Audalaxy Platform Documentation

Audalaxy

Feb 27, 2025

OVERVIEW

The home for audio cloud and learning for developers and technology professionals. Search our documentation resources for Audalaxy products, developer guides and API references.

CHAPTER

ONE

WHAT IS AUDALAXY?

Audalaxy-Audio cloud services is a large bundle of cloud-based audio services. The required orchestration for enterprise applications is included in the Audalaxy product and can be customized. The of cloud-based services bring very complex audio services with high-end features to your existing infrastructure. With many tools and real-time data, you are in control of everything. Experts and coaches with special know-how support your team in the efficient and scalable use, develop exclusive cloud features and interfaces, and give you important knowledge to save your success of the digital transformation.

For more information please open a ticket:

Visit our company website:

CHAPTER

TWO

SELF-SERVICE-CONSOLE

The Audalaxy-Self-Service-Console is a web user interface to manage and control the essential services of Audalaxy. It is possible to request and configure the following components and processes in the Audalaxy platform independently and individually. All activities within the self-service console are executed immediately and fully automatically within the Audalaxy platform.

Features:

- Create / edit / delete channel
- Configure channel
- Start / stop and configure Playout
- Configure playout sound processing
- Configure playout advanced automatic crossfading
- Start sequence editor and configure dynamic scheduler
- View / export playout protocol
- Create / delete source mounts
- Create / delete listener mounts with or without transcoding
- Create / edit / delete stream URL and configure StreamRoute
- Use stream URL generator to create stream urls for aggregators
- Start / stop and configure StreamControl
- Configure Adflow
- Configure Authentications
- Configure Metaport & Metaspreader
- Configure Channel Advanced settings & Global Advanced Settings
- Configure Connect your app
- Configure Log processing
- Configure User management

For more information please open a ticket:

Visit our company website:

CHAPTER

THREE

SPOTCONTROL

Audalaxy SpotControl is real-time dashboard for TV, Desktop and mobile devices. This service in combination with notifications results in a perfect monitoring solution to manage and control the advertisement business.

Features:

- Forecasts based on historical data
- Revenues per time unit
- Revenues / compare per time unit
- Type (preroll/midroll) per time unit
- Fill rate per time unit
- Spot requests per time unit
- Spot impressions per time unit
- Spot requests with no spot return per time unit
- Count successful beep detections per time unit
- Requested ad-duration and filled ad-duration in race
- Geo map with current spot playouts
- Filter for universal channel combinations and geo locations

For more information please open a ticket:

Visit our company website:

CHAPTER

FOUR

STREAMCONTROL

Audalaxy-Streamcontrol is a service for monitoring and controlling external audio sources with easy-to-use API. The main target is the detection and quick elimination of errors in livestream infrastructures, e.g. connection problems to the audio source for simulcast streams.

The following performance values are permanently monitored:

- Streaming connection
- Metadata
- Silence detection

Output:

- JSON
- Big Data storage

For more information please open a ticket:

Visit our company website:

CHAPTER

FIVE

NOTIFICATION SERVICE

Notification Service is a scalable and highly available system service for event-based messaging targeting Slack. Depending on the configuration, it is possible to assign a message or an alert to all events within the Audalaxy platform. This is an essential tool to safely operate the platform.

For more information please open a ticket:

Visit our company website:

CHAPTER

SIX

STREAMRECORDER

The Audalaxy Streamrecorder is a web user interface to record your own and third-party channels on the internet. The main target is analysing of quality and content of many audio streams, e.g. foreign and own program monitoring for editors and program managers. Depending on the authorization, you can log in as administrator or normal user. Administrators can edit and configure normal users and the record function.

For more information please open a ticket:

Visit our company website:

CHAPTER

SEVEN

AUDIOSTREAMER

Audalaxy Audiostreamer is a live streaming platform service for audio content own-developed by Audalaxy. It is written in a modern programming language that enables all advantages of cloud infrastructures. The software can stream in the following protocols, with ad insertion capability:

- ICY streams (MP3, AAC with included metadata)
- RAW Streaming (MP3, AAC without metadata)
- HLS streams

7.1 Features

- HTTP / HTTPs Audiostreaming
- Supports streaming ICY live streams
- Supports a variety of media formats as input
- Supports various ingestion methods (PUSH, RELAY)
- Secure token content protection
- Authenticated connections
- Data aggregations for Big Data analysis
- Automatic stream fallback to a second broadcast route or a cloud playout

7.2 Ad-Features

- Detecting ad break markers via metadata, flipbit (beep)
- Replacement of the original content with digital ads or an insertion of digital ads
- Full individual ad configuration per stream connection via tokens and parameter, e.g. personalized for a user, for individual aggregators, apps, etc.
- Ad break opener and closer before and after ad insertion
- Stream opener for promoted streams
- Keep back spots via token
- Volume adjustments of spots
- Ad insertion capabilities (VAST4)

- All Features from Audalaxy-Adflow

For more information please open a ticket:

Visit our company website:

CHAPTER

EIGHT

METAFLOW

The Metaflow is a scalable solution with easy-to-use APIs for workflows with audio metadatas. The metadata will be synchronized inside the audio stream as essential part of general system in real time for a channel in general and a unique listener session. This allows metadata to be aligned with individual listener delays by inserted ads. Inside the metadata different types of information can be transported, e.g. title, artist or cover, but also control commands and opportunities to show commercials. If use the Audalaxy-Playoutflow, then the Metaflow is fully integrated.

Required services at external audio source:

- *AudioPort*
- *MetaPort*

Recommend services:

- *Playerservices*

Receives metadata via:

- *MetaPort*

Delivers metadata via:

- *MetaSpreader*

Tip: Related topics for DevOps

- *Metadata data-set reference*
- *MetaPort API Specification*
- *MetaSpreader configuration settings*

CHAPTER

NINE

ADFLOW

Adflow is a scalable solution with easy-to-use tools for workflows with audio commercials. The commercials will be processed in real time and individually for every listener. An advertisement agency with an VAST4 compatible ad server is required. The Audalaxy-Adflow supports the following:

1. scenarios for ad delivery:

- Preroll
- Midroll

2. methods of detecting/trigger an ad break:

- Beep
- Flipbit
- Metadata

3. Ad server API:

- AdsWizz
- Adtonos
- AdAlliance (Smartclip)
- AdMediationService (AMS)
- Any VAST4 compatible AdServer

4. Big Data storage:

- Audalaxy-Big Data

5. Additional features:

- Easy configuration per web user interface in real time, e.g. zone_alias, drift, maxAds, duration, etc.
- IAB TCF v2 compliance
- Counting listening impressions of marked elements inside the streams without Ad server (Audalaxy-Big Data required)
- All ad-features from Audalaxy-Audiostreamer

For more information please open a ticket:

Visit our company website:

CHAPTER

TEN

PLAYOUTFLOW

Playoutflow is a scalable solution with easy-to-use tools for workflows with audio playouts.

Required services:

- Audiolibrary
- Scheduler
- Audio processing
- Playout
- Metaflow

Recommend services:

- Audiostreamer
- StreamRoute
- Playerservices
- Big Data control and analytics
- Business support

For more information please open a ticket:

Visit our company website:

CHAPTER

ELEVEN

AUDIOPORT

The AudioPort is a scalable and highly available system service to receive audio streams and can receive in the following protocols:

- ICY streams (MP3, AAC with included metadata)
- MP3, AAC, Ogg Vorbis, FLAC, WAV

Features:

- supports a variety of media formats as input
- supports various ingestion methods (PUSH, RELAY)

For more information please open a ticket:

Visit our company website:

CHAPTER

TWELVE

TRANSCODING

Easy-to-use scalable audio transcoding for on-demand audio and live audio. The transcoding is an automatically deployed process per channel if necessary. It can be controlled in real-time via web interface.

Input-Formats:

- Everything that is supported by ffmpeg (mp3, aac, flac, wave)

Output-Formats:

- MP3 (several bitrates)
- AAC (several bitrates, HEv2)
- FLAC (compressed lossless)

For more information please open a ticket:

Visit our company website:

CHAPTER

THIRTEEN

PLAYOUT

Easy-to-use scalable solution to create a perfect short-lived or 24/7 audio program from audio elements. It is qualified for a large and more than enough amount of playout deliveries. The playout system can be controlled via web interface in real time.

Features:

- FM/DAB-radio-quality
- Perfect mixing for personalized, individual and broadcast playouts
- Sound processing before and after playout
- Audalaxy Metaflow included
- Real-time deployment and controlling
- Easy playout report
- Replay Gain support
- Advanced crossfading/mixing - automatically or with manual values
- Voice-track on ramp
- Drop in on ramp
- Voice-track over music bed, e.g. news

Stream Targets:

- Audalaxy Streamer
- Icecast
- SHOUTcast
- Wowza
- AIS

CDN Delivery:

- Audalaxy CDN
- Privat CDN on request

For more information please open a ticket:

Visit our company website:

CHAPTER

FOURTEEN

AUDIOLIBRARY

The Audalaxy Audiolibrary is a high-performance cloud storage for audio files incl. metadata with following features:

Import:

- Schedules from MusicMaster, GSelector
- Schedules from API-based planning tools
- Schedules based on track tags
- Audiofiles upload with metadata
- FTP or S3-compatible API
- Browser upload via web frontend
- Metadata from MP3 ID3 container

Edit:

- Metadata via web frontend

Cluster:

- Static and dynamic smartblocks (folder) by audio metadata

Deliver to:

- Audalaxy-Sequence editor and dynamic scheduler
- Audalaxy-Playout

For more information please open a ticket:

Visit our company website:

CHAPTER

FIFTEEN

AUDIOTOOLS

The Audalaxy Audiotools are the universal and constantly growing in number of helpers for all situations with easy-to-use API for individual workflows with audio contents.

Features:

- get audio properties
- replay gain (using mp3gain or loudgain)
- loudness
- bpm
- quality

Features - edit audio:

- Drop-in jingle on ramp
- Voice-track on ramp
- Voice-track over music bed
- Startnext / mixpoint based on db-level or loudness
- sound processing
- stereo tool with individual config
- overlay of multiple audio elements on background track
- pre-processed fade in/out, cue in/out
- transcoding

For more information please open a ticket:

Visit our company website:

CHAPTER

SIXTEEN

SOUND PROCESSING

Good audio processing can do wonders and help to reach great harmony for playout sound. Audalaxy can use all kind of scalable software-based audio processors.

16.1 Audalaxy-Sound processing

The system standard of Audalaxy-Sound processing is a very light alignment of volume, so that the output of playout is normalized to similar level of volume.

16.2 Stereo Tool

With Audalaxy is Stereo Tool available.

Requirements:

- valid license
- Stereo Tool config file (sts)
- Audalaxy-Business support level

For more information please open a ticket:

Visit our company website:

CHAPTER

SEVENTEEN

AUDIOCACHE

Audalaxy-Audiocache is an easy-to-use proxy service for audio files that stores content with individual lifetime. For example, this service is recommended for complex audio pre-processing in front of playout.

For more information please open a ticket:

Visit our company website:

CHAPTER

EIGHTEEN

AUDIOBUTLER

Audalaxy-Audiobutler is an exploring and monitoring service for audio file web links or time-based configurations. If the content of the audio file URL changes or a configured time range is reached, the service forwards audio files with metadata as a part of:

- RSS-Feed
- JSON file

Tip: Get started Audalaxy Developers:

- audiobutlerapi

CHAPTER

NINETEEN

DYNAMIC SCHEDULING

The scheduling system is an essential tool to sequence audio contents for personalized, individual and broadcast playout processes. It supports short-lived and 24/7 audio programs. In the Audalaxy ecosystem you can choose what scheduling system should be used.

For more information please open a ticket:

Visit our company website:

CHAPTER

TWENTY

AUDALAXY-SEQUENCE EDITOR AND DYNAMIC SCHEDULER

The Audalaxy-Sequence editor is a web user interface to create an audio sequence from different audio sources, e.g. Audalaxy-Audiolibrary, podcasts, news sources, etc.

The Dynamic Scheduler is necessary to schedule personalized, individual and broadcast audio programs. The scheduler plans the program in real time. For personalized or on-demand channels, the scheduler switches to a special mode. E.g. time-dependent elements are handling separately. Because it's not possible to skip to them.

For more information please open a ticket:

Visit our company website:

CHAPTER

TWENTYONE

EXTERNAL SCHEDULING TOOLS – MUSICMASTER, GSELECTOR OR OTHERS

Audalaxy allows connectivity to external scheduling tool installations. Because the existing scheduling tools integrations are very diverse, there is not only one way to connect Audalaxy. Audalaxy offers a standard way for synchronizing with external schedules, but it can be adapted if it doesn't match to an existing workflow. The enterprise support level is required for further details.

For more information please open a ticket:

Visit our company website:

CHAPTER

TWENTYTWO

SKIP-ON-RADIO

Audalaxy-Skip-On-Radio is a special service for skipping elements inside of an audio program. The Audalaxy platform creates a simply personalized audio stream after each skipping interaction. This function requires the creation of an audio program with the Audalaxy-Playout and modified players with the skip button and a connected Audalaxy-Skip-API. All modified players can use the advanced, personalized audio streaming functions. The enterprise support level is required for further details.

For more information please open a ticket:

Visit our company website:

CHAPTER

TWENTYTHREE

TIMESHIFT

Audalaxy-Timeshift is a special service for time-shift broadcasts of audio programs. The Audalaxy platform record the audio stream for later listening. The function can be controlled with user-friendly parameters in the stream-URL. Recommended jump markers are automatically transferred via Metaflow. The enterprise support level is required for further details.

For more information please open a ticket:

Visit our company website:

CHAPTER

TWENTYFOUR

PLAYERSERVICES

The Audalaxy Playerservices are a collection of scalable and highly available system services for player clients. This system service delivers all types of metadata to player clients, e.g. web player, apps, skills, etc. With stream start using a unique listener id, the client connects to Audalaxy-Player services and receives synchronized metadata in real time for general or a unique listener session. Inside the metadata content several types of information can be delivered, e.g. title, artist or cover, but also control commands and opportunities to show commercials. By now only artist and track title together with a timestamp are garanteed to be delivered. The available data fields depened on what the playout system sends to the metaport.

Protocols:

- HTTPs, HTTP (polling with server-side cache)
- Websockets (real-time)

Tip: Related topics for DevOps

- *Playerservices Endpoints*
- *Metadata data-set reference*
- *Metaflow*

CHAPTER

TWENTYFIVE

STREAMROUTE

StreamRoute is a scalable and highly available system service for effectively connects user stream requests to Audiostreamer infrastructure running in Audalaxy. The service can be available with own custom domain and automatic https certifications.

Features:

- Generate listener IDs if necessary, based on client fingerprinting
- Responsible for aggregator detection
- Skip request specific handling
- Alias domain if required
- Forwarding parameters to Audalaxy-Audiostreamer or other Streamers
- Detour blacklist-IPs
- No pre-roll spot at specific aggregator requests, e.g. wake up alarm
- Generate PLS and M3U-Playlists on request over stream URL
- Create individual landing pages with metadata for aggregators
- Certification for listeners from own infrastructure
- GDPR no tracking option
- Special handling for special aggregators

All communications with Audalaxy user audio streams comprises of a set of parameters that can be passed in the stream request. These parameters will capture sufficient details about the available opportunity and help serve a targeted audio stream to the user.

25.1 Request listener audio streams

25.1.1 URL-Scheme

The URLs to request audio streams are systematized according to the following scheme: `https://domain/programkey/format/aggregator/listenerid/?parameters`

25.1.2 Parts of audio stream request

domain

StreamURL domain for this channel

Example `streams.audiobrand.com`

programkey

Short identifier of this stream name

Type string

Example `chill`

format

Codec and bit rate combined

Type string

Example `mp3-192`

aggregator

Optional aggregator identifier that should be displayed in the analytics dashbord

Type string

Example `mywebsite` or `tunein`

listenerid

Optional unique ID to identify the listener

Type string

Example `142e4f19b49da39529de786f81344b76`

Hint: If you use a specific provider to generate the listener ID and use dependent functions, then please use the listener IDs container.

- *Build the listener IDs container*

---

parameters

Optional URL parameters

Type string

Example `tracking=true`

25.1.3 Request audio streams via media file (m3u/pls/m3u8)

This is an extension of the standard *URL-Scheme*. To request media file playlists, add a file extension to the URL.

m3u

Example1 `https://domain/programkey/format/`*play.m3u*

Example2 `https://domain/programkey/format/aggregator/`*play.m3u*`?parameters`

Example3 `https://domain/programkey/format/aggregator/listenerid/`*play.m3u*`?parameters`

`pls`

Example1 `https://domain/programkey/format/`*play.pls*

Example2 `https://domain/programkey/format/aggregator/`*play.pls*`?parameters`

Example3 `https://domain/programkey/format/aggregator/listenerid/`*play.pls*`?parameters`

`m3u8`

Example1 `https://domain/programkey/format/`*play.m3u8*

Example2 `https://domain/programkey/format/aggregator/`*play.m3u8*`?parameters`

Example3 `https://domain/programkey/format/aggregator/listenerid/`*play.m3u8*`?parameters`

25.1.4 Parameters

`tracking`

This is a simple GDPR no tracking option. All parameters will be drop. Possible values are true and false. Default value assumed is false.

Options `true`, `false`

Default `true`

Example `tracking=false`

Tip: Get started Audalaxy Developers:

- *Integrating a TCF 2.0 CMP Consent*
- vast4integration

CHAPTER

TWENTYSIX

PROTECTED STREAMS, AUTHENTICATION

This service is useful for special content without public access, e.g. behind a paywall. Protected streams are only accessible with proper authentication. The authentication is based on user specific cryptographically secured JWT (JSON Web Token) with a defined TTL. The enterprise support level is required for further details.

For more information please open a ticket:

Visit our company website:

CHAPTER

TWENTYSEVEN

METASPREADER

The MetaSpreader is a scalable system solution to send current audio stream metadata to different receivers.

Supported receiver types:

- Audalaxy Audiostreamer
- JSON-HTTP push to external targets / URLs
- Icecast-Mounts
- AIS-Mounts
- Radioplayer
- radio.net
- FTP servers

Tip: Related topics for DevOps

- *MetaSpreader configuration settings*
- *Metadata data-set reference*
- *MetaPort API Specification*
- *Metaflow*

CHAPTER

TWENTYEIGHT

METAPORT

The MetaPort is a scalable and highly available system service to receive metadata with easy-to-use API. External playout systems or other services can push metadata to a secured endpoint. This is a required service, if you use external audio source(s) to push current metadata into your audio stream.

Supported data types:

- GET URL parameters
- POST JSON
- POST XML

Tip: Related topics for DevOps

- *MetaPort API Specification*
- *Metadata data-set reference*
- *MetaSpreader configuration settings*
- *Metaflow*
- *Delay metadata*

CHAPTER

TWENTYNINE

CLOUDCONNECT

With Audalaxy CloudConnect, existing customer infrastructure can be connected to the Audalaxy platform. This enables all advantages and services of Audalaxy on external customer infrastructure. For example, existing streaming server hardware can be controlled with Audalaxy-CloudConnect using the Audalaxy Self-Service Console.

Features:

- Full access to Audalaxy Audio cloud services
- Full access to the Audalaxy-API
- Personal customer support in local language

Required:

- Enterprise support

For more information please open a ticket:

Visit our company website:

CHAPTER

THIRTY

LOGIMPORTER

The Audalaxy-Logimporter is a (stand-alone) system service, which can be installed on your own streaming servers. The service sends pre-processed logs of these streaming servers to the Audalaxy-Audio cloud services. This is necessary to generate listener statistics and special log files, notifications and other statistical analyzations. You can use Logimporter to stream logs to Audalaxy ("tail-mode") on the fly or to send the content of a single file at once.

Cloud features:

- Can run as a docker container
- Health checks
- Anonymize ip addresses / GDPR compliant
- Supports time zones

Logimporter can work with log files from these streaming servers:

- Icecast
- Shoutcast 1
- Windows Media Server
- AIS (Ver. 7.x and 8.x)

Supported operating systems: Linux, Windows, BSD, MacOS

Tip: Get started with Audalaxy-Logimporter:

- *Logfileimport per Logimporter*

CHAPTER

THIRTYONE

LOGPROCESSING

The Audalaxy-LogProcessing service can create and provide log files for external service partners.

A solution is available for this service partner:

- ma IP Audio (Log file-based measurement of radio usage for German radio stations)
- Radioanalyzer - radioanalyzer.com
- AdsWizz Audiometricx
- Streamalyzer

It is also possible to export logs in generic Icecast, CSV or JSON formats.

For more information please open a ticket:

Visit our company website:

CHAPTER

THIRTYTWO

DATA ANALYTICS

Using Audalaxy-Data Analytics enables access to your own raw data with special big data tools. The platform generates millions of data logs per day, and store those to a high security big data cluster. The use is recommended for data analysts with special knowledge. You can book training courses and get access. The business support level is required for further details.

For more information please open a ticket:

Visit our company website:

CHAPTER

THIRTYTHREE

APIS

An increasing number of APIs enables data exchange with external big data systems.

Currently available:

- Quantyoo
- Splunk

For more information please open a ticket:

Visit our company website:

CHAPTER

THIRTYFOUR

STREAMWATCH

The Audalaxy Streamwatch is an easy-to-use statistic and reporting tool to control the usage of all channels. It was specially developed for program managers to get an overview of the most important values and is available as a desktop and mobile version. It is easy to spot historical trends and the current usage.

The following KPIs are available, among others:

- Number of simultaneous listeners (ccu) in real time
- Active sessions / stream starts
- Average Active Sessions
- Average Time Spent Listening
- Total Listening Hours
- Bounce rate
- Aggregators (website, app, radio player, skill, etc.)
- Geo information based on the IP address
- Quarter Trends
- Calculated trend for current quarter
- Evaluation of best performances / records

For more information please open a ticket:

Visit our company website:

CHAPTER

THIRTYFIVE

ADVISOR

Audalaxy offers the ability to book an advisor with in-depth knowledge of the big data pool to advise management on the use of data and recommend new work methods that focus on data-driven workflows.

For more information please open a ticket:

Visit our company website:

CHAPTER

THIRTYSIX

HEALTH

In addition to the listener usage data, data about the health status of the platform is continuously detected and stored. Audalaxy-Health is a part of Audalaxy-Cloud Connect with enterprise support level. This real-time data can be accessed after a training.

For more information please open a ticket:

Visit our company website:

CHAPTER

THIRTYSEVEN

DEVELOPERS

This section of the documentation details the Developer APIs usable to connect with audio cloud services and other details from Audalaxy.

37.1 Listener ID and Meta ID

A unique listener ID can and should be provided along with the stream URL to get best results from Adservers and other services. By now we support one listener ID per stream. If you work with adswizz this could be the adswizz listener ID.

There a multiple ways to provide a listener ID:

- As part of the virtual stream url: https://domain/programkey/format/aggregator/listenerid/?parameters
- As query parameter: https://domain/programkey/format/aggregator?listenerid=xyz

As the listener ID is especially important for ad insertion, we support a different ID that can be used along with listener specific metadata that we call "Meta ID".

You should provide a unique meta id for each listener if you want to use listener specific metadata. This works by adding a query parameter metaid=xyz to the stream url.

https://domain/programkey/format/aggregator/listenerid/?otherparameter=123&metaid=xyz

You can than use the same meta id subscribe to metadata events and get the metadata for the listener. see: *Playerservices*

37.2 Listener IDs and listener IDs containers (WORKING DRAFT)

The following description is planned to be supported in the future and works as a working draft. Audalaxy supports listener IDs. It is possible to pass one or more listener IDs in order to be able to use dependent functions.

If listener IDs have been integrated and a function requires it, Audalaxy uses the listener IDs and forwards them to the associated provider.

37.2.1 Build the listener IDs container

Use an JSON array of identifiers with value to build the listener IDs container.

```
[
    { "identifierid": "value" },
    { "identifierid": "value" }
]
```

identifierid
Unique ID to identify the provider.

The identifierid should be as short as possible, and can contain lowercase alphanumeric characters.

Type string

Example adswizz

Hint: Please note the *Predefined identifier IDs*.

The ID can also be used without an associated provider. In this case use the `identifierid = qc` or the simple listener ID transfer via stream URL: *Parts of audio stream request*

---

value
The value is the listernerID and should be a UUID.

Type string

Example ad82903a-7ba0-11ec-90d6-0242ac120003

37.2.2 Submit Listener IDs

Request a stream via *StreamRoute* and submit listener IDs container as a base64 encoded string.

idsv1
Parameter used to pass listener IDs container as a base64 encoded string through the stream uri

Example

```
idsv1=W3sicWMiOiI0ZThkNmQwNTEyOWI4ZDFjNmU2MTZiY2ZlYWIyYjlmMiJ9LHsiYWRzd2l6eiI6ICJFNDUzNTk1Mi02QkVGI
```

37.2.3 Predefined identifier IDs

37.3 Metadata data-set reference

The *Metaflow* describes how the platform handles metadata. To get metadata from Audalaxy, use the *MetaSpreader*. A metadata data-set contains the following data for each audio item:

```
{
    "id": "ebbf8b04da405f51ad6aab607a7ffbe8",
    "channelname": "Radio Brocken Lovesongs",
    "channelshortname": "RB Lovesongs",
    "time": "2021-02-18T20:55:38Z",
    "duration": "227",
```

(continues on next page)

(continued from previous page)

```
    "song": "Naked",
    "artist": "James Arthur",
    "channelkey": "fhl_rbroos7_klun3",
    "brandid": "radiobrocken",
    "cover": "https://cloudspace/cover.jpg",
    "separator": "-",
    "timezone": "UTC",
    "etype": 1001
}
```

id
: Unique ID to identify the metadata set

  Example `ebbf8b04da405f51ad6aab607a7ffbe8`

channelname
: Name of the stream

  Example `Radio Brocken Lovesongs`

channelshortname
: Short-name of the stream

  Example `RB Lovesongs`

time
: ISO 8601 combined date-time format

  Example `2021-02-18T20:55:38Z`

duration
: Duration of the audio element in seconds

  Example `227`

song
: Title of the audio element

  Example `Naked`

artist
: Artist of the audio element

  Example `James Arthur`

channelkey
: Unique key / ID of the channel

  Example `fhl_rbroos7_klun3`

brandid
: Brand ID for this channel

  Example `radiobrocken`

timezone
: Time zones represented by alphabetic abbreviations

  Example `UTC`

etype
: Content-type ID of the audio item as an integer

Default `0`

Example `1001`

Type	ID
unknown	0
Jingles	101
Promo	102
Advertising	103
News	104
Article	105
Music 1	1001
Music 2	1002
Music 3	1003
Music 4	1004
Music 5	1005
Music 6	1006
Music 7	1007
Music 8	1008

`cover`
Url of the cover image

Example `https://cloudspace/cover.jpg`

`separator`
Ability to specify the separator between artist and song. This is sometimes relevant to optimize the frontend view.

Example -

Tip: Related topics

- *MetaSpreader configuration settings*
- *MetaPort API Specification*
- *Metaflow*

37.4 MetaPort API Specification

Developer's guide to submitting external metadata to *Metaflow* and into your audio stream.

The *Metaflow* describes how the platform handles metadata. To push metadata to Audalaxy, use the *MetaPort*. This is a required service, if you are using own (encoded) external audio source(s) and **you will push current metadata into your audio stream**.

37.4.1 Push metadata

GET https://metadata.streamabc.net/metapush/(string: *channelkey*)**/**

string: *token* Send metadata to your audio stream. `channelkey` and `token` can be found in your own *Self-Service-Console*.

- `channelkey` - unique key / ID of the stream.
- `token` - MetaPort Token for this channel

Query Parameters

- **song** (*string*) – song name (can be empty)
- **artist** (*string*) – artist name (can be empty)
- **encoding** (*string*) – `utf8` oder `windows` - default (if not specified): `utf8`, for Windows ISO8859-1 is used for song/artist
- **duration** (*int*) – Length of the song in seconds
- **time** (*string|int*) – if int: unix timestamp | if string: time in RFC 3339 format, default (if not specified): current time is used
- **tracktype** (*string*) – type of record, `now` for current track, `next` for following track, default (if not specified): `now`
- **etype** (*int*) – Content-type ID of the audio item as an integer

  default `0`

Type	ID
unknown	0
Jingles	101
Promo	102
Advertising	103
News	104
Article	105
Music 1	1001
Music 2	1002
Music 3	1003
Music 4	1004
Music 5	1005
Music 6	1006
Music 7	1007
Music 8	1008

- **id** (*string*) – reference ID of the element (internal ID for evaluation, can be empty)
- **isrc** (*string*) – ISRC (can be empty)
- **separator** (*string*) – separator between artist and song, default is " - "

Example request:

Bash

```
$ curl https://metadata.streamabc.net/metapush/qc_fkvwoo65bsvz_tuzx/bPlfFlFpEw27We6a
```

POST https://metadata.streamabc.net/metapush/(string: *channelkey*)/(string: *token*)

Send metadata to your audio stream. `channelkey` and `token` can be found in your own *Self-Service-Console*.

- `channelkey` - unique key / ID of the stream.
- `token` - MetaPort Token for this channel

Example request:

Bash

```
$ curl      https://metadata.streamabc.net/metapush/qc_fkvwoo65bsvz_tuzx/bPlfFlFpEw27We6a      -X P
```

The content of `body.json` has the following structure:

```
{
    "id":"",
    "time":"2021-09-01T16:11:49Z",
    "duration":210,
    "song":"Song Name",
    "artist":"Artist Name",
    "cover":"https://cover.url",
    "images": {
        "small": {
            "url": "https://cover.url"
        },
        "medium": {
            "url": "https://cover.url"
        },
        "large": {
            "url": "https://cover.url"
        }
    },
    "album":"Album Name",
    "separator":" - ",
    "timezone":"UTC",
    "etype":0,
    "tracktype":"now"
    "isrc": "",
    "extdata": {
        "extData1": "...",
        "extDataN": "..."
    },
    "externalid": {
        "zenonid": "12344",
        "otherid": "abcdsd"
    }
}
```

Response JSON Object

- **id** (*string*) – unique ID of the metadata element, if empty a UUID will be generated
- **time** (*time.Time*) – RFC3339 format, if empty the current time will be used
- **duration** (*int*) – length of the audio item in seconds (mandantory)

- **song** (*string*) – song name (mandantory)
- **artist** (*string*) – artist name (mandantory)
- **cover** (*string*) – cover URL (can be empty)
- **images** (*object*) – object of several image sizes (can be empty)
- **album** (*string*) – album name (can be empty)
- **separator** (*string*) – chars to separate artist and song in stream metadata, default is " - "
- **timezone** (*string*) – time zone, should be UTC
- **etype** (*int*) – element type of the audio item as an integer
- **tracktype** (*string*) – type of track data, one of now or next
- **isrc** (*string*) – ISRC value of the audio item (can be empty)
- **extdata** (*object*) – object of external data, will be passed untouched
- **externalid** (*string*) – external ID of the metadata element with sub-items; you can choose this metadata element itself, but it must be uniform

Example response:


```
{
    "time":"2021-09-01T16:11:49Z",
    "start":"01.09.2021 16:11:49",
    "start_timestamp":1630512709,
    "duration":210,
    "song":"",
    "artist":"",
    "code":"qc_fkvwoo65bsvz_tuzx",
    "channelkey":"qc_fkvwoo65bsvz_tuzx",
    "timezone":"UTC",
    "channel":"qc_fkvwoo65bsvz_tuzx",
    "id":"",
    "userplayout":false,
    "separator":" - ",
    "etype":0,
    "album":"",
    "type":"now",
    "isrc":""
}
```


Response JSON Object

- **time** (*time.Time*) – start time in RFC3339 format
- **start** (*string*) – local date-time
- **start_timestamp** (*int64*) – unix timestamp
- **duration** (*int*) – Length of the audio item in seconds
- **song** (*string*) – song name
- **artist** (*string*) – artist name
- **code** (*string*) – code (legacy field for channelkey)

- **channelkey** (*string*) – channelkey - unique key / ID of the stream
- **timezone** (*string*) – time zone of the time specification
- **channel** (*string*) – channelkey - unique key / ID of the stream
- **id** (*string*) – reference ID
- **userplayout** (*bool*) – default: `false` | `true` in case of personalized audio stream
- **separator** (*string*) – separator between artist and song, default is " - "
- **etype** (*int*) – element type of the audio item as an integer
- **type** (*string*) – type of track data one of now or next
- **isrc** (*string*) – ISRC of the audio item

Tip: OpenAPI Specification

- https://metadata.streamabc.net/openapi/api.json
- https://metadata.streamabc.net/openapi/api.yaml

Related topics

- *Metadata data-set reference*
- *MetaSpreader configuration settings*
- *Metaflow*

37.5 Playerservices Endpoints

Warning: Deprecation notice

The old endpoint that start with https://playerservices.streamabc.net are deprecated and continue to work for a while but get no updates and are not high available anymore. Please migrate to the new endpoints mentioned below.

37.5.1 Polling

`https://api.streamabc.net/metadata/channel/{channelkey}.json` (Channel JSON Polling) `https://api.streamabc.net/metadata/channel/{channelkey}.txt` (Channel TXT Polling) `https://api.streamabc.net/metadata/station/{stationkey}.json` (Station JSON Polling, metadata for all channels of this station) `https://api.streamabc.net/metadata/listener/{metaid}.json` (Listener JSON Polling, metadata a specific listener, please consider using Websockets here)

37.5.2 Websockets

`wss://api.streamabc.net/metadata/listener/{metaid}` (Websockets Listener Metadaten) `wss://api.streamabc.net/metadata/channel/{channelkey}` (Websockets Channel Metadaten) `wss://api.streamabc.net/metadata/station/{stationkey}` (Websockets Station Metadaten)

To use the websockets endpoints you need to subscribe and listen for incoming JSON messages. There are PING messages that are sent to keep the connection alive and that have to be answered with PONG messages. The browser implementation of the websockets API usually does this automatically.

A simple Javascript browser example can be found here:

```
<pre id="now"></pre>
<pre id="next"></pre>
<pre id="status"></pre>
<script>
    const now = document.getElementById("now");
    const next = document.getElementById("next");
    const status = document.getElementById("status");
    const socket = new WebSocket("wss://api.streamabc.net/metadata/channel/qc_
↪fkvwoo65bsvz_tuzx");

    socket.onopen = function () {
        status.innerHTML += "Status: Connected\n";
    };

    socket.onclose = function () {
        status.innerHTML += "Status: Disconnected\n";
    };

    socket.onmessage = function (e) {
        const metadata = JSON.parse(e.data);
        console.log(metadata);
        if (metadata.type === "next") {
            next.innerHTML = "Es folgt: " + metadata.song + " von " + metadata.artist;
        } else {
            now.innerHTML = "Es läuft: " + metadata.song + " von " + metadata.artist;
        }
    };
</script>
```

37.5.3 Listener specific Metadata

In order to activate listener specific metadata you need to start a stream by providing a unique meta id for each listener as query parameter `metaid` as part of the stream url.

You than use the same meta id to either subscribe to the websocket or access the polling endpoint.

Tip: Related topics

- *Metadata data-set reference*
- Description of *Playerservices*
- *Metaflow*

- *Listener ID and Meta ID*

GET https://api.streamabc.net/metadata/channel/(string: *channelkey*).json

Example request:

Bash

```
$ curl https://api.streamabc.net/metadata/channel/qc_fkvwoo65bsvz_tuzx.json
```

Example response:


```
{
    "rawdata": "",
    "userplayout": false,
    "channel": "Audalaxy ShowCase Sequenz",
    "session": "",
    "position": "c9eb8216c09ccf4541edadee70178aa1",
    "marker": "",
    "id": "",
    "mount": "",
    "channelkey": "qc_fkvwoo65bsvz_tuzx",
    "timestamp": "2021-12-01T17:58:32Z",
    "artist": "Adele",
    "song": "Someone Like You",
    "station": "audalaxy",
    "start": "01.12.2021 17:58:32",
    "duration": 294,
    "start_timestamp": 1638381512,
    "type": "now",
    "etype": 1008
}
```


Response JSON Object

- **time** (*time.Time*) – ISO 8601 combined date-time format
- **start** (*string*) – local date-time
- **start_timestamp** (*int64*) – unix timestamp
- **duration** (*int*) – Length of the audio item
- **song** (*string*) – song name
- **artist** (*string*) – artist name
- **code** (*string*) – code
- **channelkey** (*string*) – channelkey - unique key / ID of the stream
- **timezone** (*string*) – time zone of the time specification
- **channel** (*string*) – channelkey - unique key / ID of the stream
- **id** (*string*) – reference ID
- **userplayout** (*bool*) – default: `false` | `true` in case of personalized audio stream
- **separator** (*string*) – separator between artist and song, default is " - "
- **etype** (*int*) – Content-type ID of the audio item as an integer

- **type** (*string*) – type of record
- **isrc** (*string*) – ISRC of the audio item

37.6 Integrating a TCF 2.0 CMP Consent

Audalaxy is a IAB Vendor and it is possible to use a TCF 2.0 (https://iabeurope.eu/tcf-2-0/) compliant Consent Management Platform to handle the listener's consent.

When a CMP has been integrated, Audalaxy will pass the audio user's specific consents onto any approved IAB Vendor that are part of the Global Vendor List <https://vendor-list.consensu.org/v2/vendor-list.json>.

37.6.1 Uri parameters

Request a stream via *StreamRoute* and use following parameters:

cstring
Parameter used to pass TCF2 consent string through the stream uri, if CMP is using

Example

```
cstring=CO2rS1-O2rS1-AGABCENAtCsAP_AAH_AAAwIGGNV_T5fb2vj-3Z99_tkaYwf95y3p-wzhheMs-
↪8NyYeH7BoGP2MwvBX4JiQKGRgksjKBAQdtHGhcSQgBgIhViTKMYk2MjzNKJLJAilsbO0NYCD9mnsHT3ZCY70-
↪vu__7P3ffwMMar-ny-3tfH9uz77_bI0xg_
↪7zlvT9hnDC8ZZ94bkw8P2DQMfsZheCvwTEgUMjBJZGUCAg7aONC4khADARCrEmUYxJsZHmaUSWSBFLY2doawEH7NPYOnuyEx3
↪_2fu-_gAA.YAAAAAAAAAA
```

aw_0_req.userConsentV2
Adswizz AIS compatible parameter used to pass TCF2 consent string, if CMP is using - Audalaxy *Audiostreamer* and *StreamRoute* supports this parameter

Example

```
aw_0_req.userConsentV2=CO2rS1-O2rS1-AGABCENAtCsAP_AAH_AAAwIGGNV_T5fb2vj-3Z99_
↪tkaYwf95y3p-wzhheMs-
↪8NyYeH7BoGP2MwvBX4JiQKGRgksjKBAQdtHGhcSQgBgIhViTKMYk2MjzNKJLJAilsbO0NYCD9mnsHT3ZCY70-
↪vu__7P3ffwMMar-ny-3tfH9uz77_bI0xg_
↪7zlvT9hnDC8ZZ94bkw8P2DQMfsZheCvwTEgUMjBJZGUCAg7aONC4khADARCrEmUYxJsZHmaUSWSBFLY2doawEH7NPYOnuyEx3
↪_2fu-_gAA.YAAAAAAAAAA
```

aw_0_req.gdpr
Adswizz compatible parameter used to pass GDPR handle, if no CMP is using - Audalaxy *Audiostreamer* and *StreamRoute* supports this parameter

Example `aw_0_req.gdpr=true`

CHAPTER

THIRTYEIGHT

CLOUD OPERATORS

This section of the documentation details the Cloud Operators opportunities usable to control and configure audio cloud services and other details from Audalaxy.

38.1 JSON Configuration Guide for Advanced Settings

Follow this link to various settings for ad injection (Ad Tech) and audio streaming (Playout & Processing):

JSON Configuration Guide for Advanced Settings

38.2 Ad insertion capabilities

Following is a list of special options and configuration regarding audio advertising.

38.2.1 Opener and closer - instream / midroll

Configure opener and closer of instream ads.

Supported format: mp3, wav

```
{
  "ad_instream_opener": [
      "https://streamabc-audio-content.s3.eu-central-1.amazonaws.com/testfiles/563935__
↪fester993__guitar-trombone-reverb.wav"
  ],
  "ad_instream_closer": [
      "https://streamabc-audio-content.s3.eu-central-1.amazonaws.com/testfiles/564003__
↪rasmuspnielsen__low-hum-32-hz.wav",
      "https://streamabc-audio-content.s3.eu-central-1.amazonaws.com/testfiles/564309__
↪keybal__thermal-sniper-shot-3.wav",
      "https://streamabc-audio-content.s3.eu-central-1.amazonaws.com/testfiles/563850__
↪nikplaymostories__fail-trombone-wah-wah-wah-sound-effect.mp3"
  ]
}
```

38.2.2 Opener - preroll

Configure opener for preroll.

Supported format: mp3, wav

Set `ad_force_preroll_opener` to `true` if the opener should always be included, e.g. in the case of sponsored channels, the opener is always inserted at the start of the session.

```
{
  "ad_preroll_opener": [
      "https://streamabc-audio-content.s3.eu-central-1.amazonaws.com/testfiles/563935__
↪fester993__guitar-trombone-reverb.wav"
  ],
  "ad_force_preroll_opener": "true"
}
```

38.2.3 Closer - preroll

Configure closer for preroll.

Supported format: mp3, wav

```
{
  "ad_preroll_closer": [
      "https://streamabc-audio-content.s3.eu-central-1.amazonaws.com/testfiles/563935__
↪fester993__guitar-trombone-reverb.wav"
  ],

}
```

38.2.4 Grace time

The grace time, in seconds, is the time during the Audalaxy Streamer service should ignore ad triggers. This timer starts immediately after an ad trigger with a successful impression.

grace_time

grace time between midrolls

grace_time_preroll

grace time from preroll to first midroll

grace_time_reconnect

grace time between prerolls in case of reconnect

```
{
    "grace_time": 600,
    "grace_time_preroll": 900,
    "grace_time_reconnect": 30
}
```

38.2.5 Disable Preroll Ads

Preroll spots for an active session can be suppressed using the URL query parameter context with value fHA6LTE= (AIS default parameter): http://streamurl/?context=fHA6LTE=

Further configuration for ads like ad duration, number of ads as well as disabling ads in general can be done using tokens (JWT). This is the preferred way for the most flexibility but needs more work to implement on the client side.

38.2.6 Disable All Ads

Using a special advanced configuration it is possible to disable all ads for a specific session.

```
{
    "no_ad_parameter": "noads"
}
```

This configured parameter has to be passed as value of the URL query parameter context. For the example above this would lead to: http://streamurl/?context=noads

Keep in mind that everyone that knows this value can suppress ads in your stream. So it is recommended to use this parameter for specific purposes only. It can be changed on the fly if the value is misused or leaked.

For a more secure way to do this it is also possible to use tokens (JWT). Tokens have a limited lifetime and can not be used for multiple sessions.

38.2.7 Max. ad duration per session

The max. ad duration per session, in seconds, indicates the maximum offset that the listener may have in the session. This could be necessary, for example, if the listener listens to the channel all day.

```
{
    "ad_duration_max": 900
}
```

38.3 Geoblocking / Geostreaming

In disallowed countries an audio file is played and the stream stopped.

```
{
    "geo_allow_countries": ["DE","AT"],
    "geo_fallback": "https://url.zu.einer.audiodatei",
    "geo_active_hours": [20,21]
}
```

geo_allow_countries Array specification of which countries are allowed to stream freely, all others are blocked; specify as ISO code

geo_fallback URL to a file that is played as an announcement

geo_active_hours Optional array with hours as a number when blocking is active. If it is always to be active, omit this block or specify an empty array.

38.4 Authentication

38.4.1 Basic access

The simplest technique for enforcing access controls.

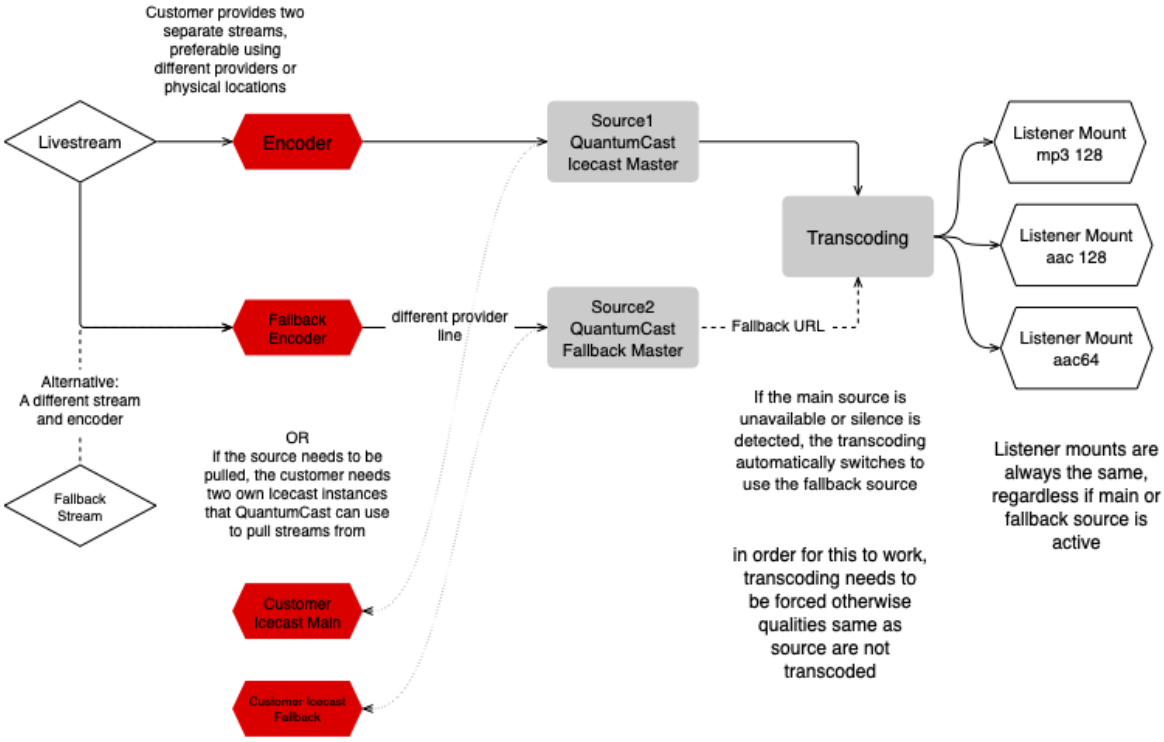
```
{
    "mountacl": {
        "username": "",
        "password": ""
    }
}
```

38.4.2 Token access

Audalaxy supports authentication by using a Token. Token-Based Authentication, relies on a signed JSON Web Token (JWT) that is sent to the server on each request.

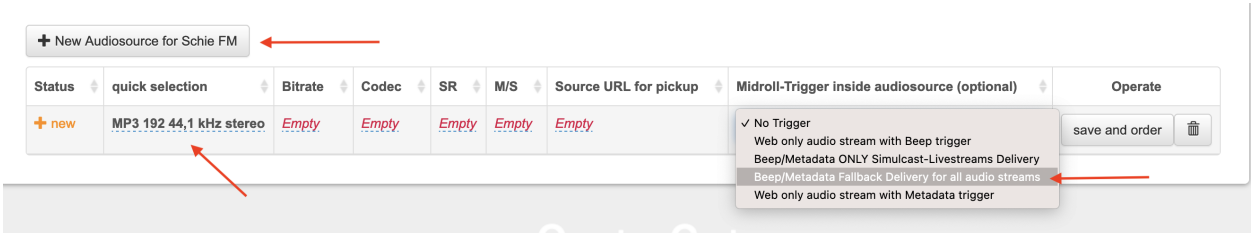
38.5 Auto switch to fallback stream

The Auto Switch function can be used to replace an audio source that is no longer available - for example, as a fallback for a simulcast.

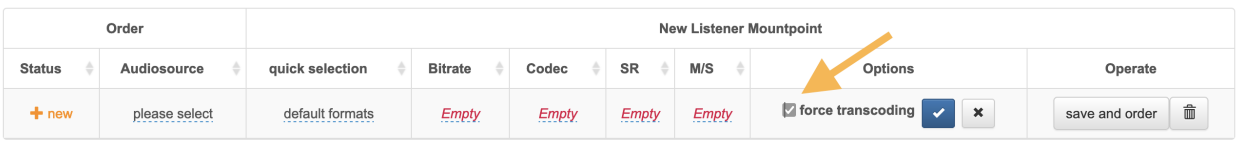


38.5.1 Workflow

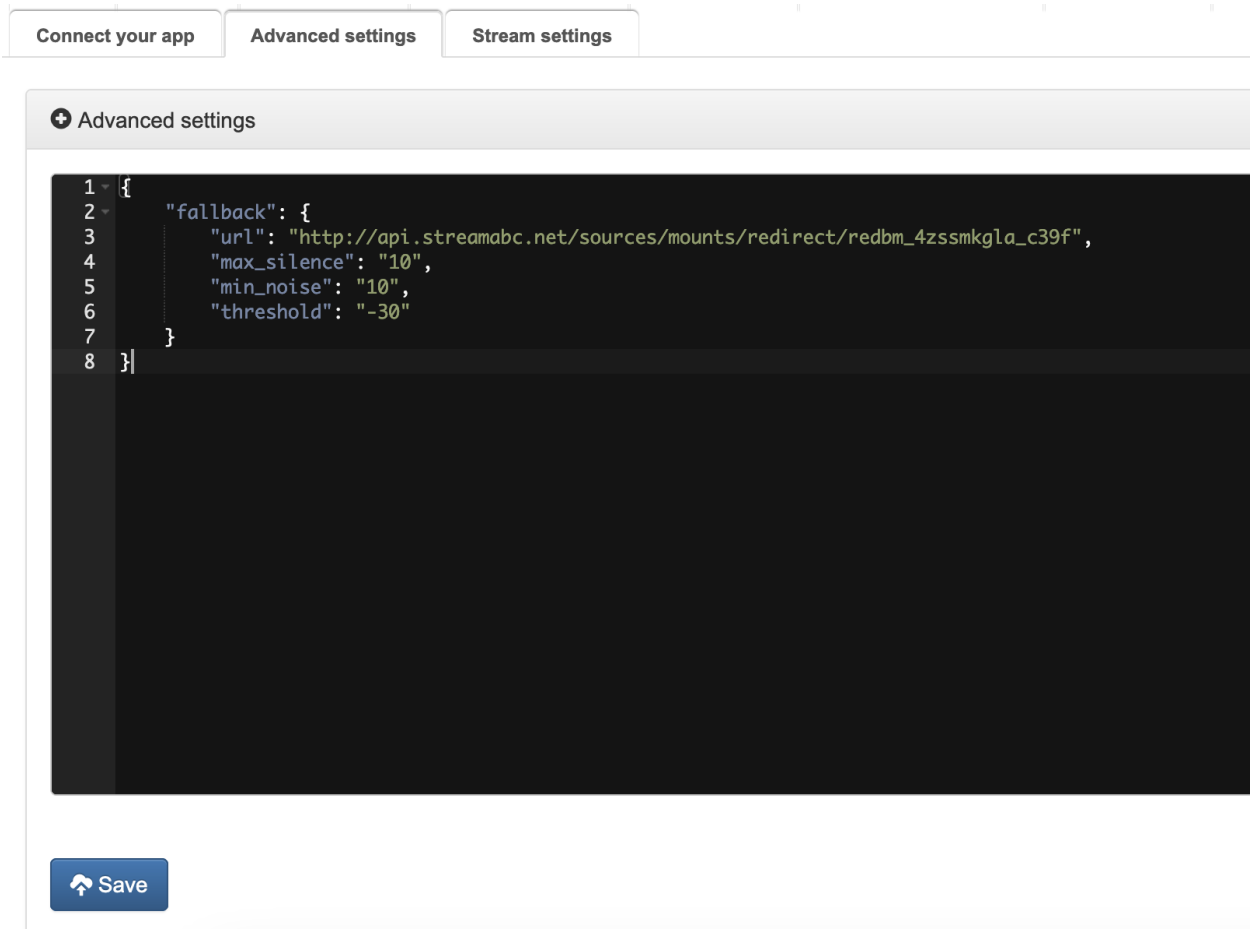
Create a new additional source for this channel with the right incoming bitrate and select the fallback master server (the naming in the UI is a bit misleading, it's not only "mid-roll trigger inside source").



When creating listener mounts transcoding should be "forced". Otherwise transcoding will be skipped if the source and listener mount have the same spec (e.g. same bitrate/format). The fallback mechanism is part of our transcodings so it needs to be enabled.



A special setting for the channel has to be created in "Advanced Settings". This is the place for all settings that don't have an UI and is written in JSON.



Hint: Please check if there is a red mark in the sidbar. In this case the JSON is not valid and can not be saved.

38.5.2 Description Structure of the JSON document

- parameters silence, noise and threshold are freely configurable
- configuration per channel
- transcoding feature necessary

```
{
    "fallback": {
        "url": "http://api.streamabc.net/sources/mounts/redirect/qc_0dm7w3mdgo3b_koaz",
        "max_silence": "10",
        "min_noise": "10",
        "threshold": "-30"
    }
}
```

`fallback` is the Key for the settings

`url` is the StreamURL of the fallback stream. You can just copy the URL of the source here. However, we have a special service that redirects to the source if a channelkey (our unique ID for a channel) is given. It auto-selects fallback

mounts. The syntax is http://api.streamabc.net/sources/mounts/redirect/<channelkey> Just use the channelkey of the current channel here. You can find it under "Setup"

`max_silence` is the number of seconds that the stream has to be silent before switching to fallback

`min_noise` is the number of seconds without silence before switching back

`threshold` is the audio level in dB that is considered silence

38.5.3 Special use cases

1. If the fallback is a Audalaxy cloud playout there is no need to add an additional source. In that case use the special URL http://api.streamabc.net/sources/mounts/redirect/<channelkey>/playout and replace <channelkey> with the channelkey of the playout channel.
2. In case of sources with integrated bitflips we need a dedicated fallback url for every codec and bitrate as it is not possible to transcode these channels (in order to not loose the bitflips)

38.6 Troubleshooting

38.6.1 Audio glitching

Audio glitching means crackling, popping, and other sound problems. They can occur for a variety of reasons.

Glitches after instream ads

Glitches after instream ads can occur if bit reservoirs are used in the encoder. If an MP3 file or stream is encoded with bit-reservoir feature enabled then the consecutive frames depend on each other. In the case of in-stream ads, the stream is cut exactly at the trigger point. During this process, dependent frames in the stream can be separated and heard as glitches.

38.7 Logfileimport per Logimporter

The Audalaxy Logimporter is a dedicated software tool, distributed as executable binary, that can be used to transmit log data in a secure and GDPR safe way to the Audalaxy big data platform. It can be used to send a log file at once or (preferred) to continously send new log entries in realtime by "tailing" files.

The latest version can be downloaded here:

- Linux 64Bit: https://streamabc-sw.s3.eu-central-1.amazonaws.com/logimporter/logimporter-linux-amd64
- Windows 64Bit: https://streamabc-sw.s3.eu-central-1.amazonaws.com/logimporter/logimporter.exe
- MacOS Intel 64Bit: https://streamabc-sw.s3.eu-central-1.amazonaws.com/logimporter/logimporter-mac

All binaries are built static and have no further system requirements and are used as a command line tool.

Logimporter supports several import formats, log format types and output types. The configuration is provided using command line values and flags (or environment variables).

The configuration is divided into 3 parts:

- *input:* defines what input type should be used v
- *parser:* defines what log file type is provided and wich parser should be used

- *output:* defines output of the parsed data and specifies an origin as identifier that you get from Audalaxy

The usage in general:

```
./logimporter input:file -path="./logs/access.log" parser:icecast output:amqp -origin=xxx
```

Hint: The value xxx in the examples as in `-origin=xxx` has to be substituted by the value provided by Audalaxy.

The -help flag shows a help note and can be used for every sub-command (input, parser, output)

```
./logimporter -help
```

38.7.1 Input Plugins for Logfiles

The input plugin is configured by the prefix "input:" followed by the wanted input type.

One of the following input types can be used:

```
input:file -path=/path/to/file
```

The provided file will be read, parsed and sent to Audalaxy. After this, the programs exits. You need to provide a path to an existing file. It is possible to use Gzip compressed files with the suffix .gz.

```
input:fileglob -path=/path/to/files*.log
```

All files that match the provided glob pattern are read, parsed and sent to Audalaxy. After all files are processed, the program exists. It is possible to use Gzip compressed files with the suffix .gz.

```
input:tail -path=/path/to/file
```

Continuously reading of new data in a file. This works like the "tail" command and keeps reading new lines until the program is stopped manually. It supports log rotation if the new file gets the same name. This is the preferred way to use in production environments to get realtime logs.

Additional flags for *input:tail*:

```
-whence=end or start
```

Start reading the file from end or beginning/start of the file. Default is end.

```
-listen=127.0.0.1:8008
```

Opens a HTTP server and binds to to provided address and port that can be used for health checks. In this example a call to http://127.0.0.1:8008 either returns a status 200 if everything is ok or a higher status if something does not work.

```
-polling
```

User polling instead of fsnotify for getting notified of new lines. Preferred is the default fsnotify.

```
-scanheader
```

Can be used to read the field definitions in W3C compatible AIS log files.

Hint: The *Healthcheck* can be use to monitor the input.

38.7.2 Parser Plugins for Streaming-Server Logs

The desired parser can be configured with the prefix "parser:". You need the right parser for the log file type you use.

```
parser:icecast
```

Parser for Icecast logs.

```
parser:ais
```

AIS 7 session log format.

```
-version=ais8
```

AIS 8 and above session format. It also tries to guess the right fields by reading the file header. Please note that the AIS session log has to be used, not the access log.

38.7.3 Output Plugins for data transmission

The output plugin is configured usingthe prefix "output:".

The available plugins are the following:

```
output:amqp
```

Logs are sent as AMQP messages to a Audalaxy message queue. This is the default and should only be changed if requested. You need to open port 5672 for outgoing TCP.

Additional Flags for *output:amqp*:

```
-origin=xx
```

Mandantory field. The value will be provided by Audalaxy and is used to assign incoming log data to the right customer.

```
-streamwatch
```

Optional. Should only be used if requested by Audalaxy.

```
output:noop
```

Test output "dry run". No data is sent and it can be used to check if everything is working.

Additional flags for *output:noop*:

```
-output
```

Output parsed log data to stdout.

```
output:http
```

Send data using HTTPS to an ingest endpoint. Can be used if AMQP is not possible due to port restrictions and is requested by Audalaxy. Please note that this is less reliable as AMQP and can lead to loss of some log data under certain circumstances.

Additional flags for *output:http*:

```
-origin=xx
```

Mandantory field. The value will be provided by Audalaxy and is used to assign incoming log data to the right customer.

```
-streamwatch
```

Optional. Should only be used if requested by Audalaxy.

If "tail" mode is used the program runs indefinitely. In this case it is adviced to use a start-up script that manages the process. For instance on modern Linux systems you can use a systemd unit file.

Example for a SystemD unit file

```
[Unit]
Description=Audalaxy Logimporter
Wants=network-online.target
After=network-online.target

[Service]
ExecReload=/bin/kill -HUP $MAINPID
ExecStart=/usr/local/bin/logimporter-linux-amd64 input:tail -path=/var/log/icecast/
↪access.log parser:icecast output:amqp -origin=xxx
User=icecast
KillMode=process
KillSignal=SIGINT
LimitNOFILE=infinity
LimitNPROC=infinity
Restart=on-failure
RestartSec=2
StartLimitBurst=3
StartLimitIntervalSec=10
TasksMax=infinity

[Install]
WantedBy=multi-user.target
```

The following parameters need to be changed and adapted to your needs:

- Path to the program itself (here /usr/local/bin/logimporter-linux-amd64)
- Path to the log file (here /var/log/icecast/access.log)
- User (here icecast)

Examples for starting the Logimporter

```
./logimporter input:tail -path=/path/to/logfile parser:ais output:amqp -
↪origin=xxx

./logimporter input:file -path=/path/to/logfile parser:ais output:amqp -
↪origin=xxx

./logimporter input:fileglob -path=/path/to/logfiles* parser:ais output:amqp -
↪origin=xxx
```

38.7.4 Re-delivery of missing logs

If some logs have not beend sent in tail mode because of errors or network issues, it is possible to use the normal file or fileglob mode to re-delivery the missing data. Since version v3.1.0 there are new flags to restrict the logs to a specific time range: `-after="YYYY-MM-DD HH:mm:ss` and `-before="YYYY-MM-DD HH:mm:ss` can be provided to restrict to data before and after the given dates.

The flags can be used individually or in combination. Both are global flags and need to go before the input defintion.

If you know the exakt time when for instance an error occured and data is missing, you can use `-after` to send all logs after this date regardless of the timespan the log covers.

If you know the point in time since when everything worked fine again, use this time with `-before`.

The full call could be something like this:

```
logimporter -after="2022-03-04 19:15:25" input:file -path="./logs/access-2022-03-04.log.
↪gz" parser:icecast output:amqp -origin=xxx
```

38.7.5 Healthcheck

To start the HTTP server and enable /health endpoint, use the -listen flag.

Example:

`-listen=127.0.0.1:8080` starts the HTTP server and connects it to IP 127.0.0.1 and port 8080. http://127.0.0.1:8080/health can be used to call the health check. It returns HTTP status 200 if everything is OK. If not enough new log lines were processed via *input plugins for logfiles* in the interval "-alertInterval 300" (default 300s), status 500 is returned.

If the IP is omitted for -listen, the health check uses all IPs that are configured `-listen :8080`

38.8 MetaPort

38.8.1 Delay metadata

In advanced settings you can specify that the incoming metadata should be delayed (in seconds).

```
{
    "metadata_delay": 30
}
```

38.9 MetaSpreader configuration settings

MetaSpreader supports configuring your metadata receivers as JSON in the Console.

Your Console > *Edit Channel* > *Metaflow* > *MetaSpreader*

Console

```
{
    "radioplayer": {
        "stationid": "--old Station ID",
        "email": "-- email of account (required)",
        "apikey": "-- apikey (required)",
        "typ": "radioplayercloud"
    },
    "radio.de": {
        "broadcast": "",
        "apikey": "",
        "typ": "radio.de"
    },
    "jsonpush_1": {
      "channelname": "my audiostream name 1",
      "url": "https://playerwebsite1/metadata-input",
      "typ": "jsonpush"
    },
    "jsonpush_2": {
      "channelname": "my audiostream name 2",
      "url": "https://playerwebsite2/metadata-input",
      "typ": "jsonpush"
    },

}
```

38.9.1 POST request with JSON payload

```
{
    "jsonpush_1": {
        "channelname": "my audiostream name 1",
        "url": "https://playerwebsite1/metadata-input",
        "typ": "jsonpush"
    },
    "jsonpush_2": {
        "channelname": "my audiostream name 2",
        "url": "https://playerwebsite2/metadata-input",
        "typ": "jsonpush"
    },

}
```

Note: The content of POST is like *Metadata data-set reference*

38.9.2 Radioplayer

```
{
    "radioplayer": {
        "stationid": "--old Station ID",
        "email": "-- email of account (required)",
        "apikey": "-- apikey (required)",
        "typ": "radioplayercloud"
    }
}
```

Radioplayer Unique ID:

```
{
    "radioplayer": {
        "rpuid": "--Radioplayer Unique ID",
        "email": "-- email of account (required)",
        "apikey": "-- apikey (required)",
        "typ": "radioplayercloud"
    }
}
```

When using the Radioplayer Unique ID, an additional "territoryid" can be entered if required; by default it is 276 (Germany).

Old:

```
{
    "radioplayer": {
        "url": "https://ingest.radioplayer.de/ingestor/metadata/v1/np/",
        "stationid": "",
        "user": "",
        "pass": "",
```

(continues on next page)

(continued from previous page)

```
        "typ": "radioplayer"
    }
}
```

38.9.3 Radio.de (radio.net)

```
{
    "radio.de": {
        "broadcast": "",
        "apikey": "",
        "typ": "radio.de"
    }
}
```

38.9.4 TuneIn

```
{
    "tunein": {
        "partnerID": "",
        "partnerKey": "",
        "stationID": "",
        "typ": "tunein"
    }
}
```

38.9.5 Cover Brands Are Live

```
{
    "brandsarelive": {
        "id": "1554",
        "default_cover": "https://url.zu.einem.cover",
        "typ": "brandsarelive"
    }
}
```

The name “brandsarelive” is freely definable.

id You get the ID for the cover from Brands Are Live.

default_cover If no cover data is available, the default cover is sent.

typ The type must be “brandsarelive” to send metadata to this provider.

38.9.6 Icecasts/AIS

```
{
    "source_icecast": {
        "icecast": "hostname:port",
            "mount": "/mountname",
            "user": "admin",
            "pass": "",
            "typ": "icecast"
    },
    "source_ais": {
        "icecast": "hostname:port",
        "mount": "/mountname",
        "user": "admin",
        "pass": "",
        "typ": "icecast"
    }
}
```

38.9.7 FTP-Server

```
{
    "myftpserver1": {
        "channelname": "my audiostream name",
        "host": "myftpserver.io:21",
        "user": "",
        "pass": "",
        "template": "myservice.xml.tmpl",
        "filename": "directory/filename.xml",
        "typ": "ftp"
    }
}
```

template
: template describes the content of filename

38.9.8 Amqp

```
{
   "amqp1": {
        "brandid": "",
        "channelkey": "",
        "typ": "amqp"
    }
}
```

Tip: Related topics

- *Metadata data-set reference*
- *MetaPort API Specification*

- *Metaflow*

38.10 MusicMaster Scheduling Workflow

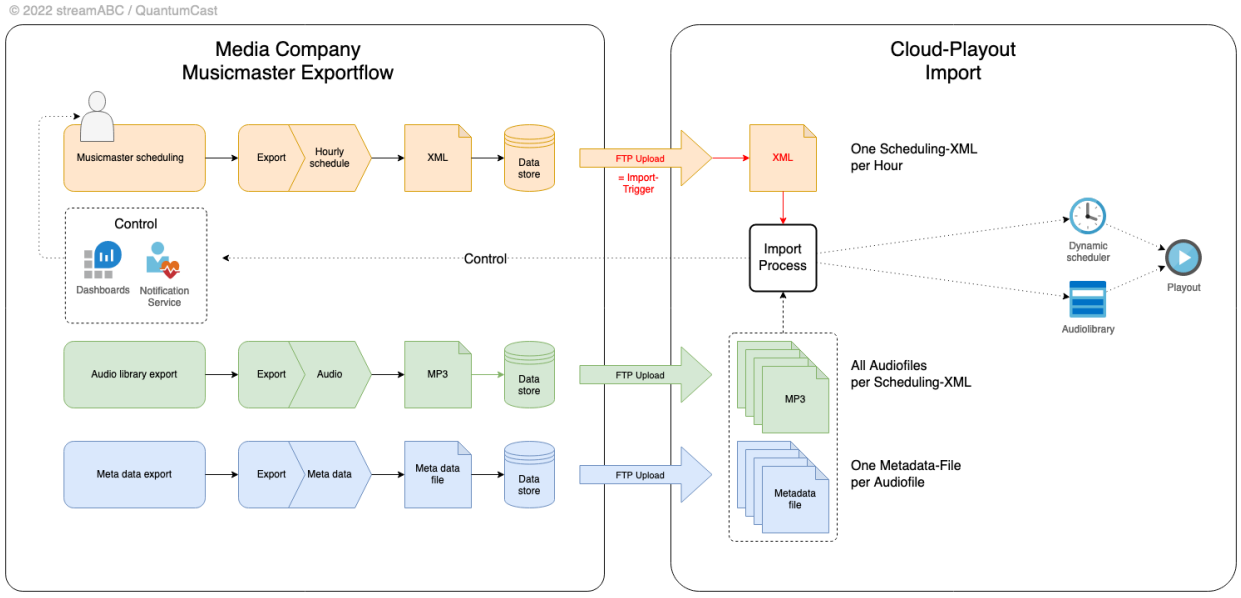
Audalaxy has created APIs that allow media companies to connect their existing software systems (e.g. MusicMaster, Dabis, Zenon) to the cloud playout.

The API can be used under the following conditions:

- the specified XML format can be exported from Musicmaster
- all required fields in the XML are filled in
- the import trigger must be triggered repeatedly if audio elements are missing

38.10.1 Workflow

1. Audio files export
2. Audio meta data export
3. MusicMaster Scheduling export



1. Audio files export:

- Export as MP3
- Upload via FTP or clients that support s3-APIs
- The name of the file has to be unique and match with the ID or ArchivNumber.

2. Meta data export:

- Export the meta data files from the broadcast system and upload via FTP or s3-client.
- Meta data must be linked to audio files by ID or ArchivNumber.

3. MusicMaster Scheduling:

- Export the MusicMaster scheduling as XML
- The XML indicates which audio files are used by referencing the ID or ArchivNumber of the audio files.
- One XML per broadcast hour OR one XML per day
- The targeted channel needs to be enabled by Audalaxy first.
- Reference the Audalaxy channel by providing the channel key as StationName in the XML file.
- Upload via FTP or s3-Client
- The finished upload automatically triggers the import process.
- Audio files will be added to the Audalaxy audio library, scheduled will be transformed to the internal format and pushed to the Audalaxy console.
- The reset of the schedules is set to midnight by default. It is possible to change the reset to any hour - this must be set up by a Audalaxy staff member.

Upload files via FTP or s3-client (rClone etc.) and wait. If the file is in the "processed"-folder, it has been processed. If you want to update the scheduling, audio files or audio metadata, simply re-upload the file again.

Attention:

- All files must have a lower case file extension (.mp3, .xml, ...)
- All audio files used in a MusicMaster scheduling must be present BEFORE, otherwise they are marked as errors and skipped
- For all audio files that are used in a schedule but can not be found are created empty files with the missing ID as name in the "requested"-folder. You can observe this folder and upload missing files. The reference in the "requested"-folder is then deleted.
- Missing audio files are not automatically included in the schedule if they are sent subsequently - the scheduling XML file must be uploaded again.

38.10.2 API

Audio files:

The system works internally with MP3. Therefore, we recommend providing MP3 files in high quality (44,100 Hz sampling rate). In principle, MP2/MUS files can also be provided. But then we convert them internally. To get the best quality, you should upload the files in the correct format.

Meta data export:

- For the audio files, the corresponding metadata is absolutely necessary.
- Two formats are supported: XML and CSV
- The file must have the same name as the audio file (ArchivNumber), but the extension .xml or .csv.
- In addition to the artist and song, the most important information is the duration, CueIn, CueOut, StartNext, FadeIn, FadeOut and Intro.
- There is no automatic calculation of transitions. Exactly the values that are transferred via the metadata file are used.

The XML already contains the names of the fields - the CSV is structured as follows:

```
"ID":        1,
"Name":      2,
"Title":     3,
"Etype":     4,
"CueIn":     5,
"CueOut":    6,
"Intro":     7,
"Outro":     8,
"Mix":       9,
"FadeIn":    10,
"FadeOut":   11,
"DbFadeIn":  12,
"DbFadeOut": 13,
"FadeType":  18,
"Type":      19,
"StartNext": 30,
```

Example XML | Example CSV

Example of a scheduling:

- Schedule files must have the channelkey included in the StationName (additional text is ok): `<Station StationName="rtlb_98i4femse6_uemx">`
- The schedule must contain only one date as AirDate tag (format is MM.DD.YYYY): `<AirDate Date="07.24.2022">`
- Attention! If there is another AirDate with the same date in the same file, it OVERWRITES the other schedule.
- All hours must be specified within the AirDate tag.
- Each element needs an ID tag that references the audio file.
- Each element needs a runtime value (format mm:ss). If the runtime is 00:00, the element is played “on ramp” of the following track.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Station StationName="radio_98i4gehse2_uenx">
    <AirDate Date="07.24.2022">
        <AirHour Hour="00">
            <Element>
                <ID>S100166</ID>
                <Artist>JINGLE</Artist>
                <Title>Dropin - Jingle MANN 05</Title>
                <AirTime>00:00:00</AirTime>
                <Runtime>00:00</Runtime>
                <NonMusic>Y</NonMusic>
                <Ende></Ende>
                <EAN></EAN>
                <ISRC></ISRC>
                <Label></Label>
                <LC></LC>
                <Verlag></Verlag>
                <Komponist></Komponist>
                <Info1></Info1>
```

(continues on next page)

(continued from previous page)

```
            <Info2></Info2>
            <Info3></Info3>
        </Element>
        <Element>
            <ID>33050M</ID>
            <Artist>Spliff</Artist>
            <Title>Carbonara</Title>
            <AirTime>00:00:00</AirTime>
            <Runtime>04:04</Runtime>
            <NonMusic>N</NonMusic>
            <Ende>F</Ende>
            <EAN>0042283987128</EAN>
            <ISRC></ISRC>
            <Label>Polyphon</Label>
            <LC>00310</LC>
            <Verlag>Edition Spliff</Verlag>
            <Komponist>Heil/Klimek/Tykwer</Komponist>
            <Info1></Info1>
            <Info2></Info2>
            <Info3></Info3>
        </Element>
    </AirHour>
    <AirHour Hour="01">
        <Element>
            <ID>S100159</ID>
            <Artist>JINGLE</Artist>
            <Title>Dropin - Jingle FRAU 01</Title>
            <AirTime>01:00:00</AirTime>
            <Runtime>00:00</Runtime>
            <NonMusic>Y</NonMusic>
            <Ende></Ende>
            <EAN></EAN>
            <ISRC></ISRC>
            <Label></Label>
            <LC></LC>
            <Verlag></Verlag>
            <Komponist></Komponist>
            <Info1></Info1>
            <Info2></Info2>
            <Info3></Info3>
        </Element>
        <Element>
            <ID>13572M</ID>
            <Artist>Amanda Lear</Artist>
            <Title>Follow Me</Title>
            <AirTime>01:00:00</AirTime>
            <Runtime>03:38</Runtime>
            <NonMusic>N</NonMusic>
            <Ende>F</Ende>
            <EAN>4013659660033</EAN>
            <ISRC></ISRC>
            <Label>Target</Label>
```

(continues on next page)

(continued from previous page)

```
                <LC>04520</LC>
                <Verlag>Arabella Musikverlag GmbH</Verlag>
                <Komponist>A.Monn, A. Lear</Komponist>
                <Info1></Info1>
                <Info2></Info2>
                <Info3></Info3>
            </Element>
        </AirHour>
        <AirHour Hour="02">
            <Element>
                <ID>S100157</ID>
                <Artist>JINGLE</Artist>
                <Title>Dropin - Jingle FRAU 03</Title>
                <AirTime>02:00:00</AirTime>
                <Runtime>00:00</Runtime>
                <NonMusic>Y</NonMusic>
                <Ende></Ende>
                <EAN></EAN>
                <ISRC></ISRC>
                <Label></Label>
                <LC></LC>
                <Verlag></Verlag>
                <Komponist></Komponist>
                <Info1></Info1>
                <Info2></Info2>
                <Info3></Info3>
            </Element>
            <Element>
                <ID>24162M</ID>
                <Artist>Klingande</Artist>
                <Title>Jubel</Title>
                <AirTime>02:00:00</AirTime>
                <Runtime>03:16</Runtime>
                <NonMusic>N</NonMusic>
                <Ende>C</Ende>
                <EAN>9715800109716</EAN>
                <ISRC>FR9W11310765</ISRC>
                <Label>B1 Recordings</Label>
                <LC>16158</LC>
                <Verlag>EMI Music Publishing France / Klingande Music / Sony ATV</Verlag>
                <Komponist>Cédric Steinmyller , Edgar Catry</Komponist>
                <Info1></Info1>
                <Info2></Info2>
                <Info3></Info3>
            </Element
        </AirHour>
    </AirDate>
</Station>
```

Description Structure of the XML document:

- **ID** (*string* mandatory) - archive number
- **Artist** (*string* mandatory) - artist name

- **Title** (*string* mandatory) - song name
- **AirTime** (*time* highly recommended) - format hh:mm:ss
- **Runtime** (*time* mandatory) - length of the audio item
- **NonMusic** (*bool* highly recommended) - is the audio item music (N) or not (Y)
- **Ende** (*bool*)
- **EAN** (*string*) - EAN code
- **ISRC** (*string* highly recommend) - International Standard Recording Code
- **Label** (*string*) - label name
- **LC** (*string*) - label code
- **Verlag** (*string*) - publisher name
- **Komponist** (*string*) - composer's name
- **Info1** (*string*) - more information about the audio item
- **Info2** (*string*) - more information about the audio item
- **Info3** (*string*) - more information about the audio item

---

38.10.3 More workflows

Automatic uploads:

- Observe the "requested"-folder for new elements and upload them to the server.
- To fill the audio library send schedules and upload the requested audio files.
- The references in the "requested"-folder will be deleted after sucesfull imports.
- Upload the schedule again until no new elements are found.

38.10.4 Troubleshooting

The mixing does not sound good:

- deactivate sound processing
- show CueIn, CueOut, FadIn, FadeOut and StartNext values in the protocol of the console and compare them with the values from the XML.

Gespielt	Länge	Artist	Titel	AlbumTitle	CueIn	fadein	startnext	fadeout	cueOut	Type	Audiofile
2022-09-01 17:05:03	00:03:02	VIZE x Tokio Hotel	White Lies	-	0	0.00	0.99	0.00	182.67	None	
2022-09-01 17:02:01	00:03:01	Jason Derulo & Nuka	Love Not War (The Tampa Beat) (THAT KIND Remix)	-	0.168	0.00	0.03	0.00	181.55	None	

- If these values differ, the transfer was not correct.

CHAPTER

THIRTYNINE

INDICES AND TABLES

- genindex

For more information please visit our company website:

HTTP ROUTING TABLE

/https:

INDEX

F

G

H

I

L

M

N

O

P

S

T

V