ESB3003 Live Ingest User Guide

Document no	EDGS-187
Version	A25-draft
Created on	2024-10-09

Contents

1 Confidentiality notice

2 About this document

This document covers an overview of Edgeware Live Ingest and the Catchup Buffer Manager, 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 ESB3003 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 the release notes [1]. Neither does this document define any applications relying on the specific product. Please refer to the application notes for Edgeware SW Origin [2].

4 References

[1] EDGS-183 - ESB3003 Release Notes - TV Content Capture

[2] Application Note - SW Origin Live And Catchup

[3] OpenCable Adaptive Transport Stream Specification OC-SP-ATS-I01-140214

[4] MPEG-2 TS - ISO/IEC 13818-1

[5] EDGS-069 Convoy Management Software User Guide

[6] EDGS-107 Confd HTTP API

[7] MPEG DASH - ISO/IEC 23009-1 (2014)

[8] Apple HLS Authoring specification for Apple devices - General authoring requirements

[9] ANSI/SCTE 35 2017 - Digital Program Insertion Cueing Message for Cable

[10] SMPTE-2022-1-2007 - Forward Error Correction for Real-Time Video/Audio Transport Over IP Networks

[11] Nielsen-ID3-Tag-Solution-Overview-V6 - Nielsen ID3 Tag Solution

[12] EDGS-235 - ESB3003 Live Ingest Auto Configuration User Guide

4.1 Third party acknowledgement

Edgeware Live Ingest uses the Bento4 C++ library to read and write MP4 files in some workflows.

5 History

Version	Date	Changes
A1	2017-10-30	First version
A2	2018-01-15	Added Catchup Management section
A3	2018-02-12	Added Monitoring section
A4	2018-03-07	Added ATS unicast input support
A5	2018-04-13	Added AC-3 and E-AC-3 support Added sections on SCTE-35 ad markers
A6	2018-04-18	Added section on channel state configuration
A7	2018-04-26	Added information on language set from PMT
A8	2018-05-14	Add section on redundancy
A9	2018-08-28	Add info on subtitles support
A10	2018-09-21	Add source-specific multicast support
A11	2018-10-10	Updated monitoring states
A12	2018-07-19	Add section on how to upgrade
A13	2019-02-20	Add section on input source configuration, including RTP and FEC
A14	2019-03-14	Change text about restarts due to configuration mismatch
A15	2019-03-25	Document change to avc3 as sample entry format for AVC. Add section about allowed input changes without configId update. Add section on Partial Catchup Buffer Blackout.
A16	2019-06-30	Virtual channels from other live channels.
A17	2019-07-15	Add information about input source problem handling.
A18	2020-06-16	Virtual channels with VOD sources. SCTE-35 scheduling for virtual channels.
A19	2021-01-21	Generate silent segments for missing audio input.
A20	2021-12-03	Added SRT input support
	2021-12-09	CBM accelerator cache
	2021-12-10	Appendix: Used files and ports
	2022-03-08	Configuration naming rules
A21	2022-04-06	Update naming rules
A22	2022-05-24	Remove chapter about virtual channels
	2022-06-01	Add chapter about DVS signaling
	2022-06-24	Update accelerator cache content
	2022-09-13	Update socket buffer memory recommend
	2022-09-21	Add chapter for Nielsen ID3 markers
A23	2022-11-18	Update cache clear in blackout
	2023-01-17	Update supported Linux distribution
	2023-02-07	Some notes about performance
	2023-02-08	Configuration of OCR
	2023-02-13	Add audio drift adjustment
A24	2023-04-21	Source-specific multicast failover

Version	Date	Changes
	2023-08-02	Handle subtitle time deviation
	2023-04-07	More details about track skew limits
	2023-09-25	Allow to disable clock drift checking
	2023-10-13	Add note about segmentation margin for late ad markers
A25	2023-11-14	Update accelerator cache content
	2023-11-16	Update OCR configuration
	2023-12-25	Toggle video DTS fluctuations auto adjustment
	2024-02-27	Update Tesseract languages installation instructions
	2024-04-15	Drop RHEL7 support
	2024-05-03	Update installation instructions
	2024-05-10	Update Used files and ports
	2024-05-13	Update Installation prerequisites
	2024-07-02	Update Redundancy

6 ESB3003 Live Ingest Overview

The Edgeware Live Ingest product is responsible for segmenting live input sources into Common Media Application Format (CMAF) based segments that later are repackaged and distributed to end user clients, in various formats, by other Edgeware products:

- Software Repackager
- Software Streamer

Edgeware Live Ingest also enables scheduling of virtual channels, i.e. channels that may be repackaged as any other live input channel, with segments from other live channels or VODs, based on a calendar schedule.

Supported input source format is Adaptive Transport Stream (ATS) [3], received as UDP multicast or unicast. Edgeware Live Ingest is configurable via the Edgeware `confd` service, that is populated with configuration for properties like channels and their video and audio track properties. See Configuration for additional details on how to configure the service.

Edgeware Live Ingest can output both video channels and radio channels. A video channel contains one or more video tracks, the first video track in the configuration is used as reference track. For a radio channel no video track shall be configured. A radio channel must contain one or more audio tracks and it can also contain subtitle tracks. The first audio track in the configuration is used as reference track. SCTE-35 is not handled in radio channels.

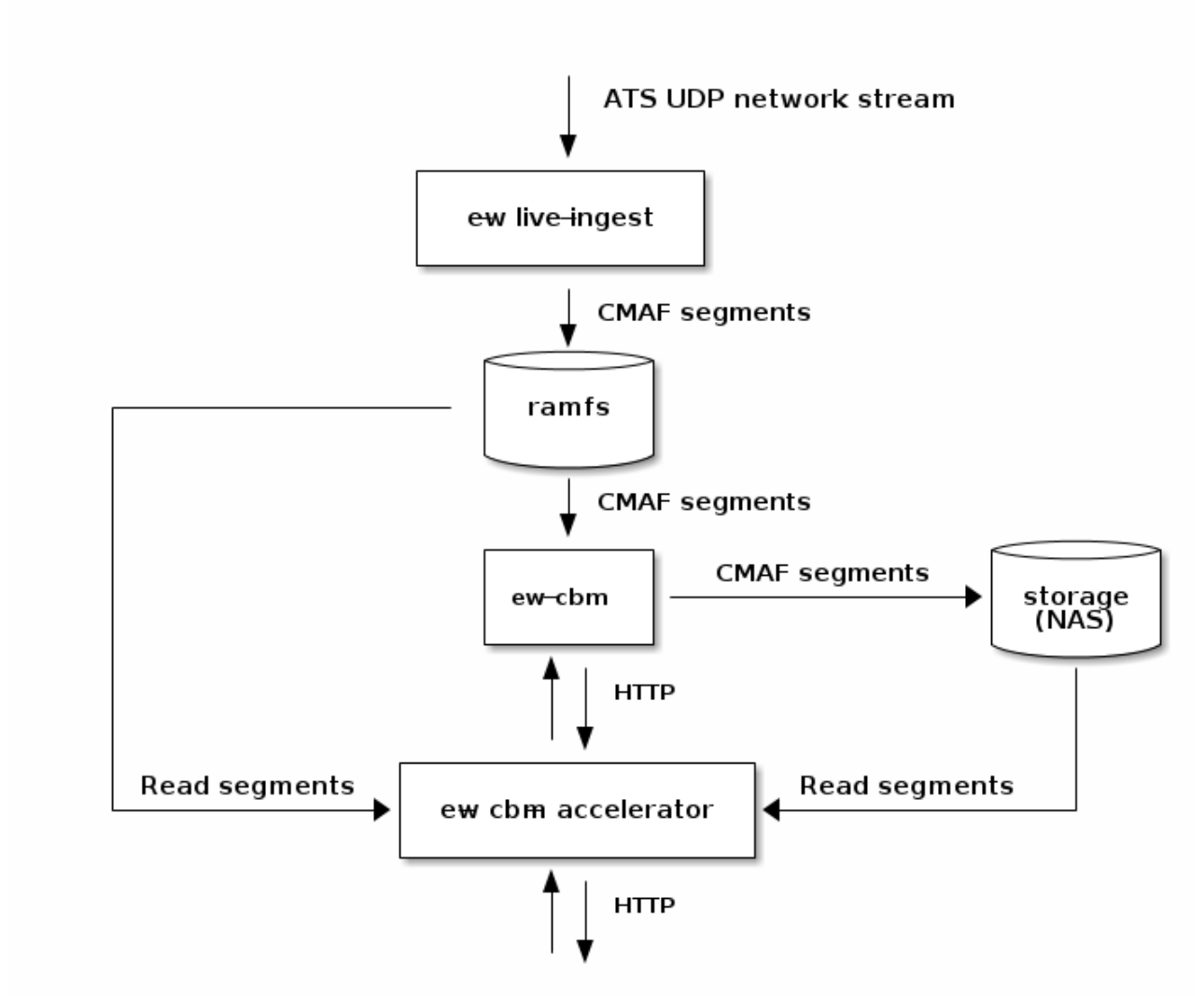
6.1 Services

The Live Ingest product consists of the following services.

Service Name	Description
ew-live-ingest	Ingests ATS UDP streams and write CMAF segments into the ramdisk.
ew-cbm	Catchup manager, copy segments from ramdisk to persistent storage and serve incoming HTTP requests from Software Repackager.
ew-cbm-accelerator	HTTP server that offloads the ew-cbm by serving all segment requests.
ew-vc-scheduler	Scheduling daemon for virtual channels.
ew-vc-content-selector	Plugin for ew-cbm-accelerator HTTP server to support virtual channels.
ew-radicale	"radicale" calDAV calendar server (https://radicale.org/) for virtual channel calendar.
confd	Configuration daemon.

6.1.1 Live ingest data flow

The following diagram shows how data flows through the system and how the different services interact with each other.



6.2 Input stream requirements

Edgeware Live Ingest supports ingest of Adaptive Transport Streams (ATS) according to Cablelabs OC-SP-ATS-I01-140214 [3].

Video frames must be PES-aligned. Audio frames are recommended to be PES-aligned for optimal performance.

ATS network input can be provided as either UDP multicast or UDP unicast network streams. Multicast may be received using any-source multicast (using IGMPv2) as well as source-specific multicast (using IGMPv3).

RTP input over UDP is supported. If forward error correction (FEC) according to SMPTE-2022-1-2007 [10] is enabled on the source, Live Ingest can be configured to take advantage of that to increase robustness against packet loss.

See Input source configuration for details on how to configure Live Ingest to use different types of sources.

6.2.1 Adaptive Transport Stream

Only EBP and RAI markers that point to an IDR picture will be considered when looking for segment boundaries (the actual marker that should be used is specified in the Live Ingest service configuration, see Configuration). If other EBP or RAI markers appear in the input stream, they will be ignored and a warning will be printed in the log.

6.2.1.1 Error condition behavior

6.2.1.1.1 Transport stream discontinuity

ISO/IEC standard 13818-1 [4] states that a transport stream (TS) should have its Continuity Counter per PID, incremented by one for each incoming TS packet. If this is not fulfilled by the input stream, this is interpreted as a packet loss, and an event is sent to the management system, followed by a configurable action. See Input source problem handling for details on configuring this action.

An announced discontinuity by the TS packet Adaptation field Discontinuity Indicator, is treated the same as any other stream discontinuity. An event is sent to the management system, followed by the configured action.

6.2.1.1.2 Clock drift

Segmentation is based on the incoming timestamps in the input stream, and for the segmentation to work correctly, these timestamps must not have an accumulated drift compared to NTP time. It is therefore recommended to have both the encoder and Live Ingest nodes synchronised to the same NTP source.

If a clock drift of more than a threshold between the input stream and the Live Ingest node is detected, an error event will be sent to the management system and the channel will be restarted. The threshold is defined in Advanced configuration:

- `maxClockDriftMs`. Default value is 5000ms. Value `-1` will disable the check.

To investigate clock drift, the tool `ew-tsanalyzer` can be run to check fluctuations in timestamps and drift in PTS/DTS and PCR clock. The tool will generate a JSON report that can be further analyzed.

6.2.1.1.3 Audio track drift

The ideal output audio has constant frame duration with fixed timestamp increases. However the source doesn't always ensure that, for example because of channel stitching or encoder issue, and the drift can be accumulated over time until there is recognizable A/V sync issue at playout.

There are two entries in Advanced configuration:

- `avSyncWarningThresholdMs` output log on drift exceeds threshold.
- `alignAudioFramesThresholdMs` adjust **incoming** drift smaller than threshold. Audio frames can be inserted (silent), removed or shifted depends on how difference actual duration is compare to ideal frame duration.

Similar to Clock drift, `ew-tsanalyzer` can help investigating.

NOTE: `alignAudioFramesThresholdMs`

- can't distinguish between small drift (i.e. encoder issue) and bigger jump of missing frames (i.e. input signal loss).
- can't adjust past drift (e.g. drift occurs before enabling it), hence it should be enabled when channel is in good state with current drift 0.
- won't handle big drift exceeds threshold, only error log is output. Manually restarting channel or fixing source are needed.

6.2.2 SCTE-35 ad markers

Edgeware Live Ingest can be configured to honor and propagate SCTE-35 [9] ad markers from the input stream to allow ad insertion to be performed downstream. See Configuration for details on how to enable this feature.

The following limitations apply to the SCTE-35 support:

- Only `splice_insert` commands are supported
- `time_signal` commands are not supported
- Component Splice Mode is not supported
- Encrypted SCTE-35 messages are not supported

SCTE-35 messages that don't adhere to the above will be ignored.

The splice time announced in the SCTE-35 message does not need to match a segmentation marker exactly. If not exact, the closest marker of the configured type (EBP or RAI) will be used.

Note that non-SCTE-35 segmentation markers must continue to appear at the same intervals throughout the ingest. This makes it possible to match a SCTE-35 splice time by moving a single segment boundary, for example to create one 3-second segment followed by one 5-second segment instead of the two 4-second segments that would normally have been created. In this way, it can be ensured that the segment numbering continues to be correct, and that no segment is shorter than 50%, or longer than 150%, of the configured segment duration (`exactAverageSegmentDuration`). This also means that splice times must be at least one ideal segment duration apart for the segmentation to work correctly.

NOTE: When SCTE-35 ad markers are used, the skew limit of audio track is affected (see Track skew)

6.3 Output format

The segments produced by the Edgeware Live Ingest product will be based on Common Media Application Format (CMAF). For video, avc1/avc3 or hvc1/hev1 sample entries are used together with inband propagation of parameter sets. The parameter sets acquired at startup are also stored in the init segments.

7 Installation

This section describes how to install Edgeware Live Ingest on a dedicated machine. As a starting point you will need the Edgeware Live Ingest installer, named according to the pattern `install-esb3003-x.y.z`. To install the software, follow the steps below.

7.1 Prerequisites

Edgeware ESB3003 Live Ingest requires RHEL 8.8+ as operating system. Oracle Linux 8.8+ are unofficially supported.

7.1.1 Hardware

Contact Edgeware for information regarding the recommended hardware setup.

7.2 Preparation

7.2.1 SELinux

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

7.2.2 Set machine id

The machine-id must be set. Use command

```
[root@esb3003 ~]#cat /etc/machine-id
```

to view the machine-id. If it is not set create a new machine id using command

```
[root@esb3003 ~]#systemd-machine-id-setup
```

and reboot the node.

7.2.3 Create a RAM disk for live segments

The live ingest service assumes that a RAM disk is available at `/mnt/ramdisk`. A RAM disk ensures that ingested data can always be written.

The default ramdisk duration is 60 seconds. It can be configured for each segmentation template with the `circularBufferS` parameter. It is not recommended that you decrease this value, and if feasible, it is better to double it.

NOTE:

- The size of the ramdisk should be based on: `circularBufferS` x total bitrate of all variants in all channels.

For example: **100 channels** with 10 Mbps total per channel for the default 60 second duration ⇒ 60s x 1000 Mbps / 8 = 7500 MB ≈ **8 GB**)

Create and mount the disk:

```
# mkdir -p /mnt/ramdisk
# mount -t tmpfs -o size=8G tmpfs /mnt/ramdisk
```

Make the RAM disk mount boot-persistent by adding it to `/etc/fstab`.

- `tmpfs` might use swap if system is running low on free memory or too high value of `vm.swappiness` and badly affect performance.

7.2.4 Create catchup location for segments

The catchup buffer manager service requires a storage location where media segments are stored. The storage can either be a local storage or a remote mounted storage. The default configuration uses `/media/cb/` as catchup location.

You must create a directory at the storage location manually, and make sure that the `edgeware` user has write permission there.

7.2.5 Increase socket receive buffer memory

In order to avoid packet loss the maximum receive buffer memory for network sockets must be **at least** 1048576 bytes.

First, check the current system value

```
# sysctl net.core.rmem_max
```

If it is lower than 1048576, then add the following line to `/etc/sysctl.conf`:

```
net.core.rmem_max = 1048576
```

To apply the changes without rebooting:

```
# sysctl -p
```

You can verify the change by running:

```
# sysctl net.core.rmem_max
net.core.rmem_max = 1048576
```

7.2.6 Firewall

The following ports must be opened in the firewall:

Port	Description
8090	HTTP API used by Software Repackager (ESB3002)
5000	Control port for the configuration service. Needed if Convoy is used for propagating the configuration.
TS input	The TS input ports that are configured in `services.liveIngest.channels.*.inputSources.*.source`

Note: Firewall also needs to be configured to allow the ATS input network streams.

7.2.7 Enable NTP

All nodes in an Edgeware installation requires the clocks to be synchronized. Make sure that an NTP service (e.g. Chrony) is installed and configured on the new machine.

7.3 Perform the installation

Install Live Ingest by running the install script:

```
chmod +x install-esb3003-x.y.z
./install-esb3003-x.y.z
```

7.4 Starting the services

To start the services, run

```
systemctl start ew-live-ingest ew-cbm
```

To enable the use of virtual channel scheduling, run

```
systemctl start ew-vc-scheduler
```

Check status with

```
systemctl status <service>
```

in which <service> is one or more of `ew-live-ingest`, `ew-cbm`, `ew-cbm-accelerator`, `ew-vc-scheduler`, and `ew-radicale`.

To automatically start the service on boot, run

```
systemctl enable ew-live-ingest ew-cbm ew-cbm-accelerator ew-vc-scheduler
```

Notes:

- Starting `ew-cbm` starts `ew-cbm-accelerator` automatically, in fact, it would also start `ew-live-ingest`.
- On system that the number of channels is many times more than CPU cores, it is recommended to separate the set of cores running ew-cbm and ew-live-ingest by using systemd CPU affinity, to avoid ew-cbm workers' peaks affects ingestion of ew-live-ingest workers run on same core.

8 Upgrade

Note: For upgrading a redundant system, see the chapter Redundancy.

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 the upgrade option `-u`:

```
chmod +x install-esb3003-x.y.z
./install-esb3003-x.y.z -u
```

Finally start the services as described above in the section Starting the services.

9 Configuration

The system is configured via the `confd` service. Configuration changes can be made either by using the command-line tool `confcli`, or by injecting JSON directly via confd's HTTP API.

Note: After a new installation it is necessary to open a new shell for `confcli` to work properly. Log out and back in again.

9.1 confcli

To add a configuration, for a previously not configured service using the `confcli` tool, issue the following command as super user for wizard mode:

```
confcli services.liveIngest -w
```

To read or change an individual attribute in the configuration, confcli may traverse the configuration hierarchy by adding a period between levels.

Example command to read a specific attribute:

```
confcli services.liveIngest.logging.general
```

Example command to write a specific attribute:

```
confcli services.liveIngest.logging.general DEBUG
```

`confcli` can also be used to get documentation on a specific configuration attribute, by using the `-h` option.

9.2 confd HTTP API

It is also possible to apply a full configuration from a JSON file, e.g. `config.json`, by issuing the following command:

```
curl -X PUT -H "Content-Type: application/json" -d @config.json http://
    ↪ localhost:5000/config/__active/
```

This will set the active configuration and write it to `/var/confd/config/_active`.

In order to work with channels on an individual basis, you can get the JSON for a channel named "x" with:

```
curl -X GET http://localhost:5000/config/__active/services/liveIngest/
    ↪ channels/x
```

A channel can be created, or updated if it already exists, with

```
curl -X PATCH -H "Content-Type: application/json" -d @x.json http://localhost
    ↪ :5000/config/__active/services/liveIngest/channels
```

where x.json contains the fields that should be created or updated. When updating, only the fields present in the supplied JSON will be affected. The file x.json should have the following structure:

```
{
    "channels": [
        {
            "name": "x",
             ...
        }
    ]
}
```

For more information about how to work with the confd API, see [6].

9.2.1 Pre-validation of configuration

To avoid errors when applying a configuration via the confd HTTP API, a configuration file may be manually pre-validated using `ew-live-ingest-tool`:

```
ew-live-ingest-tool --validate <JSON configuration file>
```

The configuration file may be a full configuration (as received by `confcli services`) or a configuration file with `liveIngest` as its top JSON level (as received by `confcli services.liveIngest`).

You can also check the input signal by using the `ew-tsanalyzer` tool. With a target segmentation duration specified, it reports AV codecs, bitrates of segments, and if ebp or rai markers have the right distance so that succesful segmentation can take place.

9.3 Channel configuration

9.3.1 Naming rules

Endline and tab are prohibited in any name.

There are two types of names:

- Name that is used as key in other parts of configuration, and in `confcli` autocomplete, such as channel name, track name, channel template name and segmentation template name, MUST NOT contain any space and MUST NOT BE empty. Also, all shell special characters MUST BE avoided.
- Name is used mostly for displaying or logging, such as channel alias name or track display name, CAN contain space, dot and CAN BE empty.

NOTE: `confcli` autocomplete doesn't work if names contain dot (`.`), which is `confcli` separator. In that case, use array index number manually to access, i.e. `services.liveIngest.channels.0`.

9.3.2 Configuration updates

Most configuration changes that affect a channel cause the channel to be automatically restarted. An exception to this rule is changing log levels (see section on logging, below), which takes effect immediately and does not cause any channel restarts.

9.3.3 Configuration ID

The `configId` parameter makes it possible to change the channel encoding and/or confd configuration and keep the history of which time interval corresponds to what output configuration. To achive this, any change in the content encoding or the channel configuration that results in a non-compatible output must be reflected in a change (increase) in the configId parameter, that separates configurations in time. This enforcement is triggered at start, confd reconfiguration, or reception of a change in media by comparing the a new generated `content_info.json` with the last generated file if found in the channel's local RAM disk area.

For example, adding or removal tracks require a change in configID, since the track list in HLS master playlists cannot be updated during streaming sessions. On the other hand, changes to pure input parameters like IP addresses and PIDs do not influence the output.

However, some changes to configuration parameters and changes in incoming media result in changes of the output "configuration" that are less critical so that players can handle them. They do therefore not require a change in configId. These cases are described in Input change without configId update and Confd config change without configId update below.

For all other changes that influences the output, the channel will be shut down. A change must also adhere to the configuration validation rules and mechanisms described in Configuration validation.

In case of an updated input stream, bringing up the channel again is done by either reverting the input stream change, or making sure that the channel configuration matches the input stream updates and updating the `configId` parameter. In case of an updated confd configuration, the channel can be brought up again by either reverting the configuration change, or making sure that the input stream matches the new configuration and updating the `configId` parameter.

9.3.3.1 Input change without configId update

Generally a change of properties in a track input stream, would require an updated `configId` in order to be able to generate correct meta data for manifests and initialization segments to players. However, many players allow correct playback even if some properties of the media segments don't match what was announced in such meta data. When this happens, a warning will be printed in the log, and an informational event sent to Convoy. In case there is a permanent change of properties for a track, the recommendation is to update the `configId`.

The following changes to the input are allowed:

- Changed channel configuration for an audio track, for example from stereo to 5.1 or vice versa.
- Changed language in PMT for audio and subtitle tracks
- Added or removed PIDs in PMT for tracks that are not included in the channel configuration
- Changes in SPS, PPS or VPS for video tracks

Note that manifests may not be entirely accurate after such a change.

9.3.3.2 Confd config change without configId update

{#confd-config-change-without-config-id-update} Similar to certain cases in the input media, there are changes to the confd configuration of the channels, either directly or via their segmentationTemplate and channelTemplate that have a relatively minor influence on the output and are therefore allowed to be changed without a `configId` change.

Currently these are:

- Changed displayName
- Changed CEA608 embedded CEA-608 signalling

Again, note that manifests may not be entirely accurate after such a change.

9.3.4 Channel name alias

A channel may be configured with an alias for the channel name, that could help making administration easier. E.g. for listing channels using `ew-live-ingest-tool`. The channel name alias is only for administrative purposes, and can not be used in e.g. request URIs. Channel name aliases may be used for and in log files, by setting the `services.liveIngest.advanced.useChannelNameAliasForLogging` configuration option to true. Changing the channel name alias for a channel, causes the channel to restart.

9.4 Input configuration

9.4.1 Channel state

Each channel has a configured state, which is controlled by setting the `state` parameter of the channel to either `enabled` (default), `catchup`, `cbm-only`, or `disabled`.

State	Ingestion	Storage	Output
Enabled	Yes	New segments stored	Live and catchup
CbmOnly	Other service	New segments stored	Live and catchup
Catchup	No	No segments stored	Catchup only
Disabled	No	No segments stored	Neither live nor catchup

In all four states, the tail of the circular catchup buffer is trimmed when falling outside the buffer window size. In order to immediately remove the channel content from the storage the channel must be removed from the configuration. In the catchup and disabled states no new data is copied to the storage. Hence content from that period will not be available for catchup even if the channel is later set to enabled.

9.4.2 Input source configuration

9.4.2.1 RTP

When an RTP source is used, that must be stated explicitly in the Live Ingest configuration by including the scheme `rtp://` in the `source` property of the input source. Example:

```
rtp://224.1.2.3:2000
```

9.4.2.2 Forward Error Correction (FEC)

For RTP sources that have FEC according to SMPTE-2022-1-2007 enabled on the encoder, forward error correction can be enabled in Live Ingest by setting the `errorCorrection` property of the input source.

If the source uses one FEC stream (also known as one-dimensional FEC, FEC-1D or Level A), set `errorCorrection` to `fec-1d`. If the source uses two FEC streams (two-dimensional, FEC-2D, Level B), set it to `fec-2d`.

Note that there is no need to specify addresses and ports for the FEC stream(s) in Live Ingest since they will be deduced from the address and port of the source RTP stream. The number of FEC rows and columns are also automatically detected.

Refer to the encoder documentation and FEC specification [10] for details on how to configure FEC to strike a good balance between bandwidth overhead, latency, and robustness. As an example, a wide FEC matrix (many columns) can handle longer bursts of lost packets at the cost of increased latency, while a smaller matrix increases robustness against random packet drops at the cost of increased bandwidth overhead.

9.4.2.3 Unicast

To receive an input source using unicast, configure the encoder to push to an IPv4 address of the Live Ingest machine and an arbitrary port. In Live Ingest, specify the same address and port in the `source` property of the input source. Alternatively, specify "0.0.0.0" as the address to make Live Ingest listen on all available IPv4 interfaces. Examples:

```
10.11.12.13:2000
0.0.0.0:2000
```

9.4.2.4 Source-specific multicast

To use source-specific multicast, specify the desired source address in the configuration property `igmpV3Source`.

`backupIgmpV3Source` can be used with `igmpV3Souce` for failover in case input source error. It strictly requires a working redis database of ingest redundacy. The last-known-good source is selected. In another words, after the failover to good backup source, then the primary is fixed, to switch to it again, the backup source has to be bad.

NOTE:

- The channel restarts to perform the failover
- The channel won't start if there is redis connection error
- The failover source might be not synchronized to the main source (different segmentation markers, dts/pts, etc.), but must be exactly compliant (tracks, pids, bitrates, markers interval, etc.)

9.4.2.5 SRT - Secure Reliable Transmission

SRT allows ingesting streams to Live Ingest with a *retransmission* mechanism to provide a more reliable communication. This provides an efficient way of recovering from missing packets in terms of bursty and random packet drops in addition to packet reordering.

To receive an input source using SRT, configure the encoder to push to an IPv4 address of the Live Ingest machine and an arbitrary port as an SRT *Caller*. Live Ingest always acts as an SRT *Listener*. Note that SRT Rendezvous is currently not supported. In Live Ingest, specify the same address and port in the `source` property of the input source. Alternatively, specify "0.0.0.0" as the address to make Live Ingest listen on all available network interfaces. Examples:

```
srt://10.11.12.13:2000
srt://0.0.0.0:2000
```

9.4.2.5.1 Encryption

Live Ingest supports encryption over SRT. Configuring encryption for a channel using SRT is done with the *optional* parameter `srtPassphrase`. `srtPassphrase` parameter is configured individually for every input source entry in a channel. Example:

```
confcli services.liveIngest.channels.channelXYZ.inputSources.0.srtPassphrase
    ↪ secret_key
```

When setting a passphrase, the passphrase **must** be between 10 to 79 characters to be considered valid. If the parameter is set to an empty string (""), or if the parameter is omitted from the channel's configuration, encryption will be disabled. The derived encryption key length will default to what the encoder is advertising in the SRT handshake. If both sides have set no preference to key length, it will default to AES-128.

Note! It's important that both the encoder and Live Ingest use an *identical* passphrase. Furthermore, if only one of the sides have encryption configured, ingest will fail. So either both sides have encryption configured, otherwise no side should have encryption configured.

9.4.2.5.2 Latency

The receive latency for SRT can be configured using `services.liveIngest.advanced.srtReceiveLatencyMs` and is provided in milliseconds. The actual value of the latency is however also affected by the RTT and the encoder (if it has a desired value for its peer's latency). The default value of this parameter is 120ms. This value determines how long data received will stay in the SRT receive buffer (until it's available for the Live Ingest worker). Increasing the value can improve reliability. Decreasing this value is **not** recommended.

9.4.2.5.3 Receive buffer length

The SRT buffer receive buffer will be set to the same parameter as with other ingest methods, `services.liveIngest.advanced.udpReceiveBufferSizeB`. However,the actual size of the receive buffer will not match the size provided exactly, but it will rather be the multiple of 1456.

9.4.2.5.4 Notes on recovery

SRT can not be expected to recover from all types of bad network conditions. For example, SRT configuration options such as bandwidth overhead configured on the encoder side will affect the ability to recover packets. Further more, packets will be considered *lost*, if they have not been received (as a normal send or retransmitted send) after `1020ms + services.liveIngest.advanced.srtReceiveLatencyMs + (~1/2 RTT)`.

9.4.2.6 DVS - Descriptive Video Service

Live Ingest supports to ingest DVS signaling content via role in channelTemplate configuration:

```
confcli services.liveIngest.channelTemplates.<channel-template-name>.tracks.<
    ↪ track-name>.role`
```

For the primary audio and video track, the default value is empty string. For the secondary audio track to support DVS signaling, the config is set to `description`. For subtitle tracks, this is unused, subtitle role is set in `teletextType` configuration.

The DVS signaling configuration will be propagated to Live Ingest output and can be used as input for ESB3002 and ESB3005 products.

9.4.2.7 Nielsen ID3 markers

Edgeware Live Ingest can be configured to honor and propagate Nielsen ID3 [11] markers from the input stream to allow audience measurements.

```
confcli services.liveIngest.channels.<channel-name>.nielsenMarkerTracking
```

Options:

"off": Nielsen markers are thrown away. "auto": Auto detect Nielsen markers and propagate them. PID value: Propagate Nielsen markers in specified PID. If there are Nielsen markers on other PIDs then they are thrown away.

The Nielsen ID3 markers will be propagated to Live Ingest output and can be used as input for ESB3002 and ESB3005 products.

9.4.3 Configuration validation

The provided configuration is subject to validation. The first step is a simple cross-check of configurable parameters. See Static validation for details. The second step verifies that the configuration can be applied to incoming data. This step is called dynamic validation. See Dynamic validation.

If the configuration fails validation, affected channels are shut down. If static validation failed, they will remain stopped until a valid configuration is sent to confd or the channel is explicitly restarted using the `ew-live-ingest-tool --start <channel>` command. If the dynamic validation failed, the channels will be periodically restarted to automatically recover in case the input source changes to match the configuration.

Channels are not allowed to restart too frequently, in order to protect the system. If a channel restarts, it will be scheduled for start using a back-off mechanism that will gradually increase the start delay for the channel, until the channel has been running stable for a long enough time period (30s). Maximum restart delay is 60 seconds.

9.4.3.1 Static validation

The following tests are carried out:

- Segmentation time scale must be compatible with the internal video time scale (180,000 ticks per second). That is, 180,000 must be an even multiple of the segmentation timescale.
- Average segment duration must be compatible with video frame rate (for example, if the video frame rate is 25 Hz, the segment duration must be an even multiple of 40 ms)

9.4.3.2 Dynamic validation

9.4.3.2.1 Video frame rate

A video frame rate in Hz should be set in configuration property `videoFrameRate`. Depending on the value set, another frame rate (double or half) will also be supported. This allows for channels to mix two or more tracks with different but compatible frame rates.

Allowed and compatible values are "25" and "50", "29.97" and "59.94", and "30" and "60". The actual frame rate used by a track will be automatically detected at run time. The used frame rate must be compatible with the selected average segment duration (`exactAverageSegmentDuration`).

9.4.3.2.2 Timestamp correction

In order to maintain high robustness against poorly encoded streams, the ingest worker will perform automatic correction of incoming dts timestamps that don't quite align with the expected values.

For example, assuming the video frame rate is 25 Hz, the frame duration should be 3600 ticks (assuming a 90 kHz clock). If the actual dts values don't agree with this ideal, they will be automatically corrected to do so. As long as the differences even out over time, there will be no problem. However, if the accumulated difference between the ideal and the actual timestamp should ever exceed 1 second, the ingest will be restarted. As an example, this will happen if the dts difference between two frames is consistently a little larger than 3600, since the difference between actual and ideal dts will then grow over time until the 1-second threshold is reached.

The ingest will also be restarted if the dts difference between two consecutive frames is off from the ideal value by more than 5 ticks.

9.4.3.2.3 PMT

The Program Map Table contains information (codec, pid, media type, etc) about all tracks present in the input stream. The ingest worker compares the PMT with configuration and halts execution on mismatch. PMT updates cause automatic restart of the affected ingest worker.

The language used for a track can be specified in the configuration or read from the PMT. If a language is configured as empty ("") the one from the PMT will be used. If a language is configured but does not match the one in the PMT, the one from the configuration will be used, but a warning will be printed in the log.

9.4.3.2.4 Track bitrate

Some adaptive media streaming formats, e.g. HLS put strict requirements on segment and/or average bitrates, see [8]. Using the wrong bitrate could cause some clients to be unable to play the segments.

The configured track bitrates should correspond to the actual bitrates of the input stream. If the momentary segment bitrate, or the longer term average track bitrate, is lower or exceeds certain limits, warnings or errors will be issued in the log, together with management system events. An error will also cause the channel to be stopped. The configuration will then need to be updated with a matching bitrate. The average bitrate is calculated as an average of segment bitrates over approximately 60 seconds (depending on configured segment duration).

- Video track levels:
  - Segment bitrate Warning level: Segment bitrate > 125%
  - Average bitrate Warning level: Average bitrate > 110% or < 50%
  - Average bitrate Error level: Average bitrate > 200%
- Audio Track levels:
  - Segment bitrate Warning level: Segment bitrate > 110% or < 80%
  - Segment bitrate Error level: Segment bitrate > 200% or < 50%
  - Average bitrate Warning level: Average bitrate > 110% or < 80%
  - Average bitrate Error level: Average bitrate > 140% or < 70%

Subtitle tracks also need to have a reasonable bitrate specified in order to give client players some idea of what to expect. Since the size of the actual text cues is quite small, the bitrate as seen by an end client can differ a lot depending on the output format and amount of cues (WebVTT subtitles used for HLS are more compact than TTML subtitles used for DASH, for instance), but specifying a bitrate of 1000 bps for subtitle tracks should work fine for most cases.

9.4.3.2.5 Segment creation

All tracks for a channel must periodically create new segments. If a Live Ingest channel hasn't been able to create a new segment for an individual track within three average segment durations, the channel will automatically restart.

9.4.3.2.6 Track skew

To avoid unnecessary latency and ensure that segments are written on time, the input streams for all tracks in a channel should be aligned properly in time. For example, if a certain PTS arrives later for audio or subtitles than for video, that will delay the segmentation since a segment cannot be published until it is available for all tracks.

If two tracks are found to be further apart than the configuration property `maxSkewForMediaComponentMs` indicates when creating a segment, a warning will be written to the log and a management system event will be sent to bring attention to a potential source configuration problem.

For subtitle tracks, `maxSkewForMediaComponentMs` also specifies the maximum amount of time to wait for incoming subtitle data before creating a segment. It therefore makes sense to set it as low as possible when subtitle tracks are present. Warnings will be printed in the log if subtitle data is being discarded because it arrives too late, so it is possible to experiment to find an optimal value.

NOTE: For audio tracks, the skew limit can also be affected by whether the SCTE-35 ad markers are used or not and silent audio is enabled or not. The segment duration for ad break can be in range from 50-150% of `exactAverageSegmentDuration`. Meanwhile, when silent audio is enabled, the live ingest service will determine if it has enough data to produce normal or silent segment. In case of both are in use, to avoid generating silent audio segments (because it treats late samples as lost), the track skew limit is based on 1/2 * `exactAverageSegmentDuration` and `exactAverageSegmentDuration` otherwise.

9.4.4 Segmentation configuration

In order to comply with the DASH specification's requirements on segment availability time [7], it may be necessary to introduce a margin to ensure that a segment corresponding to a certain wall clock time is written on time. This could be the case, for instance, if the audio and video tracks are poorly aligned in the input stream, if a video sync frame arrives right before the wall-clock time of an ideal segment border, or if a downstream CDN introduces latency that delays segment availability to clients.

To adjust the timing of segment writes, set the `minimumSegmentationMargin` attribute in the segmentation template. To find a suitable value, the reported publish margin can be used as a guide. The publish margin is available as `publish_margin_ms` in Segment Published events, and is also logged on debug level in the channel worker log when a segment a published. Negative publish margins are generally a sign that you should consider increasing `minimumSegmentationMargin`.

The practical effect of increasing the margin is that an input frame that arrives at a certain wall clock time will have its media time, which can be seen as an outgoing wall clock time, moved into the future as needed to reach the requested margin. In summary, the segment is made available early with respect to its ideal availability time.

Note that only DASH clients are affected by this setting, since for example HLS clients may play back a segment as soon as it's available to download.

9.4.4.1 AVC/HEVC sample description format

For AVC video output, the sample description format is configured by the `avcSampleDescriptionFormat` parameter. The possible values are `avc1` and `avc3`. The default value is `avc3`.

For HEVC video output, the sample description format is configured by the `hevcSampleDescriptionFormat` parameter. The possible values are `hvc1` and `hev1`. The default value is `hev1`.

9.4.5 Output base path

Configuration of ESF output is configured on service level. The output from all channels is stored under a common base path, where each channel has its own subtree structure.

The default output base path is `/mnt/ramdisk`.

The output base path is configured in `/usr/lib/systemd/system/ew-live-ingest.service`

Live ingest output directory structure example:

```
/mnt/ramdisk/
   /channel1/
      /cfg_0/
      /cfg_1/
      ...
      /cfg_n/
   /channel2/
      /cfg_0/
      /cfg_1/
      ...
      /cfg_n/
```

9.4.6 Subtitle track configuration

Each subtitle track in a channel should produce subtitles for one specific language. Because multiple subtitles often appear on the same source PID, it is necessary to specify enough information in each track to make it possible to extract only the relevant data. How to do this depends on the input format of the subtitles, as described below.

The `maxSkewForMediaComponentMs` config property has special meaning for subtitle tracks. See section Track skew for details.

A track bitrate needs to be set for subtitle tracks as well as for video and audio tracks. See section Track bitrate.

9.4.6.1 EBU Teletext

For EBU Teletext subtitle tracks, the ingest process needs to know the two teletext-specific attributes *magazine* and *page* in order to find the correct subtitle. Assuming that the track language is configured to match the desired subtitle, these numbers can often be looked up automatically in the PMT of the source. In these cases, no additional configuration is needed.

However, if multiple subtitles in the source use the same language, or there is no teletext information in the PMT, the ingester needs more information to find the correct subtitle. This can be provided by configuring a `teletextFilter` for the track in the channel template. The specifics of the source determines how much of the filter that needs to be filled in. Setting the properties `magazine` and `page` is always enough to identify the correct subtitle, but it may also be enough to specify `type` in order to distinguish between subtitles that use the same language. For magazine 8, a configured magazine number of either 0 or 8 is supported. The internal magazineNr is made to follow the configured value instead of always being 8. Parts that are left at the default values will be read from the PMT.

9.4.6.2 Cavena STU P31

For Cavena STU P31 ("Cavena") subtitle tracks, nothing more than the track language needs to be configured. (Note that because Cavena uses fixed magazine and page numbers, a single PID can only carry one Cavena subtitle per language.)

9.4.6.3 CEA-608 Closed Captions

Embedded CEA-608 captions can be *extracted* into separate subtitle tracks and/or *signalled* downstream to players.

9.4.6.3.1 Extraction of embedded CEA-608 Closed Captions

The live ingest service can extract embedded CEA-608 closed captions from a video track, and forward them as separate subtitle tracks with a captions "role". This can be achieved by configuring a separate subtitle track with the same PID as **one of** your video tracks containing captions. The caption channel to extract (`CC1, CC2, CC3, CC4`) is specified by the track parameter `cea608Channel`. To specify a language code for an *extracted* caption channel the track parameter `language` is used (unlike the case below for the signalling of *embedded* closed captions).

9.4.6.3.2 Signalling of embedded CEA-608 Closed Captions (with pass-through)

Embedded CEA-608 closed captions are always passed through the whole ingest and repackaging chain. However, the presence of CEA-608 may be hard to detect due to intermittency, and the language may be unknown. To signal the presence and languages of embedded CEA-608, the channelTemplate can be configured in the following manner:

- `CEA608.mode`: can be either `"enabled"`, `"unknown"`or `"none"`. Where `"enabled"` will enable the signalling of languages listed below. `"unknown"` prevents signalling. `"none"` explicitly signals the absence of closed captions. Some manifest standards (e.g. HLS) can signal this. Default: `"none"`

CEA-608 can carry up to four separate channels: CC1 to CC4. A language can be set for each of them. At least channel one must be set when `"mode"` is `"enabled"`.

- `CEA608.CC1Language`: the three letter language code for captions channel CC1. The language of three more channels may be configured, up to `CC4Language`.

Example:

```
$ confcli services.liveIngest.channelTemplates.ntsc.CEA608.
{
    "CEA608": {
        "CC1Language": "swe",
        "CC2Language": "",
        "CC3Language": "eng",
        "CC4Language": "",
        "mode": "enabled"
    }
}
```

Note: ESB3003 does not make any verification that the captions are present in the ingested stream. These configuration parameters simply allow downstream manfest generators to include the information.

9.4.6.4 OCR to convert DVB subtitles to text

DVB subtitles can be converted to normal subtitle tracks using OCR.

9.4.6.4.1 Language files

If you configure OCR for a language then you need to ensure that corresponding data files are available.

You can install a Tesseract **fast** data file for a language by package manager, i.e.:

```
dnf install tesseract-langpack-<language>
```

However to get better results, you can configure to use other <language>.traineddata for primary and secondary data files, which are not included in default repositories.

Files for manual installation are available here:

- Legacy https://github.com/tesseract-ocr/tessdata
- Best https://github.com/tesseract-ocr/tessdata_best
- Fast https://github.com/tesseract-ocr/tessdata_fast

Use 2 of these, usually legacy + one more.

9.4.6.4.2 OCR Engine Configuration

A subtitle track will use the `language` and `additionalLanguages` that is configured in channelTemplates.

In `services.liveIngest.ocr` you configure the OCR engine configuration by creating an item that has the same `language` and `additionalLanguages` as the track. If two tracks have the same `language` and `additionalLanguages` values then they will use the same OCR engine configuration.

The OCR engine options are:

- `firstEngineTessdataPath`: path to folder containing traineddata files for first OCR engine. Empty by default, meaning the system default, e.g. path installed by `dnf/yum`, is used instead.
- `firstEngineConfidenceOk`: if the confidence percentage of the first engine is lower than this threshold, proceed to second engine, and use the result with the highest confidence, or the first engine if the results are equal.
- `secondEngineTessdataPath`: path to folder containing traineddata files for second OCR engine. Empty by default, meaning that the second engine is unused.
- `mergeBitmaps`: decides if the bitmaps in a PES payload are to be merged or not. Default value is `true`. **Note:** Typically `false` yields better OCR results. However, in some channels bitmaps are split too close to the text. Thus `true` is safer.

9.4.6.4.3 Specific OCR configuration for a track

For each subtitle track that will use OCR you need to configure: `services.liveIngest.channelTemplates.*.tracks.ocr`.

The `additionalLanguages` option determine additional language files that Tesseract will use. It is also used to determine which "shared OCR configuration" to use, see above.

The options `italicsShear...` are used to configure handling of italics text:

- `italicsShearConfidenceOk`: if the confidence percentage of the second engine is lower than this threshold, proceed to shearing the image, and use the result with the highest confidence.
- `italicsShearX`: ratio for horizontally "un-shearing" (possibly) italic text. A typical value of 0.15 would mean that a bitmap 100 pixels high would be shifted 15 pixels to the left at the top and 0 pixels at the bottom.

9.4.6.4.4 String replacements

For each language, string replacements can be configured. These are configured in file `/etc/edgeware/ew-live-ingest/ocr/<language>-replace.config`.

The config file must be utf-8 encoded. Each line has format "from§to".

9.4.6.4.5 Debug output

Files can be saved for debugging purposes in the folder:

```
/run/edgeware/ew-live-ingest/ocr
```

If you want to get these files, create this folder and change permissions so user edgeware can save files in that folder.

9.4.7 SCTE-35 ad markers

In order to allow ad insertion to be performed downstream, the system can be configured to honor and pass on SCTE-35 cue messages that are present in the input stream. In this mode, the boundaries of the created CMAF segments will be adjusted to align with the announced SCTE-35 splice points, and the original SCTE-35 messages will be propagated downstream.

Ad marker mode is enabled by setting the property `useAsScte35Source` to true for an `inputSource` of a channel. Note that only one source can be marked as a SCTE-35 source.

The system will automatically determine which PIDs carry SCTE-35 data by examining the PMT of the source. Multiple SCTE-35 PIDs are supported as long as they are listed in the PMT.

Refer to Input stream requirements for further details on the SCTE-35 support in Edgeware Live Ingest.

9.4.7.1 Late ad marker timestamp match.

Ad markers should arrive at least half a segment before the cue out / cue in. Otherwise it may sometimes not be possible to split the segments as wanted.

If the ad doesn't start when wanted and you see that the log says `Late ad marker timestamp match`, the ad marker came too late.

In the log there are `INFO` messages that say when segments are announced and when SCTE-35 messages arrive. If a segment should be extended for an ad then the SCTE-35 message must arrive before that segment is announced.

The parameter `segmentationMarginForLateAdmarkerMs` can be used to delay segmentation.

NOTE: Setting `segmentationMarginForLateAdmarkerMs` has to be less than average GoP duration. Otherwise, the segmentation is unstable.

9.4.8 Input source problem handling

If the input source has problems such as packet loss, the resulting output will be impacted. In this situation there is a tradeoff between only creating those segments that are not impacted, or to use whatever data is available on a best effort level and get segments with lower quality content. In either case the repair feature can heal the issue if another node has better output, see Repair.

Inside `inputProblemHandling` in the segmentation template, to control what action to take:

- At limited packet loss (less than one segment worth of data lost), set the `limitedPacketLossAction` attribute. To make sure that all packet loss is detected and handled appropriately it is recommended to use RTP.
- At too late or too early subtitle sample, set the threshold of abnormality to `subtitleSampleMaxDeviation` attribute.
- Toggle DTS step fluctuations auto-correct function by `handleVideoDtsFluctuations`. Default value is `true`. But this function may cause the audio-video desync problem in redundancy mode. See Audio video desync for more details.

9.4.8.1 Silent replacement segments when audio segments are missing

If too much input is lost, so that it isn't possible to generate audio segments at all, there is a configuration option to enable generation of silent replacement segments.

To control what action to take at full segment loss, set the `missingSegmentAction.audio` attribute in the segmentation template to `silence` or `restart` (default).

In order to generate matching replacement silence for a particular track, real audio data must have been received before the outage for that track for the current config ID. As for other tracks, the PID for the audio track must stay constant and be present in the transport stream PMT all the time.

The channel will be in `RunningDegraded` state (see Live Channel Status) while silent segment generation is active on one or more audio tracks.

NOTE: Frames loss below `alignAudioFramesThresholdMs` (see Audio track drift) generate partially silent segments, instead of triggering this function.

NOTE: When `missingSegmentAction.audio` is set to silence and SCTE-35 ad markers are used, the skew limit of audio track is shortened (see Track skew)

9.4.9 Logging

Each ingested channel has a dedicated log file, which is available under `/var/log/edgeware/ew-live-ingest/↪ live-ingest-channels/worker-<channel_name>.log`.

The log contains output from several different *loggers* within the worker. The verbosity of the logs can be controlled by setting the desired log levels in the live ingest configuration. This can also be done while a channel is running by using the `confcli` command-line tool. The default log level is INFO.

Log level for three different loggers can be set both globally and on a per- channel basis: `general`, `source`, and `track`. Configuration on channel level will override the global level.

Source-specific logging covers the input stages of the worker, e.g. ATS and AVC/AAC parsing. A grep-friendly logger name will be used in the log file, to make it easy extract information about a single network source, for example.

Track-specific logging covers events that only concern one specific track, such as information about segmenting. In addition to global and channel level, it can also be set on track level by using the `logLevel` property.

9.4.9.1 Log levels

The following describes the log levels used in the worker. When a particular log level is configured, that means that all levels above it in this list are enabled as well.

FATAL Unexpected errors that lead to the exit (and subsequent restart) of the worker.

ERROR Errors that may affect the output of the worker, but will not kill it. Should normally not appear in production.

WARNING Warnings should not affect the operation of the worker, but may indicate for example non-critical imperfections in an input stream or dubious configuration.

INFO General information about the configuration and life cycle of the channel. This is the recommended level to use in a running production system.

DEBUG More detailed information to help troubleshooting. May for example show information on a per-segment basis. Using DEBUG level in a production system for a limited amount of time is perfectly acceptable, but it should be enabled only for channels or channel tracks of particular interest rather than applied globally.

9.5 Automate channel configuration

The channel configuration can be automated using the auto-conf tool. For details refer to [12]

9.6 Advanced configuration

Advanced configuration settings under `liveIngest.advanced` should normally not be updated from the default settings. The advanced parameters may be used for customized tuning and troubleshooting by Edgeware representatives.

9.7 Convoy system events

Edgeware Live Ingest may be integrated with Convoy to monitor service events. The Convoy server to receive events needs to be configured in the integration section of the configuration. This may be done by issuing the following command as super user:

```
confcli integration -w
```

For additional information, see the Convoy user guide [5].

9.8 Log rotation

Log rotation is handled by `logrotate`. The log rotation policy can be customized by changing the configuration in `/etc/logrotate.d/ew-live-ingest`.

9.9 Master

This section describes configuration of the liveingest-master, which is done through the `confd` service, meaning that the configuration can be changed using for instance `confcli`. The configuration of the master only affects the capturing network data for selected channels which are dumped in `/run/edgeware/ew-live-ingest/captures`.

9.9.1 Capturing

A channel may be configured so that a network traffic capturing process will be executed to record the incoming data by dumping the data to pcap files (with multiple rotation files and a size cap on each rotation file). If a liveingest-worker encounters errors in the input source, the content of the captured data is stored as a new file in a separate directory (/run/edgeware/captures/saves) with a timestamp. The default configuration for a channel is that this feature is disabled. Since this feature utilizes the CPU, memory, and disk IO, it is only recommended to enable this feature if you are running into problems with a specific channel and it would make sense to verify the content of the input of that channel to liveingest.

The configuration of the capturing is done in `services.liveIngest.capture`. Currently we only support one mode of capturing, "continuous", which is described in the paragraph above. Continuous mode is configured in `services.liveIngest.capture.continuous` The table below provides a description of the parameters which are configurable for continuous mode.

Parameter	Default Value	Description
channelList	[]	The list of channels to do continuous capturing for
numberOfRotateFiles	2	Number of rotation files when doing continuous capturing
sizeOfRotateFilesMB	8	Size of each rotation file

The following parameters lie under `services.liveIngest.capture` and should apply for all modes

Parameter	Default Value	Description
maxNumberOfStoredCaptures	20	Maximum number of captured dump files stored

10 Monitoring

10.1 Convoy

Edgeware Live Ingest may be integrated with Convoy to monitor service events. See Convoy Configuration for details.

10.2 The ew-live-ingest-tool

`ew-live-ingest-tool` is a tool for the ew-live-ingest service, and may be used to monitor the current state and operation of the service and its live channels.

```
# ew-live-ingest-tool --list   (left most columns)

Channel   State    Status   Sync  PID   Start Time              Restarts ...
--------------------------------------------------------------------------- ...
channel1  Enabled  Running  n/a   18841  2018-02-12 10:00:00 CET         ...
channel2  Enabled  Running  n/a   19171  2018-02-12 10:33:38 CET  2      ...

# ew-live-ingest-tool --list   (right most columns)

Channel   ... Start Time               Restarts (Not Acknowledged)
--------- ... ---------------------------------------------------
channel1  ... 2018-02-12 10:00:00 CET
channel2  ... 2018-02-12 10:33:38 CET  2        2
```

The command will list the following for all currently configured live channels:

- Channel name.

- Channel state, the current administrative state in the configuration.
- Current channel status (see Live channel states)
- Ingest redundancy synchronization (Sync) state (see Ingest redundancy monitoring)
- Channel worker process PID.
- Start time of the now running (or next to be scheduled) channel worker process
- The number of channel worker process restarts since the `ew-live-ingest` service was started.
- The number of channel worker process restarts since last acknowledgement by using the `ew-live-ingest-tool --acknowledge-restarts` command.

The `ew-live-ingest-tool` command may also be used to start a non-running channel or to restart an already running channel. The channel must be present in the configuration.

```
# ew-live-ingest-tool --start <channel>

# ew-live-ingest-tool --restart <channel>
```

The restart option should rarely be used as it will cause lost segments for a duration of a few segment durations.

The `--start` option can be used to start a non-running channel.

For low level details about specific channels, e.g statistics counters, the `--channels` option may be used:

```
# ew-live-ingest-tool --channels <channel1> [<channel2> ... ]
```

In case any virtual channels are configured, they can be listed together with their current schedule.

```
# ew-live-ingest-tool --list-virtual

Virtual channel  Default  Scheduled  Start time       End time           Live
-------------------------------------------------------------------------------
vc1              ch1                                                      *
                          ch2    2019-07-02T13:00:00Z 2019-07-02T14:00:00Z
vc2              ch2
                          ch1    2019-07-02T12:00:00Z 2019-07-02T13:00:00Z  *
                          ch1    2019-07-03T12:00:00Z 2019-07-03T13:00:00Z
```

The `Live` column indicates what content source that is active at the current live point, i.e. "now".

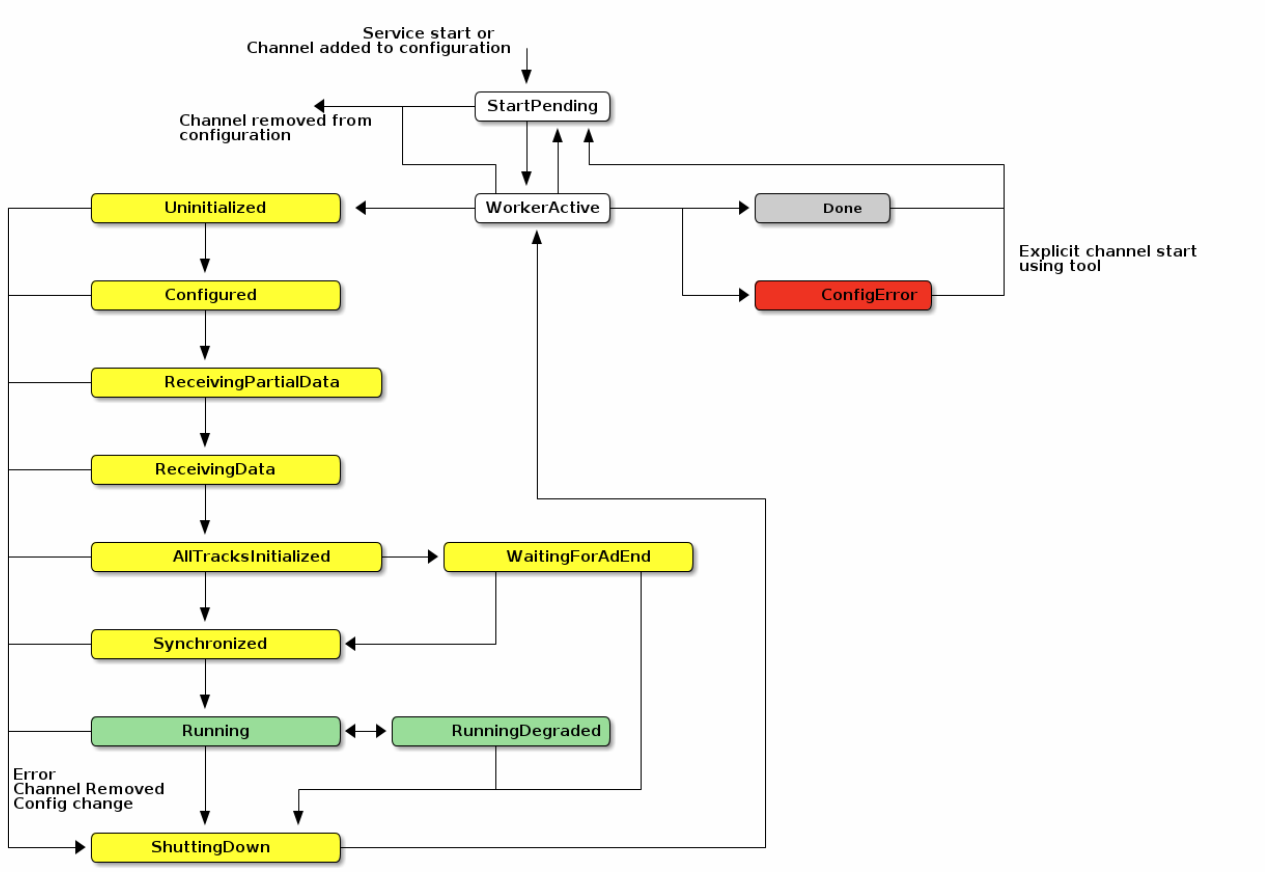
In case there are scheduled items that are using SCTE-35 markers from the default live channel for precision scheduling, items will also show if they are still "Tentative", i.e. still waiting for the configured SCTE-35 cues to arrive, and the actual scheduled times as detected from the SCTE-35 markers.

```
# ew-live-ingest-tool --list-virtual  (left most columns)

Virtual channel  Default  Scheduled  Start time       End time           Live
-------------------------------------------------------------------------------
vc3              ch3
                          ch1   2019-07-02T12:00:00Z  2019-07-02T13:00:00Z  *
                          ch1   2019-07-03T12:00:00Z  2019-07-03T13:00:00Z

# ew-live-ingest-tool --list-virtual  (right most columns)

... SCTE35-ID  Tentative  SCTE35-start          SCTE35-end
... ------------------------------------------------------
...
... 10001                 2019-07-03T12:00:10Z  2019-07-03T12:05:03Z
... 10002      *
```

10.2.1 Live channel status

Each configured live channel has an internal status, describing its current operational state.

State	Description
StartPending	The channel is present in the configuration and is enabled. Start of the channel is scheduled. Is normally only seen in case of automatic restarts of the channel.
WorkerActive	Brief transitional state where a channel worker process is running, but not available for communication.
Uninitialized	Channel worker process is running but has not yet received its configuration.
Configured	Channel has received its configuration and is listening for network input. Static configuration validation successful.
ReceivingPartialData	Data is being received on some, but not all configured input sources.
ReceivingData	Data is being received on all configured input sources.
AllTracksInitialized	All configured tracks have been found in the MPEG-TS stream(s) and the channel has been published for the `ew-cbm` service.
WaitingForAdEnd	Node syncronization is enabled, and the channel is waiting for ongoing and/or pending ads to finish.
Synchronized	A valid track synchronization point has been found and set. All tracks are now being requested to start segmenting.
Running	All tracks are now producing segments. This is the normal state for an operational channel.
RunningDegraded	Channel is running and all tracks are producing segments, but one or more tracks are producing forced segments, e.g. silent segments for audio tracks. Due to missing input on the configured input source.
ShuttingDown	The channel worker process has started to shut down. The reason could be due to some failure or due to an explicit restart or changed configuration.
ConfigError	The configuration for the channels has errors. The channel will not start until the configuration is changed, or the channel is explicitly started using `ew-live-ingest-tool --start <channel>`.
Done	The channel has terminated and will not restart until it's explicitly started using `ew-live-ingest-tool --start <channel>`.



10.2.2 Inspect network stream

`ew-live-ingest-tool` allows brief inspection of incoming UDP transport streams. Using the `--ts-stream-info` option, it is possible to get some information about what tracks and codecs are available in a transport stream.

10.3 The ew-cb-media tool

The `ew-cb-media` tool may be used to monitor the catch-up buffers of available channels. See (List catchup buffers) for details.

11 Catchup Management

The catchup buffers for live channels are managed by the catchup buffer manager (`ew-cbm`) and it is responsible for copying the segments produced by the `ew-live-ingest` to persistent storage, e.g. a local drive or an NFS mounted volume. The catchup manager will also remove segments once they fall outside the configured catchup window.

11.1 Accelerator

The accelerator (`ew-cbm-accelerator`) is an HTTP server that sits in front of the catchup manager and relieves it from doing file transfers.

The segment metadata cache can be configured through `storage.catchup.cache`:

- `enable`: toggling cache on/off. Default value is `True` (cache on).
- `capacity`: max size of cache in MB. Default value is `1024`.

NOTE: If the configured value of cache capacity exceeds `/dev/shm` ramdisk capacity, cache is turned off to not cause system malfunction, and a warning is logged into ew-cbm main process log.

11.2 Control

The catchup buffer manager is controlled using `systemctl` as follows:

To start the catchup buffer manager:

```
systemctl start ew-cbm
```

Note: When starting the catchup manager it will scan through already written catchup data and it might therefore take a while for the command to complete.

The accelerator is started automatically when starting `ew-cbm` but it can also be started independently as follows:

```
systemctl start ew-cbm-accelerator
```

To stop the catchup buffer manager and accelerator:

```
systemctl stop ew-cbm ew-cbm-accelerator
```

To check the status of catchup buffer manager and accelerator:

```
systemctl status ew-cbm ew-cbm-accelerator
```

11.3 Logging

The logs produced by the catchup manager are located in the directory `/var/log/edgeware/ew-cbm/`. See the table below for further details about each log file.

Log file	Description
access-accelerator.log	Access log for the `ew-cbm-accelerator`
error-accelerator.log	Error log for the `ew-cbm-accelerator`
ew-cbm.log	Main process log for `ew-cbm`
ew-cbm-channel-*channelName*.log	Main log for channel *channelName*
ew-cbm-writer-*channelName*.log	File writer log for channel *channelName*
ew-cbm-cleaner.log	Cleaner process log (Trimming of catchup buffer)

All log files are rotated and the log rotation policy is configured in `/etc/logrotate.d/ew-cbm`.

11.4 Configuration

Each channel can be configured to have its own catchup storage location but usually is convenient to share the same location for all channels. The location of the catchup buffer is configured in `storage.catchup.locations`. For example, to configure a new catchup storage location for a 24 hour catchup using the wizard mode in `confcli`:

```
confcli storage.catchup.locations. -w
Running wizard for resource 'Catchup Locations'
<A list of catchup locations.>

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

Catchup Locations <A list of catchup locations.>: [
  location <Catchup location configurations.>: {
    Catchup Duration (default: 0): 86400
    Catchup Base Path (default: ): /mnt/nas/cb
```

```
    Catchup Location Name (default: ): cb
  Add another 'location' element to array 'locations'? [y/N]: N
Generated config:
{
  "locations": [
    {
      "duration": 86400,
      "basePath": "/mnt/nas/cb",
      "name": "cb"
    }
  ]
}
Merge and apply the config? [y/n]: y
```

After the catchup location has been configured it can be referred to in the channel configuration using the `catchupLocation` keyword. In this example above the channels would refer to `cb` which is the name of the catchup location.

11.5 Storage format

The catchup buffer is stored in Edgeware's intermediate CMAF-based format where segments are concatenated into one minute files and placed in a directory according to the following format: *basePath/channelName/date/hour/config/track/minute*.

For example, the file `/mnt/nas/cb/channel-1/2018-01-08/10/cfg_0/video_2mbps/00.cmfv` would contain the data for the track *video_2mbps* in channel *channel-1* which was ingested at 2018-01-08 10:00.

11.6 List catchup buffers

The `ew-cb-media` command can be used to list all available channels and their catchup buffers. For example:

```
ew-cb-media
Channel    State      Start time          End time            Duration
-------------------------------------------------------------------------------
    ↪
channel-1  *enabled   Jan-08 12:00:00     Jan-09 12:09:32     1d 0h 9m 32s
channel-2  !enabled   Jan-08 12:00:00     Jan-09 12:09:32     1d 0h 9m 32s
channel-3  !disabled  Jan-08 12:00:00     Jan-09 12:09:32     0s
channel-4  -enabled   Jan-08 12:00:00     Jan-09 12:09:32     1d 0h 9m 32s
```

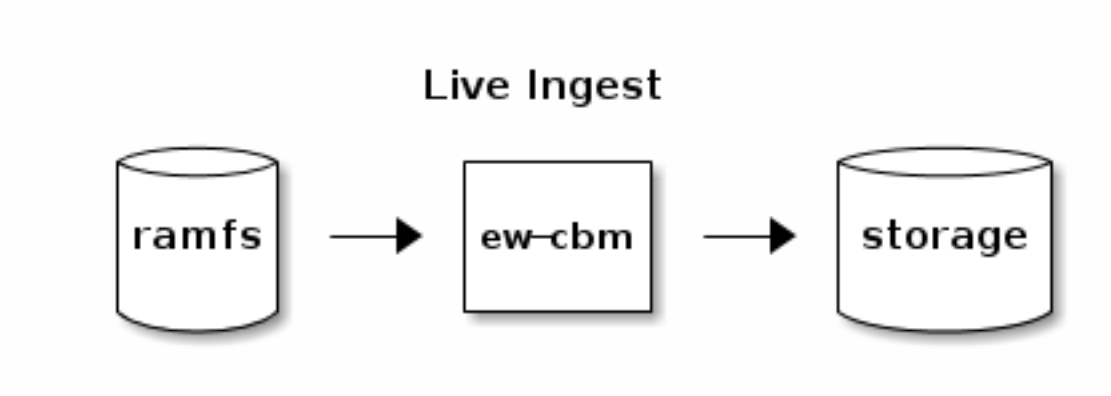
Note: `ew-cb-media` retrieves the information from the catchup buffer manager so the `ew-cbm` process must be running.

11.7 Supported storage setups

This sections describe the supported storage setups and how to configure them.

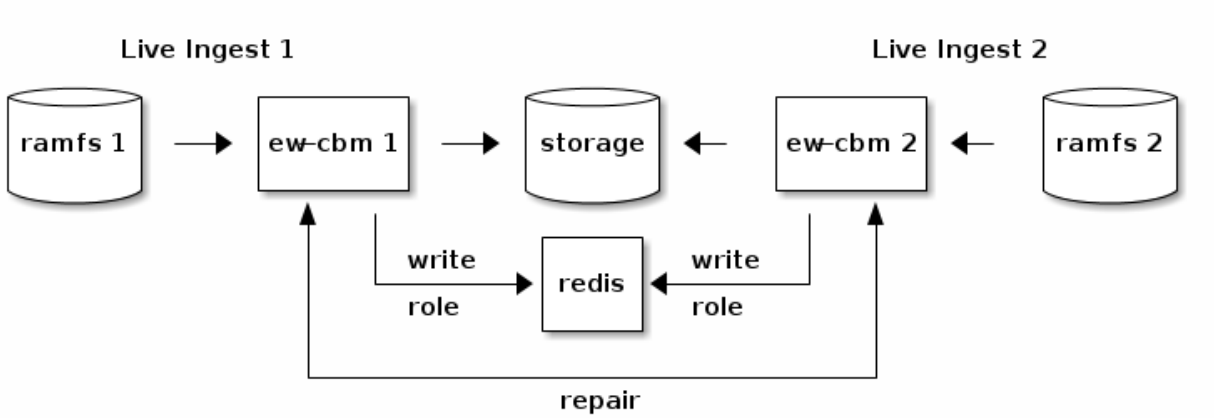
11.7.1 Single Live Ingest with single catchup buffer on single storage

This is the most basic setup where a single Live Ingest node is using a single storage (local or NAS).



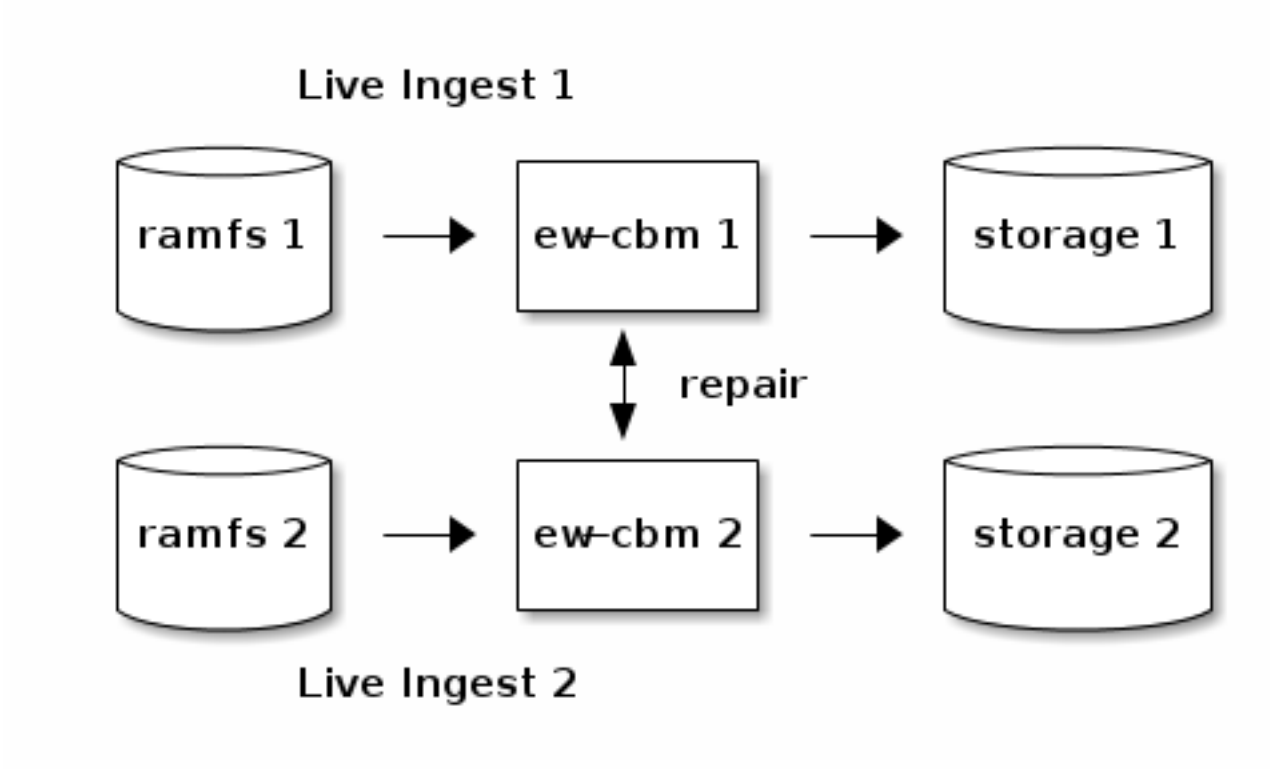
11.7.2 Multiple Live Ingest nodes with shared storage

This is a configuration where shared storage (NAS) is accessible by other nodes.



11.7.3 Multiple Live Ingest nodes with local storages

This is a configuration where each node has its own storage.



11.8 Partial Blackout of Catchup Buffer

It is possible to mark certain time intervals in a channel's catchup buffer as unavailable for delivery, for example in response to legal or regulatory requirements. This is referred to as *blacking out* part of the catchup buffer.

CBM will deny attempts by the SW Repackager to fetch media segments from a blacked-out portion of the catchup buffer. Manifests are not affected by the fact that a blacked-out interval exists. Note that in the current version of SW Origin, nothing is done to replace or skip the blacked-out content, so clients will typically simply stop when they reach the beginning of a blacked-out interval since the segment requests will fail.

The blackout operation itself covers the case where catchup requests are being made against the live channel. However, cases where content from the blacked-out interval has already been extracted from the catchup buffer must also be taken into account. For example, it may already have been stored in a downstream cache or included in a recording. The full blackout procedure detailed below covers such cases to ensure, to the extent possible, that blacked-out data doesn't reach clients.

11.8.1 Performing Blackout

Blackout operations are performed by issuing commands towards CBM's HTTP API. Note that there is no synchronization of blackout information between nodes, so in a redundant setup all blackout operations must be performed on all participating CBM nodes. This is true regardless of whether shared storage is used or not.

To black out part of the catchup buffer for a channel, issue a command like the following:

```
$ curl -X POST "http://<cbm host>:8090/<channel name>/blackout?startTime
    ↪ =<start>&stopTime=<stop>"
```

The `startTime` and `stopTime` parameters must be specified using UTC time, either as an integer number of seconds since epoch (Jan 1 1970), or using the compact version of the ISO8601 format (for example `20190331T134500Z`).

If the request is successful, the response status code will be "200 OK", and the response body will contain JSON data with information about the request and the catchup buffer intervals that were actually affected by the operation.

If the response is in the 4xx range, there is something wrong with the request, and the response body will explain what it is.

If the response is "500 Internal Server Error", an unexpected error has occurred. More information will be available in CBM's channel log. Note that in this case it is possible that the operation has been partially completed, which could cause inconsistencies with regard to the blackout status of the affected parts of the catchup buffer. It is recommended to contact Edgeware Technical Support for assistance in case this happens.

Notes:

- The interval that is actually blacked out will be aligned with segment boundaries. Therefore it may end up being slightly larger than the requested interval. If so, that is reflected in the JSON response.
- It is not possible to black out an interval that overlaps Live Ingest's live buffer (the length of which is determined by the configuration property `circularBufferS` in the segmentation template for the channel). Either wait a while or adjust the `stopTime` parameter as needed to avoid this.
- If the catchup buffer contains holes in the requested interval, the blackout operation will be performed on the parts of the interval that do exist in the buffer.
- The blackout state is persisted in the catchup buffer, and will remain even if CBM is restarted.
- It is not possible to perform a recording that overlaps a blacked-out interval (it is possible to *schedule* such a recording, but it will fail).

11.8.1.1 Clearing Blackout

It is possible to undo an incorrect or no longer valid blackout operation by clearing the blackout status for part of the catchup buffer. The specified interval does not have to match the blacked-out interval exactly, so it is also possible to change the blackout boundaries using this method.

The command is the same as for performing blackout, but with the extra parameter `clear=true` added:

```
$ curl -X POST "http://<cbm host>:8090/<channel name>/blackout?startTime
    ↪ =<start>&stopTime=<stop>&clear=true"
```

11.8.1.2 Listing Blacked-Out Intervals

To list all currently blacked-out parts of the catchup buffer, issue the blackout command using the `GET` method instead of `POST`:

```
$ curl -X GET "http://<cbm host>:8090/<channel name>/blackout"
```

The resulting JSON data will contain information about all blacked-out intervals.

If `startTime` and/or `stopTime` are provided, only the corresponding part of the buffer will be checked. It is however recommended to omit them when listing intervals to ensure nothing is missed.

11.8.2 Full Step By Step Procedure

This section describes the steps needed to black out part of the catchup buffer and ensure that the blackout is enforced as far as possible by deleting copies of the blacked-out interval that may have been made before the blackout operation was performed.

1. **Determine the interval to black out.** The assumption is that an incoming blackout request will specify a time interval to black out from a certain channel. Due to differences in time zones and the inherent latencies of the ingest chain, the incoming time interval may need to be adjusted before it can be applied.

   One method of determining what `startTime` and `stopTime` to use in the blackout request to CBM is to issue catchup requests for the channel in question via the repackager and watching the resulting video. Simply adjust the start and stop times until only the desired content is covered. Since the repackager and CBM accept the same formats and names for these parameters, they can then be used directly in the blackout request.

   Note that if the same client is later used to verify that the blackout operation was successful, it may be necessary to clear its local cache to avoid being served cached copies of the blacked-out segments (this is especially likely if the client runs inside a web browser).

2. **Perform the blackout.** Issue the blackout request as described in the previous section, and verify that it completes successfully. Note that identical requests must be sent to all CBM nodes in a redundant setup for the blackout to be complete.
3. **Purge downstream caches.** The blacked-out content may already have been cached downstream, and for the blackout to be effective such caches must be invalidated.

   On live ingest nodes, the accelerator cache can be purged by: $ rm -rf /dev/shm/cache/* Similar can be applied repackager node, but on path `/dev/shm/cache-repack/*`.

   If a Convoy-managed Edgeware CDN is used to distribute content from SW Origin, issue the following command on a Convoy management node to purge the channel from the streamers' caches:

   ```
   $ ew-cache -k <account API key> content-purge <channel name>
   ```

   Note that purging the caches will result in temporarily increased load on SW Origin as the caches are refilled.

   If the channel is distributed by other means, such as a third-party CDN, the channel needs to be purged manually using whatever methods and APIs are offered by the vendor.
4. **Delete existing recordings.** If there are any existing recordings that overlap the blacked-out interval, they must be deleted. Use the `ew-content list` command or the CCMI API to determine the names of the recordings from the channel in question that overlap the blacked-out interval. Then use commands like the following to delete each of them:

   ```
   $ ew-content -k <account API key> delete <recording content name>
   ```

   Note that it is not possible to reschedule the recording using its original start and stop times, since recording across a blacked-out interval is not possible. Part of the recording may be possible to recreate by adjusting the start and stop times, however (assuming the data is still available in the catchup buffer).

   If the Private Copy feature is used (see "EDGS-182 - Convoy Private Copy Application Note"), any private copies based on affected recordings must be deleted separately. They will be included in the output from `ew-content list`/CCMI, so just make sure to delete the private copy content as well as the original recording.

11.8.3 Logging

CBM will respond with "451 Unavailable For Legal Reasons" when a blacked-out media segment is requested. This will be shown on WARNING level in CBM's channel log ("Media segment is blacked out, will respond with 451: ..."), and is also visible in `access-accelerator.log`.

CBM logs blackout operations on INFO level in the channel log for traceability ("Blackout request performed: ...").

12 Redundancy

Making the setup redundant requires a number of steps to be performed:

- The network must have no single point of failure. This can be achieved with bonded network interfaces and redundant switches.
- The downstream requests towards the Live Ingest nodes must fail over to another Live Ingest node should the first one fail. This requires adequate configuration in the Repackager nodes.
- Since video segments consumed by a single client may have been produced by two different live ingest nodes it is required that the segmentation is synchronized between the nodes. Otherwise a glitch may be experienced during failover.
- The storage that holds the catchup buffers must have a redundancy solution. When the storage has a built-in redundancy solution and exposes a single integration point, the Live Ingest nodes must synchronize the writing to the storage to guarantee the integrity of the catchup buffer.
- Live ingest nodes should be configured in cluster mode to enable the repair of the catchup buffers.

The first two points require no change on the Live Ingest nodes. The last three points require configuration changes and are explained in detail in the sections below.

Both ingest and storage redundancy rely on a Redis server running on a separate node. The Redis servers that run on Convoy management nodes can be used for this purpose. That requires the Redis listening port in the Convoy firewall to be opened for access from Live Ingest nodes. By default the port number is TCP port 6379.

The Redis node must be up and available for channels to start. When the channels are up and running, the Redis node can be restarted, e.g. during an upgrade of Convoy.

Repair functionality relies on Catchup Buffer Manager accessibility between the nodes. That requires TCP port 8090 to be open for traffic between all Live Ingest nodes.

Note: It is crucial that a redundant channel always has an identical configuration on both nodes, see Changing configuration below for details.

12.1 Prerequisites

The machine-id on a redundant node must be unique. Use command

```
[root@esb3003 ~]#cat /etc/machine-id
```

to view the machine-id. If it is not a unique value compared to the other node, delete the file and create a new machine id using command and reboot the node.

```
[root@esb3003 ~]#systemd-machine-id-setup
```

12.2 Ingest redundancy

Ingest redundancy is accomplished by letting two Live Ingest nodes ingest the same channel and enabling synchronized segmentation. By the nature of the segmentation algorithm, the segments will likely be identical without synchronization but it cannot be guaranteed. To enable the synchronization of segments, first specify a Redis server to use for the coordination:

```
confcli services.liveIngest.synchronizedSegmentation.coordinator redis://<ip
    ↪ address>[:port]/[dbindex]
```

then turn synchronization on by running:

```
confcli services.liveIngest.synchronizedSegmentation.mode enabled
```

If the redis password authentication is used, insert it:

```
confcli services.liveIngest.synchronizedSegmentation.password <password>
```

Alternative modes are `disabled` and `passive`. In passive mode the segmentation on that node follows other nodes but will never decide on segmentation boundaries for others. In disabled mode the node operates independently of any other node.

NOTE:

- When ad marker mode is enabled, a node that is joining an already running node will not start segmenting until any ongoing and already announced upcoming ads are finished. This is to ensure that the new node has access to all relevent SCTE-35 information before it starts segmenting. As soon as conditions allow, the channel will automatically start segmenting and enter the "Running" state.
- Do not specify `dbindex` when using Redis Cluster. See redis doc.

12.2.1 Monitoring with `ew-live-ingest-tool`

The synchronization status can be viewed in the "Sync" column by running:

```
ew-live-ingest-tool -l
Channel    State    Status   Sync     PID   Start Time          Restarts
-------------------------------------------------------------------------
channel-1  Enabled  Running  Yes (2x) 1438  2018-05-15 16:41:43 CEST  12
channel-3  Enabled  Running  No (1x)  6573  2018-05-15 13:13:13 CEST
...
```

The possible values in the Sync column are

Value	Meaning
`Yes (nx)`	Fully redundant ingest, *n* synchronizing nodes
`No (nx)`	Ingest not fully redundant, *n* synchronizing nodes
`???`	Failed to get information, likely Redis is unreachable
`n/a`	Synchronization is disabled

12.3 Storage redundancy

To support redundancy for the catchup buffers when two Live Ingest nodes are connected to the same NAS the nodes must ensure that exactly one node writes at all times. To enable this functionality, specify a coordinator:

```
confcli storage.catchup.sharedStorage.coordinator redis://<ip address>[:port]
```

and enable the coordination of the writing:

```
confcli storage.catchup.sharedStorage.enable true
```

If the redis password authentication is used, insert it:

```
confcli storage.catchup.sharedStorage.password <password>
```

If there is no connectivity to the coordinator while restarting or adding a new channel, the channel's data will not be written to the catchup location and will be served as live only. This includes restart of the `ew-cbm` service.

Note: If two nodes are configured with the same catchup location and shared storage is not enabled the behavior is undefined.

12.3.1 Monitoring with `ew-cb-media`

The command line option `--storage` will make `ew-cb-media` display the shared storage configuration and test the coordinator connectivity:

```
[user@host ~]# ew-cb-media --storage

Shared storage:
    coordinator: redis://172.16.1.2
    enabled    : True
    ping       : OK
```

In a redundant setup where two Live Ingest nodes write to a shared storage, the decision to write to the storage is made individually for each channel. For example if a node does not have a live input signal for a channel, it will lose the right to write that channel to the storage and the other node will take over.

When inspecting a channel, `ew-cb-media` will indicate the status for writing to the storage with one of the following additional symbols prepended to the channel status:

Symbol	Meaning
*	Node writes to the NAS (leader)
~	Node is the leader, but does not have any segments to write
-	Node does not write to the NAS (follower)
!	Channel does not have any live data
' ' (space)	The role for this node is undecided

12.3.2 Changing configuration

To update the configuration in a redundant setup, the recommended procedure is to disable the affected channels on all nodes while the configuration changes are being made, and then enabling them on one node at a time. This means that there will be some downtime in the service, and configuration changes should therefore be planned accordingly.

In more detail (using a single channel as an example):

1. Disable the channel on all nodes:

```
# confcli services.liveIngest.channels.<my_channel>.state disabled
```

1. Update the configuration on the first node:

```
# confcli ...
# confcli services.liveIngest.channels.<my_channel>.configId <new config id>
```

1. Enable the channel on the first node:

```
# confcli services.liveIngest.channels.<my_channel>.state enabled
```

1. Verify that the Live Ingest and CBM services operate as expected. See sections Monitoring with `ew-live-ingest-tool` and Monitoring with `ew-cb-media` for details.
2. Repeat steps 2-4 for the other nodes in the redundant setup, and verify that the service is running as expected after enabling the channel on each new node.

Note that minor configuration changes that don't require a new configId can be made independently on each node without restarting anything. A typical example would be changing log levels.

12.4 Repair

Repair functionality gives Catchup Buffer Manager the ability to repair missing segment information from other nodes in the redundant setup.

To enable catchup buffer repair functionality each node should have other nodes listed in configuration under:

```
# confcli storage.catchup.clusterMembers -w
```

For example, if we have installation with three nodes with IPs `10.10.10.11`, `10.10.10.12` and `10.10.10.13`, then the value for `storage.catchup.clusterMembers` on all nodes is `[ "10.10.10.11:8090", "10.10.10.12:8090", "10.10.10.13:8090" ]`. It is not mandatory to have the node's own address in the list, but it helps with keeping the configuration same on all nodes.

Repair functionality works differently depending on whether the storage is configured as shared or not. That is controlled with `storage.catchup.sharedStorage.enable`.

Configuring cluster members enables repair functionality in the following areas:

- **Channel start repair**
  - When the storages are configured as local, the channel bootstrap process will first read from local storage and then try to repair any missing data from other nodes in the cluster.

  - When the storage is configured as shared, channel bootstrap process will first ask other nodes for the state of the channel. Then, the bootstrap will read segment metadata from storage just for hours that are not completely known. This is mainly an optimization for a faster channel startup time.
- **Live buffer repair** When segments in live buffer are not in a complete state, repair asks for a list of segments from other nodes, and presents merged information to the clients. Live buffer repair will make CBM fetch missing segment data from other nodes when the segments are stored in storage. A complete state is defined as a continuous list of segments in a single configuration, without worker restarts, and without segments with errors.
- **Storage repair** While the live signal is lost for a channel, the repair process will start a loop that repairs the catchup buffer from other nodes.
  - In the case of shared storage, repair will update segment metadata from other nodes.
  - In the case of local storages, repair will update segment metadata and also download and store the segment data from one of other nodes. Repair loop will stop on first successful repair after the live signal is back. A successful repair is a repair that was successful on at least one of nodes in the cluster.

Repaired segments that need segment data from other nodes are queued in the channel's writer process. The writer process will process those items in the idle periods between storing of live segments. The size of the store queue for a channel can be monitored with `ew-cb-media` with verbose flag (`-v`).

Segments that failed to be repaired are not retried. In the case of such errors, manual repair command can be issued to fix the state of the channel. Manual repair is done with `ew-cb-media --repair <channel_name>` command.

Segments that are queued for storing will be reported as available in the `ew-cb-media` tool, and in API calls from repackagers. Though, if the segment data is requested, CBM will respond with `404 Not found`. Downstream nodes should be configured in a way to ask other CBM nodes for such segments, as the segment data is available on other nodes.

12.5 Upgrade

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.

To upgrade Live Ingest and Catchup Buffer Manager on a redundant setup, make sure to switchover traffic to the redundant node by stopping Live Ingest and Catchup Buffer Manager services before upgrade.

```
[root@esb3003 ~]# systemctl stop ew-live-ingest ew-cbm-accelerator ew-vc-
   ↪ scheduler ew-radicale ew-cbm confd
```

Remove the previous version of Live Ingest and Catchup Buffer Manager

```
[root@esb3003 ~]# dnf autoremove esb3003
```

Install the new version of Live Ingest and Catchup Buffer Manager

```
[root@esb3003 ~]# chmod +x install-esb3003-x.y.z
[root@esb3003 ~]# ./install-esb3003-x.y.z
```

Start confd service:

```
[root@esb3003 ~]# systemctl start confd
```

Temporarily set synchronization mode to `Passive` to ensure that no synchronization problems can stop the other nodes:

```
[root@esb3003 ~]# confcli services.liveIngest.synchronizedSegmentation.mode
   ↪ passive
```

Start Live Ingest service:

```
[root@esb3003 ~]# systemctl start ew-live-ingest
```

Verify Live Ingest service:

Verify Live Ingest status and make sure the channel Status is "Running" and Sync is "Yes" for the configured channels as follows:

```
[root@esb3003 ~]# systemctl status ew-live-ingest
[root@esb3003 ~]# ew-live-ingest-tool --list

Channel       State    Status   Sync     PID  Start Time
   ↪ Restarts
-------------------------------------------------------------------------------
   ↪
test-channel  Enabled  Running  Yes (1x) 245  2018-06-12 10:30:33 CEST
```

Start Catchup Buffer Manager service:

```
[root@esb3003 ~]# systemctl start ew-cbm ew-cbm-accelerator
```

Verify Catchup Buffer Manager service:

Verify CBM status and make sure the role of the channels is follower (-) as follows:

```
[root@esb3003 ~]# systemctl status ew-cbm ew-cbm-accelerator
[root@esb3003 ~]# ew-cb-media

Channel         State     Start time        End time          Duration
-------------------------------------------------------------------------
test-channel   -enabled  Jun-12 10:00:00   Jun-12 10:05:24    05m 24s
```

Set synchronization mode to `Enabled`:

```
[root@esb3003 ~]# confcli services.liveIngest.synchronizedSegmentation.mode
   ↪ enabled
```

To automatically start the services on system boot:

```
[root@esb3003 ~]# systemctl enable ew-live-ingest ew-cbm ew-cbm-accelerator
```

If virtual channels is used:

```
[root@esb3003 ~]# systemctl start ew-vc-scheduler
[root@esb3003 ~]# systemctl enable ew-vc-scheduler
```

Repeat the Upgrade procedure on the other node.

13 Troubleshooting

This chapter describes how to identify and resolve problems related to the Live Ingest software. Depending on the nature of the problem, you can use confd tools or logs to identify and solve problems.

13.1 Overview of Best Practices

We recommend the following general best practices:

- See the ESB3003 release notes and ESB3002 release notes for the latest features, guidelines and limitations.
- Understand the data flow as described in ESB3003 Live Ingest Overview
- Choose a channel to troubleshoot.

13.2 Troubleshooting Basics

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/* do often contain valuable information about problems related to the *HTTP service*. It is recommended to check these logs when investigating a problem.

Besides just observing the network traffic, tools such as *curl* and *wget* can be very helpful to see how the video server responds to different requests. For example in order to determine at what stage a complex process fails.

If all the above components are working as expected, the downstream machines ie:Repackager, CDN or client would need troubleshooting.

13.2.1 Input validation

The tool `ew-tsanalyzer` can be used to analyze the input when it comes to timestamps, drift and the ebp/rai marker distances used for segmentation. The tool can be run on a TS file, a PCAP capture, or live by running on a machine that can receive a multicast or unicast UDP or UDP/RTP channel.

13.3 Validation and Troubleshooting

This section provides information on how to validate and troubleshoot the ESB3003 software package installation

13.3.1 Verifying the Software Version

To verify the software version, perform the following:

```
root@esb3003:~#  dnf list esb3003
```

13.3.2 Troubleshooting installation

If the software version is not as expected, see the sections in upgrade for details to update the version.

13.3.3 Validating and Troubleshooting Configuration and Bringup

Choose a channel to troubleshoot and run the command below to identify problems and perform recommended steps mentioned in the tool to resolve issues.

```
[root@esb3003:~]# ew-diagnose-tool -c <channel-name>
```

13.3.4 Input source packet loss issues

In order to ensure correct segmentation and alignment between tracks, Live Ingest require that input stream(s) are received without packet loss. So it is essential to ensure good network conditions for a perfect service.

One mechanism that Live Ingest use in order to detect packet loss, is by monitoring the received transport stream continuity counter for each TS packet. In case of any lost TS packets, Live Ingest will by default restart the channel to re-synchronize. The reported restart cause will then be "TS continuity error".

Live Ingest may be configured to attempt to proceed in case of limited packet loss without restarting. This is done by updating the segmentation template `inputProblemHandling.limitedPacketLossAction` parameter. The resulting behavior is then very much dependent on the nature of the packet loss. But may be able to provide an uninterrupted service in case of limited packet loss. See input problem handling for more details.

13.3.5 Subtitle samples come too early or too late

Subtitle source might produce late or early samples compare to previous sample. Live Ingest is able to adjust their PTS to make the output timeline looks normal.

However, if the difference is too big, Live Ingest buffer can be filled up without creating segment from later data, results in text hangs. It is recommended to just drop those suspicious samples by using segmentation template `inputProblemHandling.subtitleSampleMaxDeviation`.

See input problem handling for more details.

13.3.6 Audio video desync in redundancy mode without information in logs

Small fluctuations in video DTS samples may be corrected by setting the `handleVideoDtsFluctuations` to `true` (see inputProblemHandling). The AV drift accumulates change may cause the audio-video desync problem, it will be monitored and if exceeding the threshold `maxDtsDriftVideoTimescale` (in Advanced configuration), the segmenter is restarted. Note that in redundancy mode, the drift on reference track is kept, even if a single node is restarted, in order to synchronize the segment information between all nodes. To reset that, restarting of all nodes at the same time is required. In case of video DTS fluctuates around the ideal value, consider set `handleVideoDtsFluctuations` to `false` to avoid the AV desync problem.

13.3.7 Input source captures

For troubleshooting input source issues, acquiring and investigating a network capture for the channel could sometimes be required. `ew-live-ingest-tool` provides options to allow this. It's recommended to use this over tools like `tcpdump` directly, in order to ensure proper multicast membership on the network, as well as capturing all sources for a channel. Examples:

Basic 60 second capture:

```
$ ew-live-ingest-tool --capture <channel> --capture-duration-s 60
```

Circular capture with max 200MB of data. Runs until command is cancelled with `ctrl-c`:

```
$ ew-live-ingest-tool --capture <channel> --capture-max-size 200
```

The capture may then be provided to an Edgeware representive and/or support tickets, together with the full configuration (output from `confcli` command). Together they will allow full re-creation and analysis of the issue.

13.3.7.1 RTP

To detect all packet loss and to allow better robustness against e.g. re-ordered packets, and to be able to tell the nature of the packet loss, it is recommended to enable RTP for the input streams. This will also enable a user to get brief reports, by either enabling periodic reports in the log (default disabled):

```
[root@esb3003 ~]$ confcli services.liveIngest.advanced.
   ↪ rtpLogSummaryPeriodS <period in seconds>
```

or get it asynchronously via `ew-live-ingest-tool`:

```
[root@esb3003 ~]$ ew-live-ingest-tool --periodic-report <channel>
```

This will for instance show the number of consecutive packets that are lost at each occurrence and can help to tell how frequent the packet loss is. It will report both all events from channel start as well as events in the last period only (in case of running `ew-live-ingest-tool`, since the command was last run).

See input source configuration for how to configure with RTP.

13.3.7.2 FEC

Enabling Forward Error Correction (FEC) for RTP streams is recommended to be able to repair limited packet loss. FEC statistics will then also be available in the reports described for RTP above.

It is also possible to set on which log level recovered packets should be logged (default DEBUG).

```
[root@esb3003 ~]$ confcli services.liveIngest.advanced.
   ↪ fecRecoveredPacketLogLevel <level>
```

The information that can be retrieved from the logs and periodic report may be used to configure the appropriate FEC dimension settings to reduce the FEC bandwidth overhead, while still being able to repair lost packets.

See input source configuration for more information for how to configure with FEC.

13.4 Core dumps

In case of severe errors in any of the esb3003 applications, core dumps may be generated. The core dumps will then be stored in `/run/edgeware` as well as via the Automatic Bug Reporting Tool (ABRT) system service. Core dumps in `/run/edgeware` will be removed when the system is rebooted. Main configuration of the ABRT service is done via `/etc/abrt/abrt.conf`. It should be configured to allow sufficient space to hold a number of core dumps. The recommendation is to allow at least 1GB (default).The contents of the ABRT report directory (`/var/spool/abrt`), and `/run/edgeware` will be included in Edgeware tech support dumps when using `ew-tech-support`. See RedHat documentation for further information about the ABRT service.

14 Contacting Edgeware TAC or Customer Support

If you are unable to solve a problem after using the troubleshooting suggestions in this guide, contact a customer service representative for assistance and further instructions. Before you call, have the following information ready to help your service provider assist you as quickly as possible:

- Version of ESB3003 and ESB3002 software that you are running
- Brief description of the problem
- Brief explanation of the steps taken to isolate and resolve the problem
- There is a script called `ew-tech-support` which is used to collect all relevant logs and information about the esb3003 into a zip archive. To run, simply type `ew-tech-support esb3003`

  NOTE: As the amount of data collected may be quite large, the `--reduced` flag is by default used to limit the amount of data, to only include the most recent data in e.g. logs and to exclude large files. To reduce the dump size even further the `ew-tech-support` tool may be run for selected channels only. For a full dump of all data, the `--full` option may be specified.

```
[root@esb3003:~]# ew-tech-support esb3003 [--channel <channel1> --
   ↪ channel <channel2>]
[root@esb3003:~]# ew-tech-support esb3003 [--full]
```

NOTE: For faster compression, the `pbzip2` tool should be installed on the system.

15 Appendix A: Supported Input Formats and Codecs

15.1 Video

Codec Support:

- H.264 (AVC)
- H.265 (HEVC)

Note that support is limited to one instance of each parameter set (SPS/PPS/VPS) for both H.264 and H.265.

15.2 Audio

Codec Support:

- AAC in ADTS transport syntax (stream_type = 0x0f). Supported profiles: AAC-LC, HE-AACv1, HE-AACv2.
- AC-3, a.k.a. Dolby Digital, and Enhanced AC-3 (E-AC-3), a.k.a. Dolby Digital Plus. Both System A (ATSC) and System B (DVB) MPEG-2 Transport Streams are supported as transports.

AAC tracks with 24 kHz sampling frequency are automatically detected as HE-AAC. One-channel tracks are assumed to be HE-AACv2, and other channel configurations are assumed to be HE-AACv1. A consequence of this auto-detection mechanism is that 24 kHz AAC-LC audio can not be used as input.

15.3 Subtitles

Supported subtitle formats:

- EBU Teletext
- Cavena STU P31
- CEA-608/CEA-708 pass-through in SEI NAL units

EBU Teletext subtitles are supported up to Presentation Level 1.5, with the exception of Packet M/29/1, which is not supported.

16 Appendix B: Used files and ports

This is an overview of resources used by esb3003. This can be useful for security audits.

16.1 Files

16.1.1 Written at install/upgrade time

These files do not change at runtime.

Files/directories	Purpose
`/etc/bash_completion.d/ew-live-ingest-tool-completion.bash`	Bash completion
`/etc/edgeware/*`	Configuration
`/etc/init.d/openresty/*`	Configuration
`/etc/sudoers.d/ew-*`	Configuration

Files/directories	Purpose
`/etc/sysconfig/edgeware/*`	Configuration
`/usr/bin/ew-*`	Executables
`/usr/bin/openresty`	Executable
`/usr/lib/systemd/system/ew-*.service`	Configuration
`/usr/libexec/ew-tech-support/*`	Executable
`/usr/lib64/ew-*`	Executables
`/usr/lib/python3.11/site-packages/ew*`	Libraries
`/usr/local/openresty/*`	Openresty files
`/usr/share/edgeware`	Empty folder

16.1.2 Changed at runtime when you reconfigure

Files/directories	Purpose
`/etc/cron.hourly`	Log rotation
`/etc/logrotate.d/ew-*`	Log rotation configuration

16.1.3 Application output files, changed at runtime.

Directories	Purpose
`/mnt/ramdisk/*`	Live Ingest output
`/var/lib/edgeware/*`	Virtual channel files
`/var/log/edgeware/*`	Logs
`/run/edgeware/*`	Configuration, sockets, pids, etc.

Directories configured by confd	Purpose
`services.storage.catchup.locations.*.basePath`	CBM storage

16.2 Ports

16.2.1 Server (HTTP Listen)

Port	Service
5232	radicale
8889	live-ingest
8090	cbm-accelerator
8091	cbm (only for localhost clients)

16.2.2 Client/Server

ESB3003 will connect to network addresses that are specified by these confd configuration parameters:

Ports configured by confd	Purpose
`integration.convoy.events.servers.*`	Events
`services.liveIngest.synchronizedSegmentation.coordinator`	Node synchronization
`storage.catchup.sharedStorage.coordinator`	CBM coordinator
`storage.catchup.clusterMembers`	CBM nodes (repair)
`services.liveIngest.channels.*.inputSources.*.source` (*)	Channel input

(*) Both multicast/unicast addresses can be configured. If a unicast address is specified Live Ingest will open and listen on the configured port. If a multicast address is specified Live Ingest will join and open the configured port.

If FEC error correction is configured by `services.liveIngest.channels.*.inputSources.*.errorCorrection` ↪ then Live Ingest open additional ports.

Error correction	Input source	ports used
fec-1d	rtp://address:port	address:port address:port+2
fec-2d	rtp://address:port	address:port address:port+2 address:port+4

16.3 File/Port used by virtual channels calendar

Port/file configured by confd	Default value
`services.virtualChannels.calendar.url`	`http://localhost:5232/edgeware/virtual_channels/`

16.4 Monitord

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

16.4.1 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

16.4.2 Ports

Port	Purpose
12345	HTTP listen (monitord server)

16.5 Confd/confcli

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

16.5.1 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

16.5.2 Ports

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