Injecting one stream into another

Overview

Since build 5.2.841 it is possible to iject one stream published on server into another. This feature can be used, for example, to add advertising material into a stream. The original stream content will be fully replaced by injected stream one until injected stream is stopped or injection is terminated.

Supported codecs

Video:

- H264
- VP8

Audio:

- Opus
- AAC
- G711

Known limits

1. Both streams to which injection is applied must be encoded with the same audio and video codecs.
2. Audio tracks in both streams should have the same samplerate and channels number.
3. Injection cannot be applied to SIP call streams. Use the special audio and video injection technologies for SIP call streams.
4. Only one stream can be injected into the stream simultaneously, but one stream can be injected into multiple streams.
5. Cyclic injection is not supported. It is not possible to inject `stream1` into `stream2` and then `stream2` into `stream1` without terminating the previous injection.

Injection implementation in builds before 5.2.1618

REST API

REST query must be HTTP/HTTPS POST request as follows:

- HTTP: `http://test.flashphoner.com:8081/rest-api/stream/inject/startup`
- HTTPS: `https://test.flashphoner.com:8444/rest-api/stream/inject/startup`

Where:

- `test.flashphoner.com` - WCS server address
- `8081` - standard REST / HTTP port of WCS server
- `8444` - standard HTTPS port
- `rest-api` - mandatory URL part
- `/stream/inject/startup` - REST method used

REST methods and responses

/stream/inject/startup

Inject stream2 into stream1

REQUEST EXAMPLE

```
POST /rest-api/stream/inject/startup HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
    "localStreamName": "stream1",
    "remoteStreamName": "stream2"
}
```

RESPONSE EXAMPLE

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

RETURN CODES

Code	Reason
200	OK
400	Bad request
404	Not found
409	Conflict
500	Internal error

/stream/inject/find_all

Find all injections on the server

REQUEST EXAMPLE

```
POST /rest-api/stream/inject/find_all HTTP/1.1
Host: localhost:8081
Content-Type: application/json
```

RESPONSE EXAMPLE

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

[
    {
        "localStreamName": "stream1",
        "remoteStreamName": "stream2"
    }
]
```

RETURN CODES

Code	Reason
200	OK
404	Not found

`stream/inject/terminate

Stop injection into stream1

REQUEST EXAMPLE

```
POST /rest-api/stream/inject/terminate HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
    "localStreamName": "stream1"
}
```

RESPONSE EXAMPLE

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

RETURN CODES

Code	Reason
200	OK
400	Bad request
404	Not found
500	Internal error

Parameters

Name	Description	Example
localStreamName	Stream name to inject to	`stream1`
remoteStreamName	Stream name to be injected	`stream2`

Injecting a VOD stream from file

Since build 5.2.1535 VOD stream directly from a file may be injected while sending the REST query `/stream/inject/startup`:

```
{
    "localStreamName":"host",
    "remoteStreamName":"vod-live://advertising.mp4"
}
```

In this case, injected file will play without a delay from the first key frame. The file can be injected to another stream, in this case the file also will be played from the beginning in that stream.

This feature is useful, for example, to inject advertising video into a stream being viewed.

Configuration

Since build 5.2.1235 the parameter is added to set a time interval to wait for a keyframe in injected stream

```
inject_wait_keyframe_ms=1000
```

By default, the interval is 1000 milliseconds. If no keyframes arrived in injected stream during this time, server will generate a black picture (by default) or a picture from a file set by `custom_watermark_filename` parameter. This behaviour may be switched off by the following parameter

```
inject_wait_keyframe_ms=-1
```

In this case, the stream to be injected to will be played until keyframe arrives in the injected stream.

Injection implementation in build 5.2.1618 and newer

Configuration

Since build 5.2.1618 a new injector implementation is added allowing to choose what exactly to inject: audio, video or both. The feature may be enabled by the following parameter

```
use_new_injector=true
```

REST API

REST query must be HTTP/HTTPS POST request as follows:

- HTTP: `http://test.flashphoner.com:8081/rest-api/stream/inject2/startup`
- HTTPS: `https://test.flashphoner.com:8444/rest-api/stream/inject2/startup`

Where:

- `test.flashphoner.com` - WCS server address
- `8081` - standard REST / HTTP port of WCS server
- `8444` - standard HTTPS port
- `rest-api` - mandatory URL part
- `/stream/inject2/startup` - REST method used

REST methods and responses

/stream/inject2/startup

Inject stream2 into stream1

REQUEST EXAMPLE

```
POST /rest-api/stream/inject2/startup HTTP/1.1
Host: localhost:8081
Content-Type: application/json
```

```
{
    "localStreamName": "stream1",
    "remoteStreamName": "stream2",
    "video": true,
    "audio": true,
    "muteIfAbsent": true
}
```

RESPONSE EXAMPLE

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

RETURN CODES

Code	Reason
200	OK
400	Bad request
404	Not found
409	Conflict
500	Internal error

/stream/inject2/find_all

Find all injections on the server

REQUEST EXAMPLE

```
POST /rest-api/stream/inject2/find_all HTTP/1.1
Host: localhost:8081
Content-Type: application/json
```

RESPONSE EXAMPLE

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

[
  {
    "streamName": "test",
    "videoInjectorInfo": {
      "targetStreamName": "test2",
      "rootStreamName": "test2",
      "startTime": 1683344295099
    },
    "audioInjectorInfo": {
      "targetStreamName": "test2",
      "rootStreamName": "test2",
      "startTime": 1683344295056
    }
  }
]
```

RETURN CODES

Code	Reason
200	OK
404	Not found

/stream/inject2/terminate

Stop injection into stream1

REQUEST EXAMPLE

```
POST /rest-api/stream/inject2/terminate HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
    "localStreamName": "stream1",
    "video": true,
    "audio": true
}
```

RESPONSE EXAMPLE

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

RETURN CODES

Code	Reason
200	OK
400	Bad request
404	Not found
500	Internal error

Parameters

Name	Description	Example
localStreamName	Stream name to inject to	`stream1`
remoteStreamName	Stream name to be injected	`stream2`
video	Replace video when injecting	`true`
audio	Replace audio when injecting	`true`
muteIfAbsent	Replace a track which is absent in a source stream to black picture or silence	`true`
videoInjectorInfo	Video information from injected stream	`{ "targetStreamName": "stream2", "rootStreamName": "stream2", "startTime": 1683344295099 }`
audioInjectorInfo	Audio information from injected stream	`{ "targetStreamName": "stream2", "rootStreamName": "stream2", "startTime": 1683344295056 }`

Injecting a VOD stream from file

Since build 5.2.1719 VOD stream directly from a file may be injected while sending the REST query `/stream/inject2/startup`:

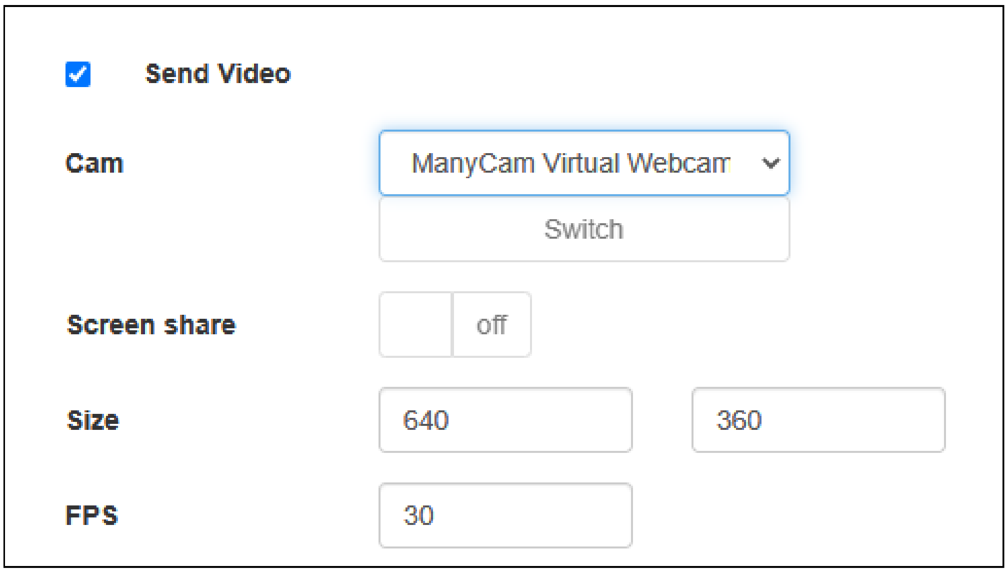
```
{
    "localStreamName":"host",
    "remoteStreamName":"vod-live://advertising.mp4",
    "video":true,
    "audio":true
}
```

In this case, injected file will play without a delay from the first key frame. The file can be injected to another stream, in this case the file also will be played from the beginning in that stream.

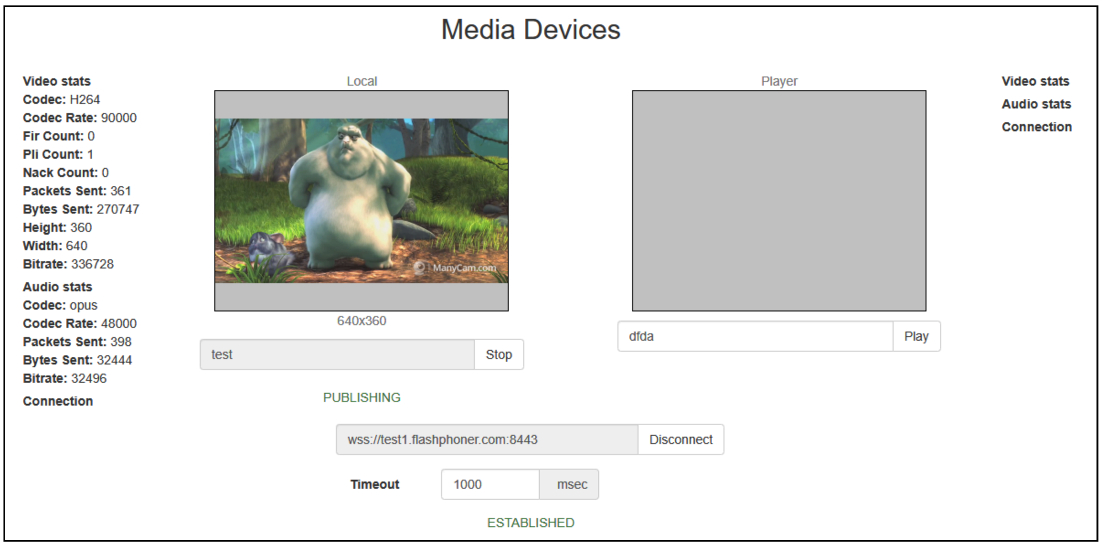
This feature is useful, for example, to inject advertising video into a stream being viewed.

Quick testing

1. For test we use:
   - WCS server;
   - Media Devices web application to publish stereams;
   - Two webcams, or two different PCs to publish streams;
   - Player web application to play stream to be injected to;
   - Chrome browser and REST client to send queries to the server
2. Open Media Devices aplication page, publish stream `test` in resolution 640x360



✓ Send Video		
Cam	ManyCam Virtual Webcam	
	Switch	
Screen share	off	
Size	640	360
FPS	30	



Media Devices

Video stats
Codec: H264
Codec Rate: 90000
Fir Count: 0
Pli Count: 1
Nack Count: 0
Packets Sent: 361
Bytes Sent: 270747
Height: 360
Width: 640
Bitrate: 336728

Audio stats
Codec: opus
Codec Rate: 48000
Packets Sent: 398
Bytes Sent: 32444
Bitrate: 32496

Connection

Local: 640x360, test, PUBLISHING

Player: dfda

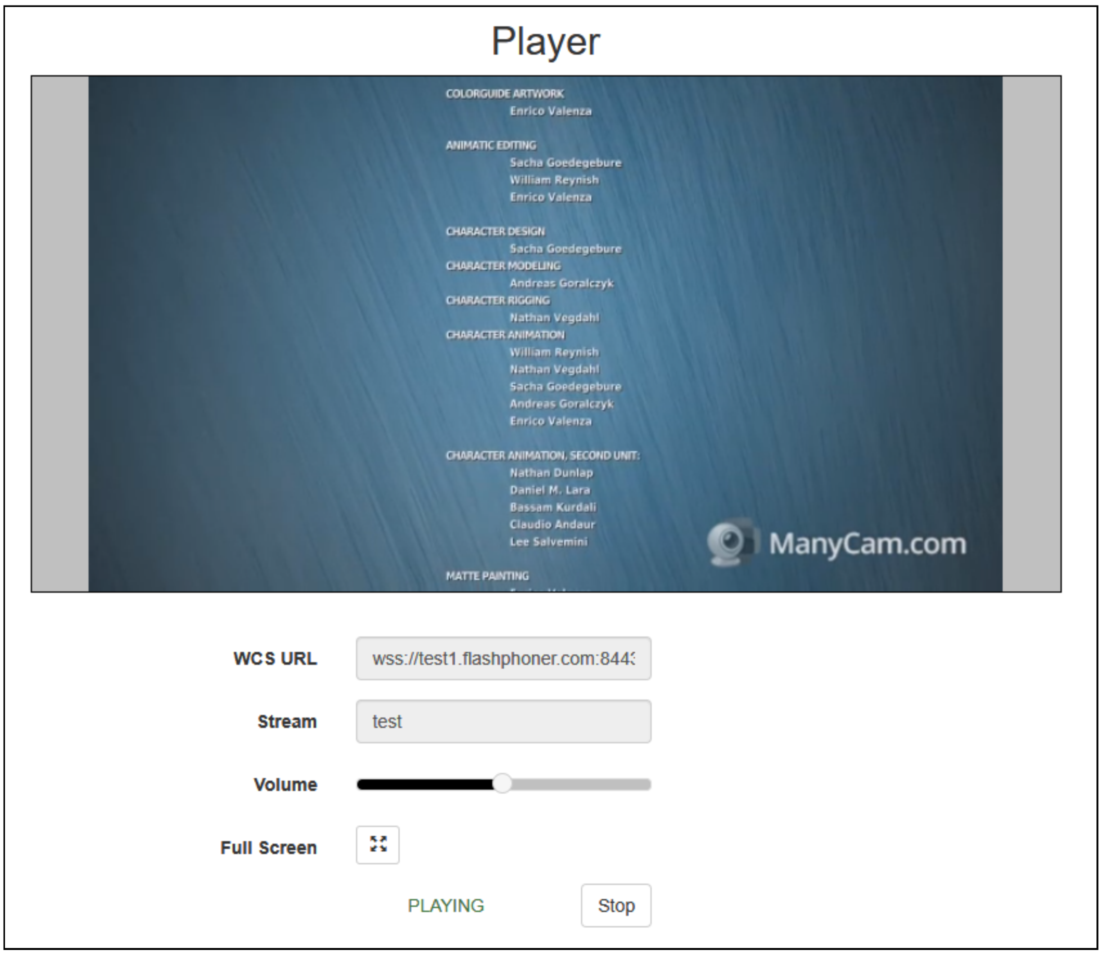
Video stats
Audio stats
Connection

wss://test1.flashphoner.com:8443

Timeout: 1000 msec

ESTABLISHED

3. Play the stream `test` in Player example



Player

WCS URL: wss://test1.flashphoner.com:8443

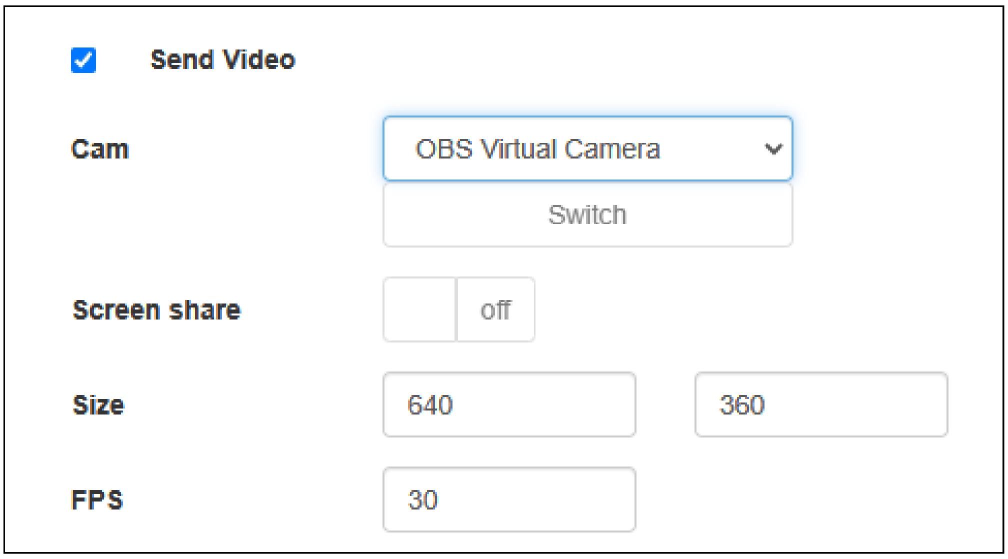
Stream: test

Volume

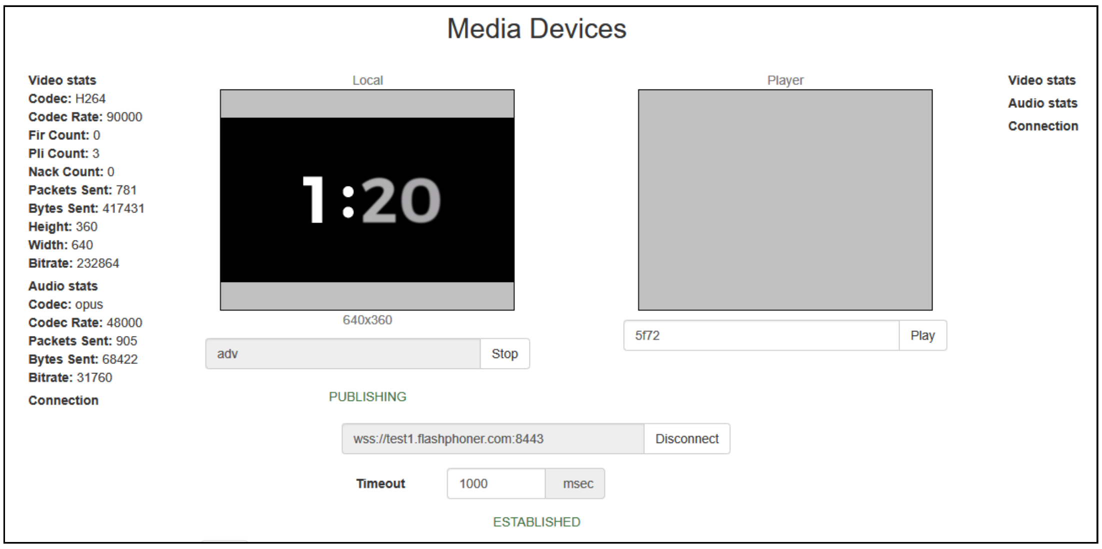
Full Screen

PLAYING

4. Publish `adv` stream in Media Devices example using another browser tab, another webcam or another PC



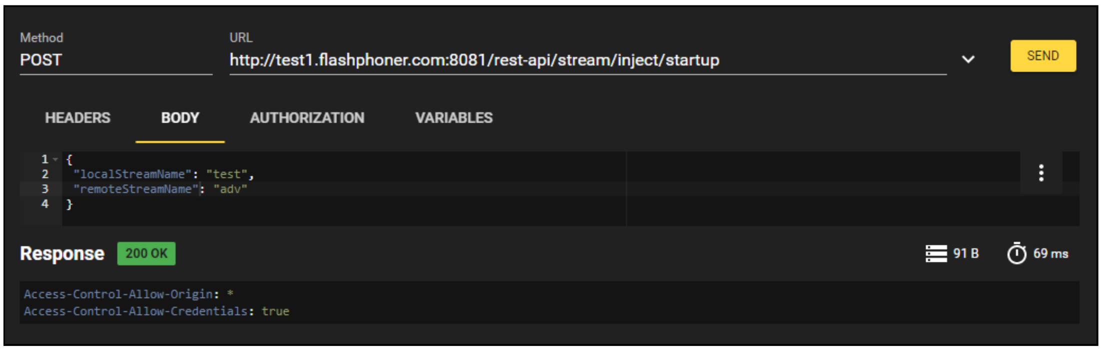
✓ Send Video		
Cam	OBS Virtual Camera	
	Switch	
Screen share	off	
Size	640	360
FPS	30	



Media Devices

Video stats
Codec: H264
Codec Rate: 90000
Fir Count: 0
Pli Count: 3
Nack Count: 0
Packets Sent: 781
Bytes Sent: 417431
Height: 360
Width: 640
Bitrate: 232864
Audio stats
Codec: opus
Codec Rate: 48000
Packets Sent: 905
Bytes Sent: 68422
Bitrate: 31760
Connection

5. Open REST client, send `/stream/inject/startup` query



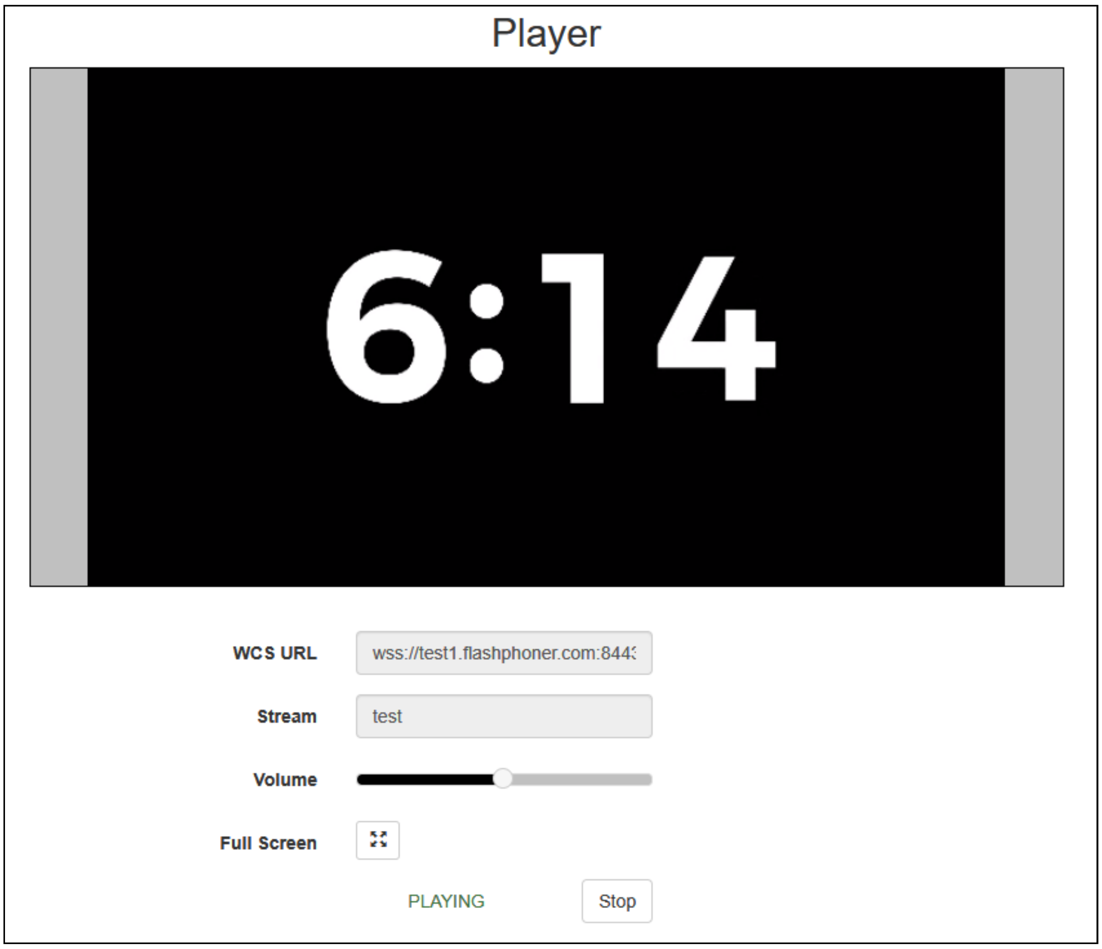
Method: POST
URL: http://test1.flashphoner.com:8081/rest-api/stream/inject/startup

```
{
 "localStreamName": "test",
 "remoteStreamName": "adv"
}
```

Response 200 OK 91 B 69 ms

```
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
```

6. `adv` stream content is playing in `test` stream

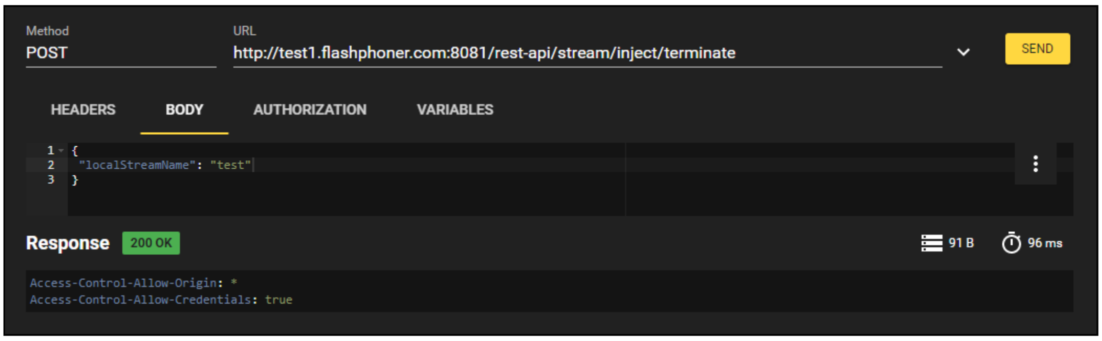


Player

WCS URL	wss://test1.flashphoner.com:844
Stream	test
Volume	
Full Screen	

PLAYING Stop

7. Send `/stream/inject/terminate` query



Method: POST

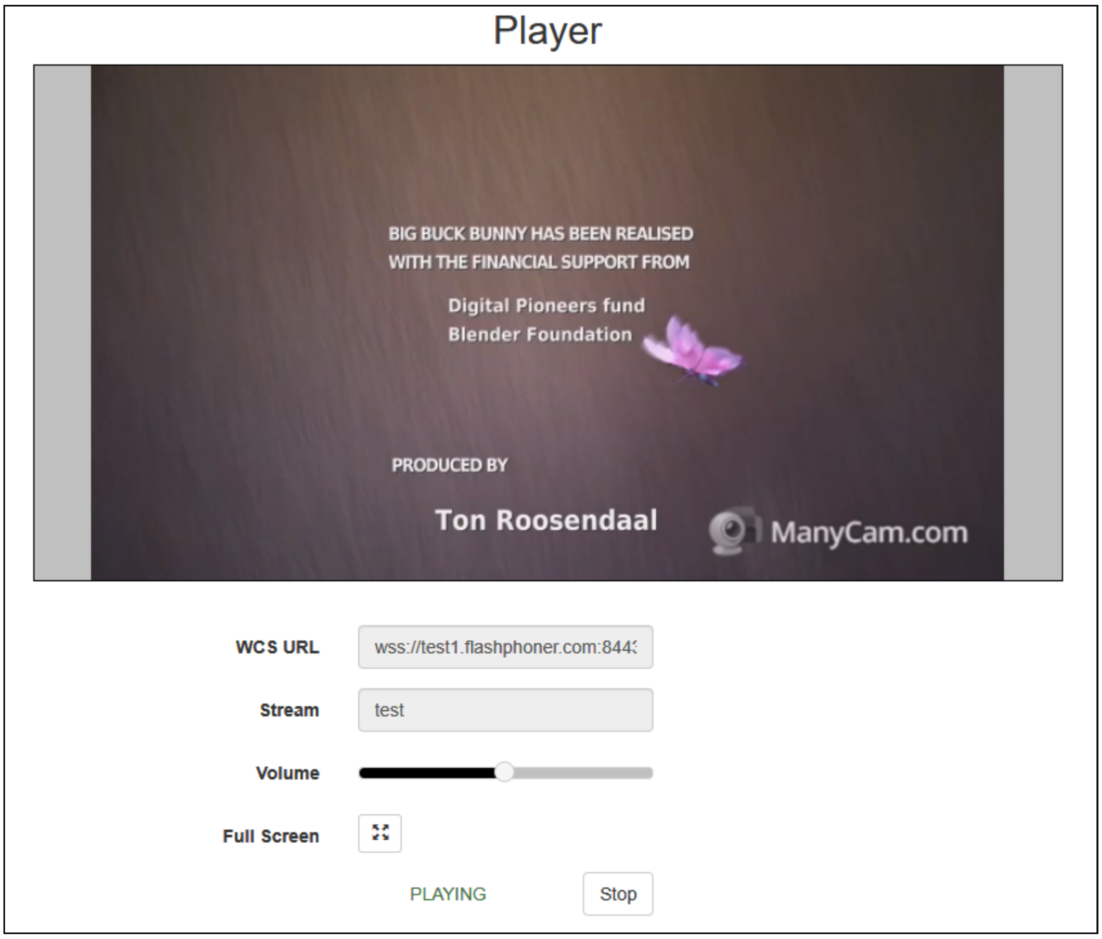
URL: http://test1.flashphoner.com:8081/rest-api/stream/inject/terminate

```
{
 "localStreamName": "test"
}
```

Response 200 OK 91 B 96 ms

```
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
```

8. Original `test` stream content is playing again



WCS URL: wss://test1.flashphoner.com:844

Stream: test

Known issues

1. Video and audio may be out of sync after stopping injection of one RTMP stream into another

Symptoms

When one RTMP stream is injected into another, the original RTMP stream may play with a strong audio/video unsync after injected stream stops

Solution

Enable RTMP incoming streams bufferization

```
rtmp_in_buffer_enabled=true
```