ДОКУМЕНТАЦИЯ ПОЛЬЗОВАТЕЛЯ

Программное обеспечение «Радиобокс»

Версия 1.0

Санкт-Петербург
2026

1. Общая информация

1.1. ПО "Радиобокс" предназначено для преобразования исходных аудиопотоков в выходные аудиопотоки различных форматов и раздачи их конечным слушателям.

1.2. Пользователь указывает данные исходного потока и параметры необходимых выходных потоков.

1.3. Полученные выходные потоки могут быть транслированы на широкий круг потребителей (в пределах используемой лицензии).

2. Термины и определения

2.1. Входящий поток - источник потокового аудио, из которого формируются выходные потоки.

2.2. Выходной поток - поток, формируемый системой из входящего потока.

2.3. Mount - имя точки публикации выходного потока.

2.4. Кодек выходного потока - аудиокодек, определяющий формат потока на выходе..

2.5. API-токен - значение Bearer-токена для авторизации пользователя в API, используется для контроля прав доступа.

3. Общая схема работы системы

3.1. Работа с системой строится по следующему сценарию:

3.1.1. Пользователь указывает входящий поток - адрес источника звука, который нужно обрабатывать.

3.1.2. Пользователь настраивает один или несколько выходных потоков для этого источника в необходимых ему форматах и качестве.

3.1.3. После сохранения настроек система автоматически подготавливает выходные потоки, они становятся доступными для прослушивания.

3.1.4. Пользователь распространяет ссылки на потоки, к ним подключаются слушатели и слушают их.

3.1.5. При необходимости пользователь может в любой момент изменить параметры, временно отключить поток или удалить его.

3.2. Таким образом, пользователь настраивает необходимые ему потоки, а система обеспечивает их формирование и трансляцию конечным слушателям.

4. Требования к входящим потокам

4.1. Оптимально — поток MP3, 320 кбит/с, стерео: при настройке эфиров с более низким качеством звук меньше теряется.

4.2. Адрес источника (URL) должен нормально открываться по сети без постоянных обрывов и зависаний.

4.3. Другие форматы, например AAC или WAV, тоже допустимы, но менее оптимальны.

5. Принципы взаимодействия

5.1. Пользователи работают в рамках организаций: у одной организации может быть несколько учётных записей, при этом каждый пользователь управляет только потоками тех организаций, к которым ему предоставлен доступ.
5.2. Все операции выполняются через программный интерфейс (API) посредством обычных HTTP-запросов к серверу системы. В каждом запросе передаётся индивидуальный ключ доступа (API-токен) для авторизации пользователя.
5.3. После сохранения изменений в настройках потоков система приводит внутренние механизмы в соответствие с новыми данными, этот процесс может занимать некоторое время. Поэтому проверять изменения рекомендуется через 1-2 минуты после отправки команд.

6. Доступ и авторизация

6.1. При регистрации организации в системе для неё формируется персональный URL для доступа к API Радиобокс и получения выходных потоков. Это адрес вида `https://<your-company-name>.hostingradio.ru`, в данной инструкции будем называть его базовым URL, и обозначать как `<Base-URL>`.
6.2. Каждый запрос должен содержать HTTP-заголовок Authorization. Формат: `Authorization: Bearer <API-Token>` (один пробел после слова `Bearer`). Регистр Bearer значения не имеет; токен указывается без кавычек.
6.3. Если заголовок авторизации отсутствует или токен неверный — сервер ответит кодом 401 (Unauthorized).

7. Формат ответов API

7.1. Все ответы от сервера приходят в формате JSON. Каждый запрос содержит в ответе поле `status`, отображающее успешность выполнения запроса.
7.2. Пример успешного ответа:
{ "status": "ok", "data": { ... } }
7.3. Пример ответа с ошибкой:
{ "status": "error", "error": "Текст причины" }

8. Описание доступных методов API

8.1. Список доступных методов приведён в таблице:

GET /incoming	Список входящих потоков пользователя
GET /incoming/<id>	Один входящий поток по идентификатору
POST /incoming	Создание входящего потока
PATCH /incoming/<id>	Изменение входящего потока
DELETE /incoming/<id>	Удаление входящего потока
GET /output	Список выходных потоков пользователя

GET /output/<id>	Один выходящий поток по идентификатору
POST /output	Создание выходящего потока
PATCH /output/<id>	Изменение выходящего потока
DELETE /output/<id>	Удаление выходящего потока

8.2. GET /incoming

Возвращает массив входящих потоков, доступных текущему пользователю (по связи пользователь — организация).

curl -sS -H *"Authorization: Bearer <API-Token>" "<Base-URL>/incoming"*

8.3. GET /incoming/<id>

Позволяет получить данные входящего потока по его идентификатору.

curl -sS -H *"Authorization: Bearer <API-Token>" "<Base-URL>/incoming/<id>"*

8.4. POST /incoming

Создаёт входящий поток.

Обязательные поля:

- *companyId* - идентификатор организации;
- *name* - техническое имя потока латинницей;
- *sourceUrl* - URL потока;

Опционально:

- *description* - описание (для пользователей);
- *isEnabled (0/1)* - включен или выключен.

При успешном добавлении возвращает статус *201 Created* с данными добавленного потока в теле ответа.

```
curl -sS -X POST "<Base-URL>/incoming" \
  -H "Authorization: Bearer <API-Token>" \
  -H "Content-Type: application/json" \
  -d '{
    "companyId": <companyId>,
    "name": "my-station",
    "sourceUrl": "https://source.example.com/live.mp3",
    "description": "Основной эфир",
    "isEnabled": 1
  }'
```

8.5. PATCH /incoming/<id>

Частичное обновление данных входящего потока. Допустимые поля в теле: *name, sourceUrl, description, isEnabled* (см. описание полей выше). Для *sourceUrl* выполняется проверка как для URL.

При успешном добавлении возвращает статус *200 OK* с данными изменённого потока в теле ответа.

```
curl -sS -X PATCH "<Base-URL>/incoming/<id>" \
  -H "Authorization: Bearer <API-Token>" \
  -H "Content-Type: application/json" \
  -d '{"name": "my-station-renamed", "isEnabled": 1}'
```

8.6. DELETE /incoming/<id>

Удаляет входящий поток. Каскадно затрагивает связанные выходные потоки. При успешном удалении возвращает статус *200 OK*

```
curl -sS -X DELETE "<Base-URL>/incoming/<id>" \
  -H "Authorization: Bearer <API-Token>"
```

8.7. GET /output

Список выходных потоков, доступных пользователю (через привязку к организации входящего потока).

```
curl -sS -H "Authorization: Bearer <API-Token>" "<Base-URL>/output"
```

8.8. GET /output/<id>

Получение данных выходного потока по его идентификатору.

```
curl -sS -H "Authorization: Bearer <API-Token>" "<Base-URL>/output/<id>"
```

8.9. POST /output

Создание выходного потока на основе входящего с идентификатором *incomingStreamId*. В результате появляется выходной поток под именем *mount*.

Обязательные поля:
incomingStreamId - идентификатор входящего потока;
*moun*t - имя выходного потока, уникальное в пределах организации;
encoder - кодек выходного потока, может принимать одно из следующих значений:

- *hls* - сегментированный выходной поток HLS;
- *libmp3lame* - поток в mp3;
- *libfdk_aac -profile:a aac_he, libfdk_aac -profile:a aac_he_v2, libfdk_aac -cutoff 20000, libfdk_aac* - различные варианты кодирования для потоков aacp.

bitrate - битрейт выходного потока (kbit), для HLS можно не указывать или указать 0, не должен превышать битрейт входящего потока;
samplerate - частота дискретизации выходного потока (kHz);
channels - число каналов выходного потока (1 или 2);

Опционально:
isEnabled (0/1) - включен или выключен.

При успешном добавлении возвращает статус *201 Created* с данными добавленного потока в теле ответа.

```
curl -sS -X POST "<Base-URL>/output" \
  -H "Authorization: Bearer <API-Token>" \
  -H "Content-Type: application/json" \
  -d '{
    "incomingStreamId": <incomingStreamId>,
    "mount": "station128.mp3",
    "encoder": "libmp3lame",
    "bitrate": 128,
    "samplerate": 44100,
    "channels": 2,
    "isEnabled": 1
  }'
```

Пример создания HLS-выхода:

```
curl -sS -X POST "<Base-URL>/output" \
  -H "Authorization: Bearer <API-Token>" \
  -H "Content-Type: application/json" \
  -d '{
    "incomingStreamId": <incomingStreamId>,
    "mount": "station-hls",
    "encoder": "hls",
    "samplerate": 44100,
    "channels": 2,
    "isEnabled": 1
  }'
```

8.10. PATCH /output/<id>

Частичное обновление данных выходного потока. Допустимые поля в теле: *mount, encoder, bitrate, samplerate, channels, isEnabled* (см. описание полей выше).

При успешном добавлении возвращает статус 200 OK с данными изменённого потока в теле ответа.

```
curl -sS -X PATCH "<Base-URL>/output/<id>" \
  -H "Authorization: Bearer <API-Token>" \
  -H "Content-Type: application/json" \
  -d '{"bitrate": 192, "mount": "station192.mp3"}'
```

8.11. DELETE /output/<id>

Удаляет выходной поток. При успешном удалении возвращает статус *200 OK*

```
curl -sS -X DELETE "<Base-URL>/output/<id>" \
  -H "Authorization: Bearer <API-Token>"
```

9. Типовые проблемы и способы их устранения

9.1. Ответ 401 и сообщения про токен

Проверьте: заголовок передан как Authorization: Bearer <токен> (пробел после Bearer), токен передаётся целиком и совпадает со значением в учётной записи.

9.2. Ответ 403 при создании потока или выхода

Для POST /incoming проверьте, что companyId совпадает с организацией, к которой привязан пользователь токена. Для POST /output — что incomingStreamId ссылается на существующий входящий поток, доступный этому пользователю (в той же организации).

9.3. Ответ 404 на GET/PATCH/DELETE по id

Идентификатор мог быть удалён, ошибочно указан или поток относится к другой организации и недоступен текущему токену. Сверьте списки через GET /incoming и GET /output.

9.4. Запись в API успешна, но слушатели не слышат поток

После изменений сервер выполняет синхронизацию конфигурации. Проверьте isEnabled у входа и выхода и доступность sourceUrl. Некорректные числовые поля, пустой mount или неподдерживаемый encoder приводят к исключению валидации — смотрите текст ошибки в теле ответа и логи сервера.