Stream mixer

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.

Supported protocols of input streams

- WebRTC
- RTMP
- RTSP

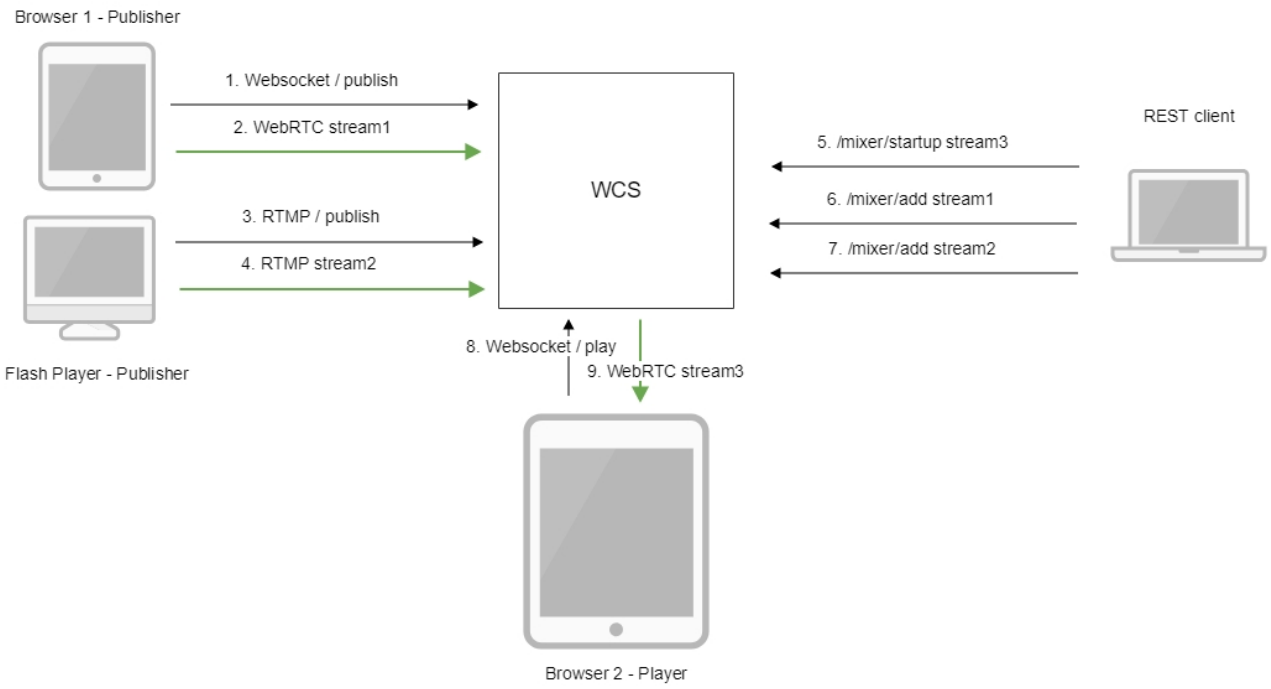
Output stream control capabilities

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	`{ "uri": "mixer://mixer1", "localStreamName": "stream3", "hasVideo": "false" }`		200 - OK 409 - Conflict 500 - Internal error	Creates a mixer the provided stream is published for

/mixer/add	`{ "uri": "mixer://mixer1", "remoteStreamName": "stream1" }`		200 - OK 404 - Mixer not found 404 - Stream not found 500 - Internal error	Add the RTMP stream to the mixer
/mixer /remove	`{ "uri": "mixer://mixer1", "remoteStreamName": "stream1" }`		200 - OK 404 - Mixer not found 404 - Stream not found 500 - Internal error	Remove the RTMP stream from the mixer
/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	Find all mixers
/mixer /terminate	`{ "uri": "mixer://mixer1" }`		200 - OK 404 - Not found 500 - Internal error	Terminate operation of the mixer
/stream /startRecording	`{ "mediaSessionId": "23d07fa1-3c74-4d6f-a0de-9b4bf83ce665" }`		200 - OK 404 - Not found 500 - Internal error	Start recording of the stream in the given media session
/stream /stopRecording	`{ "mediaSessionId": "23d07fa1-3c74-4d6f-a0de-9b4bf83ce665" }`		200 - OK 404 - Not found 500 - Internal error	Stop recording the stream in the given media session

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) or audio only (false)	false

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

Sending the REST query to the WCS server

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

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
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
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.

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 bandwith 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 outout stream bitrate to 3-5 Mbps

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

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 syncronizing 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 realtime perfomance.

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

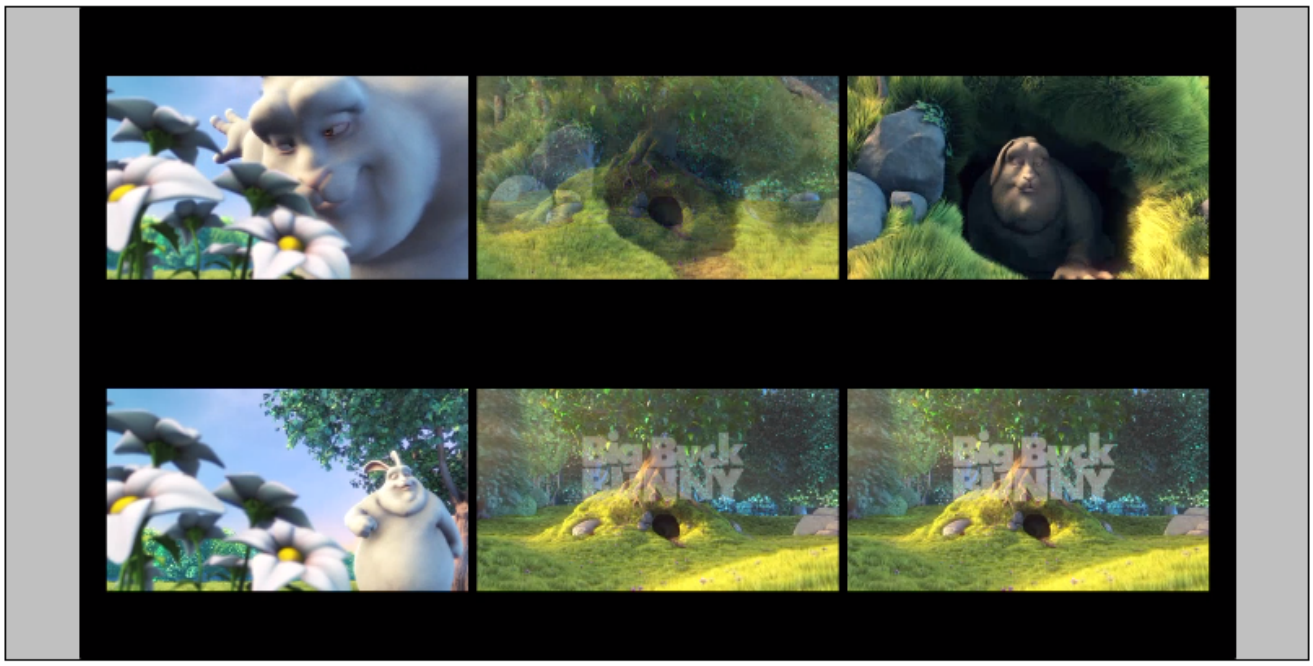
```
mixer_idle_timeout=60000
```

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:

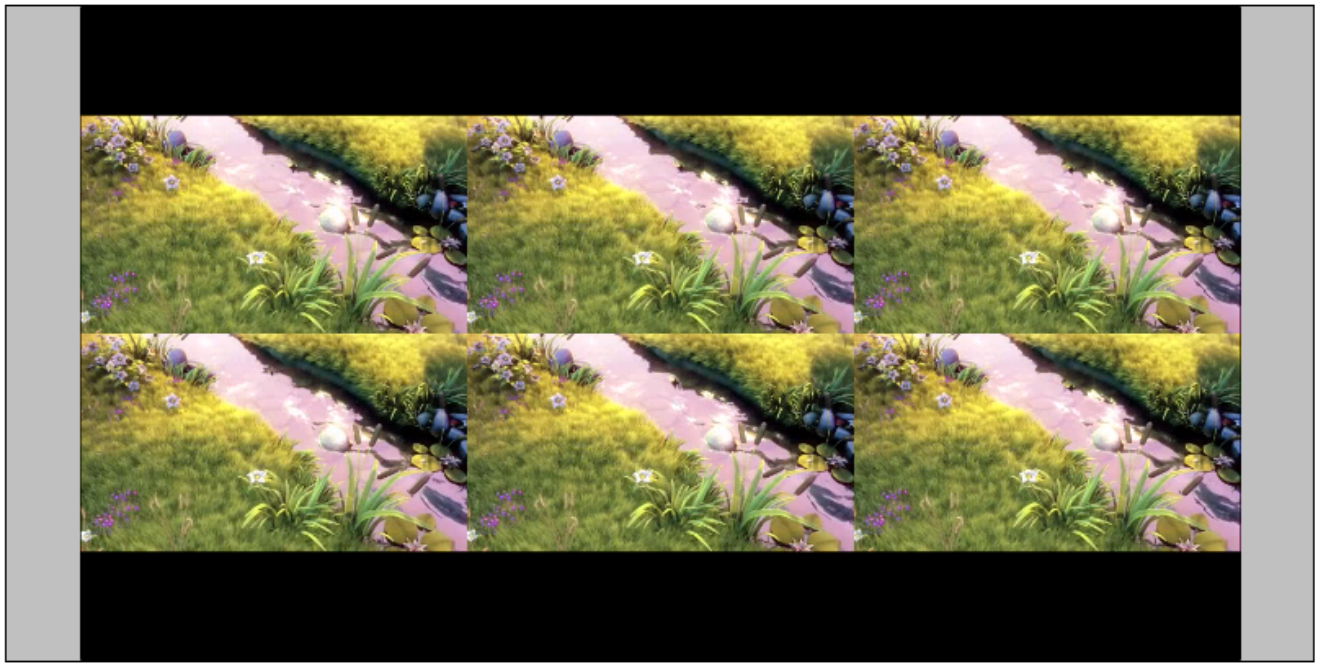
1. Grid layout



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

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

2. 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

3. Desktop (screen sharing) layout

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

```
mixer_video_layout_desktop_key_word=desktop
```

By default, `desktop` name is used for screen sharing stream.

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 com.flashphoner.mixerlayout;

import com.flashphoner.sdk.media.IVideoMixerLayout;
import com.flashphoner.sdk.media.YUVFrame;
import java.awt.*;
import java.util.ArrayList;

public class TestLayout implements IVideoMixerLayout {

    private static final int PADDING = 5;

    @Override
    public Layout[] computeLayout(YUVFrame[] yuvFrames, String[] strings, int canvasWidth, int canvasHeight) {
        ArrayList<IVideoMixerLayout.Layout> layout = new ArrayList<>();
        for (int c = 0; c < yuvFrames.length; c++) {
            Point prevPoint = new Point();
            Dimension prevDimension = new Dimension(canvasWidth, canvasHeight);
            if (layout.size() > 0) {
                prevPoint.setLocation(layout.get(c-1).getPoint());
                prevDimension.setSize(layout.get(c-1).getDimension());
            }
            Point currentPoint = new Point((int) (prevPoint.getX()+prevDimension.getWidth()+PADDING),
                                           (int)(prevPoint.getY()+prevDimension.getHeight()));
            layout.add(new IVideoMixerLayout.Layout(currentPoint, new Dimension(canvasWidth/yuvFrames.length,
                                                                  canvasHeight/yuvFrames.length), yuvFrames[c]));
        }
        return layout.toArray(new IVideoMixerLayout.Layout[layout.size()]);
    }
}
```

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

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

and execute the command

```
javac -cp /usr/local/FlashphonerWebCallServer/lib/tbs-flashphoner.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.

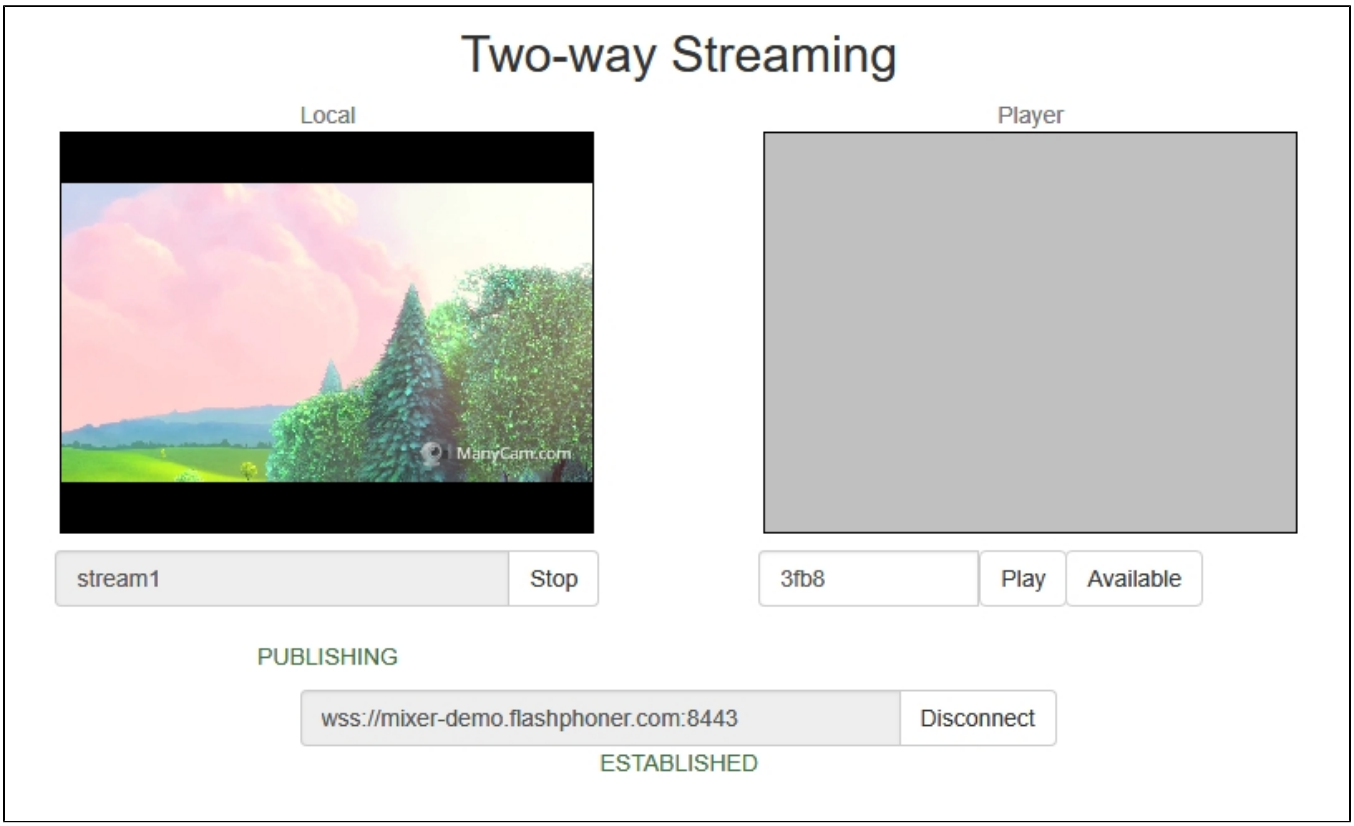
With this custom layout, mixer output stream for three input streams will look like:

Quick manual on testing

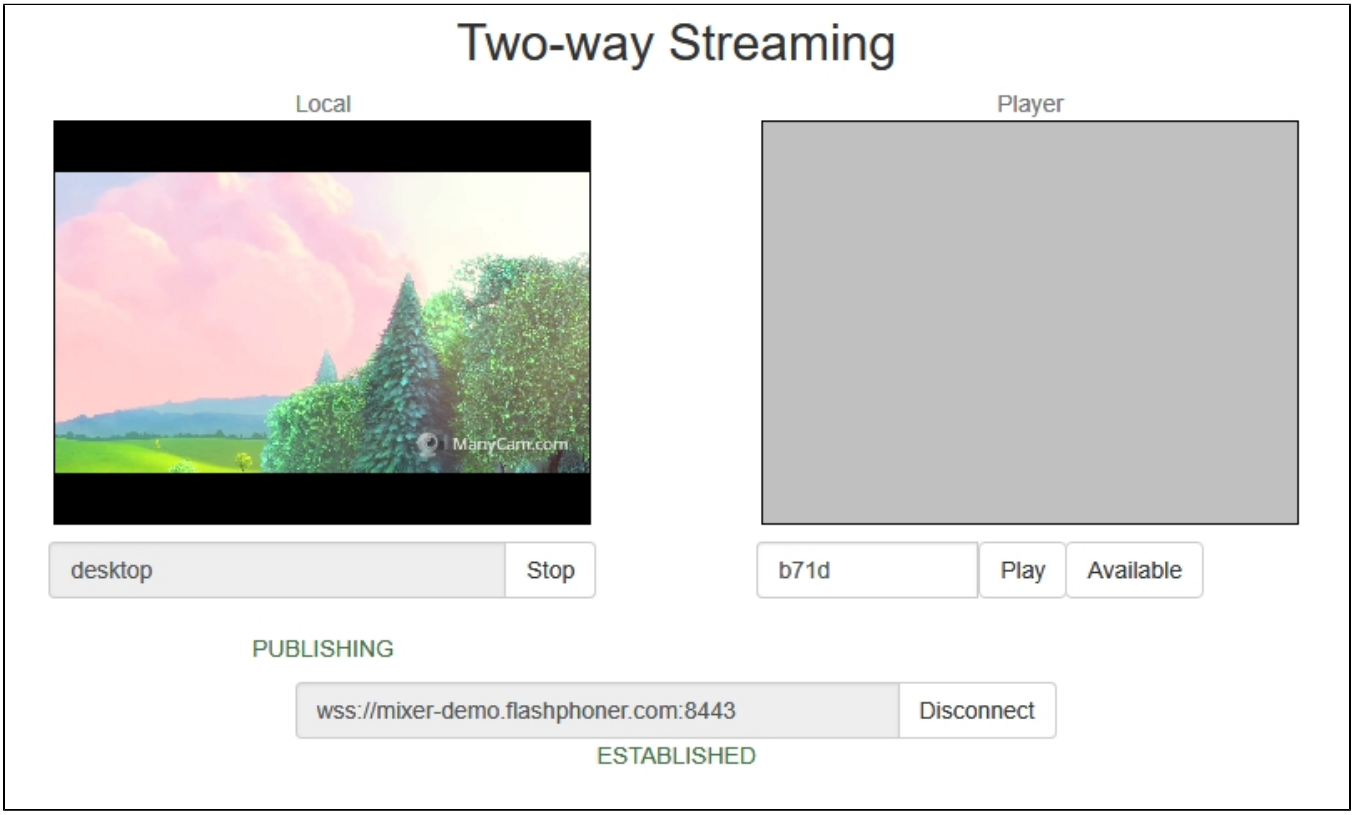
1. For this test we use:

- 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:

Method
POST

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

SEND

Parameters

Headers | Body | Variables

Body content type
application/json

Editor view
Raw input

FORMAT JSON MINIFY JSON

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

200 OK 411.20 ms

DETAILS

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

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

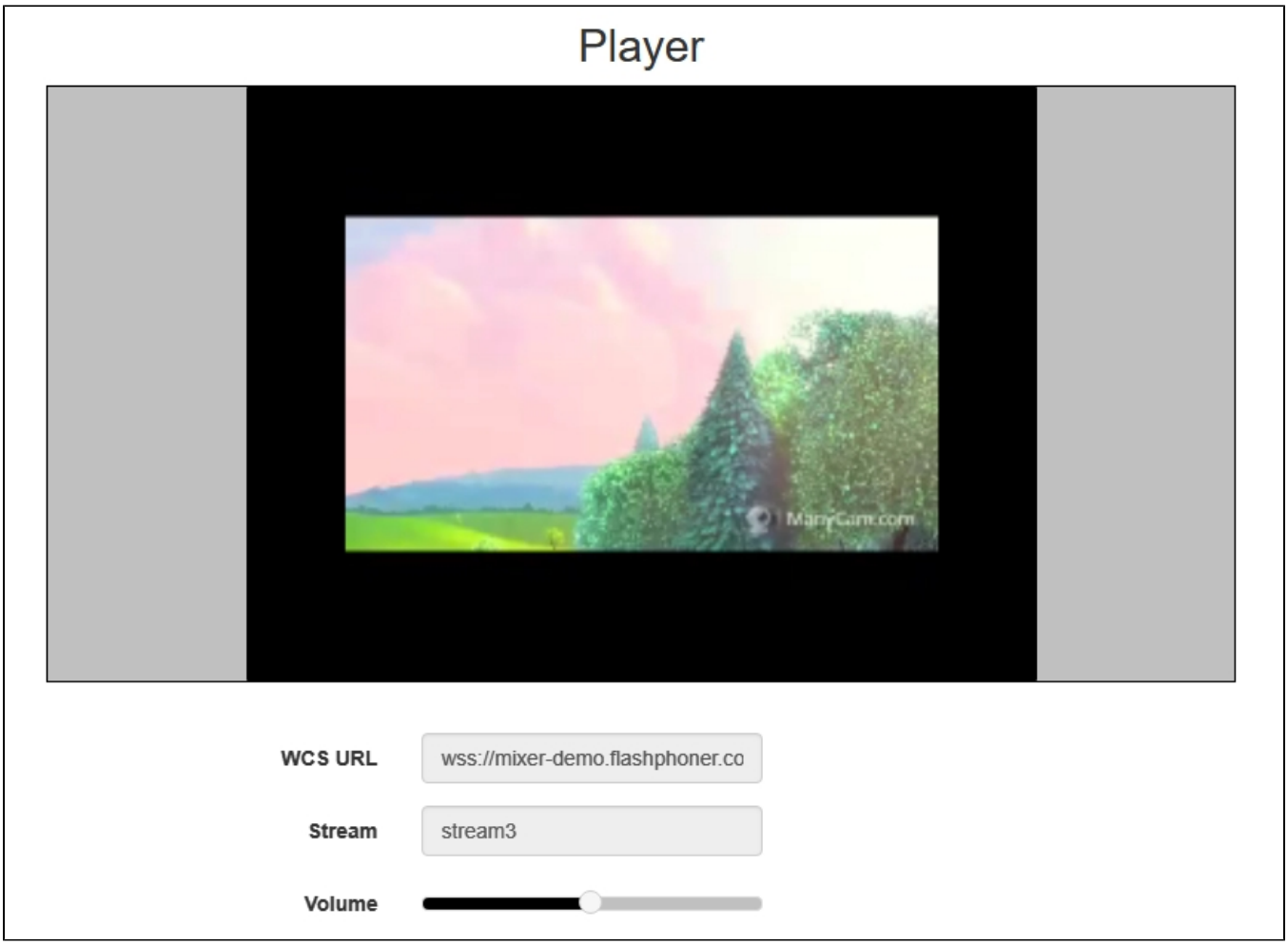
FORMAT JSON MINIFY JSON

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

200 OK 396.40 ms

DETAILS

6. 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://mixer1 and the input stream name desktop in its parameters:

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

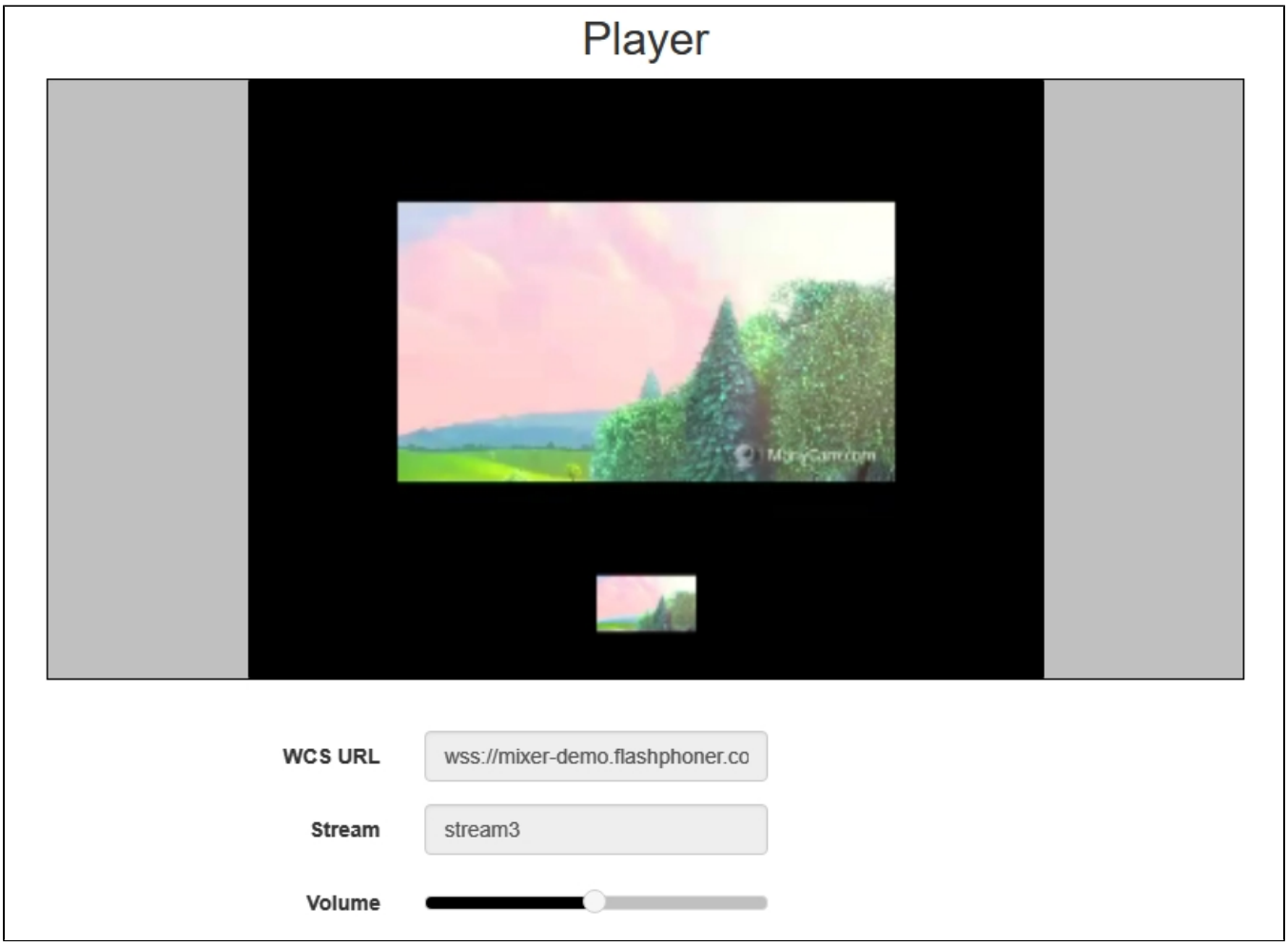
FORMAT JSON MINIFY JSON

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

200 OK 381.40 ms

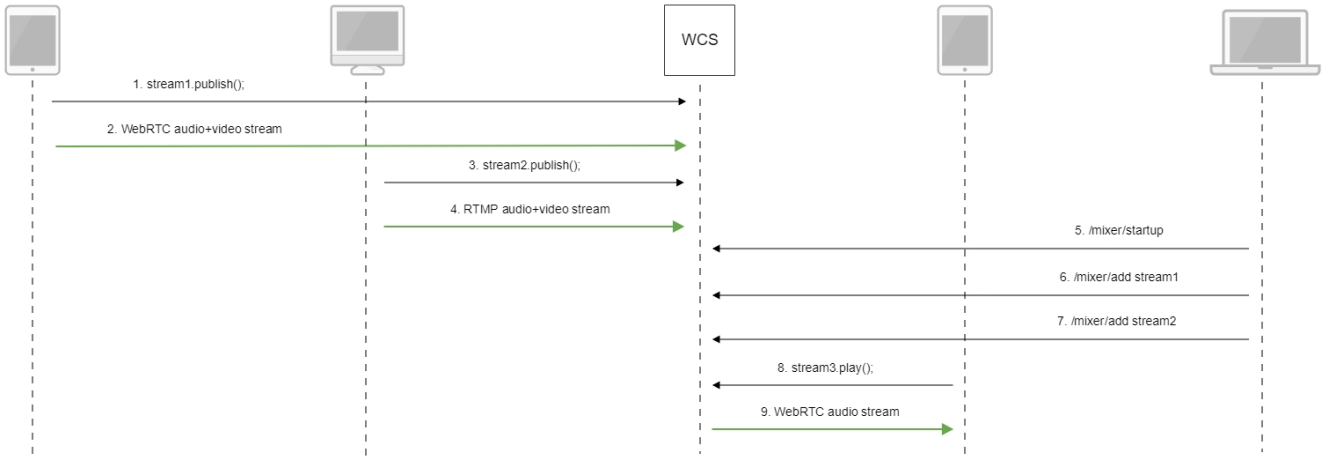
DETAILS

8. 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 mixer with 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

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
```