Stream mixer

- Overview
  - Supported protocols of input streams
  - Output stream control capabilities
  - Automatically create a mixer when publishing the stream
  - Operation flowchart
- REST queries
  - REST-methods and response statuses
  - Parameters
    - Mixer configuration while creating the instance using REST API
    - Changing mixer parameters on the fly by REST API
  - Sending the REST query to the WCS server
  - Known limits
- Configuration
  - Automatic mixer creation configuration
  - Audio and video mixing configuration
    - Publishing a stream without audio or video track to AV mixer
  - Mixer output stream bufferization
  - Changing bitrate of mixer output stream
  - Mixer output stream video codec management
  - Mixer output stream sound management
  - Using custom lossless videoprocessor for incoming streams handling
  - Mixer output stream layout management
    - Grid layout
    - Zero padding grid layout
    - Crop around center zero padding grid layout
    - Desktop (screen sharing) layout
    - Picture in picture
    - Custom mixer layout implementation
      - Cropping pictures in custom layout
      - A separate folder for custom Java libraries
    - Mixer layout management while creating mixer
  - Mixer output stream encoding profile management
  - Mixer background management and watermarking
  - Adding and changing stream watermark dynamically
  - Stereo sound in mixer output stream
  - Characters decoding in input stream name
- MCU support
- Incoming streams audio and video management
  - Incoming audio and video management while adding stream to mixer
- Quick manual on testing
- Call flow
- Mixer REST hooks
- REST hook for adding stream to mixer or removing it
- Known issues

Overview

WCS allows mixing streams of active broadcasts. The output stream of the mixer can be recorded,playedorrepublishedusing any of technologies supported by WCS.
Mixing is controlled usingsettingsandREST API.

> Stream transcoding is applied while mixing streams. The recommended server configuration is 2 CPU cores per 1 mixer.

Supported protocols of input streams

- WebRTC
- RTMP
- RTSP

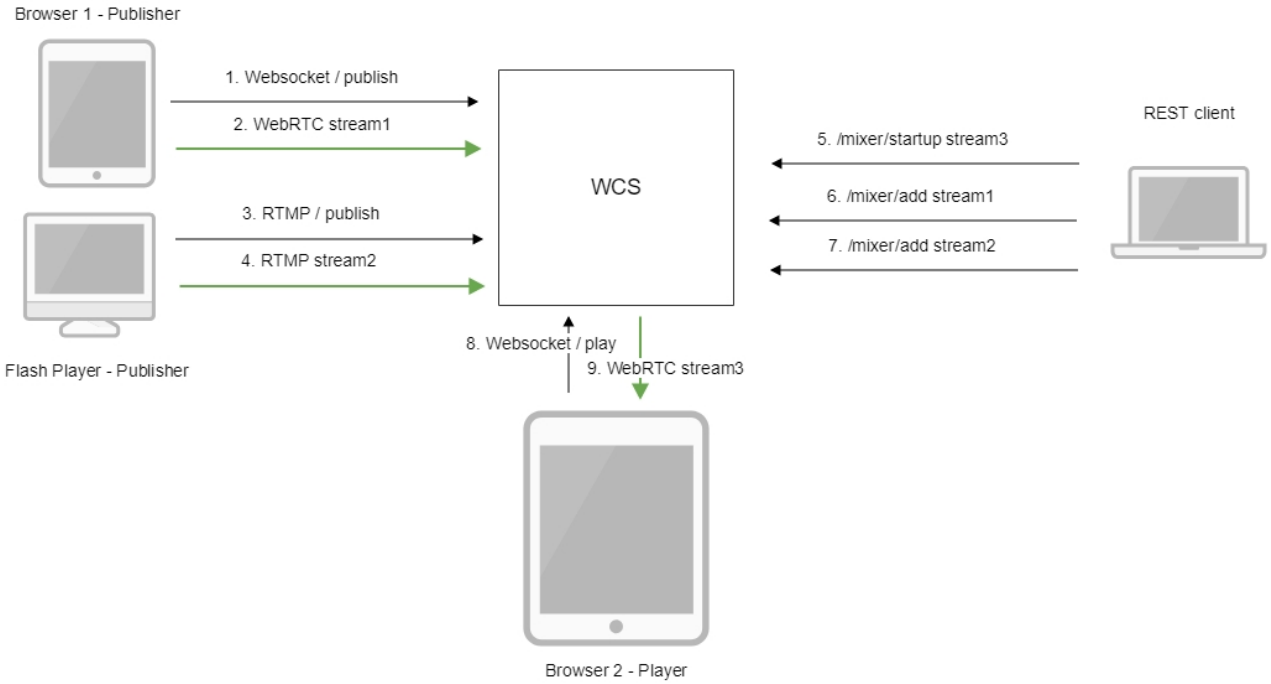
Output stream control capabilities

The mixer allows custom placing of video streams in the output frame. The stream with a certain name (by default desktop) is seen as screensharing and hence is placed in the center of the frame:

Automatically create a mixer when publishing the stream

If the name of the published RTMP stream has the '#' symbol, the server treats everything after that symbols as the name of the mixer that will be created when the stream is published. For instance, for the user1#room1 stream, the room1 mixer is created, and the stream is added to this mixer then. The stream name can also include the screen sharing keyword, for example, user1#room1#desktop

Operation flowchart



1. The browser connects to the server via the Websocket protocol and sends the publish command.
2. The browser sends the WebRTC stream1 to the server.
3. Flash Player establishes a connection via RTMP and sends the publish command.
4. Flash Player sends the RTMP stream2 to the server.
5. The REST client creates a mixer with the output stream3 using the query: /mixer/startup
6. The REST client adds stream1 to the mixer
7. The REST client adds stream2 to the mixer
8. The second browser establishes a connection via Websocket and sends the play command.
9. The second browser receives the WebRTC audio stream stream3 and plays that stream on the page.

REST queries

A REST-query must be an HTTP/HTTPS POST query in the following form:

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

Here:

- streaming.flashphoner.com- is the address of the WCS server
- 8081 - the standard REST / HTTP port of the WCS server
- 8444- the standard HTTPS port
- rest-api- the required prefix
- mixer/startup- the REST-method used

REST-methods and response statuses

REST-method	Example of REST query	Example of response	Response statuses	Description

/mixer
/startup
	<pre>{
	  "uri": "mixer://mixer1",
	  "localStreamName":
	"stream3",
	  "hasVideo": "true",
	  "hasAudio": "true",
	  "watermark": "watermark.
	png",
	  "background": "background.
	png",
	  "mixerLayoutClass": "com.
	flashphoner.mixerlayout.
	TestLayout"
	}</pre>
			200 - OK
			
			400 - Bad request
			
			409 - Conflict
			
			500 - Internal error
				Creates a mixer the provided stream is published for

/mixer/add	<pre>{
	  "uri": "mixer://mixer1",
	  "remoteStreamName":
	"stream1",
	  "hasVideo": "true",
	  "hasAudio": "true",
	  "streamLabel": "John Doe",
	  "avatar":
	"https://mystorage/storage
	/avatar.png",
	  "videoPositionId":
	"speaker"
	}</pre>
			200 - OK
			
			404 - Mixer not found
			
			404 - Stream not found
			
			500 - Internal error
				Add the RTMP stream to the mixer

/mixer
/remove
	<pre>{
	  "uri": "mixer://mixer1",
	  "remoteStreamName":
	"stream1"
	}</pre>
			200 - OK
			
			404 - Mixer not found
			
			404 - Stream not found
			
			500 - Internal error
				Remove the RTMP stream from the mixer

/mixer
/find_all
		<pre>{
		"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"
		],
		}</pre>
			200 - OK
			
			404 - Not found
			
			500 - Internal error
				Find all mixers

/mixer
/terminate
	<pre>{
	  "uri": "mixer://mixer1"
	}</pre>
			200 - OK
			
			404 - Not found
			
			500 - Internal error
				Terminate operation of the mixer

/stream /startRecording	<pre>{
	  "mediaSessionId": "23d07fa1-3c74-4d6f-a0de-9b4bf83ce665"
	}</pre>
			200 - OK
			404 - Not found
			500 - Internal error
				Start recording of the stream in the given media session

/stream /stopRecording	<pre>{
	  "mediaSessionId": "23d07fa1-3c74-4d6f-a0de-9b4bf83ce665"
	}</pre>
			200 - OK
			404 - Not found
			500 - Internal error
				Stop recording the stream in the given media session

/mixer /setAudioVideo	<pre>{
	"uri": "mixer://m1",
	"streams": "^stream.*",
	"audioLevel": 0,
	"videoMuted": true
	}</pre>
			200 - OK
			400 - Bad request
			404 - Not found
			500 - Internal error
				Mute/unmute video or change audio level for mexer incoming stream

/mixer /set_body_watermark	<pre>{
	"uri":"mixer://m1",
	"watermark":"/opt/media/logo.png",
	"x":10,
	"y":10,
	"marginTop":5,
	"marginLeft":5,
	"marginBottom":5,
	"marginRight":5
	}</pre>
			200 - OK
			400 - Bad request
			404 - Not found
				Add watermark to mixer output stream picture

/mixer /set_stream_watermark	<pre>{
	"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
	}</pre>
			200 - OK
			400 - Bad request
			404 - Not found
				Add watermark to one of the mixer input streams pictures in the mixer output stream

/mixer /set_stream_label	<pre>{
	"uri":"mixer://m1",
	"remoteStreamName":"stream1",
	"streamLabel": "Mr. John Doe"
	}</pre>
			200 - OK
			404 - Not found
				Set or change mixer participant stream label to display

/mixer /set_parameter	<pre>{
	"uri":"mixer://m1",
	"mixerLayoutDir":"/opt/GridLayout"
	}</pre>
			200 - OK
			400 - Bad request
			404 - Not found
				Change the mixer parameter

/mixer /set_stream_avatar	<pre>{
	"uri":"mixer://m1",
	"remoteStreamName":"stream1",
	"avatar":"https://mystorage/storage/avatar.png"
	}</pre>
			200 - OK
			400 - Bad request
			404 - Not found
			500 - Internal server error
				Set avatar picture to audio stream in mixer

/mixer /remove_stream_avatar	<pre>{
	"uri":"mixer://m1",
	"remoteStreamName":"stream1"
	}</pre>
			200 - OK
			404 - Not found
			500 - Internal server error
				Remove avatar picture from audio stream in mixer

/mixer /set_position	<pre>{
	"uri":"mixer://m1",
	"remoteStreamName":"stream1",
	"videoPositionId":"speaker"
	}</pre>
			200 - OK
			404 - Not found
			500 - Internal server error
				Move the stream picture to the position (for custom XML layouts only)

Parameters

Parameter name	Description	Example
uri	Unique identifier of the mixer	mixer://mixer1
localStreamName	Name of the output stream of the mixer	stream3
hasVideo	Mix video	true
hasAudio	Mix audio	true
remoteStreamName	Name of the stream added to the mixer	stream1 rtmp://rtmp.flashphoner.com:1935/live/rtmp_stream1
mediaSessionId	Media session identifier	ce92b134-2468-4460-8d06-1ea3c5aabace
status	Stream status	PROCESSED_LOCAL
background	Mixer background	background.png
watermark	Mixer watermark	watermark.png
mixerLayoutClass	Mixer layout	com.flashphoner.mixerlayout.TestLayout
streams	Streams list or regular expression for search	^stream.* ["stream1", "stream2"]
audioLevel	Incoming stream audio level	0
videoMuted	Mute video	true
streamLabel	Mixer participant stream label to display	John Doe

avatar	Picture URI in PNG, JPG, BMP formats	https://mystorage.com/storage/avatar.jpg
videoPositionId	Position identifier in custom XML layout to place a stream picture	speaker

Mixer configuration while creating the instance using REST API

Since build5.2.872it is possible to pass most of mixer parameters corresponding toflashphoner.propertiesmixer settings while creating the mixer using /mixer/startup REST query. In this case, parameters will be applied to the created mixer instance only. For example, the following query

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

will create the mixer with output stream resolution 640x360, fps 24 and bitrate 500 kbps, no matter what settings are set inflashphoner.propertiesfile

The mixer parameters fukll list can be retrieved using /mixer/find_all REST query, for example

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"
  }
]
```

Changing mixer parameters on the fly by REST API

Since build 5.2.1480, the following parameters can be changed for an active mixer using `/mixer/set_parameter` REST query:

```
[
  {
    ...
    "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": ""
  }
]
```

If the REST query contains a parameters which cannot be set for active mixer, it will return `400 Bad request` with unsupported parameters list.

Sending the REST query to the WCS server

To send the REST query to the WCS server use aREST-client.

Known limits

Mixer output stream name must be defined while creating mixer by REST API (localStreamName parameter). Otherwise, server returns 400Bad requestwith message "No localStreamName given".

Configuration

Mixing can be configured using the following parameters in theflashphoner.propertiessettings file

Parameter	Default value	Description
mixer_video_desktop_layout_inline_padding	10	Distance (padding) between windows of video streams in the lower line (below the screen sharing window)
mixer_video_desktop_layout_padding	30	Distance (padding) between the screen sharing window and the lower line (the rest streams)
mixer_video_enabled	true	Enables (by default) or disables video mixing

mixer_video_grid_layout_middle_padding	10	Distance between windows of video streams in one line (without screen sharing window)
mixer_video_grid_layout_padding	30	Distance between lines of windows (without screen sharing window)
mixer_video_height	720	The image height of the mixer output stream,should have an even value. An uneven value will be auto decremented by one: e.g., if set to 481, mixer height will be 480.
mixer_video_layout_desktop_key_word	desktop	Keyword for the screen sharing stream
mixer_video_width	1280	The image width of the mixer output stream, should have an even value. An uneven value will be auto decremented by one: e.g., if set to 641, mixer width will be 640.
record_mixer_streams	false	Turns on or off (default) recording of all mixer output streams

Automatic mixer creation configuration

Automatic creation of mixers for streams with the '#' symbol in their name requires the application that handles input streams to register the handler: 'com.flashphoner.server.client.handler.wcs4.FlashRoomRecordingStreamingHandler'. Registering the handler can be done using thecommand line interface. For instance, for the flashStreamingApp application used to publish incoming RTMP streams this can be done with the following command:

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

You can read more about managing applications using the command line of the WCS serverhere.

Audio and video mixing configuration

By default, both video and audio streams are mixed. If audio only mixing is necessary, it should be set on mixer creation

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

To switch off video mixing for all streams, this parameter should be set in flashphoner.propertiesfile

```
mixer_video_enabled=false
```

In this case video mixing can be switched on for certain mixer on its creation.

Since build5.2.689audio mixing can be enabled o disabled on mixer creation

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

or in server settings for all the streams

```
mixer_audio_enabled=false
```

Publishing a stream without audio or video track to AV mixer

When a stream with audio or video track only is published to AV mixer, it is necessary to disable RTP activity control with the following parameters

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

Mixer output stream bufferization

In some cases, mixer output stream bufferization is needed. This feature is enabled with the following parameter in flashphoner.propertiesfile

```
mixer_out_buffer_enabled=true
```

The buffer size is defined in milliseconds with parameter

```
mixer_out_buffer_start_size=400
```

In this case, the buffer size is 400 ms.

Stream data fetching from buffer and sending period is defined in milliseconds with parameter

```
mixer_out_buffer_polling_time=20
```

In this case, the period is 20 ms.

Changing bitrate of mixer output stream

When OpenH264 codec is used for transcoding, it is possible to change bitrate of mixer output stream with the following parameterinflashphoner.propertiesfile

```
mixer_video_bitrate_kbps=2000
```

By default, mixer output stream bitrate is set to 2 Mbps. If a channel bandwidth between server and viewer is not enough, bitrate can be reduced, for example

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

If picture quality with default bitrate is low, or distortion occurs, it is recommended to rise mixer output stream bitrate to 3-5 Mbps

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

Mixer output stream video codec management

Since build 5.2.1075 it is possible to set mixer output stream video codec. H264 is used by default

```
video_mixer_output_codec=h264
```

VP8 codec can be set with the following parameter

```
video_mixer_output_codec=vp8
```

Note that keyframes in this case will be sent less frequently comparing to H264 stream because VP8 key frame size is significally bigger. This can lead to a slight delay playing the first frame in browser comparing to H264 stream with the same parameters.

Mixer output stream sound management

By default, mixer output stream sound is encoded to Opus with sample rate 48 kHz. These settings may be changed using the parameters inflashphoner.propertiesfile. For example, to use mixer output stream in SIP call the following value can be set:

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

In this case, sound will be encoded to PCMA (alaw) with sample rate 8 kHz.

Using custom lossless videoprocessor for incoming streams handling

To handle mixer incoming streams, if additional bufferizing or audio and video tracks synchronizing is required for example, the custom lossless videoprocessor may be used. This feature is enabled with the following parameter inflashphoner.propertiesfile

```
mixer_lossless_video_processor_enabled=true
```

The maximum size of mixer buffer in milliseconds is set with this parameter

```
mixer_lossless_video_processor_max_mixer_buffer_size_ms=200
```

By default, maximum mixer buffer size is 200 ms. After filling this buffer, the custom lossless videoprocessor uses its own buffer and waits for mixer buffer freeing. The period of mixer buffer checking is set in milliseconds with this parameter

```
mixer_lossless_video_processor_wait_time_ms=20
```

By default, the mixer buffer checking period is 20 ms.

Note that using the custom lossless videoprocessor may degrade real-time performance.

When custom lossless videoprocessor is used, it is necessary to stop mixer with REST query /mixer/terminate to free all consumed resources. Mixer can be stopped also by stopping all incoming streams, in this case mixer will stop when following timeout in milliseconds expires

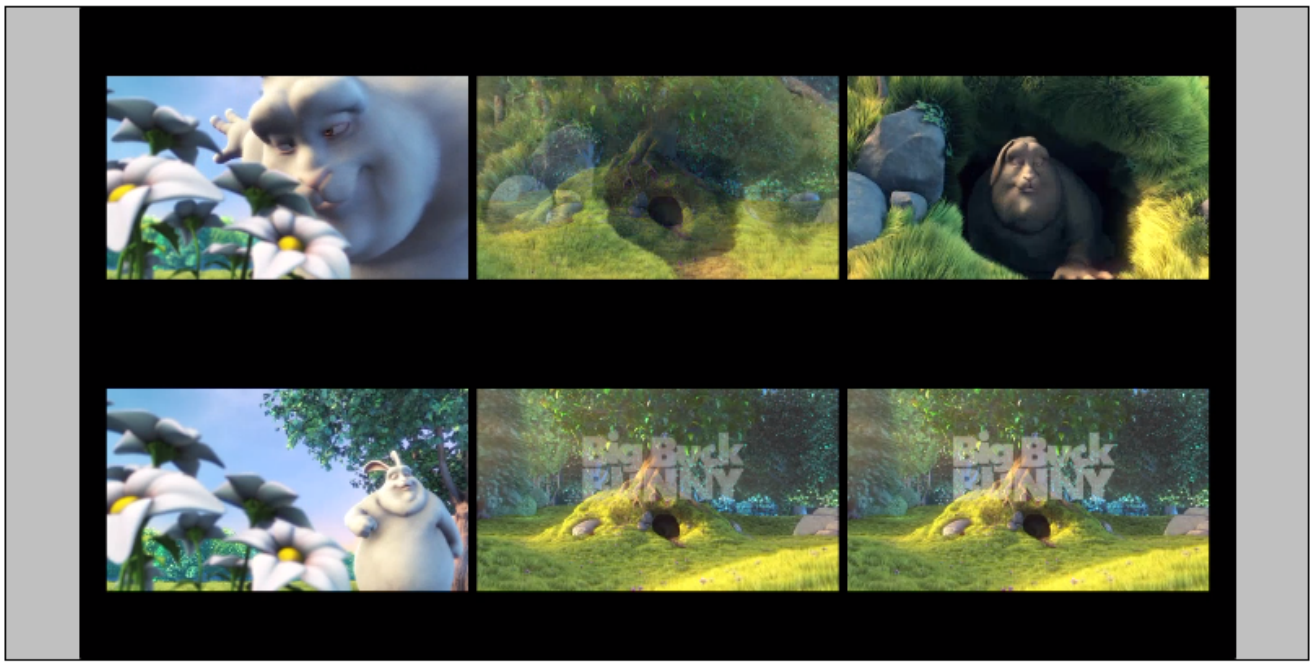
```
mixer_idle_timeout=60000
```

By default, mixer will stop after 60 seconds if there are no active incoming streams.

Mixer output stream layout management

By default, three mixer output stream layouts are implemented:

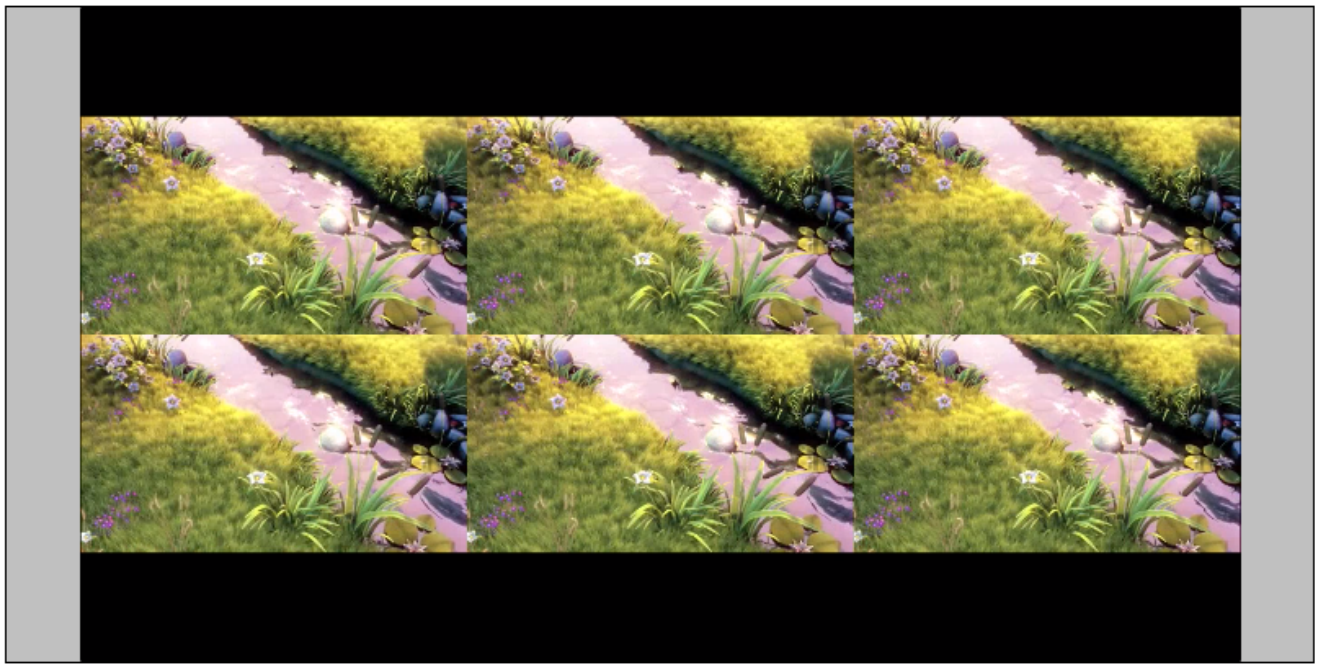
Grid layout



This layout can be enabled with the following parameter inflashphoner.propertiesfile

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

Zero padding grid layout



This layout can be enabled with the following parameter

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

and works for input streams of equal resolution with the same aspect ratio only.

Crop around center zero padding grid layout

Since build5.2.842, it is possible to crop participant's video around its center. That can be useful in case of video conference, when participant's face is located in the centeral part.

This layout can be enabled with the following parameter

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

Desktop (screen sharing) layout

This layout is enabled if one of mixer input streams has a name containing keyword defined in the following parameter

```
mixer_video_layout_desktop_key_word=desktop
```

By default, a stream should have `desktop` in its name to use it as screen sharing stream, for example `user1_desktop`

Since build5.2.710it is possible to change screen sharing picture placement with the following parameter:

```
mixer_desktop_align=TOP
```

By default, screen sharing picture is placed above the other pictures

The following placements are supported:

Placement	Description
TOP	Screen above
LEFT	Screen on left side
RIGHT	Sreen on right side
BOTTOM	Screen below
CENTER	Screen in center surronded by other pictures

For example, the following parameter

```
mixer_desktop_align=RIGHT
```

places the screen picture on the right side of mixer output stream picture

If a number of screen sharing streams are published to the same mixer, the first published stream take the main place. If the stream stops after that, the main place will be taken by the following screen share stream in alphabetical order.

Picture in picture

This layout is added since build5.2.852. In this case a stream that has a name with the keyword defined in the following parameter

```
mixer_video_layout_desktop_key_word=desktop
```

for example `user1_desktop`, will be background for other streams in mixer. This layout can be enabled with the following parameter

```
mixer_video_desktop_fullscreen=true
```

Custom mixer layout implementation

For more fine tuning of mixer layout, custom Java class should be developed to implement `IVideoMixerLayout` interface, for example

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()]);
    }
}
```

To support caption text location above or below stream picture, a special Box class should be used to calculate pictures placement since build5.2.878, for example

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) {
    //}
}
```

Main Box class methods:

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();
```

Possible positions to place Box object:

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
}
```

Then the class should be complied into byte code. To do this, create folder tree accordind to TestLayout class package name

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

and execute the command

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

Now, pack the code compiled to jar file

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

and copy this file to WCS libraries folder

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

To use custom mixer layout class, set it to the following parameter in flashphoner.properties file

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

and restart WCS.

With this custom layout, mixer output stream for three input streams will look like:

Cropping pictures in custom layout

To crop a picture around a central point like CropNoPaddingGridLayout,use Box.fillParentNoScale() instead of Box.fillParent() method. The following example places two pictures by name with cropping around center:

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) {
    //}
}
```

A separate folder for custom Java libraries

Since build 5.2.1512, custom layout Java libraries (jar files) should be placed to the folder `/usr/local/FlashphonerWebCallServer/lib/custom`

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

This folder is kept while updating WCS to a newer builds. A jar files do not need to be copied again after updating.

Mixer layout management while creating mixer

Since build5.2.693mixer layout can be defined when creating mixer with REST API, for example

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

Then, layout can be defib=ned for every mixer separately

Mixer output stream encoding profile management

Some browsers do not support playback for H264 streams encoded by certain profiles. To solve it, the following parameter is added since build5.2.414to set mixer output stream encoding profile

```
mixer_video_profile_level=42c02a
```

By default, constrainted baseline level 4.2 profile is set.

Mixer background management and watermarking

Since build5.2.693mixer backgroung can be defined and watermark can be added when creating mixer with REST API, for example

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

By default, files should be placed to /usr/local/FlashphonerWebCallServer/conf folder. Full path to the files can also be set, for example

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

Adding and changing stream watermark dynamically

Since build5.2.1349in is possible to dynamically add or change stream watermark without stopping the mixer. A watermark can be added, changed or moved to another picture location according to coordinates defined using REST API queries:

- `/mixer/set_body_watermark` - to the whole mixer output stream picture

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

- `/mixer/set_stream_watermark` - to a certain input stream picture

```
{
 "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
}
```

Where

- watermark - watermark file name
- x, y - top left watermark corner coordinates on the stream picture
- marginTop, marginLeft, marginBottom, marginRignt - watermark margins from stream picture borders

If watermark coordinates are out of stream picture bounds, the watermark will be scaled to the bounds using margins.

To move watermark to another location on the stream picture, send the query with the same file name and a new coordinates. To remove watermark from the stream picture, send the query with empty`watermark`field

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

Stereo sound in mixer output stream

By default, mixer converts stereo audio from incoming streams to mono to reduce the amiunt of processed data. This allows to minimize a possible delay while using realtime mixer in video conferencing case.

Mixer can be switched to stereo sound processing if necessary, for example, in case of online music radio. Since build5.2.922it is possible to set mixer audio channels count usig the following parameter

```
audio_mixer_output_channels=2
```

Characters decoding in input stream name

Since build 5.2.1802 it is possible to decode a characters encoded by encodeURIComponent() in input stream name. The feature may be enabled by the following parameter

```
mixer_decode_stream_name=true
```

or by the following `/mixer/startup` query parameter

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

In this case a decoded characters available in the font used will be displayed, or a similar characters.

MCU support

Mixer MCU support for audio can be enabled with the following parameter

```
mixer_mcu_audio=true
```

In this case for three input streams stream1, stream2, stream3 the following output streams will be generated in mixer mixer1:

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

Thus, each of the additional streams contains audio of all streams in the mixer, except for one. This allows for example to eliminate echo for conference participants.

This feature can be also enabled for videowith the following parameter

```
mixer_mcu_video=true
```

In this case for three input streams stream1, stream2, stream3 the following output streams will be generated in mixer mixer1:

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

Thus, each of the additional streams contains audio and video of all streams in the mixer, except for one. This allows to arrange a full-fledged chat room based on mixer.

In this case, if mixer recording is enabled with the parameter

```
record_mixer_streams=true
```

the main mixer output stream will be only recorded (mixer1 in the example above).

If video MCU support is enables, channel losses can affect mixer output stream quality. Custom lossless videoprocessor can be used to improve quality,that can make an additional latency.

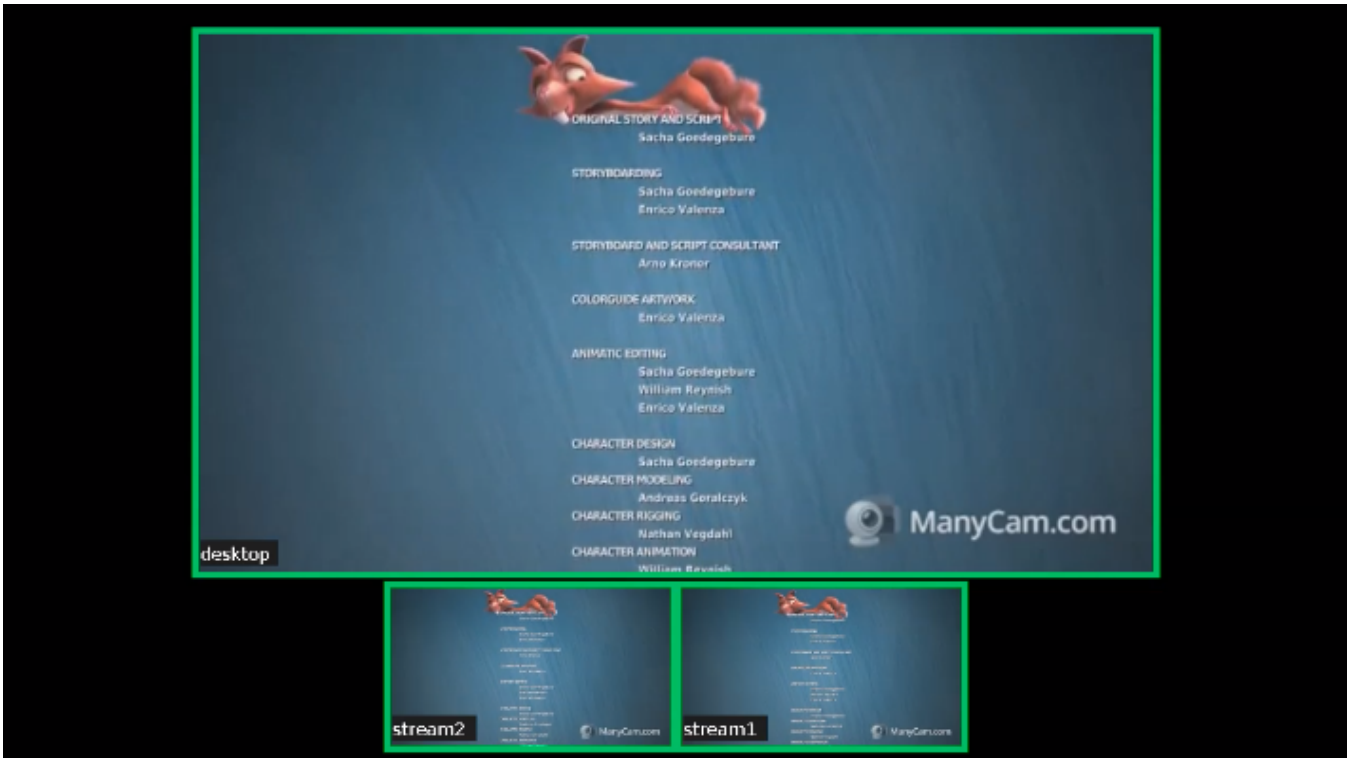
Incoming streams audio and video management

Since build5.2.835it is possible to change audio level and mute video for mixer incoming streams. In this case, the original stream remains unchanged. Video track can be muted (black screen) and then unmuted. For audio track, volume level can be set in percent up to 100, or sound can be muted by setting level to 0.

Incoming streams are managed using REST API query /mixer/setAudioVideo.

For example, create a mixer and add 3 streams to it: 2 participants and 1 speaker

```
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"}'
```



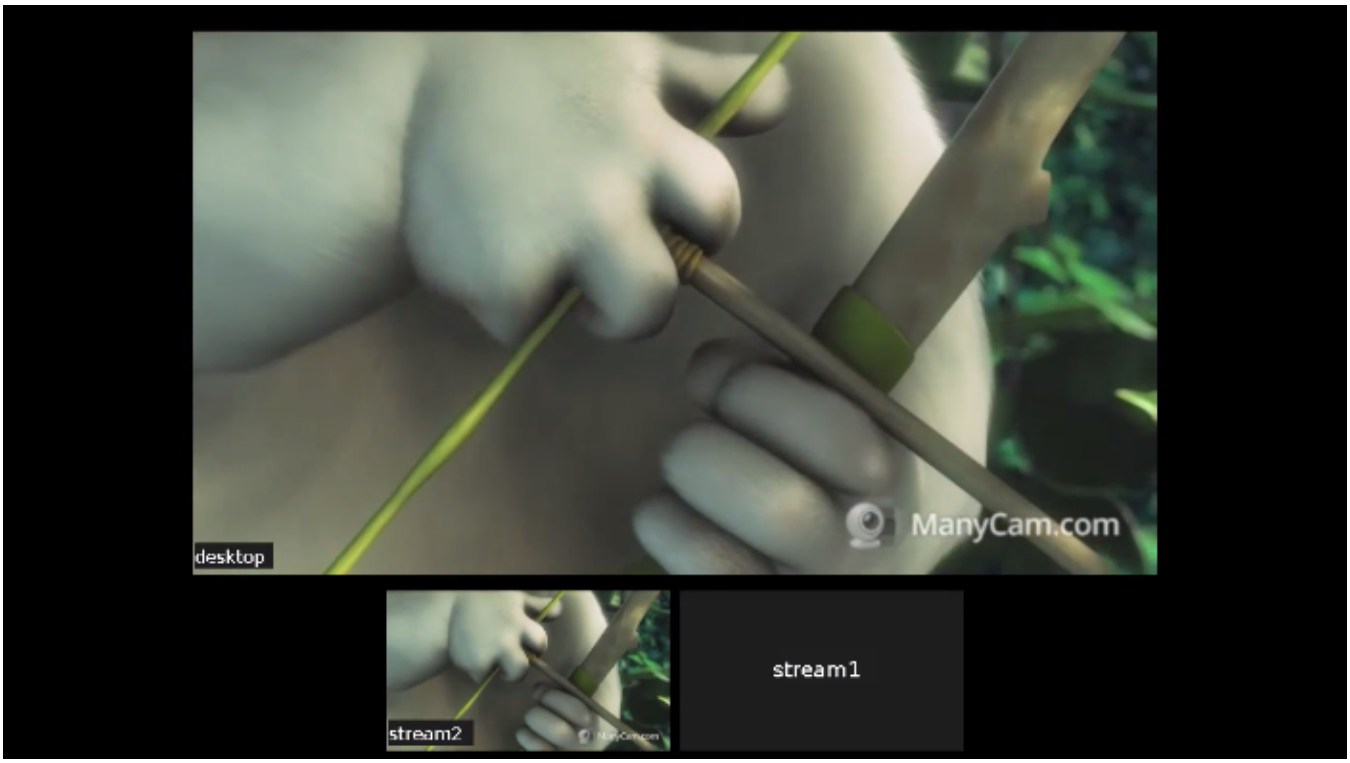
Mute all participants excluding speaker

```
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
}
```

REST query `streams` parameter may be set either as regular expression to match streams by name, or as streams list. Use regular expression syntax supported by Java, the examples can be foundhere.

Mute stream1 video

```
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
}
```



Check streams state by /mixer/find_all query

```
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"
      }
    ]
  }
]
```

Incoming audio and video management while adding stream to mixer

Since build5.2.982it is possible to mute audio or change audio level and mute video while adding stream to mixer. To do this, the additional parameters should be set in /mixer/add REST query like in /mixer/setAudioVideo one

```
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
}
```

Note that if a stream was added to mixer with `hasVideo: false` parameter

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

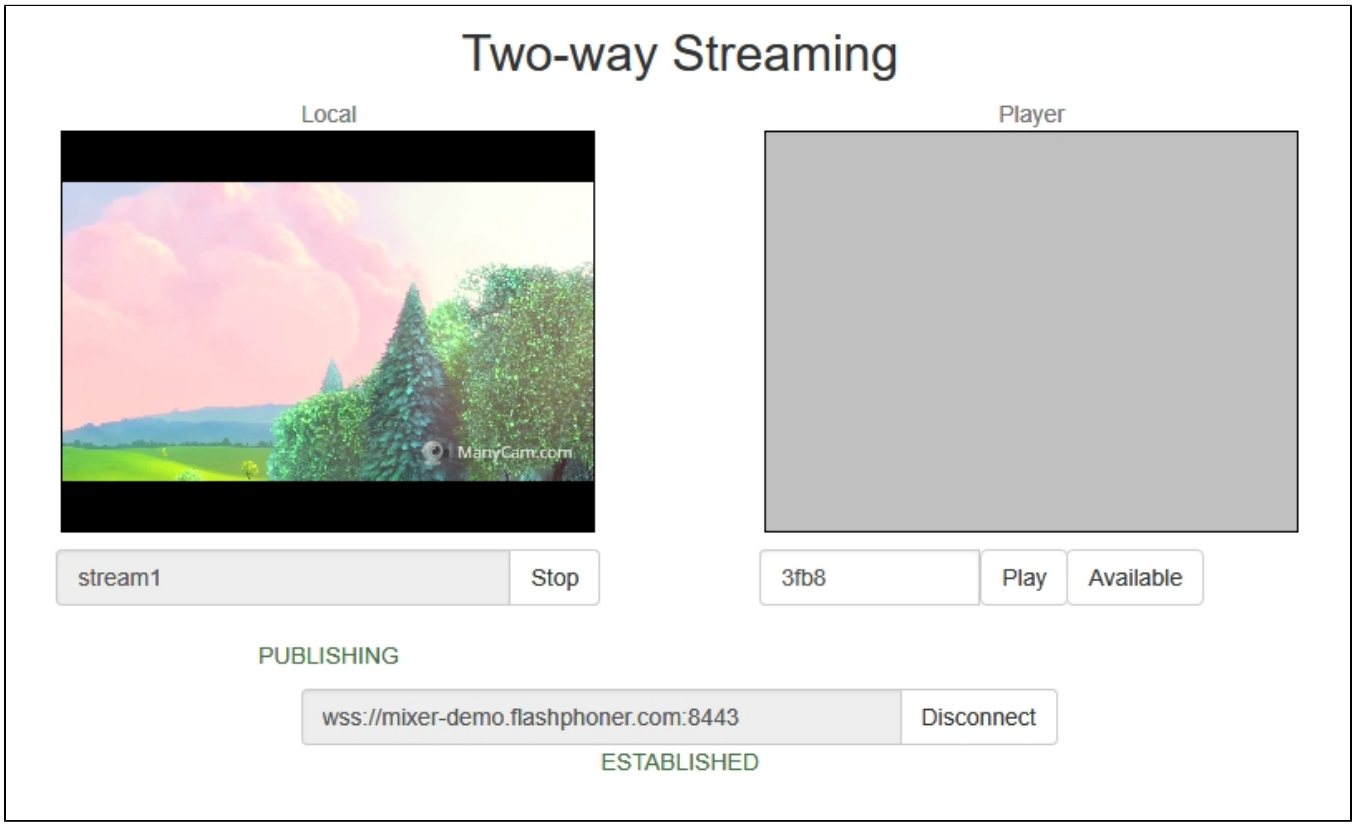
then `videoMuted` parameter will be ignored, the stream will always be audio only in the mixer.

Quick manual on testing

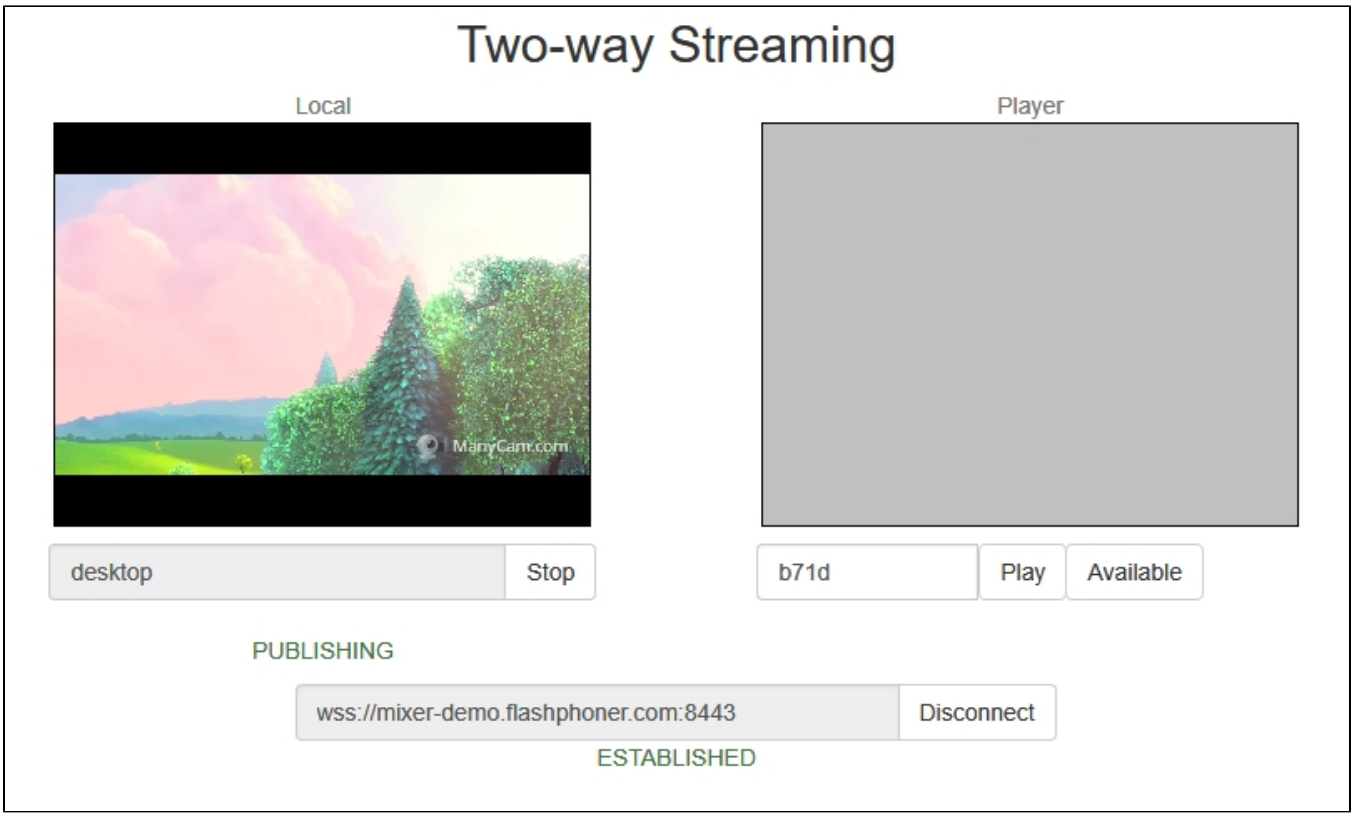
1. For this test we use:

- the demo server at demo.flashphoner.com;
- the Chrome browser and theREST-clientto send queries to the server;
- theTwo Way Streamingweb application to publish input streams of the mixer;
- thePlayerweb application to play the output stream of the mixer.

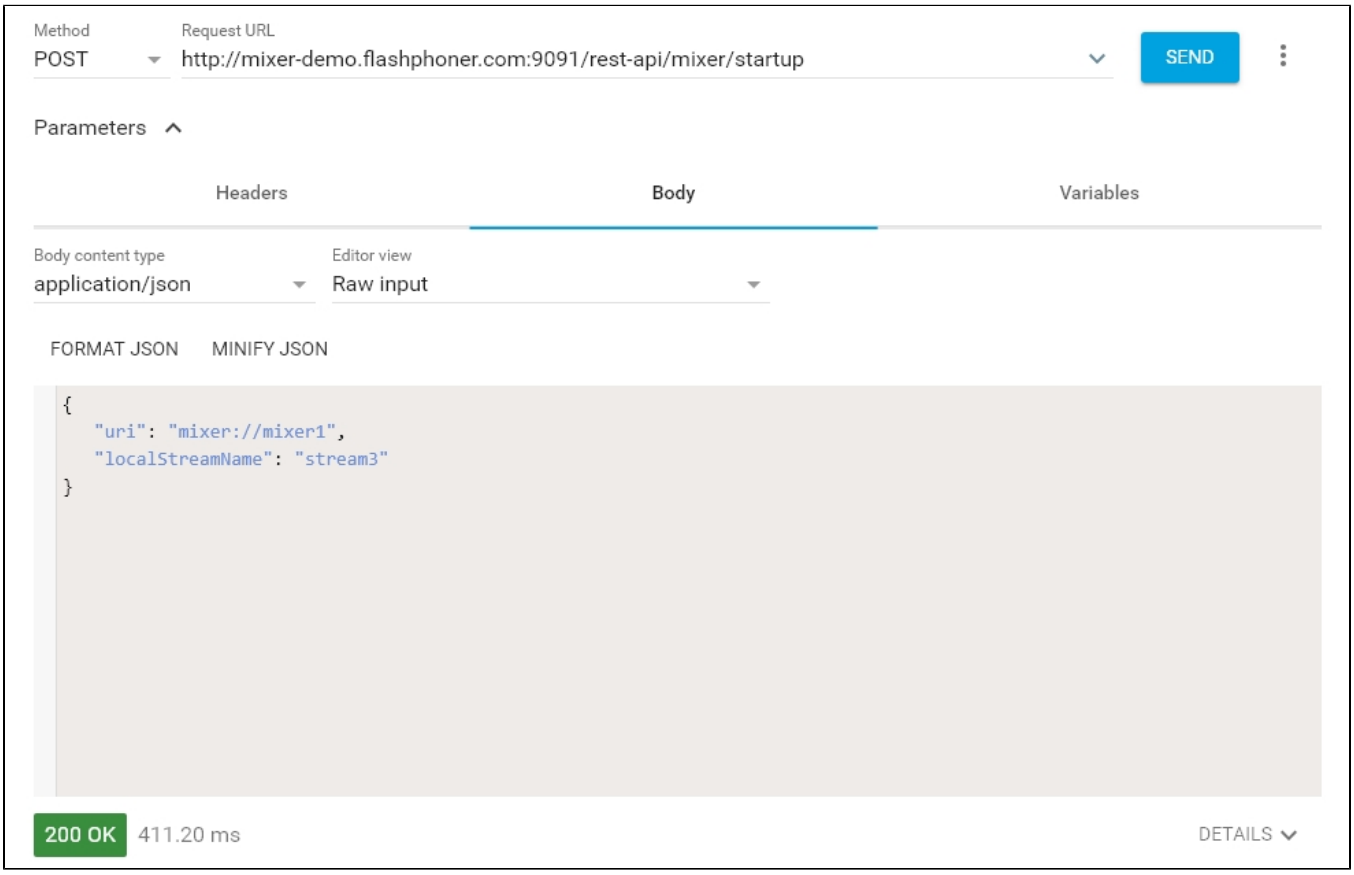
2. Open the page of the Two Way Streaming application. Publish the stream named stream1:



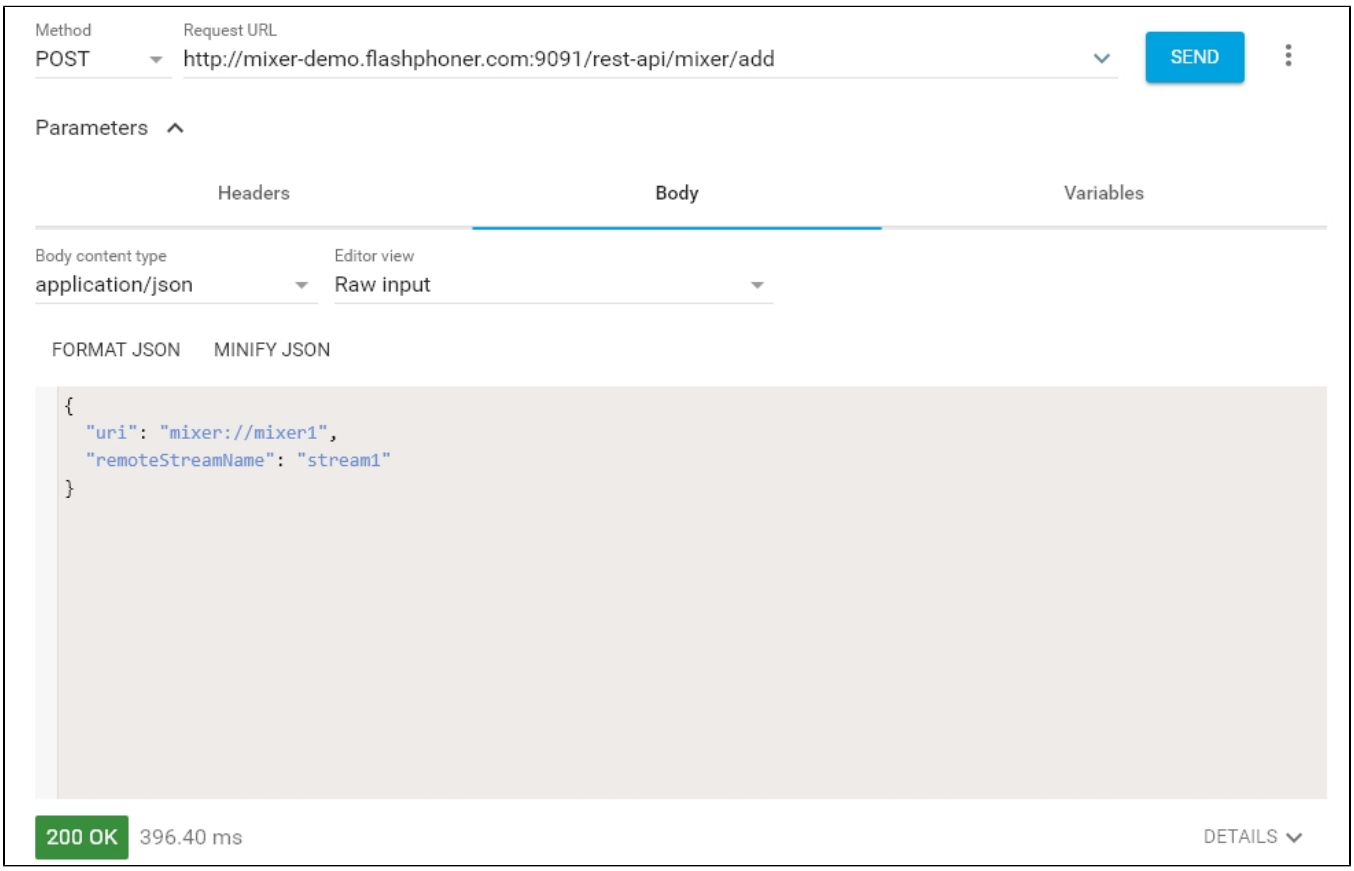
3. In another tab open the page of the Two Way Streaming application. Publish the stream named desktop:



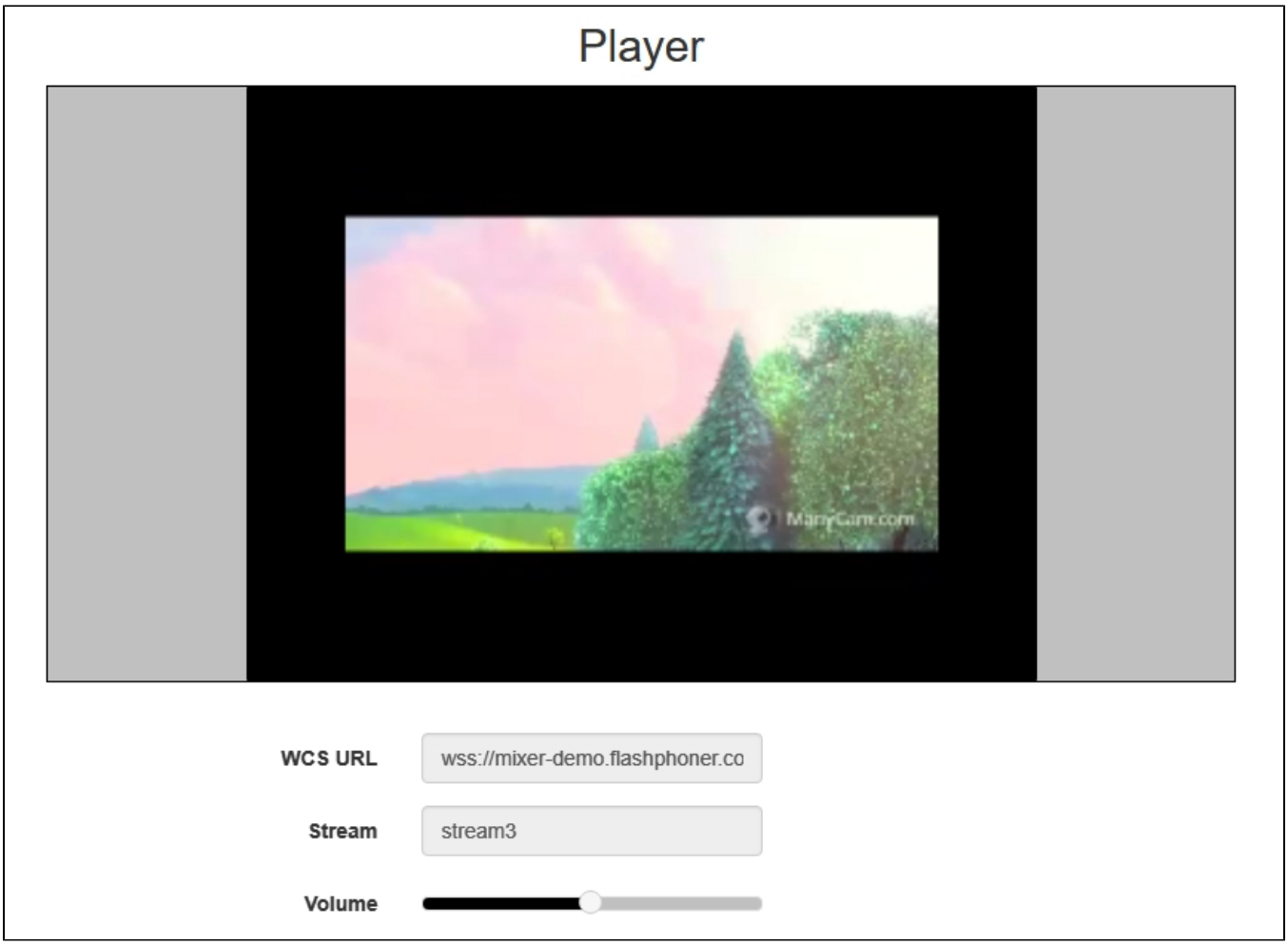
4. Open the REST client. Send the /mixer/startup query and specify the URI of the mixer mixer://mixer1 and the output stream name stream3 in its parameters:



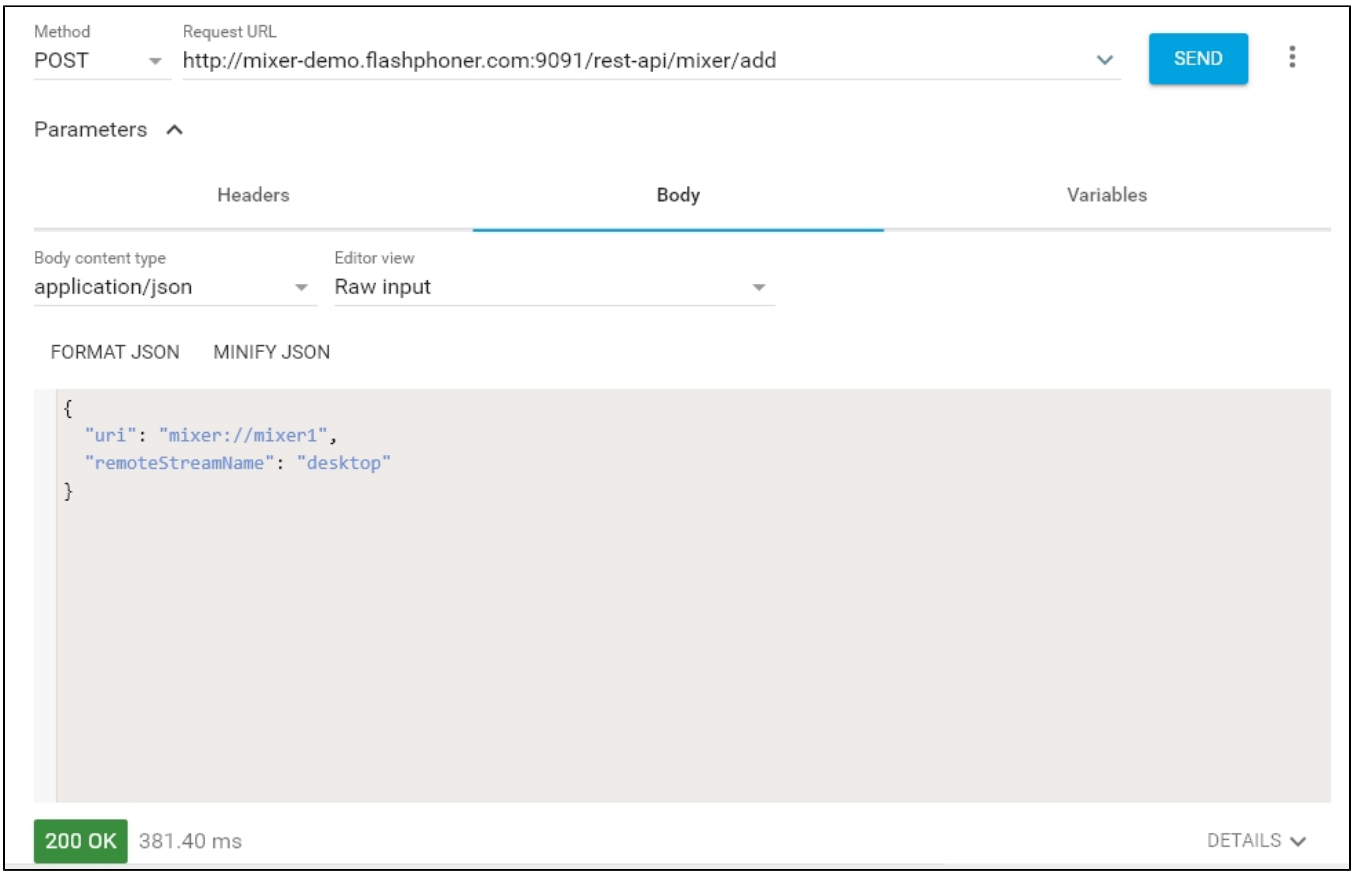
5. Send the /mixer/add query and specify the URI of the mixer mixer://mixer1 and the input stream name stream1 in its parameters:



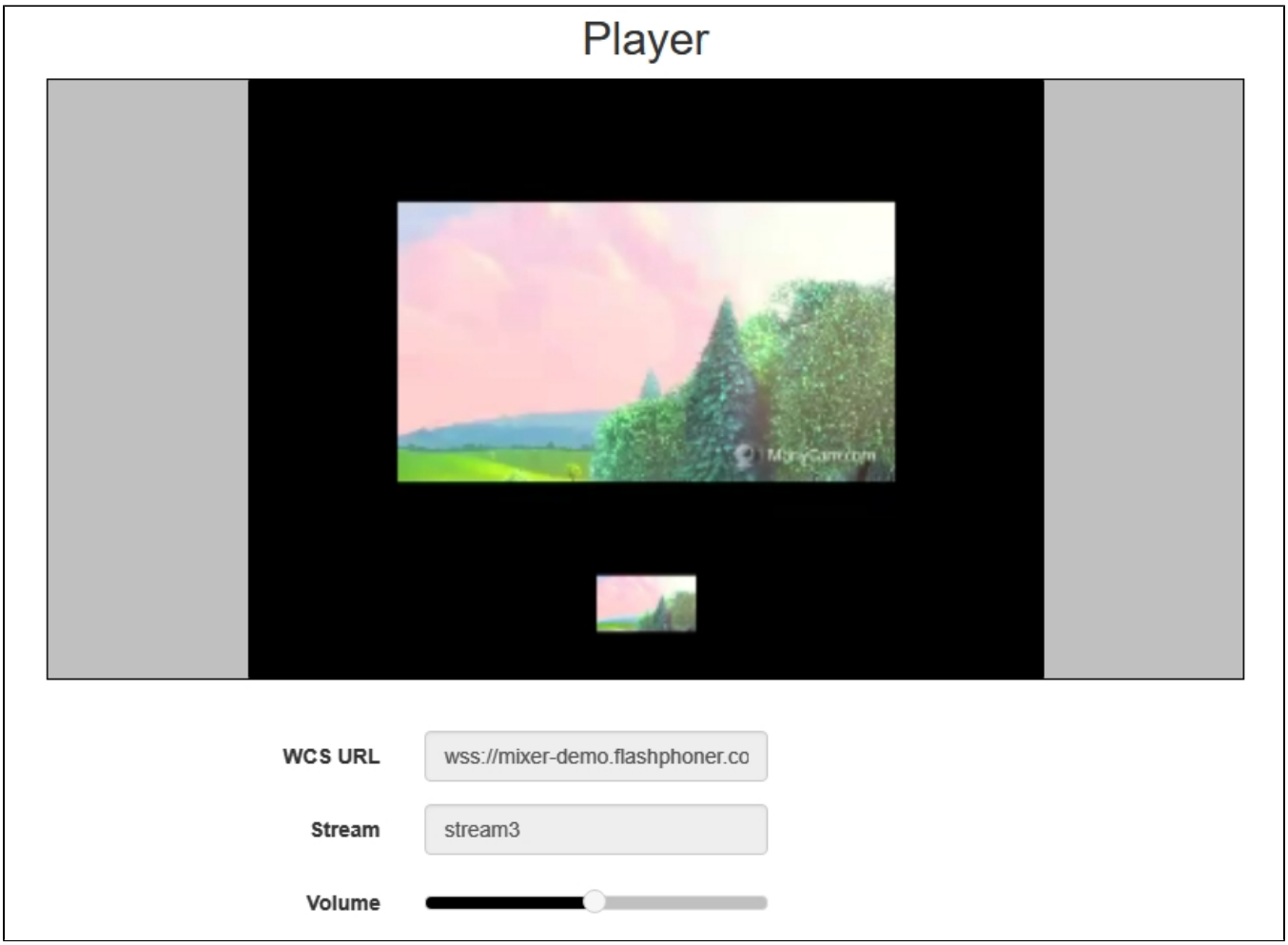
6. Open the Player web application, specify the name of the output stream of the mixer stream3 in the Stream field and click Start:



7. Send /mixer/add and specify the URI of the mixer mixer://mixer1andthe input stream name desktop in its parameters:

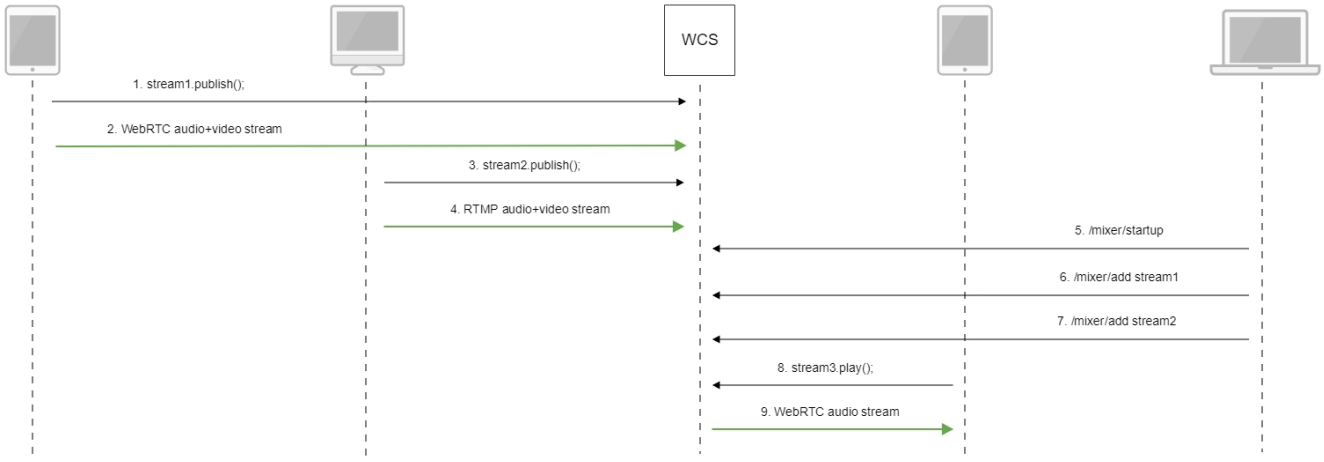


8. In the output stream of the mixer you should see the desktop stream that imitates screen sharing and the stream stream1:



Call flow

Below is the call flow when using the mixer.



1. Publishing of the WebRTC stream stream1
2. Sending the WebRTC stream to the server
3. Publishing the RTMP stream stream2

4. Sending the RTMP stream to the server

5. Sending the /mixer/startup query to create the mixer://stream3 mixerwith the output stream3

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

6. Sending the /mixer/add query to add stream1 to the mixer://stream3 mixer

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

7. Sending the /mixer/add query to add stream2 to the mixer://stream3 mixer

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

8. Playing the WebRTC stream stream3

9. Sending the WebRTC audio stream to the client

Mixer REST hooks

By default, all the standard REST hooks are invoked for mixer stream. The /connect REST hook is invoked while creating a mixer.

A separate server application can be created for mixer REST hooks handling if necessary. To send REST hooks to this application, the following parameter can be used since build5.2.634

```
mixer_app_name=defaultApp
```

By default, mixer REST hooks are handled by defaultApp like another streams REST hooks.

REST hook for adding stream to mixer or removing it

Since build 5.2.1416 it is possible to receive an events about a certain stream is added or removed to/from mixer. WCS sends to a backend server the REST hook `/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"
  }
}
```

when stream is added to a mixer and

```
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"
  }
}
```

when stream is removed from mixer.

The `StreamEvent` method must be added to a backend application configuration when updating WCS from previous builds

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

Known issues

1. A mixer is not created is the name of the mixer contains symbols restricted for URI.

Symptoms: a mixer with the name like test_mixer does not create.

Solution: do not use disallowed symbols in the name of a mixer or a stream, especially if automatic mixer creation option is enabled. For instance, the name

```
user_1#my_room
```

cannot be used.
If streams of chat rooms are mixed, room names also cannot use restricted symbols.

2. Mixer output stream will be empty if transcoding is enabled on server on demand only.

Symptoms: video streams mixer created successfully, but black screen is played in mixer output stream.

Solution: for stream mixer to work transcoding should be enabled on server with the following parameterinflashphoner.propertiesfile

```
streaming_video_decoder_fast_start=true
```

3. The same stream cannot be added to two or more mixers if non-realtime mixer is used

Symptoms: when the stream is added to second mixer, the first mixer output stream stops

Solution: do not use the same stream in more than one mixer

4. Encoding quality settings cannot be applied if OpenH264 is used

Symptoms: picture quality is not changing when using different `mixer_video_quality` values, for example

```
mixer_video_quality=5
```

does not differ from

```
mixer_video_quality=20
```

Solution: do not use OpenH264 encoder because it does not support CRF

```
encoder_priority=FF
```

5. Default watermark or background (black picture) will be used if PNG file is damaged or is not a PNG

Symptoms: black picture in output stream when watermark is added, there is a message in server log

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

output stream background is still black, there is a message in server log

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

Solution: use only correct PNG file to add watermark or background