playlists will contain 1 I-frame per GOP (instead of 1 I-frame per segment). The default value is `false`.

parameters.hls.thumbnail - Whether or not to include link to image media playlist in HLS master playlist. The default value is `false`. Only effective if upstream supports thumbnail generation. Must be set to `false` when `segmentFormat` is `muxedTS` due to software limitation.

parameters.hls.segmentFormat - The `segmentFormat` configuration parameter determines the file format used for producing HLS segments in this presentation. The parameter can be set to either `TS`, `fMP4`, `fMP4WebVTT` or `muxedTS` with `TS` being the default value.

In general the `TS` and `muxedTS` setting produces MPEG-2 Transport Stream segments, whereas the `fMP4` and `fMP4WebVTT` setting produces CMAF fMP4 segments, but there are some exceptions. Based on the value of `segmentFormat`, the output file format for each codec and media type is listed in the table below.

The `muxedTS` setting will produce a single stream of both audio and video segments.

Segment type	TS	fMP4	fMP4WebVTT	muxedTS
AAC / HE-AAC	MPEG-2 TS	CMAF fMP4	CMAF fMP4	MPEG-2 TS
AC-3 / EC-3	Packed Audio	CMAF fMP4	CMAF fMP4	N/A
h.264 segment	MPEG-2 TS	CMAF fMP4 (**)	CMAF fMP4 (**)	MPEG-2 TS
HEVC segment	MPEG-2 TS (*)	CMAF fMP4 (**)	CMAF fMP4 (**)	N/A
h.264 I-frame	MPEG-2 TS	CMAF fMP4 (**)	CMAF fMP4 (**)	N/A
HEVC I-frame	MPEG-2 TS (*)	CMAF fMP4 (**)	CMAF fMP4 (**)	N/A
Subtitles	WebVTT	CMAF fMP4	WebVTT	N/A

(*) Not supported by HLS authoring specification for Apple devices. (**) `avc1` and `hvc1` are recommended by HLS authoring specification for Apple devices. This can be configured in `avcSampleDescriptionFormat` and `hevcSampleDescriptionFormat` of Output Profiles or live ingest upstream's `segmentationTemplates`.

parameters.hls.sessionKey - The `sessionKey` configuration parameter determines whether or not to include an `EXT-X-SESSION-KEY` tag with DRM information in HLS master playlists. The default value is `false`.

parameters.hls.keyPerChunk - The `keyPerChunk` determines whether to include `EXT-X-KEY` with DRM information to each chunk in HLS media playlist, or only include after previous key expired.

parameters.hls.maxInitialGAPLength - The `maxInitialGAPLength` configuration parameter determines the maximum total duration of `EXT-X-GAP` segments to include at the beginning of the media playlist. A playlist may start with very many `EXT-X-GAP` segments if e.g. the live channel has recently started and does not have segments to fill the `liveWindow`. The default value is `0`.

parameters.hls.liveTargetDuration - A positive `liveTargetDuration` value overrides the HLS `EXT-X-TARGETDURATION` value for live and catchup HLS playlist. This can be used to trigger more frequent playlist requests in HLS players. If the default value `0` is used, no modification is done and the value is derived from the maximum segment duration reported upstream.

10.2.12 Integration

In addition, the **integration.convoy.events** node is also honored by the SW Repackager. This node denotes the Convoy instance to send statistics and alarms to. Note that only the first entry in **integration.convoy.events.servers** is used.

```
[root@esb3002 ~]# confcli integration.convoy.events.enable true
[root@esb3002 ~]# confcli integration.convoy.events.servers.
[root@esb3002 ~]$ confcli integration.convoy.events.servers. -w
Running wizard for resource 'Kafka Servers'
<List of kafka servers that events will be sent to>

Kafka Servers <List of kafka servers that events will be sent to>: [
  Kafka Server (default: ): 1.2.3.4
    Add another 'server' element to array 'servers'? [y/N]: N
Generated config:
{
  "servers": [
    "10.11.12.13"
  ]
}
Merge and apply the config? [y/n]: y
```

integration.counters are counters that report running metrics to an internal module which aggregates data for monitor GUI. **integration.counters.http** reports HTTP response status codes.

10.2.13 Advanced

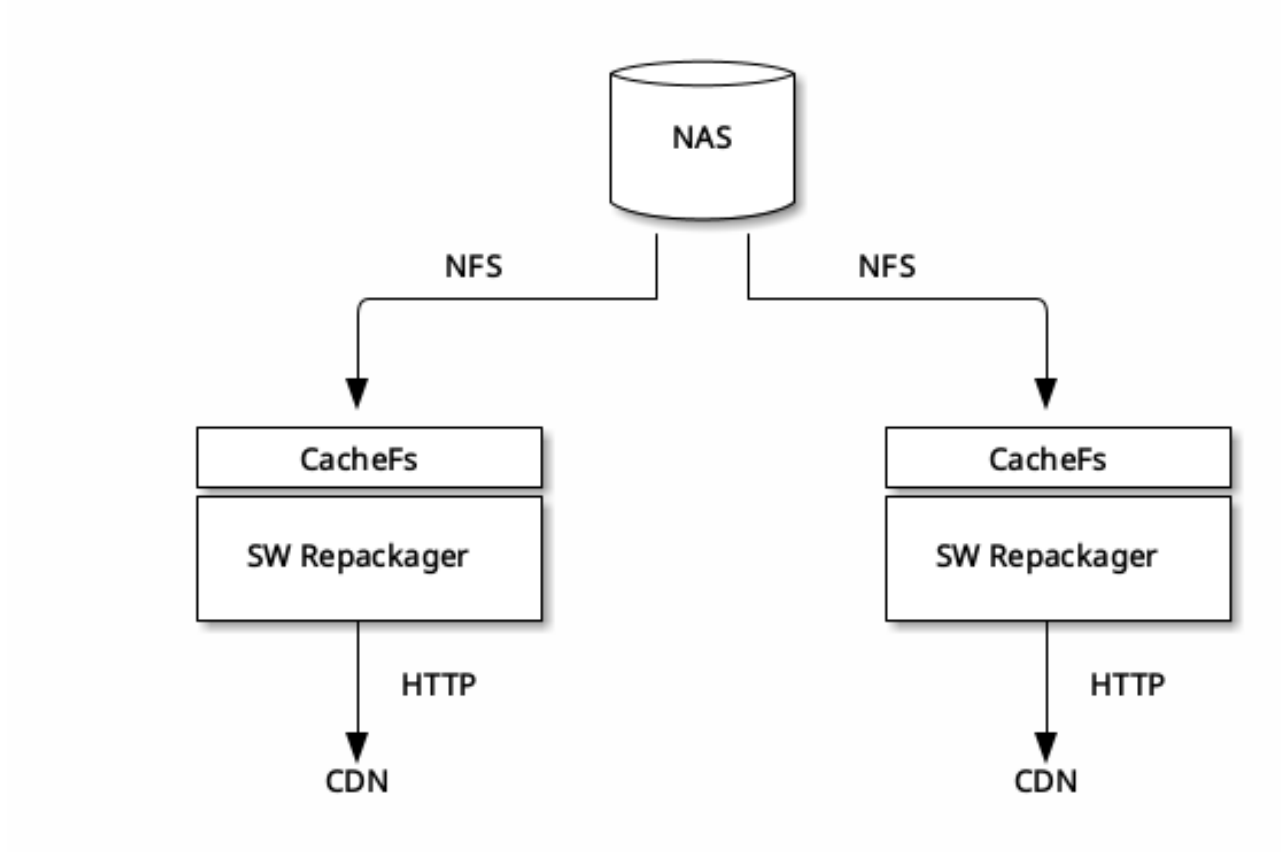
Parameters that you normally do not change.

hlsTsPcrOffsetMs Configure PCR offset in HLS output, to avoid issues reported by players. This should ideally be 0. Some players, notably VLC 3.1.1 seem to have a bug that reports an error that DTS is smaller than PCR when they are the same, and force a change of timing that results in bad lip sync. We have found that a small value of this parameter like -45 can remedy that. A better way for VLC, is to run VLC with the parameter `--no-ts-pcr-offsetfix`.

11 Performance Tuning

11.1 CacheFS

Normally the SW Repackager only caches meta data (see Cache) in RAM while the actual media data is never cached and it often becomes the bottleneck when SW Repackager serves content from an NFS server. In addition, when scaling out the number of SW Repackagers it yields more load on the NFS server which results in even slower reads. AgileTV therefore recommends the use of `CacheFS` since it both improves the read performance over NFS and also reduce load on the NFS server.



11.1.1 Install CacheFS

Install `cachefilesd`

```
dnf install cachefilesd
```

The default location of the cache is `/var/cache/fscache` but it can be changed in the configuration file `/etc/cachefilesd.conf`. See `/usr/share/doc/cachefilesd-0.10.9/README` which is available after installation for more details about configuration.

Start `cachefilesd` as follows:

```
systemctl start cachefilesd
```

Enable `cachefilesd` to automatically start after reboot

```
systemctl enable cachefilesd
```

11.1.2 Mount NFS volume using CacheFS

Add the `fsc` option when mounting the NFS volume. For example:

```
mount -t nfs -o fsc nas:/export/vod/ /mnt/vod
```

11.1.3 Test CacheFS

The easiest way of verifying that `CacheFS` is enabled it to measure the time it takes to copy a asset from the NFS Server. For example:

```
time cp /mnt/vod/big_asset.mp4 /tmp/first
```

```
real    0m7.288s
user    0m0.003s
sys 0m1.353s
```

Then copy the same file again:

```
time cp /mnt/vod/big_asset.mp4 /tmp/second

real    0m0.885s
user    0m0.000s
sys 0m0.885s
```

The first copy took 7 seconds to complete while the second took less than 1 second.

12 Troubleshooting

12.1 General troubleshooting tips

Presented here are a few general tips on what to do when a problem occurs. They all deal with gathering information, which is often the biggest part of troubleshooting.

The log files located under `/var/log/edgeware/ew-repackager/` do often contain valuable information about problems related to the repackaging service. Make it a habit to check these logs when investigating a problem.

12.1.1 Collecting logs and information

There is a script called `ew-tech-support` which is used to collect all relevant logs and information about the server into a zip archive. The script needs to be run as super user, for example:

```
user@repackager:~ ]# sudo ew-tech-support esb3002
Saving tech support info in edgeware-tech-support-repackager-2017-10-25
    ↪ T19-23.tar.bz2
```

The default output directory is `/root`. It is possible to specify the output directory using the `-o` or `--output-directory` option. The bzip2 archive `edgeware-tech-support-repackager-2017-10-25T19-23.tar.bz2` can then be sent to AgileTV for further troubleshooting.

13 Tools

13.1 Monitor services

It is possible to request statistics counters from the ESB3002. These counters include HTTP status codes and counters for DRM requests. The primary purpose of the services is to provide data for the ESB3011 Web GUI. But any other client could also consume the data.

13.1.1 Operation

Enable counters and configure monitor interval:

```
# confcli integration.counters.http.enable true
# confcli integration.counters.http.timerInterval 10
```

Action on the services:

Start, auto-start on boot or check service status as follows

```
[root@esb3002 ~]# systemctl start ew-repackager-monitord
[root@esb3002 ~]# systemctl enable ew-repackager-monitord monitord
[root@esb3002 ~]# systemctl status ew-repackager-monitord monitord
```

Details about the services:

The `monitord` service listens for HTTP requests on port 12345 and responds with statistics counters in JSON format.

The `ew-repackager-monitord` service logs to the folder `/var/log/edgeware/ew-repackager-monitord`.

The `monitord` service logs to the folder `/var/log/edgeware/monitord/`.

You can get the JSON schema with:

```
# curl localhost:12345/services/Repackager/repackager_status?schema
```

You can get the actual counters with:

```
# curl localhost:12345/services/Repackager/repackager_status
```

Sample response:

```
{
    "repackager_status": {
        "accounts": [
            {
                "drm_requests": {
                    "value": 42
                },
                "name": "default",
                "status_codes": [
                    {
                        "name": 200,
                        "value": 101
                    },
                    {
                        "name": 404,
                        "value": 2
                    }
                ]
            }
        ]
    }
}
```

13.2 Health check service

ESB3002 includes a separate service for performing health checks of the SW Repackager. The service can be started, stopped or auto-started on system boot as follows:

```
[root@esb3002 ~]# systemctl start ew-repackager-health
[root@esb3002 ~]# systemctl stop ew-repackager-health
[root@esb3002 ~]# systemctl enable ew-repackager-health
```

The health check service listens for HTTP requests on port 2000 and responds with "200 OK" if the repackaging service is considered healthy. If a problem that could cause the repackager to malfunction is detected the status "503 Service Unavailable" is returned.

The health check service logs to `/var/log/edgeware/ew-repackager-health/repackager-health.log`.

The log level of the service can be set with confcli:

```
[root@esb3002 ~]# confcli services.repackaging.healthCheck.logLevel info
```

The available log levels are "debug", "info", "warning", "error", and "fatal". The default log level is "info".

13.3 Repackager Tool

The `ew-repackager-tool` that is installed along with the SW Repackager is a utility tool that allows the user to test parts of the functionality inside the SW Repackager from the command line.

```
Usage: ew-repackager-tool [--version] [--help] <command> [<args>]
Available commands:
 request     Make a Just-In-Time Packaging request for an asset
 vod         Validate VOD assets for Just-In-Time Packaging
For further information about a command, see 'ew-repackager-tool <command> --
   ↪ help'
```

13.3.1 Make a JITP request for a managed content (ESF) using ew-repackager-tool

The following example shows how to request the master playlist (`index.m3u8`) from a managed (ESF) content called `asset`.

```
[root@esb3002 ~]# ew-repackager-tool request /media/asset/ index.m3u8
#EXTM3U
#EXT-X-VERSION:4
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",NAME="sub_chi_1",...
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",NAME="sub_eng_2",...
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",NAME="sub_chs_0",...
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="audio",LANGUAGE="und",NAME="audio_und_0
   ↪ ",...
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=4000000,CODECS="avc1.640029,...
video_0/index.m3u8
```

In the same way it is also possible to request a media playlist or segment, for example:

```
[root@esb3002 ~]# ew-repackager-tool request /media/asset/ video_0/index.m3u8
[root@esb3002 ~]# ew-repackager-tool request /media/asset/ video_0/1.ts
```

13.3.2 Test SW-Repackager configuration using ew-repackager-tool

By using the `--config` switch the `ew-repackager-tool` can be instructed to use the same configuration as the SW Repackager service. This comes very handy when trying out new configuration, e.g. presentations. (`/run/edgeware/ew-repackager-confd/ew-repackager.json`). For example:

```
[root@esb3002 ~]# ew-repackager-tool request --config /run/edgeware/ew-
   ↪ repackager-confd/ew-repackager.json /mnt/esf/asset/ index.m3u8
#EXTM3U
#EXT-X-VERSION:4
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",NAME="zho",DEFAULT=YES,..."
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="audio",LANGUAGE="und",NAME="und",..."
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1500000,CODECS="avc1.640029,..."
000001152792_1/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1000000,CODECS="avc1.640029,..."
000001152791_1/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=848000,CODECS="avc1.4d401f,..."
000001152790_1/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=448000,CODECS="avc1.42c01e,..."
000001152789_1/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=248000,CODECS="avc1.42c01e,..."
000001152788_1/index.m3u8
```

14 Alarms

These events are sent to Convoy and can be found under the "Events" tab in the Convoy GUI. See Integration for details on how to specify the event receiver.

Severity ranges from Info, Minor, Major to Critical.

Title	Severity	Description
Internal Server Error	Major	Client request could not be fulfilled due to an internal error.

15 Appendix A: Used files and ports

This is an overview of resources used by the esb3002 and related services. It can be useful for security audits.

15.1 Files

15.1.1 Written during install/upgrade

These files do not change at runtime.

Files/directories	Purpose
`/etc/edgeware/*`	Configuration
`/etc/ew-repackager/*`	Configuration etc.
`/etc/ew-repackager-health/*`	Configuration etc.
`/etc/streamviewer/*`	StreamViewer monitoring
`/usr/bin/ew-*`	Executables
`/usr/bin/telegraf`	Executable
`/usr/lib/systemd/system/ew-*.service`	`systemd` unit files
`/usr/lib/systemd/system/telegraf.service*`	StreamViewer monitoring
`/usr/lib/python3.11/site-packages/ew*`	Libraries
`/usr/libexec/ew-local-cache/*`	Executable for DRM cache
`/usr/libexec/ew-streamviewer-esb3003/*`	Executables for StreamViewer monitoring
`/usr/libexec/ew-tech-support/*`	Executable for tech support
`/var/lib/edgeware`	Empty (used by other products)
`/usr/share/edgeware`	Empty (used by other products)

15.1.2 Written at runtime when you reconfigure

Files/directories	Purpose
`/etc/logrotate.d/ew-*`	Log rotation configuration
`/etc/ew-repackager/ew-repackager*`	nginx and repackager configuration

15.1.3 Written during runtime

Files/directores	Purpose
`/var/log/edgeware/*`	Logging
`/run/redis_*.pid`	Redis pidfile (written at startup)

Note: several logfiles could hypothecially be reconfigured to other paths, but this is unadvisable. Configuration path: `services.repackaging.logging`

15.2 Ports

15.2.1 Server (HTTP listen)

Configuration path (file & json)	Default	Purpose
`/etc/ew-repackager/nginx.conf` `http{ server{ listen ...`	80	The repackaging service HTTP
`/etc/ew-repackager/nginx.conf` `https{ server{ listen ...`	disabled	The repackaging service HTTPS
`/etc/ew-repackager/nginx.conf` `http{ server{ listen ...` `↪ location = /error-log`	4999	error log fetching
fixed	2000	Repackager health check
fixed	12345	Monitord

15.2.2 Client

Configuration path (confd)	Default	Purpose
`services.repackaging.` `↪ locations.ingestServers`	8090	CBM upstreams
`services.repackaging.` `↪ locations.azureContainers`	any	Other media upstreams
`services.repackaging.` `↪ locations.httpServers`	any	Other media upstreams
`services.repackaging.` `↪ locations.s3Buckets`	any	Other media upstreams
`[...]outputProfiles.*.drm.` `↪ keyServer`	any	DRM upstream (typically 4443)

15.2.3 Local unix sockets

Socket path	Purpose
`/run/edgeware/*`	Communication
`/run/redis/cache-interface.sock`	Redis, DRM cache (EPEL8 only)

15.3 Monitord

The monitoring service `monitord`, common to several products, uses the following resources:

15.3.1 Files

These files do not change during runtime.

Files/directories, may change during installation/upgrade	Purpose
`/usr/lib/systemd/system/monitord.service`	Service file
`/usr/bin/monitord`	Executable
`/usr/lib/python3.11/site-packages/monitord*`	Library
`/etc/edgeware/monitord/*`	Configurations

These change when you reconfigure.

Files/directores	Purpose
`/var/log/edgeware/monitord/*`	Logging

Local unix socket	Purpose
`/run/edgeware/monitord/service-interface.socket`	Communicating with collector

15.3.2 Ports

Port	Purpose
12345	HTTP listen (monitord server)

15.4 Confd/confcli

The configuration service `confd`, common to several products, uses the following resources:

15.4.1 Files

These files do not change during runtime.

Files/directories, may change during installation/upgrade	Purpose
`/etc/bash_completion.d/confcli`	bash completion
`/etc/edgeware/ew-confd/cert/ew-confd.crt`	HTTPS certificate
`/etc/edgeware/ew-confd/private/ew-confd.key`	HTTPS private key
`/etc/confd/*`	Configuration schemas
`/etc/sysconfig/confd`	Configuration schemas
`/usr/lib/systemd/system/confd.service`	Service file
`/usr/bin/confcli`	Executable
`/usr/bin/confd`	Executable
`/usr/lib64/confd/*`	Executables

These change when you reconfigure.

Files/directores	Purpose
`/var/confd/*`	Current configuration
`/var/log/confd/*`	Logging

Local unix socket	Purpose
`/var/confd/service-interface.socket`	Notifications to subscribers

15.4.2 Ports

Port	Purpose
5000	HTTP listen (confd server)
5443	HTTPS listen (confd server)