Микширование потоков

- Описание
  - Поддерживаемые протоколы входных потоков
  - Возможности управления выходным потоком
  - Автоматическое создание микшера при публикации потока
  - Схема работы
- REST-вызовы
  - REST-вызовы и статусы ответа
  - Параметры
    - Настройка микшера при создании по REST API
    - Изменение параметров микшера в процессе его работы по REST API
  - Отправка REST-запроса к WCS-серверу
  - Ограничения
- Настройка
  - Настройка автоматического создания микшера при публикации потока
  - Настройка микширования аудио и видео
    - Публикация в аудио-видео микшер потока без одной из составляющих
  - Буферизация выходного потока микшера
  - Управление битрейтом выходного потока микшера
  - Управление кодеком видео выходного потока микшера
  - Управление звуковой дорожкой выходного потока микшера
  - Собственный lossless видеопроцессор для входящих потоков
  - Управление размещением картинок в выходном потоке микшера
    - Картинки сеткой
    - Картинки сеткой с минимальным расстоянием между ними
    - Картинки сеткой с минимальным расстоянием между ними и обрезкой картинок вокруг центра
    - Демонстрация экрана (screen sharing)
    - Картинка в картинке
    - Реализация собственного варианта размещения картинок
      - Обрезка картинок вокруг центра при собственном варианте размещения картинок
      - Отдельный каталог для собственных Java библиотек
    - Управление размещением картинок при создании микшера
  - Управление профилем кодирования выходного потока микшера
  - Управление фоном микшера и добавление водяного знака при создании микшера
  - Динамическое добавление и изменение водяного знака
  - Стерео звук в выходном потоке микшера
  - Декодирование символов в имени входящего потока
- Поддержка MCU
- Управление аудио и видео входного потока
  - Управление аудио и видео входного потока при добавлении его в микшер
- Краткое руководство по тестированию
- Последовательность выполнения операций (Call flow)
- REST hooks для микшера
- Контроль добавления потока в микшер на бэкенд сервере
- Известные проблемы

Описание

WCS позволяет микшировать потоки активных трансляций. Выходной поток микшера может быть записан,воспроизведенилиретранслированпо любой из технологий, поддерживаемых WCS.

Микширование управляется при помощинастроекиREST API.

> При микшировании потоков на сервере включается транскодинг. Рекомендуемая конфигурация сервера: 2 ядра CPU на 1 микшер.

Поддерживаемые протоколы входных потоков

- WebRTC
- RTMP
- RTSP

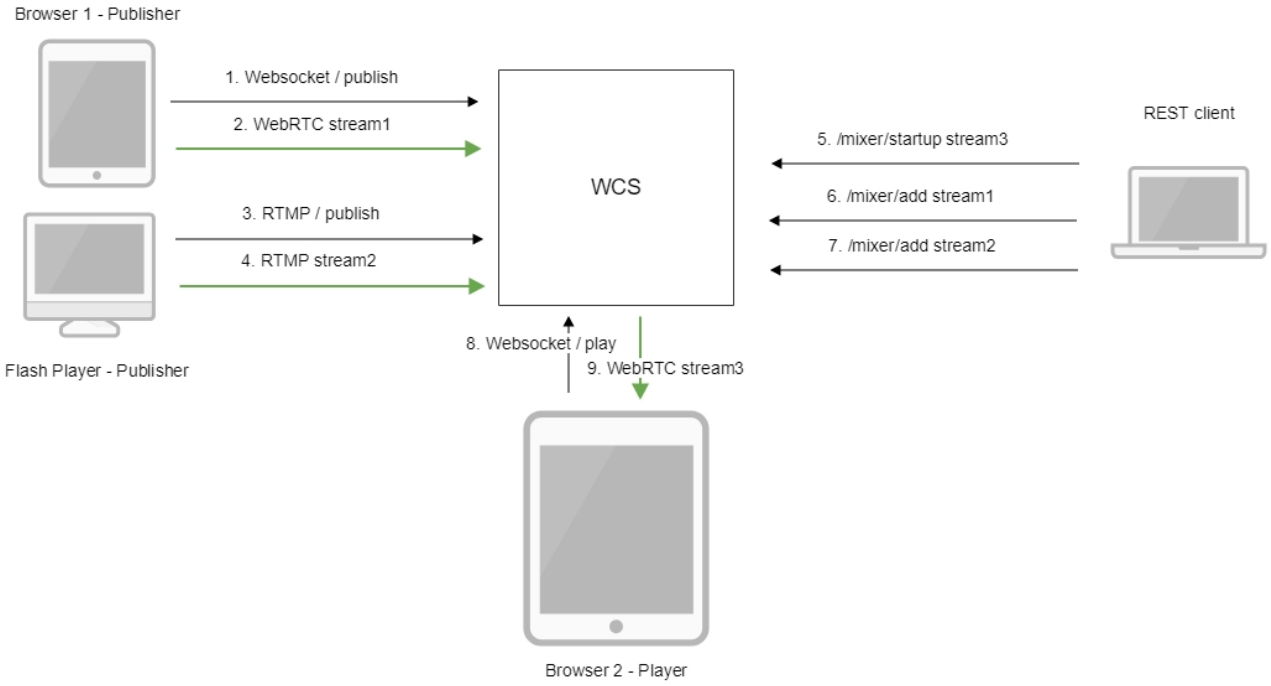
Возможности управления выходным потоком

Микшер позволяет задать размещение видеопотоков в выходном кадре. Поток с определенным именем (по умолчанию desktop) рассматривается, как демонстрация экрана (screen sharing), и размещается в центре кадра:

Автоматическое создание микшера при публикации потока

Если в имени публикуемого RTMP-потока есть символ '#', сервер рассматривает все, что после данного символа, как имя микшера, который будет создан автоматически при публикации потока. Например, для потока user1#room1 будет создан микшер room1, и поток будет в этот микшер добавлен. В имени потока может быть указано и ключевое слово демонстрации экрана, например user1#room1#desktop

Схема работы



1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду publish.
2. Браузер отправляет WebRTC поток stream1 на сервер.
3. Flash Player устанавливает соединение по RTMP и отправляет команду publish.
4. Flash Player отправляет RTMP поток stream2 на сервер.
5. REST-клиент создает микшер с выходным потоком stream3 запросом /mixer/startup
6. REST-клиент добавляет к микшеру поток stream1
7. REST-клиент добавляет к микшеру поток stream2
8. Второй браузер устанавливает соединение по Websocket и отправляет команду play.
9. Второй браузер получает WebRTC аудиопоток stream3 и воспроизводит этот поток на странице.

REST-вызовы

REST-запрос должен быть HTTP/HTTPS POST запросом в таком виде:

- HTTP:http://streaming.flashphoner.com:8081/rest-api/mixer/startup
- HTTPS:https://streaming.flashphoner.com:8444/rest-api/mixer/startup

Здесь:

- streaming.flashphoner.com- адрес WCS-сервера
- 8081 - стандартный REST / HTTP порт WCS-сервера
- 8444- стандартный HTTPS порт
- rest-api- обязательный префикс
- mixer/startup- используемый REST-вызов

REST-вызовы и статусы ответа

REST-метод	Пример тела REST-запроса	Пример ответа	Статусы ответа	Описание

/mixer /startup	{ "uri": "mixer://mixer1", "localStreamName": "stream3", "hasVideo": "true", "hasAudio": "true", "watermark": "watermark.png", "background": "background.png", "mixerLayoutClass": "com.flashphoner.mixerlayout.TestLayout" }		200 - OK 400 - Bad request 409 - Conflict 500 - Internal error	Создает микшер, для которого публикуется поток с указанным именем
/mixer/add	{ "uri": "mixer://mixer1", "remoteStreamName": "stream1", "hasVideo": "true", "hasAudio": "true", "streamLabel": "John Doe", "avatar": "https://mystorage/storage/avatar.png", "videoPositionId": "speaker" }		200 - OK 404 - Mixer not found 404 - Stream not found 500 - Internal error	Добавить RTMP-поток в микшер
/mixer /remove	{ "uri": "mixer://mixer1", "remoteStreamName": "stream1" }		200 - OK 404 - Mixer not found 404 - Stream not found 500 - Internal error	Убрать RTMP-поток из микшера
/mixer /find_all		{ "localMediaSessionId": "ce92b134-2468-4460-8d06-1ea3c5aabace", "remoteMediaSessionId": null, "localStreamName": "mixer1", "remoteStreamName": null, "uri": "mixer://mixer1", "status": "PROCESSED_LOCAL", "mediaSessions": ["95bf2be8-f459-4f62-9a7f-c588f33e0ad3", "693781de-cada-4589-abe1-c3ee55c66901"] }	200 - OK 404 - Not found 500 - Internal error	Найти все микшеры

/mixer /terminate	`{ "uri": "mixer://mixer1" }`		200 - OK 404 - Not found 500 - Internal error	Завершить работу микшера
/stream /startRecording	`{ "mediaSessionId": "23d07fa1-3c74-4d6f-a0de-9b4bf83ce665" }`		200 - OK 404 - Not found 500 - Internal error	Начать запись потока в указанной медиасессии
/stream /stopRecording	`{ "mediaSessionId": "23d07fa1-3c74-4d6f-a0de-9b4bf83ce665" }`		200 - OK 404 - Not found 500 - Internal error	Завершить запись потока в указанной медиасессии
/mixer /setAudioVideo	`{ "uri": "mixer://m1", "streams": "^stream.*", "audioLevel": 0, "videoMuted": true }`		200 - OK 400 - Bad request 404 - Not found 500 - Internal error	Заглушить/возобновить видео или изменить уровень громкости аудио входного потока микшера
/mixer /set_body_watermark	`{ "uri":"mixer://m1", "watermark":"/opt/media/logo.png", "x":10, "y":10, "marginTop":5, "marginLeft":5, "marginBottom":5, "marginRight":5 }`		200 - OK 400 - Bad request 404 - Not found	Добавить водяной знак к картинке выходного потоку микшера
/mixer /set_stream_watermark	`{ "uri":"mixer://m1", "watermark":"/opt/media/logo.png", "mediaSessionId": "030bb470-185c-11ed-9fad-918e05233ae9", "x":10, "y":10, "marginTop":5, "marginLeft":5, "marginBottom":5, "marginRight":5 }`		200 - OK 400 - Bad request 404 - Not found	Добавить водяной знак к картинке одного из входящих потоков микшера

/mixer /set_stream _label	`{ "uri":"mixer://m1", "remoteStreamName":" stream1", "streamLabel": "Mr. John Doe" }`		200 - OK 404 - Not found	Установить или изменить отображаемое имя участника в микшере
/mixer /set_param eter	`{ "uri":"mixer://m1", "mixerLayoutDir":"/opt /GridLayout" }`		200 - OK 400 - Bad request 404 - Not found	Изменить параметр микшера
/mixer /set_stream _avatar	`{ "uri":"mixer://m1", "remoteStreamName":" stream1", "avatar": "https://mystorage /storage/avatar.png" }`		200 - OK 400 - Bad request 404 - Not found 500 - Internal server error	Установить картинку аватара на аудио поток в микшере
/mixer /remove_str eam_avatar	`{ "uri":"mixer://m1", "remoteStreamName":" stream1" }`		200 - OK 404 - Not found 500 - Internal server error	Убрать картинку аватара с аудио потока в микшере
/mixer /set_position	`{ "uri":"mixer://m1", "remoteStreamName":" stream1", "videoPositionId":" speaker" }`		200 - OK 404 - Not found 500 - Internal server error	Переместить картинку потока в указанную позицию (только для собственных вариантов размещения картинок)

Параметры

Имя параметра	Описание	Пример
uri	Уникальный идентификатор микшера	mixer://mixer1
localStreamName	Имя выходного потока микшера	stream3
hasVideo	Микшировать видео	true
hasAudio	Микшировать аудио	true
remoteStreamName	Имя потока, добавляемого в микшер	stream1 rtmp://rtmp.flashphoner.com:1935/live/rtmp_stream1
mediaSessionId	Идентификатор медиасессии	ce92b134-2468-4460-8d06-1ea3c5aabace
status	Статус потока	PROCESSED_LOCAL
background	Фоновое изображение	background.png
watermark	Водяной знак	watermark.png

mixerLayoutClass	Класс размещения картинок в потоке	com.flashphoner.mixerlayout.TestLayout
streams	Список потоков или регулярное выражение для поиска	^stream.* ["stream1", "stream2"]
audioLevel	Уровень громкости аудио для входного потока	0
videoMuted	Заглушить видео	true
streamLabel	Отображаемое имя участника в микшере	John Doe
avatar	URI картинки в форматах PNG, JPG, BMP	https://mystorage.com/storage/avatar.jpg
videoPositionId	Идентификатор позиции в описании варианта размещения картинок	speaker

Настройка микшера при создании по REST API

В сборке5.2.872добавлена возможность передать параметры, аналогичные настройкам микшера вфайлеflashphoner.properties, при создании микшера по запросу /mixer/startup. В этом случае параметры будут применены только к этому экземпляру микшера. Например, запрос

```
{
 "uri": "mixer://mixer1",
 "localStreamName": "stream3",
 "mixerVideoWidth": 640,
 "mixerVideoHeight": 360,
 "mixerVideoFps": 24,
 "mixerVideoBitrateKbps": 500
}
```

создаст микшер с выходным потоком 640x360, fps 24 и битрейтом 500 кбит/с, независимо от того, какие настройки указаны в файлеflashphoner.properties.

Полный список параметров микшера можно получить в ответе на запрос /mixer/find_all, например

Mixer parameters full list

```
[
  {
    "localMediaSessionId": "7c9e5353-8680-4ad2-8a47-1a366091785c",
    "localStreamName": "m1",
    "uri": "mixer://m1",
    "status": "PROCESSED_LOCAL",
    "hasAudio": true,
    "hasVideo": true,
    "record": false,
    "mediaSessions": [],
    "mixerLayoutClass": "com.flashphoner.media.mixer.video.presentation.GridLayout",
    "mixerLayoutDir": "/opt/mixer1-layout",
    "mixerActivityTimerCoolOffPeriod": 1,
    "mixerActivityTimerTimeout": -1,
    "mixerAppName": "defaultApp",
    "mixerAudioOpusFloatCoding": false,
    "mixerAudioSilenceThreshold": -50,
    "mixerAudioThreads": 4,
    "mixerAutoScaleDesktop": true,
    "mixerDebugMode": false,
    "mixerDesktopAlign": "TOP",
    "mixerDisplayStreamName": false,
    "mixerDecodeStreamName": false,
    "mixerFontSize": 20,
    "mixerFontSizeAudioOnly": 40,
    "mixerIdleTimeout": 20000,
    "mixerInBufferingMs": 200,
    "mixerIncomingTimeRateLowerThreshold": 0.95,
    "mixerIncomingTimeRateUpperThreshold": 1.05,
    "mixerMcuAudio": false,
    "mixerMcuVideo": false,
    "mixerMcuMultithreadedMix": false,
    "mixerMinimalFontSize": 1,
```

```
    "mixerMcuMultithreadedDelivery": false,
    "mixerOutBufferEnabled": false,
    "mixerOutBufferInitialSize": 2000,
    "mixerOutBufferStartSize": 150,
    "mixerOutBufferPollingTime": 100,
    "mixerOutBufferMaxBufferingsAllowed": -1,
    "mixerShowSeparateAudioFrame": true,
    "mixerTextAutoscale": true,
    "mixerTextColour": "0xFFFFFF",
    "mixerTextBulkWriteWithBuffer": true,
    "mixerTextBulkWrite": true,
    "mixerTextBackgroundOpacity": 100,
    "mixerTextBackgroundColour": "0x2B2A2B",
    "mixerTextPaddingLeft": 5,
    "mixerVoiceActivitySwitchDelay": 0,
    "mixerVoiceActivityFrameThickness": 6,
    "mixerVoiceActivityFramePositionInner": false,
    "mixerVoiceActivityColour": "0x00CC66",
    "mixerVoiceActivity": true,
    "mixerVideoWidth": 1280,
    "mixerVideoThreads": 4,
    "mixerVideoStableFpsThreshold": 15,
    "mixerVideoQuality": 24,
    "mixerVideoProfileLevel": "42c02a",
    "mixerVideoLayoutDesktopKeyWord": "desktop",
    "mixerVideoHeight": 720,
    "mixerVideoGridLayoutPadding": 30,
    "mixerVideoGridLayoutMiddlePadding": 10,
    "mixerVideoFps": 30,
    "mixerVideoDesktopLayoutPadding": 30,
    "mixerVideoDesktopLayoutInlinePadding": 10,
    "mixerVideoBufferLength": 1000,
    "mixerVideoBitrateKbps": 2000,
    "mixerUseSdpState": true,
    "mixerType": "NATIVE",
    "mixerThreadTimeoutMs": 33,
    "mixerTextPaddingTop": 5,
    "mixerTextPaddingRight": 4,
    "mixerTextFont": "Serif",
    "mixerTextPaddingBottom": 5,
    "mixerTextDisplayRoom": true,
    "mixerTextCutTop": 3,
    "mixerRealtime": true,
    "mixerPruneStreams": false,
    "audioMixerOutputCodec": "alaw",
    "audioMixerOutputSampleRate": 48000,
    "audioMixerMaxDelay": 300,
    "mixerAudioOnlyHeight": 360,
    "mixerAudioOnlyWidth": 640,
    "videoOutputCodec":"vp8",
    "mixerTextAlign": "TOP_CENTER"
  }
]
```


Изменение параметров микшера в процессе его работы по REST API

Начиная со сборки 5.2.1480, следующие параметры можно менять для активного микшера при помощи запроса `/mixer/set_parameter` :

```
[
  {
    ...
    "mixerLayoutClass": "com.flashphoner.media.mixer.video.presentation.GridLayout",
    "mixerAutoScaleDesktop": true,
    "mixerDesktopAlign": "TOP",
    "mixerDisplayStreamName": true,
    "mixerFontSize": 20,
    "mixerFontSizeAudioOnly": 40,
    "mixerMinimalFontSize": 1,
    "mixerShowSeparateAudioFrame": true,
    "mixerTextAutoscale": true,
    "mixerTextColour": "0xFFFFFF",
    "mixerTextBulkWriteWithBuffer": true,
    "mixerTextBulkWrite": true,
    "mixerTextBackgroundOpacity": 100,
    "mixerTextBackgroundColour": "0x2B2A2B",
    "mixerFrameBackgroundColour": "0x2B2A2B",
    "mixerTextPaddingLeft": 5,
    "mixerVoiceActivitySwitchDelay": 0,
    "mixerVoiceActivityFrameThickness": 6,
    "mixerVoiceActivityFramePositionInner": false,
    "mixerVoiceActivityColour": "0x00CC66",
    "mixerVoiceActivity": true,
    "mixerVideoLayoutDesktopKeyWord": "desktop",
    "mixerVideoGridLayoutPadding": 30,
    "mixerVideoGridLayoutMiddlePadding": 10,
    "mixerVideoDesktopLayoutPadding": 30,
    "mixerVideoDesktopLayoutInlinePadding": 10,
    "mixerTextPaddingTop": 5,
    "mixerTextPaddingRight": 4,
    "mixerTextFont": "Serif",
    "mixerTextPaddingBottom": 5,
    "mixerTextDisplayRoom": true,
    "mixerTextCutTop": 3,
    "mixerTextAlign": "BOTTOM_LEFT",
    "mixerVideoDesktopFullscreen": false,
    "mixerLayoutDir": ""
  }
]
```

Если запрос содержит параметры, которые не могут быть изменены для активного микшера, такой запрос вернет `400 Bad request` с указанием неподдерживаемого параметра.

Отправка REST-запроса к WCS-серверу

Для отправки REST-запроса к WCS-серверу необходимо использовать REST-клиент.

Ограничения

При создании микшера при помощи REST API необходимо указать имя его выходного потока (параметр localStreamName). Если этого не сделать, сервер вернет 400Bad request с сообщением "No localStreamName given".

Настройка

Основные настройки микширования задаются при помощи следующих параметров файла настроек flashphoner.properties

Параметр	Значение по умолчанию	Описание
mixer_video_desktop_layout_inline_padding	10	Расстояние между окнами видеопотоков в нижней строке (под окном демонстрации экрана)
mixer_video_desktop_layout_padding	30	Расстояние между окном видеопотока демонстрации экрана и нижней строкой (остальные видеопотоки)

mixer_video_enabled	true	Включает (по умолчанию) или отключает микширование видео
mixer_video_grid_layout_middle_padding	10	Расстояние между окнами видеопотоков в одной строке (при отсутствии демонстрации экрана)
mixer_video_grid_layout_padding	30	Расстояние между строками видеопотоков (при отсутствии демонстрации экрана)
mixer_video_height	720	Высота изображения выходного потока микшера, значение должно быть четным. Нечетное значение будет автоматически уменьшено на единицу: например, при 481 высота микшера будет 480.
mixer_video_layout_desktop_key_word	desktop	Ключевое слово для потока демонстрации экрана (screen sharing)
mixer_video_width	1280	Ширина изображения выходного потока микшера, значение должно быть четным. Нечетное значение будет автоматически уменьшено на единицу: например, при 641 ширина микшера будет 640.
record_mixer_streams	false	Включает или отключает (по умолчанию) запись всех выходных потоков микшера

Настройка автоматического создания микшера при публикации потока

Для того, чтобы включить возможность автоматического создания микшера для потоков, содержащих в имени символ '#', необходимо, чтобы приложение, которое обрабатывает входящие потоки, зарегистрировало обработчик 'com.flashphoner.server.client.handler.wcs4.FlashRoomRecordingStreamingHandler'. Регистрация обработчика производится при помощиинтерфейса командной строки. Например, для приложения flashStreamingApp, используемого для публикации входящих RTMP потоков, это делается командой

```
update app -m com.flashphoner.server.client.handler.wcs4.FlashRoomRecordingStreamingHandler -c com.flashphoner.server.client.handler.wcs4.FlashStreamingCallbackHandler flashStreamingApp
```

Подробнее об управлении приложениями из командной строки WCS-сервера можно узнатьздесь.

Настройка микширования аудио и видео

По умолчанию микшируются видео и аудиопотоки. Если необходимо микшировать только аудио, необходимо указать это при создании микшера

```
{
 "uri": "mixer://mixer1",
 "localStreamName": "stream3",
 "hasVideo": "false"
}
```

Если необходимо отключить микширование видео для всех потоков, необходимо указать параметр в файлеflashphoner.properties

```
mixer_video_enabled=false
```

При этом видео может быть включено для определенного микшера при его создании.

В сборке5.2.689добавлена возможность включить или отключить микширование аудио при создании микшера

```
{
 "uri": "mixer://mixer1",
 "localStreamName": "stream3",
 "hasAudio": "false"
}
```

или в настройках сервера для всех потоков

```
mixer_audio_enabled=false
```

Публикация в аудио-видео микшер потока без одной из составляющих

Если в аудио-видео микшер публикуется поток без видео или без аудио дорожки, необходимо отключать контроль активности аудио и видео

```
rtp_activity_audio=false
rtp_activity_video=false
```

Буферизация выходного потока микшера

В некоторых случаях необходима буферизация выходного потока микшера. Эта функция включается настройкой в файлеflashphoner.properties

```
mixer_out_buffer_enabled=true
```

Размер буфера задается в миллисекундах настройкой

```
mixer_out_buffer_start_size=400
```

В данном случае размер буфера составит 400 мс.

Период отправки данных потока из буфера в миллисекундах указывается настройкой

```
mixer_out_buffer_polling_time=20
```

В данном случае период составляет 20 мс.

Управление битрейтом выходного потока микшера

Если в качестве транскодера используется кодек OpenH264, появляется возможность управлять битрейтом выходного потока микшера при помощи настройкив файлеflashphoner.properties

```
mixer_video_bitrate_kbps=2000
```

По умолчанию, битрейт выходного потока установлен в 2 Мбит/с. При недостаточной пропускной способности канала между сервером и зрителем битрейт может быть понижен, например

```
encoder_priority=OPENH264
mixer_video_bitrate_kbps=1500
```

Если качество картинки с битрейтом по умолчанию низкое, присутствуют искажения, рекомендуется увеличить битрейт выходного потока микшера до 3-5 Мбит/с

```
encoder_priority=OPENH264
mixer_video_bitrate_kbps=5000
```

Управление кодеком видео выходного потока микшера

В сборке 5.2.1075 добавлена возможность указать кодек видео выходного потока микшера. По умолчанию, используется кодек H264

```
video_mixer_output_codec=h264
```

Кодек VP8 можно указать при помощи настройки

```
video_mixer_output_codec=vp8
```

Отметим, что в этом случае ключевые фреймы в потоке идут реже по сравнению с H264, поскольку размер ключевого фрейма существенно больше. Это может привести к тому, что первый кадр видео при проигрывании видео в браузере может отобразиться с задержкой относительно потока H264 с такими же параметрами.

Управление звуковой дорожкой выходного потока микшера

По умолчанию, звуковая дорожка в выходном потоке микшера кодируется в Opus с частотой дискретизации 48 кГц. Эти параметры могут быть изменены при помощи настроекв файлеflashphoner.properties. Например, для использования выходного потока в SIP, можно установить следующие значения:

```
audio_mixer_output_codec=pcma
audio_mixer_output_sample_rate=8000
```

В этом случае звук будет кодироваться в PCMA (alaw) с частотой дискретизации 8 кГц.

Собственный lossless видеопроцессор для входящих потоков

Для обработки входящих потоков микшера, например, если необходима дополнительная буферизация или синхронизация аудио и видео дорожек, может быть использован собственный lossless видеопроцессор. Этот видеопроцессор включается настройкой в файлеflashphoner.properties

```
mixer_lossless_video_processor_enabled=true
```

Максимальный размер буфера микшера в миллисекундах устанавливается настройкой

```
mixer_lossless_video_processor_max_mixer_buffer_size_ms=200
```

По умолчанию, размер буфера микшера составляет 200 мс. При его заполнении, видеопроцессор использует свой собственный буфер и ожидает освобождения буфера микшера. Периодичность проверки буфера микшера устанавливается в миллисекундах настройкой

```
mixer_lossless_video_processor_wait_time_ms=20
```

По умолчанию, периодичность проверки составляет 20 мс.

Необходимо отметить, что использование lossless видеопроцессора может вносить задержку в трансляции в реальном времени.

При использовании lossless видеопроцессора, чтобы освободить ресурсы, занятые микшером, нужно принудительно остановить микшер при помощи REST запроса /mixer/terminate, либо остановить все входящие в микшер потоки, в этом случае микшер остановится по истечении времени, заданного в миллисекундах настройкой

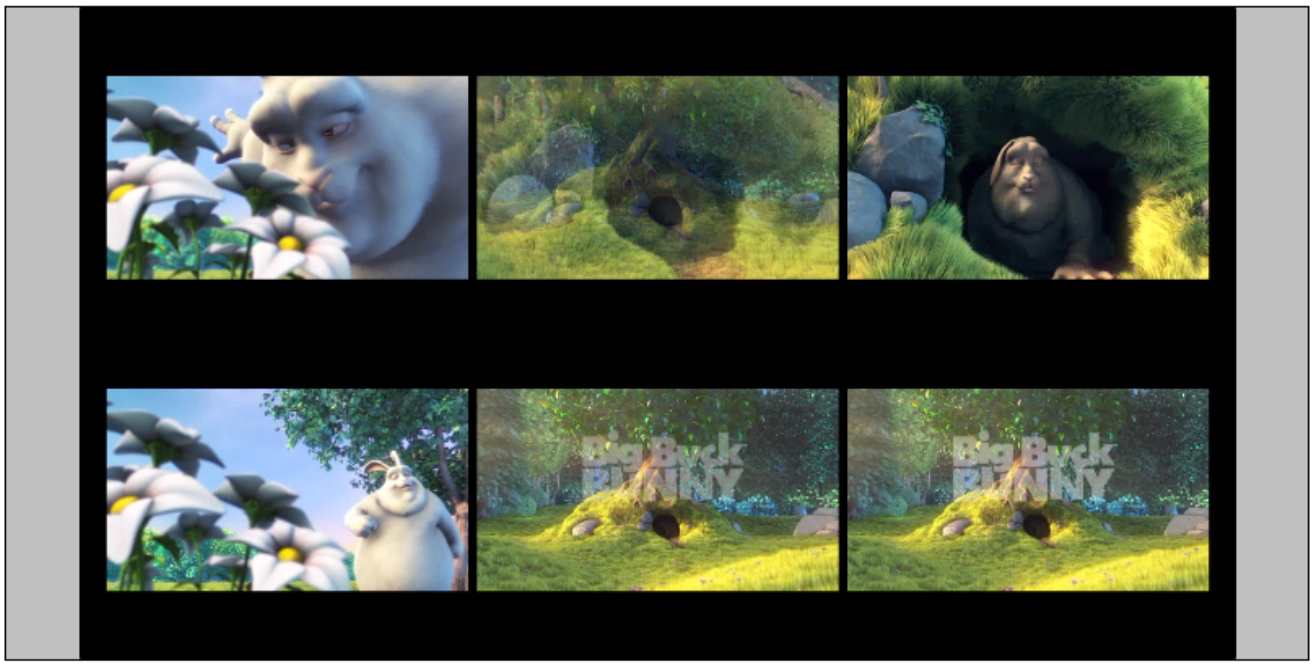
```
mixer_idle_timeout=60000
```

По умолчанию, при отсутствии входящих потоков, микшер останавливается через 60 секунд.

Управление размещением картинок в выходном потоке микшера

По умолчанию, микшер предусматривает три варианта размещения картинок входных потоков в выходном потоке:

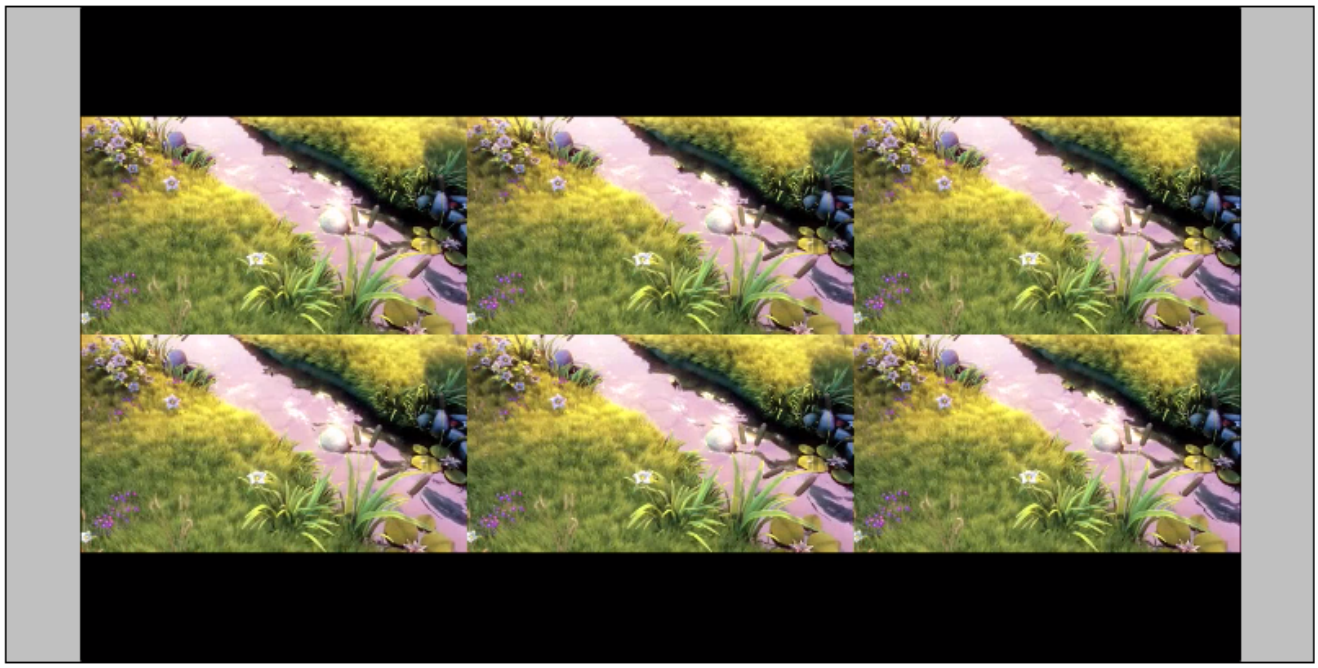
Картинки сеткой



Этот вариант включается настройкойв файлеflashphoner.properties

```
mixer_layout_class=com.flashphoner.media.mixer.video.presentation.GridLayout
```

Картинки сеткой с минимальным расстоянием между ними



Этот вариант включается настройкой

```
mixer_layout_class=com.flashphoner.media.mixer.video.presentation.CenterNoPaddingGridLayout
```

и обеспечивается только для входных потоков одинакового разрешения, с одинаковым соотношением сторон.

Картинки сеткой с минимальным расстоянием между ними и обрезкой картинок вокруг центра

В сборке5.2.842добавлена возможность обрезки картинок с тем, чтобы в микшере был виден только центр картинки. Это может быть полезно при видеоконференции, когда лицо участника находится в центре картинки.

Этот вариант включается настройкой

```
mixer_layout_class=com.flashphoner.media.mixer.video.presentation.CropNoPaddingGridLayout
```

Демонстрация экрана (screen sharing)

Этот вариант включается, если на вход микшеру подается поток с именем, содержащим ключевое слово, указанное в настройке

```
mixer_video_layout_desktop_key_word=desktop
```

По умолчанию, для потока демонстрации экрана имя должно включать слово `desktop` , например `user1_desktop`

В сборке5.2.710добавлена возможность настройки размещения картинки экрана (screen sharing):

```
mixer_desktop_align=TOP
```

По умолчанию, картинка экрана размещается над остальными картинками

Поддерживаются следующие варианты размещения:

Вариант	Описание
TOP	Экран вверху
LEFT	Экран слева
RIGHT	Экран справа
BOTTOM	Экран внизу
CENTER	Экран в центре (картинки потоков вокруг)

Например, при настройке

```
mixer_desktop_align=RIGHT
```

экран будет помещен справа от остальных картинок

Если опубликовано несколько потоков с экрана, главным потоком будет выбран тот, который опубликован первым. Если затем этот поток будет завершен, его место займет поток, следующий по алфавиту.

Картинка в картинке

Этот вариант размещения добавлен в сборке 5.2.852. В этом случае поток с ключевым слово в имени, указанным в настройке

```
mixer_video_layout_desktop_key_word=desktop
```

например, `user1_desktop` , будет фоном для картинок остальных потоков. Такое размещение картинок включается при помощи настройки

```
mixer_video_desktop_fullscreen=true
```

Реализация собственного варианта размещения картинок

Для более тонкой настройки размещения картинок в выходном потоке микшера необходимо разработать класс на языке Java, реализующий интерфейсIVideoMixerLayout, например

TestLayout.java

```
// Package name should be strictly defined as com.flashphoner.mixerlayout
package com.flashphoner.mixerlayout;

// Import mixer layout interface
import com.flashphoner.sdk.media.IVideoMixerLayout;
// Import YUV frame description
import com.flashphoner.sdk.media.YUVFrame;

// Import Java packages to use
import java.awt.*;
import java.util.ArrayList;

/**
 * Custom mixer layout implementation example
 */
public class TestLayout implements IVideoMixerLayout {

    // Pictures padding, should be even (or zero if no padding needed)
    private static final int PADDING = 4;

    /**
     * Function to compute layout, will be called by mixer before encoding output stream picture
     * This example computes grid layout
     * @param yuvFrames - incoming streams raw pictures array in YUV format
     * @param strings - incoming streams names array
     * @param canvasWidth - mixer output picture canwas width
     * @param canvasHeight - mixer output picture canwas heigth
     * @return array of pictures layouts
     */
    @Override
    public Layout[] computeLayout(YUVFrame[] yuvFrames, String[] strings, int canvasWidth, int canvasHeight) {
        // Declare picture layouts list to fill
        ArrayList<IVideoMixerLayout.Layout> layout = new ArrayList<>();

        // Find canvas center height
        int canvasCenter = canvasHeight / 2;
        // Find frame center
        int frameCenter = canvasCenter - (canvasHeight / yuvFrames.length) / 2;

        // Find every picture dimensions (this are the same for all the pictures because this is grid layout)
        int layoutWidth = canvasWidth / yuvFrames.length - PADDING;
        int layoutHeight = canvasHeight / yuvFrames.length;

        // Iterate through incoming stream pictures array
        // Note: streams pictures order corresponds to stream names array, use this to reorder pictures if
needed
        for (int c = 0; c < yuvFrames.length; c++) {
            // Use Java AWT Point and Dimension
            Point prevPoint = new Point();
            Dimension prevDimension = new Dimension();
            if (layout.size() > 0) {
                // Find previous picture location to calculate next one
                prevPoint.setLocation(layout.get(c - 1).getPoint());
                prevDimension.setSize(layout.get(c - 1).getDimension());
            }
            // Set starting point of the picture
            Point currentPoint = new Point((int) (prevPoint.getX() + prevDimension.getWidth() + PADDING),
                    frameCenter);
            // Create the picture layout passing starting point, dimensions and raw picture YUV frames
            layout.add(new IVideoMixerLayout.Layout(currentPoint, new Dimension(layoutWidth,
                    layoutHeight), yuvFrames[c]));
        }
        // Return pictures layouts calculated as array
        return layout.toArray(new IVideoMixerLayout.Layout[layout.size()]);
    }
}
```

Для того, чтобы собственный класс поддерживал размещение текста над или под картинкой потока, начиная со сборки 5.2.878необходимо использовать класс Box для расчета расположения картинок, например

TestLayoutWithBox.java

```
// Package name should be strictly defined as com.flashphoner.mixerlayout
package com.flashphoner.mixerlayout;

// Import mixer layout interface
import com.flashphoner.media.mixer.video.presentation.BoxPosition;
import com.flashphoner.sdk.media.IVideoMixerLayout;
// Import YUV frame description
import com.flashphoner.sdk.media.YUVFrame;
// Import Box class for picture operations
import com.flashphoner.media.mixer.video.presentation.Box;
// Uncomment this if using build 5.2.878-5.2.976
// import com.flashphoner.server.commons.rmi.data.impl.MixerConfig;

// Import Java packages to use
import java.awt.*;
import java.util.ArrayList;

/**
 * Custom mixer layout implementation example
 */
public class TestLayout implements IVideoMixerLayout {

    // Pictures padding, should be even (or zero if no padding needed)
    private static final int PADDING = 4;

    /**
     * Function to compute layout, will be called by mixer before encoding output stream picture
     * This example computes one-line grid layout
     * @param yuvFrames - incoming streams raw pictures array in YUV format
     * @param strings - incoming streams names array
     * @param canvasWidth - mixer output picture canwas width
     * @param canvasHeight - mixer output picture canwas heigth
     * @return array of pictures layouts
     */
    @Override
    public Layout[] computeLayout(YUVFrame[] yuvFrames, String[] strings, int canvasWidth, int canvasHeight) {
        // This object represents mixer canvas
        Box mainBox = new Box(null, canvasWidth, canvasHeight);

        // Find every picture dimensions (this are the same for all the pictures because this is grid layout)
        int frameBoxWidth = canvasWidth / yuvFrames.length - PADDING;
        int frameBoxHeight = canvasHeight / yuvFrames.length;

        // Container to place stream pictures
        Box container = new Box(mainBox, canvasWidth, frameBoxHeight);
        container.setPosition(BoxPosition.CENTER);

        // Iterate through incoming stream pictures array
        // Note: streams pictures order corresponds to stream names array, use this to reorder pictures if
needed
        boolean firstFrame = true;
        for (int c = 0; c < yuvFrames.length; c++) {
            // Stream picture parent rectangle
            // This rectangle can include stream caption if mixer_text_outside_frame setting is enabled
            Box frameBox = new Box(container, frameBoxWidth, (int)container.getSize().getHeight());
            frameBox.setPosition(BoxPosition.INLINE_HORIZONTAL_CENTER);
            // Add padding for subsequent pictures
            if (!firstFrame) {
                frameBox.setPaddingLeft(PADDING);
            }
            firstFrame = false;
            // Compute picture frame placement including stream caption text
            Box frame = Box.computeBoxWithFrame(frameBox, yuvFrames[c]);
            // Adjust picture to the parent rectangle
```

```
            frame.fillParent();

        }
        // Prepare an array to return layout calculated
        ArrayList<IVideoMixerLayout.Layout> layout = new ArrayList<>();
        // Calculate mixer layout
        mainBox.computeLayout(layout);
        // Return the result
        return layout.toArray(new IVideoMixerLayout.Layout[layout.size()]);
    }

    /**
     * The function for internal use.
     * Uncomment this if using build 5.2.878-5.2.976
     */
    //@Override
    //public void setConfig(MixerConfig mixerConfig) {
    //}
}
```

Основные методы класса Box:

Box methods available

```
    /**
     * Set paddings
     * @param padding padding value to set
     **/
    public void setPaddingLeft(int paddingLeft);

    public void setPaddingRight(int paddingRight);

    public void setPaddingTop(int paddingTop);

    public void setPaddingBottom(int paddingBottom);

    /**
     * Set position
     * @param position box position to set
     **/
    public void setPosition(BoxPosition position);

    /**
     * Compute location and size of this box and all of it's content
     * @param layouts aggregator collection to put computed layouts to
     */
    public void computeLayout(ArrayList<IVideoMixerLayout.Layout> layouts);

    /**
     * Compute box location inside the parent Box including caption text placement
     * @param parent parent box
     * @param yuvFrame frame to compute location
     * @return frame with location computed
     */
    public static Box computeBoxWithFrame(Box parent, YUVFrame yuvFrame);

    /**
     * Scale box to the size closest to size of the parent, preserving aspect ratio
     */
    public void fillParent();

    /**
     * Sets box size without saving aspect ratio, crops image from center
     */
    public void fillParentNoScale();
```

Возможные позиции для размещения объекта класса Box:

BoxPosition

```
public enum BoxPosition {
    //attach box to the left
    LEFT,

    //attach box to the right
    RIGHT,

    //attach box to the top
    TOP,

    //attach box to the bottom
    BOTTOM,

    //center box in the parent box
    CENTER,

    //bottom and center box in the parent box
    BOTTOM_CENTER,

    //top and center box in the parent box
    TOP_CENTER,

    //attach box to the right-top corner of left adjoining box (in the same parent box)
    INLINE_HORIZONTAL,

    //same as INLINE_HORIZONTAL but additionally adds vertical CENTER
    INLINE_HORIZONTAL_CENTER,

    //attach box to the left-bottom corner of upper adjoining box (in the same parent box)
    INLINE_VERTICAL,

    //same as INLINE_VERTICAL but additionally adds horizontal CENTER
    INLINE_VERTICAL_CENTER
}
```

Затем следует скомпилировать класс в байт-код. Для этого создаем дерево каталогов, соответствующее названию пакета написанного класса

```
mkdir -p com/flashphoner/mixerlayout
```

и выполняем команду

```
javac -cp /usr/local/FlashphonerWebCallServer/lib/wcs-core.jar ./com/flashphoner/mixerlayout/TestLayout.java
```

Теперь упакуем скомпилированный код в jar-файл

```
jar -cf testlayout.jar ./com/flashphoner/mixerlayout/TestLayout.class
```

и скопируем его в каталог, где размещены библиотеки WCS сервера

```
cp testlayout.jar /usr/local/FlashphonerWebCallServer/lib
```

Для того, чтобы использовать разработанный класс, необходимо указать его в настройкев файлеflashphoner.properties

```
mixer_layout_class=com.flashphoner.mixerlayout.TestLayout
```

и перезапустить WCS.

Выходной поток микшера для приведенного примера и трех входящих потоков будет выглядеть так:

Обрезка картинок вокруг центра при собственном варианте размещения картинок

Для обрезки картинок с выделением центральной части, аналогично CropNoPaddingGridLayout,при создании собственного варианта размещения картинок, необходимо вместо метода Box.fillParent() использовать Box.fillParentNoScale(). Например, вариант размещения картинок по имени с обрезкой вокруг центра:

SideBySideLayout.java

```
package com.flashphoner.mixerlayout;

// Import mixer layout interface
import com.flashphoner.media.mixer.video.presentation.BoxPosition;
import com.flashphoner.sdk.media.IVideoMixerLayout;
// Import YUV frame description
import com.flashphoner.sdk.media.YUVFrame;
// Import Box class for picture operations
import com.flashphoner.media.mixer.video.presentation.Box;
// Uncomment this if using build 5.2.878-5.2.976
// import com.flashphoner.server.commons.rmi.data.impl.MixerConfig;

// Import Java packages to use
import java.awt.*;
import java.util.ArrayList;

public class SideBySideLayout implements IVideoMixerLayout {
    // Owner's stream name
    private static final String USERFOR = "user_for";
    // Challenger's stream name
    private static final String USERAGAINST = "user_against";

    /**
     * Function to compute layout, will be called by mixer before encoding output stream picture
     * This example computes grid layout
     * @param yuvFrames - incoming streams raw pictures array in YUV format
     * @param strings - incoming streams names array
     * @param canvasWidth - mixer output picture canwas width
     * @param canvasHeight - mixer output picture canwas heigth
     * @return array of pictures layouts
     */
    @Override
    public Layout[] computeLayout(YUVFrame[] yuvFrames, String[] strings, int canvasWidth, int canvasHeight) {
        // This object represents mixer canvas
```

```
        Box mainBox = new Box(null, canvasWidth, canvasHeight);

        // Container to place CHALLENGER stream pictures
        Box userForContainer = new Box(mainBox, canvasWidth / 2, canvasHeight);
        userForContainer.setPosition(BoxPosition.LEFT);
        // Container to place OWNER stream pictures
        Box userAgainstContainer = new Box(mainBox, canvasWidth / 2, canvasHeight);
        userAgainstContainer.setPosition(BoxPosition.RIGHT);

        // Iterate through incoming stream pictures array
        for (int c = 0; c < yuvFrames.length; c++) {
            String name = strings[c];
            Box container;

            // Chhose container depending on stream name
            if (name.contains(USERFOR)) {
                container = userForContainer;
            } else if (name.contains(USERAGAINST)) {
                container = userAgainstContainer;
            } else {
                // Wrong stream name
                continue;
            }
            // Fill the container by the stream picture
            Box frameBox = Box.computeBoxWithFrame(container, yuvFrames[c]);
            frameBox.fillParentNoScale();
        }
        // Prepare an array to return layout calculated
        ArrayList<IVideoMixerLayout.Layout> layout = new ArrayList<>();
        // Calculate mixer layout
        mainBox.computeLayout(layout);
        // Return the result
        return layout.toArray(new IVideoMixerLayout.Layout[layout.size()]);
    }

    /**
     * The function for internal use.
     * Uncomment this if using build 5.2.878-5.2.976
     */
    //@Override
    //public void setConfig(MixerConfig mixerConfig) {
    //}
}
```

Отдельный каталог для собственных Java библиотек

Начиная со сборки 5.2.1512, Java библиотеки (jar файлы) собственных вариантов размещения картинок должны помещаться в каталог `/usr/local/FlashphonerWebCallServer/lib/custom`

```
cp testlayout.jar /usr/local/FlashphonerWebCallServer/lib/custom
```

Этот каталог сохраняется при дальнейших обновлениях сервера к более новым сборкам. Таким образом, нет необходимости снова копировать jar файлы после установки обновления.

Управление размещением картинок при создании микшера

В сборке5.2.693появилась возможность указать размещение картинок при создании микшера по REST API, например

```
{
 "uri": "mixer://mixer1",
 "localStreamName": "mixer1",
 "mixerLayoutClass": "com.flashphoner.mixerlayout.TestLayout"
}
```

Таким образом, размещением картинок можно управлять для каждого микшера отдельно

Управление профилем кодирования выходного потока микшера

Некоторые браузеры не поддерживают воспроизведение H264 потока, закодированного с определенным профилем. В связи с этим, в сборке5.2.414добавлена настройка для указания профиля кодирования выходного потока микшера

```
mixer_video_profile_level=42c02a
```

По умолчанию, задан профильconstrainted baseline level 4.2.

Управление фоном микшера и добавление водяного знака при создании микшера

В сборке5.2.693появилась возможность указать фон микшера и добавить водяной знак при создании микшера по REST API, например

```
{
 "uri": "mixer://mixer1",
 "localStreamName": "mixer1",
 "watermark": "watermark.png",
 "background": "background.png"
}
```

По умолчанию, файлы должны располагаться в каталоге /usr/local/FlashphonerWebCallServer/conf. Можно указать и полный путь к файлам, например

```
{
 "uri": "mixer://mixer1",
 "localStreamName": "mixer1",
 "watermark": "/opt/media/watermark.png",
 "background": "/opt/media/background.png"
}
```

Динамическое добавление и изменение водяного знака

В сборке5.2.1349добавлена возможность динамически добавлять и изменять водяной знак, не останавливая микшер и не публикуя поток заново. Водяной знак может быть добавлен, изменен или перемещен в соответствии с указанными координатами при помощи REST API запросов:

- `/mixer/set_body_watermark` - к картинке всего выходного потока микшера

```
{
 "uri":"mixer://m1",
 "watermark":"/opt/media/logo.png",
 "x":0,
 "y":0,
 "marginTop":0,
 "marginLeft":0,
 "marginBottom":0,
 "marginRight":0
}
```

- `/mixer/set_stream_watermark` - к картинке одного из входных потоков микшера

```
{
 "uri":"mixer://m1",
 "watermark":"/opt/media/logo.png",
 "mediaSessionId": "030bb470-185c-11ed-9fad-918e05233ae9",
 "x":0,
 "y":0,
 "marginTop":0,
 "marginLeft":0,
 "marginBottom":0,
 "marginRight":0
}
```

Здесь

- watermark - имя файла водяного знака
- x, y - координаты верхнего левого угла водяного знака на картинке потока
- marginTop, marginLeft, marginBottom, marginRignt - отступы от границ картинки потока

Если координаты выходят за границы картинки потока, водяной знак будет вписан в эти границы с учетом отступов.

Для того, чтобы переместить водяной знак в другое место на картинке, необходимо отправить запрос с тем же файлом, но новыми координатами. Чтобы убрать водяной знак с картинки, необходимо отправить запрос с пустым полем`watermark`

```
{
 "uri":"mixer://m1",
 "watermark":""
}
```

Стерео звук в выходном потоке микшера

По умолчанию, микшер преобразует стерео аудио из входящих потоков в моно, чтобы снизить количество обрабатываемых данных. Это позволяет уменьшить возможные задержки при использовании микшеров реального времени для видеоконференций.

При необходимости, микшер может быть переключен в режим обработки стерео звука, например, в случае онлайн-радиостанции для трансляции музыки. Для этого в сборке5.2.922добавлена настройка, позволяющая задать количество аудио каналов микшера

```
audio_mixer_output_channels=2
```

Декодирование символов в имени входящего потока

В сборке 5.2.1802добавлена возможность декодирования символов в имени потока, закодированных на стороне клиента при помощи encodeURIComponent(). Эта возможность включается настройкой

```
mixer_decode_stream_name=true
```

или добавлением параметра при создании микшера запросом `/mixer/startup`

```
{
  "uri": "mixer://mixer1",
  "localStreamName": "mixer1",
  ...,
  "mixerDecodeStreamName": true
}
```

В этом случае в изображении будут отображаться декодированные символы, если такие символы есть в используемом шрифте, или близкие к ним по начертанию.

Поддержка MCU

Поддержка MCU в микшере включается для аудио настройкой

```
mixer_mcu_audio=true
```

В этом случае для трех входящих потоков stream1, stream2, stream3 на выходе микшера mixer1 будут следующие потоки:

Output stream name	stream1		stream2		stream3	
	audio	video	audio	video	audio	video
mixer1	+	+	+	+	+	+
mixer1-stream1	-	-	+	-	+	-
mixer1-stream2	+	-	-	-	+	-
mixer1-stream3	+	-	+	-	-	-

Таким образом, каждый из дополнительных потоков содержит аудио всех потоков в микшере, кроме одного. Это позволяет, например, устранить эхо для участников конференции.

Аналогичная возможность для видео включается настройкой

```
mixer_mcu_video=true
```

В этом случае для трех входящих потоков stream1, stream2, stream3 на выходе микшера mixer1 будут следующие потоки:

Output stream name	stream1		stream2		stream3	
	audio	video	audio	video	audio	video
mixer1	+	+	+	+	+	+
mixer1-stream1	-	-	+	+	+	+
mixer1-stream2	+	+	-	-	+	+

mixer1-stream3	+	+	+	+	-	-

Таким образом, каждый из дополнительных потоков содержит аудио и видео всех потоков в микшере, кроме одного. Это позволяет организовать полноценную чат-комнату на базе микшера.

При этом, если включена запись микшера при помощи настройки

```
record_mixer_streams=true
```

будет записываться только основной выходной поток микшера (mixer1 в вышеприведенном примере).

При включении MCU для видео, потери на канале могут сказываться на качестве выходного потока микшера. Для улучшения качества видео может быть использовансобственный lossless видеопроцессор, что, в свою очередь, может внести дополнительную задержку.

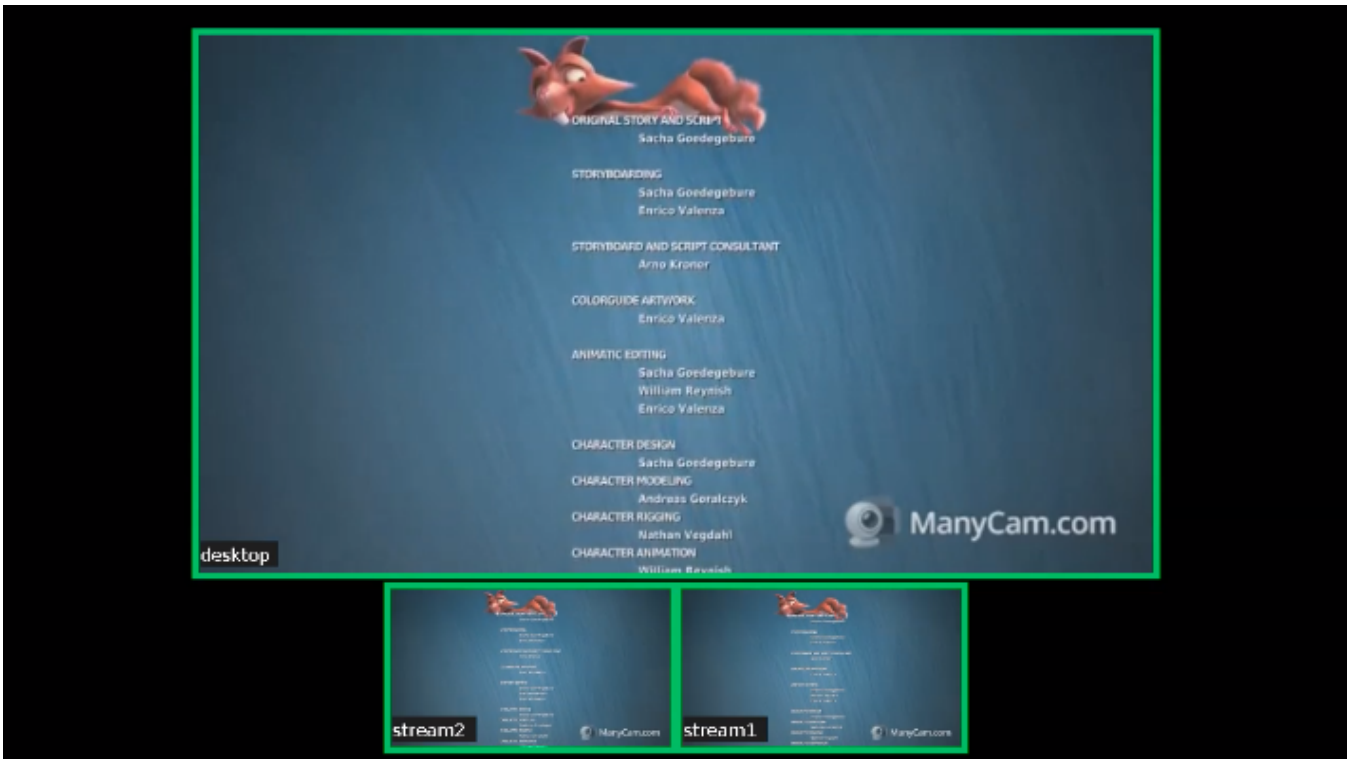
Управление аудио и видео входного потока

В сборке5.2.835добавлена возможность управлять уровнем аудио и заглушать видео для входных потоков в микшере. Оригинальный поток при этом не изменяется. Видео может быть заглушено (черный экран) и затем восстановлено. Для аудио может быть указан уровень громкости, либо звук может быть заглушен снижением уровня до 0.

Входные потоки управляются при помощи REST API запроса /mixer/setAudioVideo.

Например, создаем микшер и добавляем туда три потока: два потока участников и один поток ведущего

```
curl -H "Content-Type: application/json" -X POST http://localhost:8081/rest-api/mixer/startup -d '{"uri":
"mixer://m1", "localStreamName":"m1"}'
curl -H "Content-Type: application/json" -X POST http://localhost:8081/rest-api/mixer/add -d '{"uri":
"mixer://m1", "remoteStreamName": "stream1"}'
curl -H "Content-Type: application/json" -X POST http://localhost:8081/rest-api/mixer/add -d '{"uri":
"mixer://m1", "remoteStreamName": "stream2"}'
curl -H "Content-Type: application/json" -X POST http://localhost:8081/rest-api/mixer/add -d '{"uri":
"mixer://m1", "remoteStreamName": "desktop"}'
```



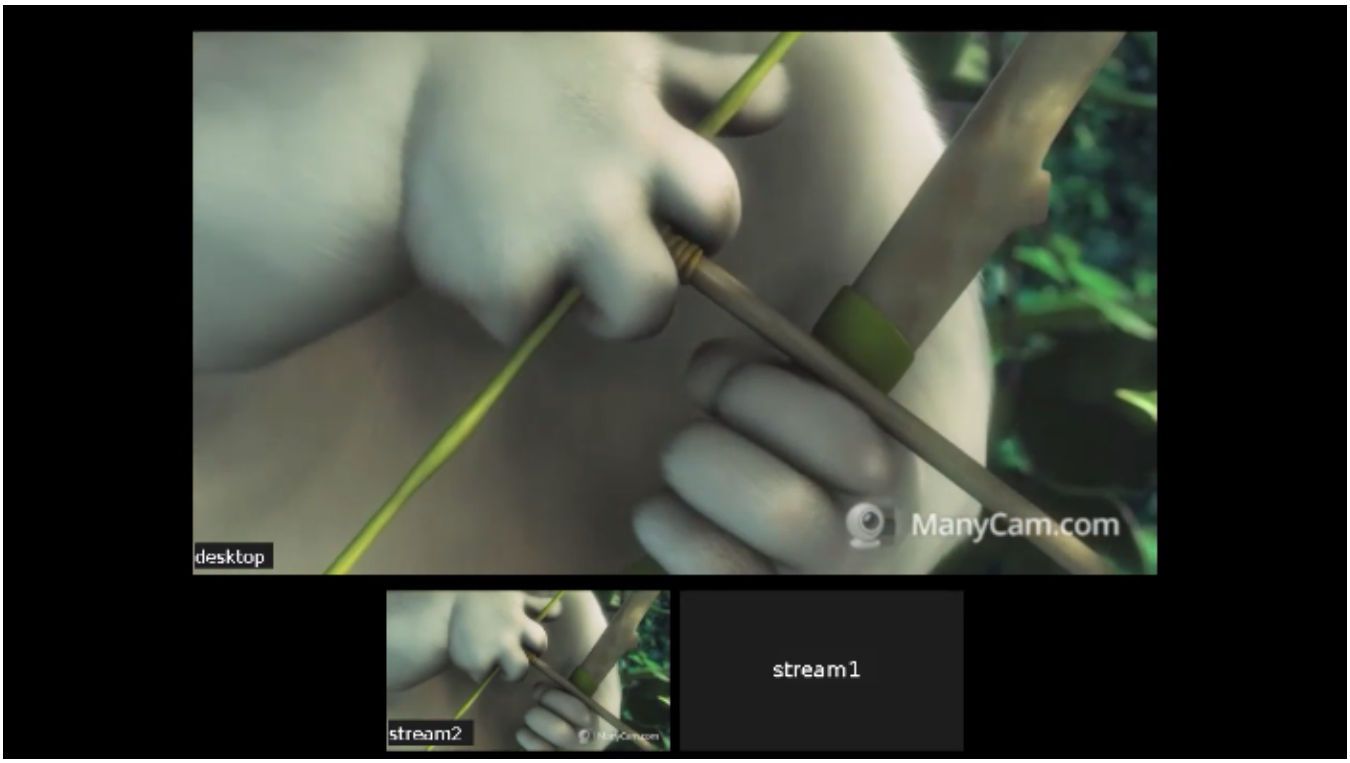
Уберем звук у всех участников, кроме ведущего

```
POST /rest-api/mixer/setAudioVideo HTTP/1.1
User-Agent: curl/7.29.0
Host: localhost:8081
Accept: */*
Content-Type: application/json
Content-Length: 62

{
 "uri": "mixer://m1",
 "streams": "^stream.*",
 "audioLevel": 0
}
```

Параметр `streams`может быть задан либо как регулярное выражение для поиска потоков по имени, либо как список потоков. Синтаксис регулярных выражении соответствует поддерживаемому в языке Java, примеры приведены здесь.

Уберем видео у потока stream1

```
POST /rest-api/mixer/setAudioVideo HTTP/1.1
User-Agent: curl/7.29.0
Host: localhost:8081
Accept: */*
Content-Type: application/json
Content-Length: 65

{
 "uri": "mixer://m1",
 "streams": ["stream1"],
 "videoMuted": true
}
```



Проверим состояние потоков запросом /mixer/find_all

```
POST /rest-api/mixer/find_all HTTP/1.
1
User-Agent: curl/7.29.0
Host: localhost:8081
Accept: */*
Content-Type: application/json
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Content-Type: application/json
Content-Length: 574

[
  {
    "localMediaSessionId": "e2fa5c8b-16f3-4917-9d5f-557dde75db07",
    "localStreamName": "m1",
    "uri": "mixer://m1",
    "status": "PROCESSED_LOCAL",
    "hasAudio": true,
    "hasVideo": true,
    "record": false,
    "mediaSessions": [
      {
        "localMediaSessionId": "3dd763b0-2ae7-11eb-aa72-
37b2cbcbf6b9",
        "audioLevel": 0,
        "videoMuted": true,
        "localStreamName": "stream1"
      },
      {
        "localMediaSessionId": "8af64760-2ae7-11eb-b086-
cdf035231b9d",
        "audioLevel": 100,
        "videoMuted": false,
        "localStreamName": "desktop"
      },
      {
        "localMediaSessionId": "7cc4b410-2ae7-11eb-b34c-
a5240fe9f151",
        "audioLevel": 0,
        "videoMuted": false,
        "localStreamName": "stream2"
      }
    ]
  }
]
```

Управление аудио и видео входного потока при добавлении его в микшер

В сборке5.2.982добавлена возможность заглушить или изменить уровень аудио и заглушить видео при добавлении потока в микшер. Для этого в запросе /mixer/add необходимо указать дополнительные параметры аналогично запросу /mixer/setAudioVideo

```
POST /rest-api/mixer/add HTTP/1.1
User-Agent: curl/7.29.0
Host: localhost:8081
Accept: */*
Content-Type: application/json
Content-Length: 85

{
 "uri": "mixer://m1",
 "remoteStreamName": "stream1",
 "audioLevel": 0,
 "videoMuted": false
}
```

Отметим, что, если поток в микшер был добавлен с параметром `hasVideo: false`

```
{
 "uri": "mixer://m1",
 "remoteStreamName": "stream1",
 "hasVideo": false,
 "hasAudio": true
}
```

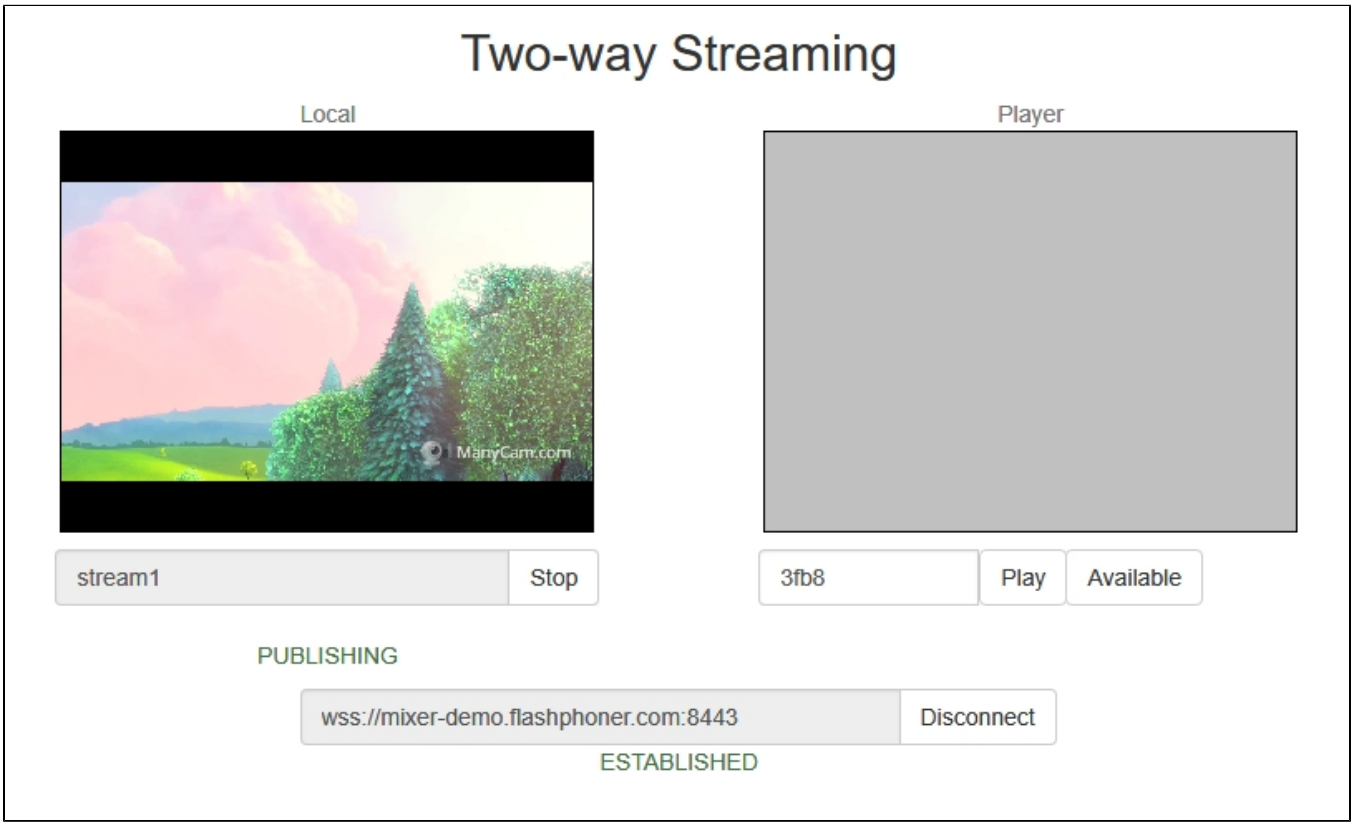
то параметр `videoMuted` будет игнорироваться, такой поток в микшере останется только с аудио.

Краткое руководство по тестированию

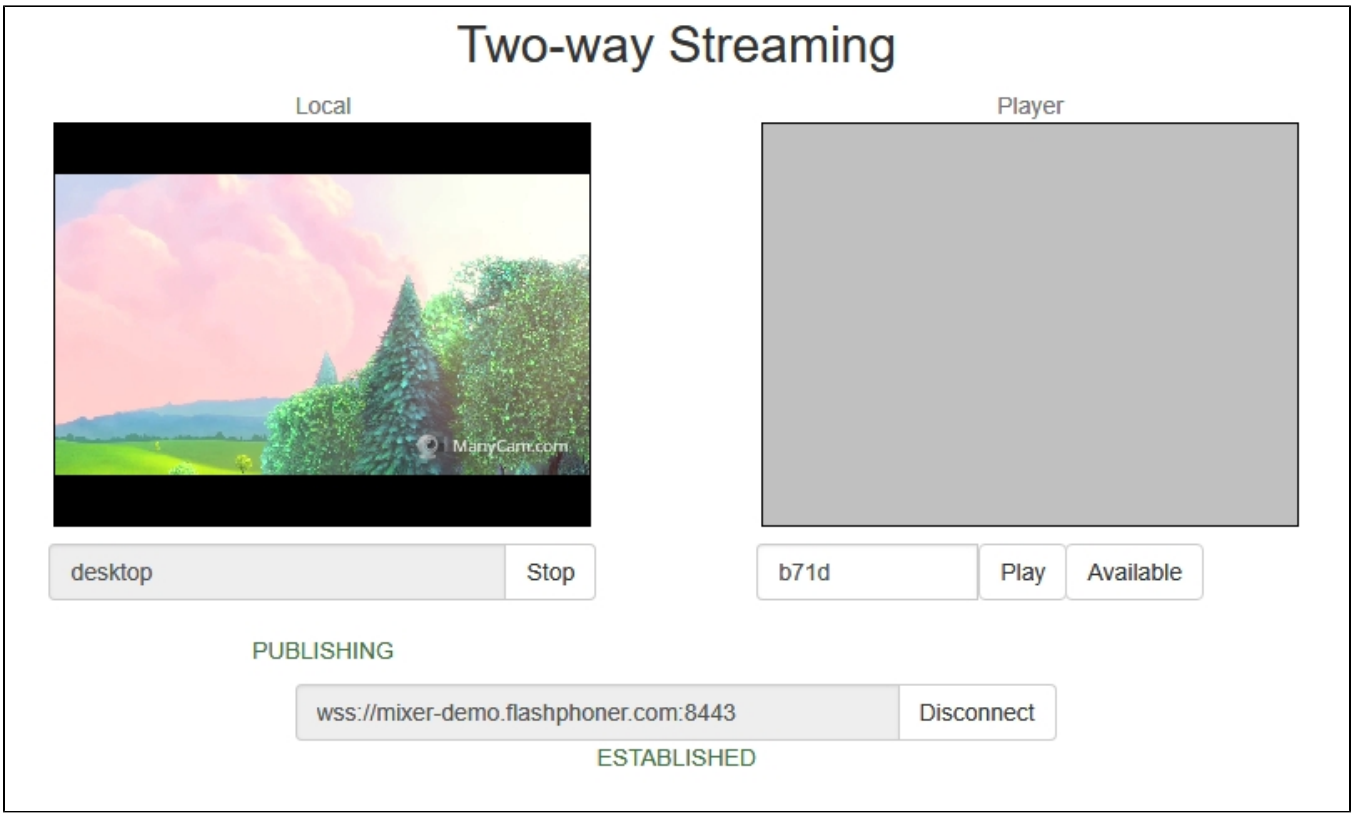
1. Для теста используем:

- демо-сервер demo.flashphoner.com;
- браузер Chrome иREST-клиентдля отправки запросов на сервер;
- веб-приложениеTwo Way Streamingдля публикации входных потоков микшера;
- веб-приложениеPlayerдля воспроизведения выходного потока микшера.

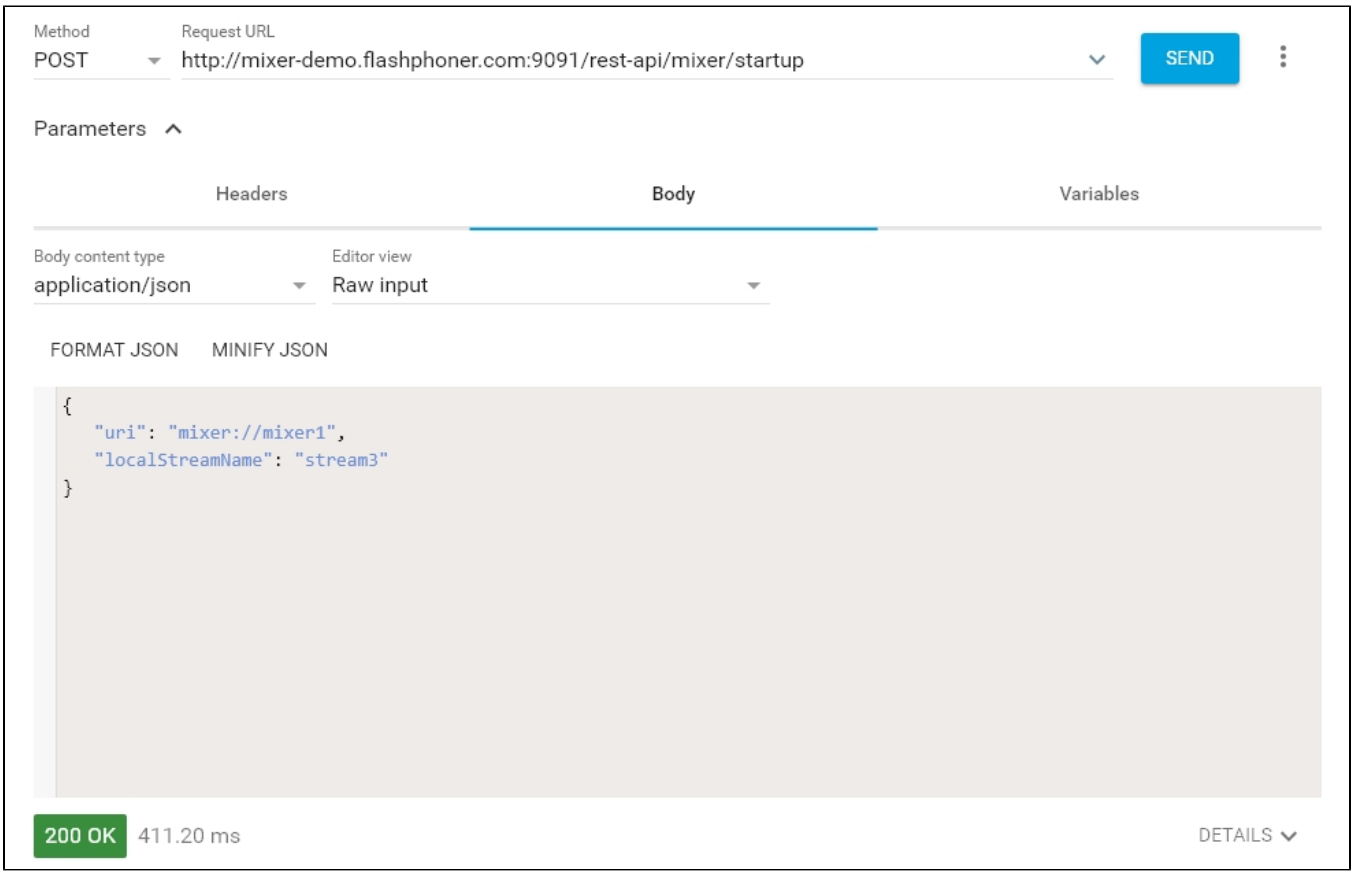
2. Откройте страницу веб-приложения Two Way Streaming. Опубликуйте поток с именемstream1:



3. В другой вкладке откройте страницу веб-приложения Two Way Streaming. Опубликуйте поток с именемdesktop:



4. Откройте REST-клиент. Отправьте запрос /mixer/startup, указав в параметрах URI микшераmixer://mixer1и имя выходного потокаstream3:



5. Отправьте запрос /mixer/add, указав в параметрах URI микшераmixer://mixer1и имя входного потокаstream1:

Method
POST
Request URL
http://mixer-demo.flashphoner.com:9091/rest-api/mixer/add
SEND

Parameters

Headers | Body | Variables

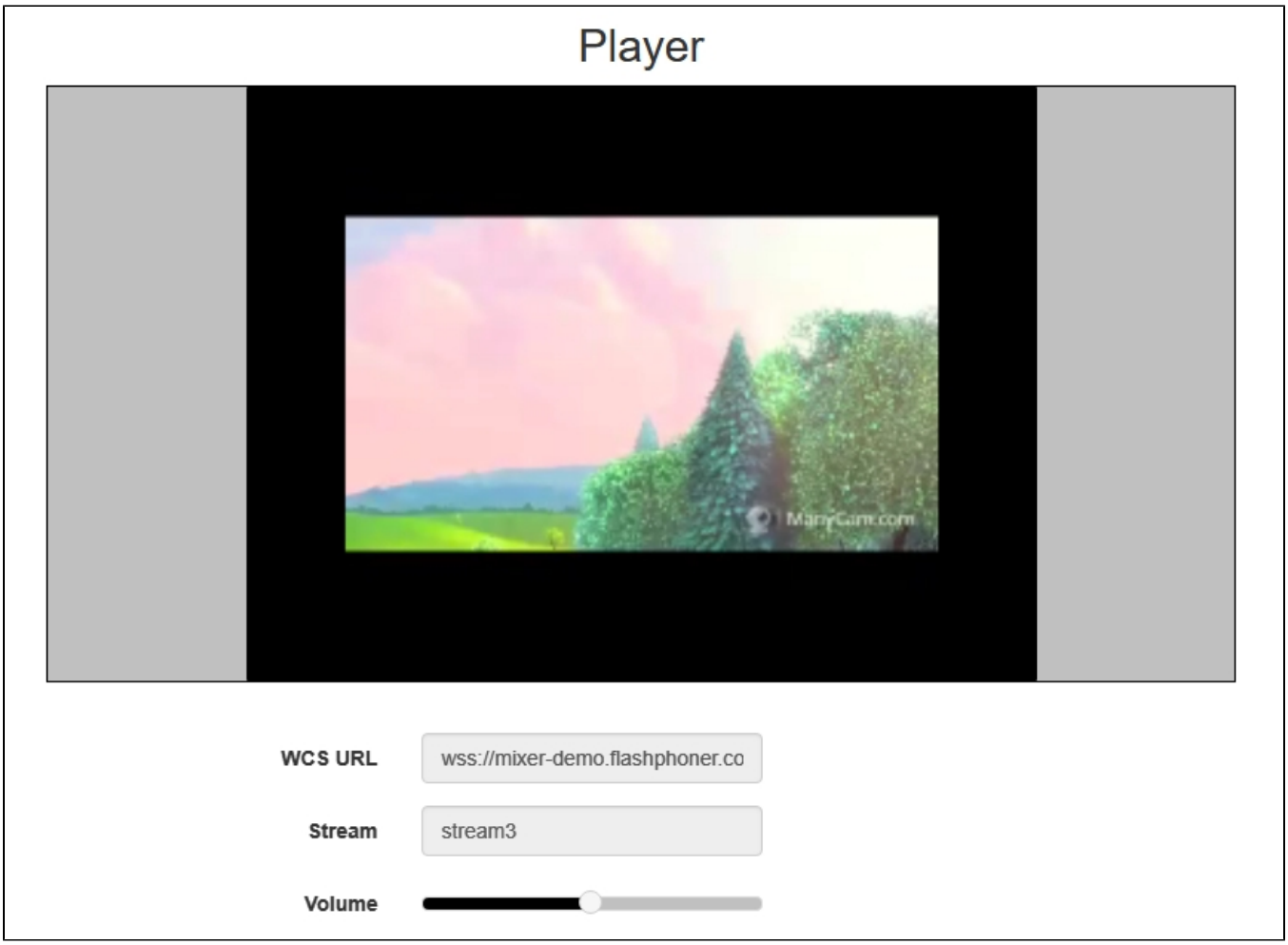
Body content type
application/json
Editor view
Raw input

FORMAT JSON MINIFY JSON

```
{
  "uri": "mixer://mixer1",
  "remoteStreamName": "stream1"
}
```

200 OK 396.40 ms
DETAILS

6. Откройте веб-приложение Player, укажите в поле Stream имя выходного потока микшераstream3и нажмите Start:



7. Отправьте запрос /mixer/add, указав в параметрах URI микшераmixer://mixer1и имя входного потокаdesktop:

Method
POST

Request URL
http://mixer-demo.flashphoner.com:9091/rest-api/mixer/add

SEND

Parameters

Headers | Body | Variables

Body content type
application/json

Editor view
Raw input

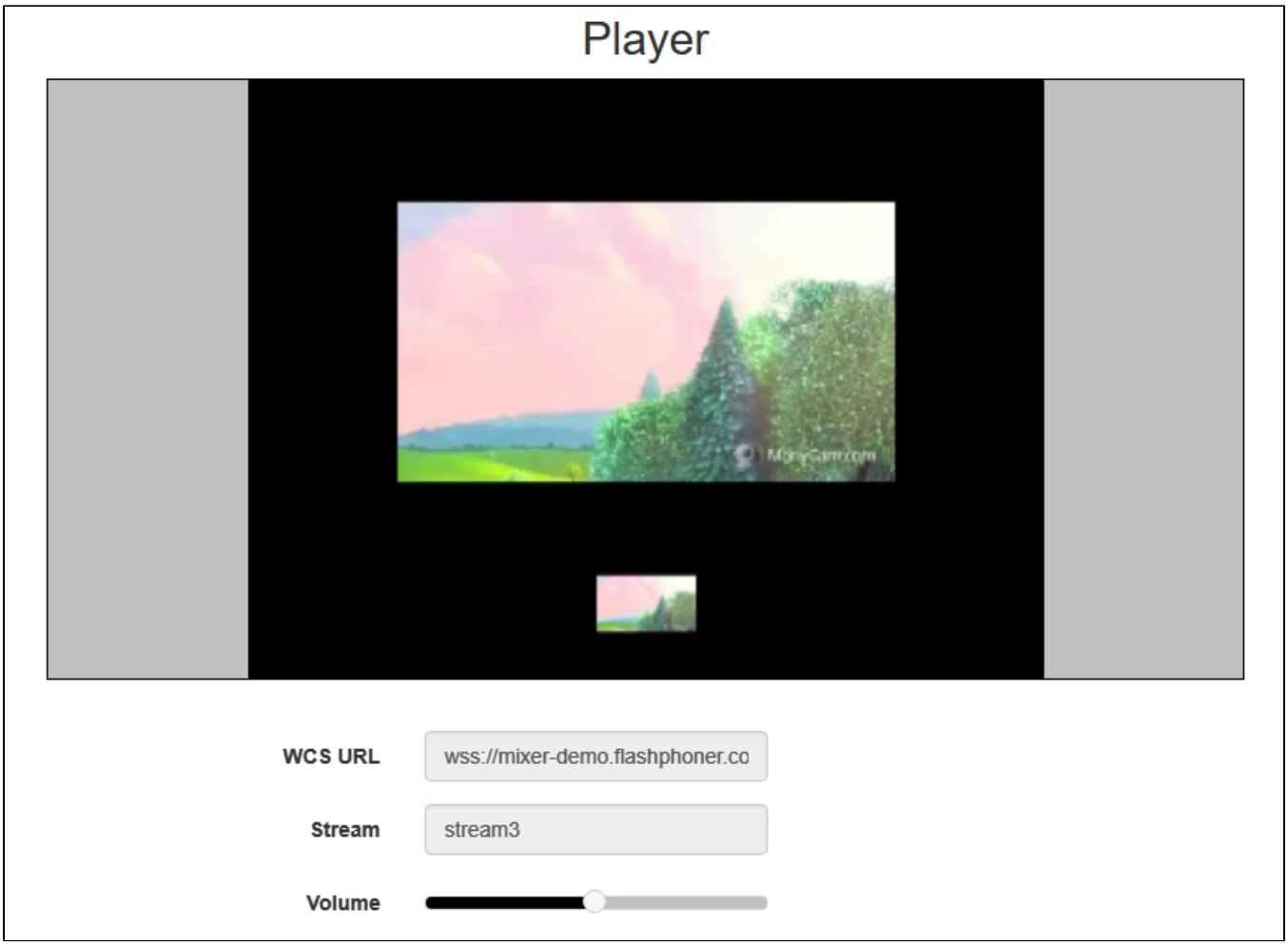
FORMAT JSON MINIFY JSON

```
{
  "uri": "mixer://mixer1",
  "remoteStreamName": "desktop"
}
```

200 OK 381.40 ms

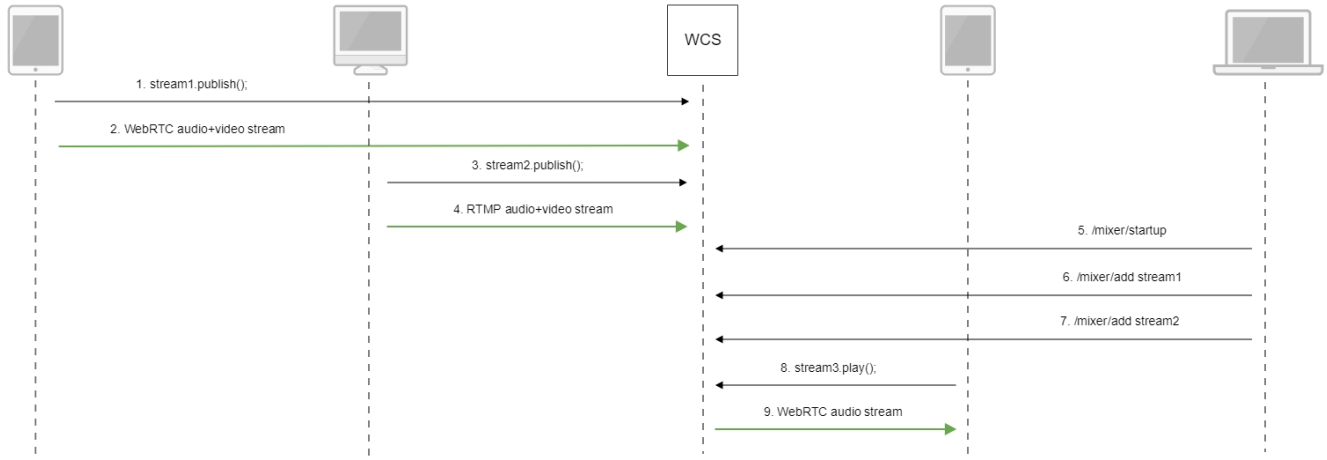
DETAILS

8. В выходном потоке микшера отобразится потокdesktop, имитирующий демонстрацию экрана, и потокstream1:



Последовательность выполнения операций (Call flow)

Ниже описана последовательность вызовов при использовании микшера.



1. ПубликацияWebRTC-потокаstream1

2. ОтправкаRTMP-потока на сервер

3. ПубликацияRTMP-потокаstream2

4. Отправка RTMP-потока на сервер

5. Отправка запроса /mixer/startup на создание микшера mixer://stream3 с выходным потоком stream3

```
http://demo.flashphoner.com:9091/rest-api/mixer/startup
{
 "uri": "mixer://stream3",
 "localStreamName": "stream3"
}
```

6. Отправка запроса /mixer/add на добавление к микшеру mixer://stream3 потока stream1

```
http://demo.flashphoner.com:9091/rest-api/mixer/add
{
 "uri": "mixer://stream3",
 "localStreamName": "stream3"
 "remoteStreamName": "stream1"
}
```

7. Отправка запроса /mixer/add на добавление к микшеру mixer://stream3 потока stream2

```
http://demo.flashphoner.com:9091/rest-api/mixer/add
{
 "uri": "mixer://stream3",
 "localStreamName": "stream3"
 "remoteStreamName": "stream2"
}
```

8. Воспроизведение WebRTC-потока stream3

9. Отправка WebRTC-аудиопотока клиенту

REST hooks для микшера

По умолчанию, для выходного потока микшера могут быть обработаны все стандартные REST методы. При этом, метод /connect приходит на бэкенд сервер при создании микшера.

При необходимости, для микшера может быть создано отдельное REST-приложение. Чтобы направить вызовы от микшера к этому приложению, в сборке5.2.634добавлена настройка

```
mixer_app_name=defaultApp
```

По умолчанию, вызовы от микшера обрабатывает приложение defaultApp, как и для остальных потоков.

Контроль добавления потока в микшер на бэкенд сервере

В сборке 5.2.1416 добавлена возможность получения событий, сигнализирующих о том, что поток добавлен или удален из микшера. Для этого WCS отправляет на бэкенд сервер запрос `/StreamEvent`

```
URL:http://localhost:8081/apps/EchoApp/StreamEvent
OBJECT:
{
  "nodeId" : "d2hxbqNPE04vGeZ51NPhDuId6k3hUrBB@192.168.1.39",
  "appKey" : "defaultApp",
  "sessionId" : "/192.168.1.83:49977/192.168.1.39:8443-591009c4-e051-4722-b34d-71cf2ade3bed",
  "mediaSessionId" : "15de2290-4089-11ed-88fe-d78a87cf3386",
  "type" : "addedToMixing",
  "payload" : {
    "uri" : "mixer://m1"
  }
}
```

при добавлении потока и

```
URL:http://localhost:8081/apps/EchoApp/StreamEvent
OBJECT:
{
  "nodeId" : "d2hxbqNPE04vGeZ51NPhDuId6k3hUrBB@192.168.1.39",
  "appKey" : "defaultApp",
  "sessionId" : "/192.168.1.83:49977/192.168.1.39:8443-591009c4-e051-4722-b34d-71cf2ade3bed",
  "mediaSessionId" : "15de2290-4089-11ed-88fe-d78a87cf3386",
  "type" : "removedFromMixing",
  "payload" : {
    "uri" : "mixer://m1"
  }
}
```

при его удалении из микшера.

При обновлении WCS с предыдущих сборок в конфигурацию бэкенд приложения необходимо добавить метод `StreamEvent`

```
add app-rest-method defaultApp StreamEvent
add app-rest-method MyAppKey StreamEvent
```

Известные проблемы

1. Микшер не создается, если имя микшера или имя выходного потока содержит символы, недопустимые для указания в URI

Симптомы: не создается микшер с именем вида test_mixer.

Решение: не использовать в имени микшера или имени выходного потока недопустимые символы, в особенности, если включена возможность автоматичекого создания микшера. Например, имя

```
user_1#my_room
```

использовать нельзя.

Если микшируются потоки чат-комнат, в именах комнат также нельзя использовать недопустимые символы.

2. Выходной поток микшера будет пустым, если на сервере включен транскодинг только по требованию.

Симптомы: микшер видеопотоков успешно создается, но в выходном потоке черный экран.

Решение: для работы микшера на сервере должен быть включен транскодинг при помощи следующего параметра в файле flashphoner.properties

```
streaming_video_decoder_fast_start=true
```

3. Один поток не может быть добавлен в два микшера одновременно, если не используется микшер реального времени

Симптомы: при добавлении потока во второй микшер выходной поток первого микшера останавливается

Решение: не использовать один поток более чем в одном микшере

4. Настройка качества кодирования не применяется при использовании OpenH264

Симптомы: качество картинки не изменяется при различных значениях настройки `mixer_video_quality` , например

```
mixer_video_quality=5
```

не отличается от

```
mixer_video_quality=20
```

Решение: не использовать кодирование на базе OpenH264, поскольку управление CRF в нем не поддерживается

```
encoder_priority=FF
```

5. При добавлении водяного знака или фона, если PNG файл поврежден, либо это не PNG файл, используется водяной знак по умолчанию (черная картинка)

Симптомы: при добавлении водяного знака в выходном потоке черный экран, в серверном логе сообщение

```
Wrong watermark file format. Should be PNG
```

при добавлении фона в выходном потоке фон остается черным, в серверном логе сообщение

```
Custom mixer background file wrong format. Should be PNG
```

Решение: использовать только PNG файл с корректной структурой для добавления водяного знака или фона.