Stream recording

- Overview
- Quick manual on testing
    - Stream recording
- Configuration
    - Server side
        - Turning stream recording on and off
        - MP4, WebM, MKV containers support
        - Recording to MPEG-TS container
            - Known limits
        - Forming the name of the stream record file
        - Record files rotation
        - Recording start time, end time and duration time calculation
        - Record files handling script
        - Directory for saving recorded files
        - Adjusting record audio sample rate
        - moov atom placement configuration in recording metadata
        - Audio track bitrate configuration for FDK AAC codec
        - Audio channels configuration
        - Recording perfomance tuning under high load
        - VP8 streams recording to webm container
        - A separate folder for temporary files
        - Minimal available disk space checking
        - Recording stopping on errors
    - Client side
- Stream recording on demand
    - REST methods and response statuses
    - Parameters
- How to get recording file name
- How to play or download recording file
    - Downloading and playing a certain part of a recording file
        - Fragments folder configuration
        - Known limits
- Multiple stream recording to one file with subsequent mixing
    - REST methods and responses
        - Parametes
    - Recording file name
    - Multiple streams recording folder
    - Multiple streams mixing tool
    - Getting tracks information from multiple recording file
    - Multiple recording file handling script
    - Multithreaded encoding while mixing multiple stream recordings
    - Multiple recording data callback
    - Minimal available disk space checking for multiple recordings mixing
- Known issues

Overview

A media stream captured by WCS can be recorded during publishing.

Supported protocols:

- WebRTC
- RTMP
- RTSP

Recording formats:

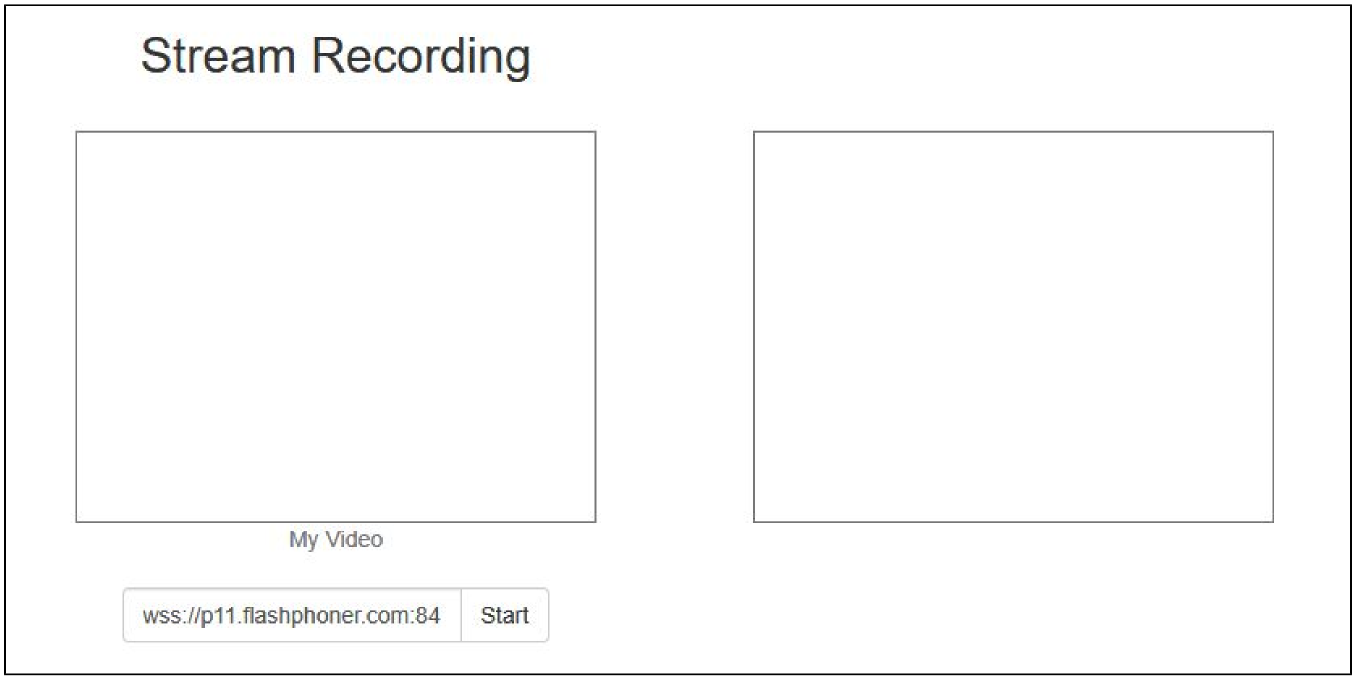
- MP4 for H.264 + AAC codecs
- WebM for VP8 + Vorbis codecs
- TS for H.264 + ADTS
- MKV (since WCS build 5.2.1190)

Quick manual on testing

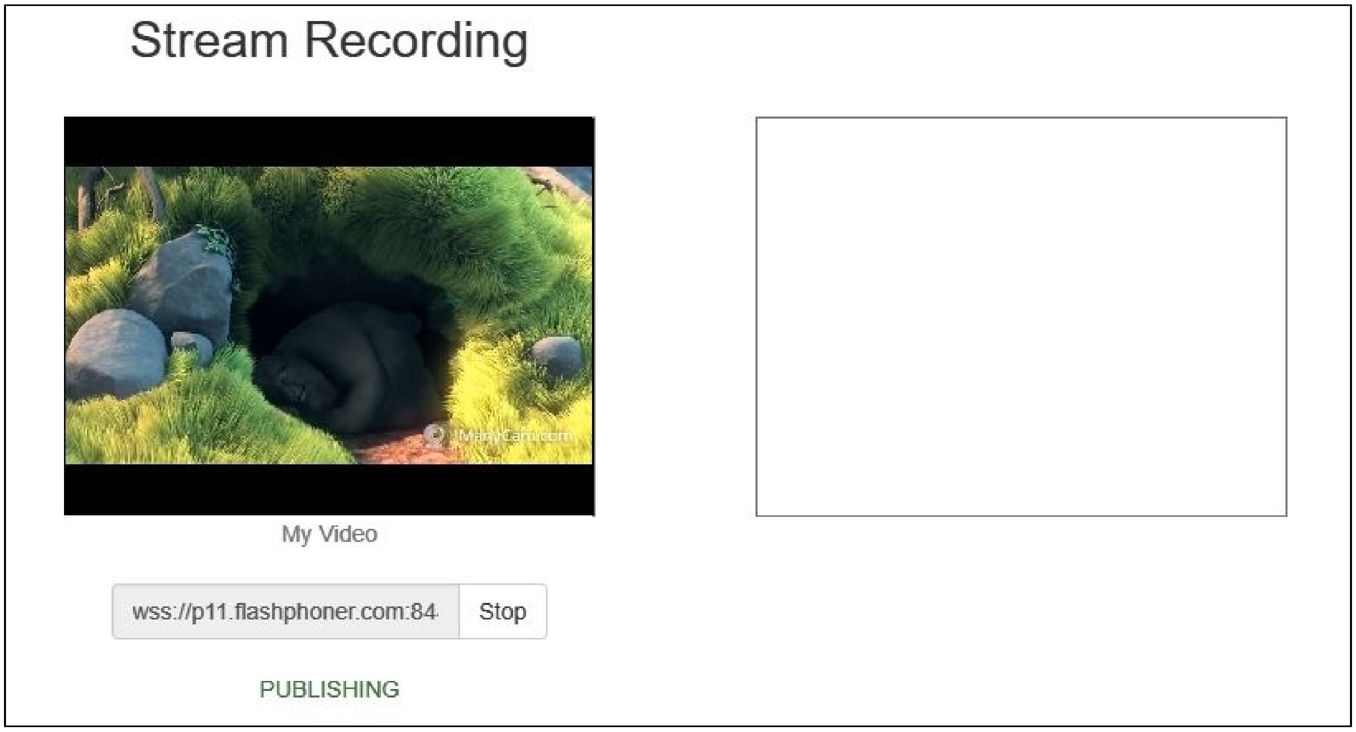
Stream recording

1. For this test we use the demo server at demo.flashphoner.com and the Stream Recording web application

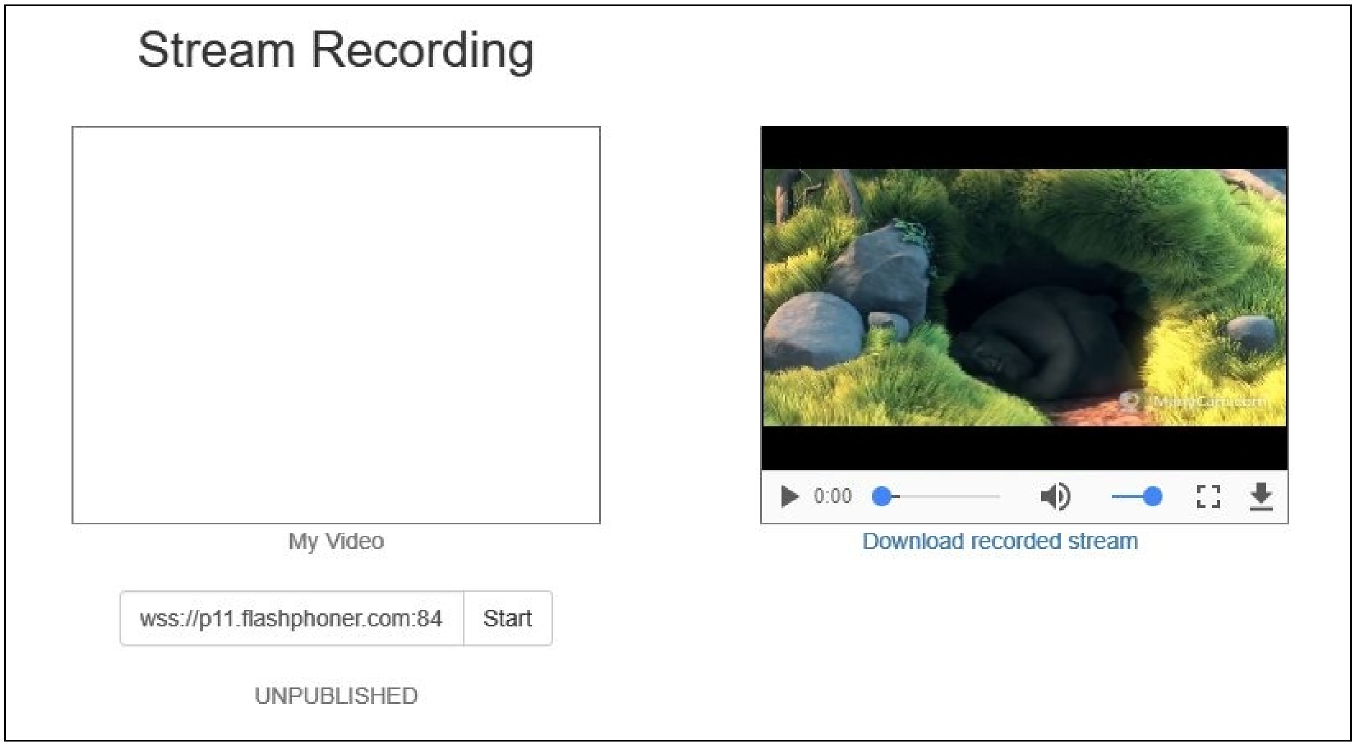
https://demo.flashphoner.com/client2/examples/demo/streaming/stream_recording/recording.html



2. Click the "Start" button. Capturing and publishing of the stream starts.



3. Click the "Stop" button. Broadcasting stops, and a link to play and download the recorded fragment appears.



Configuration

Server side

Turning stream recording on and off

By default, stream recording is turned on.
To turn recording off, add the following line to the config /usr/local/FlashphonerWebCallServer/conf/flashphoner.properties:

```
record_streams=false
```

Parameter

```
record_flash_published_streams=true
```

turns on recording for the streams published with Flash, RTMP encoder or republished from another RTMP server.

Parameter

```
record_rtsp_streams=true
```

turns on recording for the streams captured from RTSP IP cameras.

MP4, WebM, MKV containers support

By default, H264 streams are recorded to MP4 container and VP8 streams to WebM. Since WCS build 5.2.1190, MKV contaner is supported which is a swiss army knife at codecs point.

The following parameter is used to set recording containers

```
record_formats=h264-mp4,vp8-webm
```

MKV recording can be enabled as follows

```
record_formats=h264-mkv,vp8-mkv
```

One of the codecs published may be recorded to MKV container (VP8 for example)

```
record_formats=h264-mp4,vp8-mkv
```

Note that audio only streams are always recorded to MP4 container (using AAC codec).

Recording to MPEG-TS container

Since build 5.2.610, H264 streams can also be recorded to TS container using the following parameter

```
record_h264_to_ts=true
```

or since build 5.2.1190

```
record_formats=h264-ts,vp8-webm
```

The `record_h264_to_ts` parameter has a priority above `record_formats` , i.e.

```
record_formats=h264-mp4,vp8-mkv
record_h264_to_ts=true
```

makes H264 streams to be recorded to MPEG-TS.

Known limits

1. VLC before 3.0.8 can not play TS recordings.

2. TS recordings can not berewound while playing in VLC.

Forming the name of the stream record file

By default, the file name is formed by template specified with stream_record_policy_template parameter:

```
stream_record_policy_template=stream-{mediaSessionId}-{login}
```

The following elements can be used in template:

Element	Description	Maximum size
{streamName}	Stream name	
{duration}	File duration, for MP4 recordings only	
{startTime}	Recording start time	20 characters
{endTime}	Recording end time	20 characters
{startTimeMillis}	Recording start time based on server clock	20 characters
{endTimeMillis}	Recording end timebased on server clock	20 characters
{sessionId}	Session ID in BASE64 encoding	60 characters
{mediaSessionId}	Media session ID	36 characters
{login}	Login	32 characters
{audioCodec}	Audiocodec	4 characters
{videoCodec}	Videocodec	4 characters

For example,

```
stream_record_policy_template={streamName}
```

means that the file name will match the stream name. So, the stream published with ffmpeg

```
ffmpeg -re -i BigBuckBunny.mp4 -preset ultrafast -acodec aac -vcodec h264 -strict -2 -f flv rtmp://test1.
flashphoner.com:1935/live/stream_ffmpeg
```

will be written to file stream_ffmpeg.mp4.

File extension is set depending on stream parameters and container used: mp4 for H264+AAC and webm for VP8+opus.

When the file name matches the stream name, it may contain characters that are not allowed in file names, slash '/' for example. In that case, the file name should be encoded using the parameter

```
encode_record_name=true,HEX
```

Then, the file name will be encoded with a hexadecimal number. The parameter

```
encode_record_name=true,BASE64
```

will encode the file name with BASE64 encoding.

Another way to escape invalid characters is to remove them usingexclude_record_name_characters parameter. By default

```
exclude_record_name_characters=/
```

For example, to remove colons, commas, periods and slashes set

```
exclude_record_name_characters=:.,/
```

Record files rotation

Stream records can be splitted to parts of a given duration using record_rotation parameter. For example, the setting

```
record_rotation=20
```

specifies a fragment duration as 10 seconds andsetting

```
record_rotation=10M
```

defines a fragment maximum volume as 10 megabytes.

If recording file name template contains {startTime} element, recording fragment start timestamp will be inserted into file name. If template contains {endTime} element, recording fragment end timestamp will be inserted into file name. For example, if the following settings are used

```
record_rotation=20
stream_record_policy_template={streamName}-{startTime}-{endTime}
```

the`test`stream recording fragments will be named as follows

```
test-1553577643961-1553577663988_1.mp4
test-1553577663989-1553577683997_2.mp4
test-1553577683997-1553577701626_3.mp4
...
```

Recording fragments are numbered continuously from 1. For every new mediasession (even if stream is published with the same name) indexing starts again, i.e. old fragments will be overwritten if static part of file name template is not unic (stream name only for example).

Indexing can be disabled if necessary with the following parameter

```
record_rotation_index_enabled=false
```

In this case recording fragments are not numbered and will be named exactly as template sets. If template does not provide unic names, old fragments may be overwritten.

Recording start time, end time and duration time calculation

Since build5.2.458, recording start time, end time and duration time are calculated by stream frames time stamps. note that RTMP stream timestamps are always starting from 0, but WebRTC publisher usually sets full time stamp by its own clock for ever frame.

```
stream_record_policy_template={streamName}-{startTime}-{endTime}-{duration}
```

Since build5.2.635it is possible to use start time and end time based on server clock

```
stream_record_policy_template={streamName}-{startTimeMillis}-{endTimeMillis}
```

In general, stream time stamps are different from server clock time stamps.

To calculate recording times more precisely, audio data should be buffered to keep synchronization. To do this, the following parameter is added

```
record_audio_buffer_max_size=100
```

By default, audio buffer size is set to 100 packets.

Record files handling script

The on_record_hook_script setting points to the shell script that is invoked when stream recording finishes.

The script is placed to the /usr/local/FlashphonerWebCallServer/bin folder by default:

```
on_record_hook_script=/usr/local/FlashphonerWebCallServer/bin/on_record_hook.sh
```

but it can be placed to any folder with any name, for example:

```
on_record_hook_script=/opt/on_record.sh
```

This script can be used to copy or move the stream record from the WCS_HOME/records directory to another location after recording completes.

Example:

```
STREAM_NAME=$1
SRC_FILE=$2
SRC_DIR="/usr/local/FlashphonerWebCallServer/records/"
REPLACE_STR="/var/www/html/stream_records/$STREAM_NAME-"
DST_FILE="${SRC_FILE/$SRC_DIR/$REPLACE_STR}"
sudo cp $SRC_FILE $DST_FILE
```

Here

- $1 - stream name
- $2 - absolute path and file name of the stream record
- when stream recording ends, the record file is copied to /var/www/html/stream_records/

It is necessary to take into account the length of the absolute file name (including folder path) that will be formed when copying record file. If the absolute name of the target file exceeds 255 characters limit, copy command will fail with error, so the handling script will not work as expected.

Since build5.2.801,WCS is running from 'flashphoner' user for security reasons. Therefore, if recording files are moved to other folder, it is necessary to allow writing to the folder.For example, if files are copied to/opt/mediafolder

```
STREAM_NAME=$1
FILE_NAME=$2

echo $STREAM_NAME:$FILE_NAME >> /usr/local/FlashphonerWebCallServer/logs/record.log
cp $FILE_NAME /opt/media
```

writing permissions must be granted to this folder

```
sudo chmod o+w /opt/media
```

Directory for saving recorded files

By default, stream records are saved to folder WCS_HOME/records. Starting from build 5.2.687, the folder for saving records can be changed using the following parameter

```
record_dir=/usr/local/FlashphonerWebCallServer/records
```

If this parameter defines a non-default folder, files can not be downloaded in Stream Recording example. In this case, it is recommended to deploy a separate web server for recording files downloading from a custom folder.

Since build5.2.801,WCS is running from 'flashphoner' user for security reasons. Therefore, when this parameter is changing, write permissions must be granted to the folder as described above.

Adjusting record audio sample rate

By default, audio track is recorded with sample rate 44.1 kHz. This value can be changed using the following parameter if necessary

```
record_audio_codec_sample_rate=48000
```

In this case, record audio sample rate will be set to 48 kHz.

moov atom placement configuration in recording metadata

To play a recording file while downloading (progressive downloading), `moov` atom must be placed before `mdat` atom in recording metadata. To do this, the following default setting is added in latest builds

```
mp4_container_moov_first=true
```

The place for `moov` atom can be reserved when recording file is created to optimize disk operations while saving the file. This feature can be enabled with the following parameter

```
mp4_container_moov_first_reserve_space=true
```

A space size to reserve should be set in kilobytes with the following parameter

```
mp4_container_moov_reserved_space_size=2048
```

By default, 2048 kilobytes will be reserved. Therefore, a recording file size will not be less than reserved space if `moov` atom reservation is enabled, this must be taken into account while setting up record rotation by size.

Audio track bitrate configuration for FDK AAC codec

Since 5.2.428, the ability is added to set audio track bitrate mode for recording with FDK AAC codec. By default, mode 5 is set (VBR 112 kbps). This value can be changed with the following parameter

```
record_fdk_aac_bitrate_mode=5
```

The following bitrate modes are possible:

- 0 - CBR

- 1-5 - VBR

Note that recording files playback by file timestamps using nginx modulengx_http_mp4_moduleis possible for VBR recordings only.

Audio channels configuration

Since build5.2.610, it is possible to set audio channels count for recordings using the following parameter

```
record_audio_codec_channels=2
```

By default, audio channels count set to 2 (stereo). To record stream with monophonic audio, the following should be set

```
record_audio_codec_channels=1
```

Recording perfomance tuning under high load

Recording files writing to disk may take a much time when a number of published streams are recorded simultaneously. To speed up writing, it is possible since build5.2.639to set CPU threads number to work with stream recording using the following parameter

```
file_recorder_thread_pool_max_size=4
```

By default, 4 threads are used to work with stream recording. This number can be increased if necessary. Note that it is not recommended to set recording threads number more than CPUs number on the server.

VP8 streams recording to webm container

Since build5.2.905java implementatiion is used to record VP8 streams to webm container

```
webm_java_writer_enable=true
```

A cluster duration (in milliseconds) and size (in bytes) limits settings are available for this implementation. A stream data will be flushed to a recording file when one of the limits is reached

```
webm_cluster_duration_limit=100000
webm_cluster_size_limit=131072
```

If some issues occur, the native code recorder implementation based on ffmpeg may be enabled

```
webm_java_writer_enable=false
```

A separate folder for temporary files

While recording a stream, media data are written to a temporary file, then this file is copying to a recording file named according to template. Since build5.2.963, it is possible to set a separate folder for temporary files using the following parameter

```
record_tmp_dir=/tmp
```

This allows, for example, to place temporary files to RAM drive to speed up recording process.

The folder to place temporary files must be writable by WCS process, so permissions must be set to flashphoner user. For example, if the temporary files folder is set to

```
record_tmp_dir=/opt/wcs
```

then permissions must be set to flashphoner

```
sudo chown -R flashphoner:flashphoner /opt/wcs
```

By default, all the temporary files are placed to /usr/local/FlashphonerWebCallServer/records folder.

Minimal available disk space checking

Since build 5.2.1209 a minimal available disk space is checked when recording a stream. If the space available is less then a limit, the recording will stop, or will not be started. In this case, the following message will be written to server log

```
Not enough available disk space
```

The space limit is set by the following parameter (1 G by default)

```
file_recorder_min_space=1g
```

It is possible to set the limit value in gigabytes (`g` suffix) or in megabytes (`m` suffix), for example

```
file_recorder_min_space=1024m
```

Note that if the recording was stopped due to space limit, a post processing script still will be launched for the recording file.

Recording stopping on errors

Since build 5.2.1236 the parameters were added to define an errors handling while recording a stream. By default, stream recording will be stopped if 3 errors occur during 60 minutes:

```
file_recorder_error_interval=60
file_recorder_max_errors_per_interval=3
```

The frame which raise an exception and subsequent frames will not be recorded until keyframe is received and parsed successfully.

Client side

If stream recording is enabled on the server, whether the stream is recorded or not is determined by the value of record parameter passed into the createStream function in the script of the publisher client:

- true - the stream published by this client is recorded;
- false - the stream is not recorded.

For instance, the script of the Stream Recording application recording.html, recording.js, contains the following code:

```
function publishStream(session) {
    var streamName = $('#url').val().split('/')[3];
    session.createStream({
        name: streamName,
        display: localVideo,
        record: true,
        receiveVideo: false,
        receiveAudio: false
        ...
    }).publish();
}
```

Stream recording on demand

Sometimes, it is necessary to record the stream that already exists on server, mixer output stream for example. This can be done with REST API. Note that only streams in"PUBLISHING" state can be recorded.

REST query must be HTTP/HTTPS POST query like this:

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

Where:

- streaming.flashphoner.comis WCS server address
- 8081 is a standard WCS REST / HTTP port
- 8444is a standard WCS REST / HTTPS port

- rest-apiis mandatory prefix
- /recorder/startupis REST method

REST methods and response statuses

REST method	Example of REST query	Example of REST response	Response statuses	Description
/stream /startRecording, /recorder /startup	`{ "mediaSessionId": "5a072377-73c1-4caf-abd3", "config": { "fileTemplate": "{streamName}-{startTime}-{endTime}", "rotation": "20M" } }`		404 - Not found 500 - Internal error	Start stream recording in specified mediasession
/stream /stopRecording, /recorder /terminate	`{ "mediaSessionId": "5a072377-73c1-4caf-abd3" }`		404 - Not found 500 - Internal error	Stop stream recording in specified mediasession
/recorder /find_all		`[ { "fileName": "9c3e-test-1563776083752-{endTime}.mp4", "mediaSessionId": "5a072377-73c1-4caf-abd3" } ]`	404 - Not found 500 - Internal error	Find session recordings

Parameters

Parameter name	Description	Example
mediaSessionId	Media session identificator	5a072377-73c1-4caf-abd3
config	Record settings that redefine server settings	
fileTemplate	Recording file name template	{streamName}-{startTime}-{endTime}
rotation	Enables/disables rotation and duration/volume of fragments	`20M`

Recording on demand works as follows:

- When REST API/recorder/startup query is called, current recording will be stopped.
- New recording starts with settings passed in REST query.
- If some setting is not defined in REST query, the server setting will be applied.

For example if recording should have exact file name with rotation disabled, the following query should be passed:

```
/stream/startRecording
{
  "mediaSessionId":"1234567890abcdefgh",
  "config": {
    "fileTemplate": "test",
    "rotation": "disabled"
  }
}
```

REST query /recorder/find_all returns session recordings list. The list shows both recordings on demand started via REST API and recordings initiated with WebSDK:

```
[
    {
        "fileName": "003f-1563776713987-{endTime}.mp4",
        "mediaSessionId": "5af9c820-ac49-11e9-9f06-693cb47c4042"
    },
    {
        "fileName": "stream-57882100-ac49-11e9-afdd-6752f5be57a9-jtdvnittjkrd8rsc3dnfbger2o.mp4",
        "mediaSessionId": "57882100-ac49-11e9-afdd-6752f5be57a9"
    }
]
```

How to get recording file name

There are the following ways to get recording name, for example, to download it:

1.Record files handling script on server receives the recording file name right after it recorded

2. If it is necessary to know recording name in browser, file name template should be formed from parameters which can be obtained by REST API, for example

```
stream_record_policy_template={streamName}-{mediaSessionId}
```

3. When WebSDK is used, recording name can be obtained with getRecordInfo() function

```
    ...
    }).on(STREAM_STATUS.UNPUBLISHED, function (stream) {
        setStatus(stream.status());
        showDownloadLink(stream.getRecordInfo());
        onStopped();
    })
    ...
```

Note that STREAM_STATUS.UNPUBLISHED may be received considerable time after stopping a stream if recording file is big. Since build 5.2.673this time may be limited using the following parameter (15 seconds by default)

```
record_stop_timeout=15
```

How to play or download recording file

The recording file is available via WCS internal web server using the following link

```
https://test.flashphoner.com:8444/client/records/stream.mp4
```

Where

- test.flashphoner.com - WCS server URL
- stream.mp4 - recording file name

By default, WCS returns HTTP header

```
Content-Disposition: inline;filename="stream.mp4"
```

in this case, browser tries to play the file. This behaviour is enabled with the following parameter

```
record_response_content_disposition_header_value=inline
```

To make a browser to download the recording file without playing it, the following parameter should be set

```
record_response_content_disposition_header_value=attachment
```

Downloading and playing a certain part of a recording file

Since build5.2.894it is possible to download and play a certain part of a recording file. To do this, request the file with start time and end time in seconds

```
https://test.flashphoner.com:8444/client/records/stream.mp4?start=11&end=60
```

Only start time or only end time can be used.

The fragments requested are stored to the same folder where recording files are placed. Timestamps will be added to the fragment name, for example

```
stream-s11-e60.mp4
```

Such files are not removed after downloading: if the same fragment is requested again, server will send the existing file.

Since build5.2.899partial download and playback is supported for audio only recordings too.

Fragments folder configuration

By default, fragments are written to the folder

```
/usr/local/FlashphonerWebCallServer/records
```

(the same folder as recording files).

Since build5.2.957it is possible to set a separate folder to store fragments using the following parameter

```
mp4_cutter_dir=/tmp
```

Known limits

1. A partial download and playback is supported for MP4 container only. If webm file is requested, it will be always fully downloaded and played.

2. Playback can start slightly earlier if start time is set, depending on closest key frame in the recording file.

Multiple stream recording to one file with subsequent mixing

Since build5.2.1012it is possible to record multiple streams to one file. Then streams can be extracted from this file and mixed by a special tool. Multiple streams can be recorded only toMP4 container (H264 + AAC). This feature is intended, for example, to record a video conference. In this case, unlike toMCU mixer, stream mixing works while recording file is post-processed, this allows to aquire a less of server resources during the conference itself.

Multiple stream recording is managed by REST API.

REST query must be HTTP/HTTPS POST query like this:

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

Where:

- streaming.flashphoner.comis WCS server address
- 8081 is a standard WCS REST / HTTP port
- 8444is a standard WCS REST / HTTPS port
- rest-apiis mandatory prefix
- /multipleRecorder/startupis REST method

REST methods and responses

REST query	Example of REST query	Example of REST response	Response states	Description

/multipleRecorder/startup	`{`
	` "uri": "multi-recorder://test-record"`
	`}`
			409 - Conflict
			
			500 - Internal error
				Launch multiple streams recorder

/multipleRecorder/add	`{`
	` "uri": "multi-recorder://test-record",`
	` "mediaSessionId": "866a9910-fbfe-11eb-aae4-6f99b0c80a3a"`
	`}`
			404 - Not found
			
			409 - Conflict
			
			500 - Internal error
				Add the stream with mediaSessionId to the recorder

/multipleRecorder/find_all		`[`
		`  {`
		`    "mediaSessionsId": [`
		`      "866a9910-fbfe-11eb-aae4-6f99b0c80a3a",`
		`      "9f1e2530-fbfe-11eb-9ec1-77172ac14d86",`
		`      "a970d0a0-fbfe-11eb-8fcc-912807bab442"`
		`    ],`
		`    "uri": "multi-recorder://test-record",`
		`    "fileName": "multi-recorder___test-record.mp4"`
		`  }`
		`]`
			404 - Not found
			
			500 - Internal error
				Find all recorders

/multipleRecorder/remove	`{`
	` "uri": "multi-recorder://test-record",`
	` "mediaSessionId": "866a9910-fbfe-11eb-aae4-6f99b0c80a3a"`
	`}`
			404 - Not found
			
			500 - Internal error
				Remove stream with mediaSessionId from recorder

/multipleRecorder/terminate	`{`
	` "uri": "multi-recorder://test-record"`
	`}`
			404 - Not found
			
			500 - Internal error
				Stop multiple streams recorder

Parametes

Parameter name	Description	Example
uri	Recorder URI	multi-recorder://test-record
mediaSessionId	Stream mediasession Id	866a9910-fbfe-11eb-aae4-6f99b0c80a3a
filename	Recording file name	multi-recorder___test-record.mp4

Recording file name

Multiple streams recording file name is formed by template. In this case:

1.{streamName} parameter is set according to recorder URI, replacing all the characters not allowed to use in file system to underline.

2. {startTime}, {endTime} parameters cannot be set because they depend on stream timestamps, and we have a multiple streams with a different timestamps simultaneously. So it is recommended to use {startTimeMillis},{endTimeMillis} parameters to add servers clock timestamps to file name.

For example, with the following template

```
stream_record_policy_template={streamName}-{startTime}-{startTimeMillis}-{endTime}-{endTimeMillis}
```

the file name for recorder with URI

```
"uri": "multi-recorder://test-record"
```

will look as follows:

```
multi-recorder___test-record--1-1628821032180--1-1628821151750.mp4
```

Here {startTime},{endTime} are replaced by -1.

Multiple streams recording folder

By default, multiple streams recording files are stored in WCS_HOME/records forlder. Since build 5.2.1088, multiple streams recording folder can be set using the following parameter

```
multi_record_dir=/usr/local/FlashphonerWebCallServer/records
```

This folder should be writalble, For example, if the folder is set to

```
multi_record_dir=/opt/media
```

the folder access privilegies sholud be set as

```
sudo chmod o+w /opt/media
```

Multiple streams mixing tool

Only one stream can be played by default from multiple streams recording file. The streams should be mixed to view them all. The OfflineMP4Mixer tool should be used to mix, launching as follows:

```
cd /usr/local/FlashphonerWebCallServer/tools
./offline_mixer_tool.sh multi-recorder___test-record--1-1628821032180--1-1628821151750.mp4
```

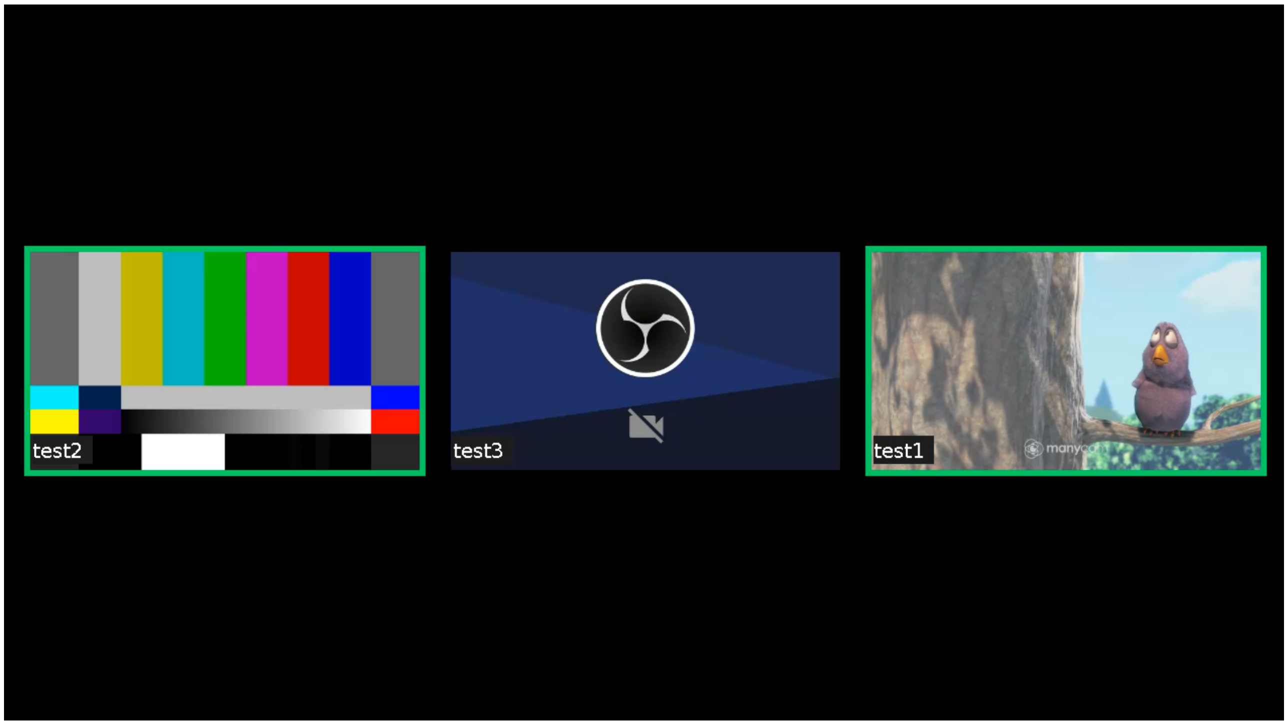
Offline mixer settings should be defined in /usr/local/FlashphonerWebCallServer/conf/offline_mixer.json file. By default, the following settings are used:

```
{
  "hasVideo": "true",
  "hasAudio": "true",
  "mixerDisplayStreamName": true
}
```

A mixed file is placed to the same folder as original one, with _mixed suffix addition, for example

```
multi-recorder___test-record--1-1628821032180--1-1628821151750_mixed.mp4
```

A picture sample from the mixed file



Getting tracks information from multiple recording file

Since build 5.2.1049 tracks information from multiple recording file may be extracted using multiple stream mixing tool. To do this, the tool should be launched as follows:

```
./offline_mixer_tool.sh --show-tracks-info ../records/multi-recorder___test-record.mp4
```

In this case, the tool will print tracks information in JSON form. See two participants room recording information example below:

Two participants track information example

```
[
  {
    "durationInMS": "37282",
    "trackEdits": [
      {
        "endInMs": "14",
        "type": "pause",
        "startInMs": "0"
      },
      {
        "endInMs": "37282",
        "type": "media",
        "startInMs": "14"
      }
    ],
    "channels": "2",
    "trackType": "AUDIO",
    "trackId": "1",
    "timescale": "44100",
    "streamName": "room-09f012-user1-e6ff",
    "trackCodec": "mp4a",
    "sampleRate": "44100",
    "mediaSessionId": "e6ff54e0-1c2a-11ec-90e8-79a2a32f3d9d"
  },
  {
    "durationInMS": "37336",
    "trackEdits": [
      {
        "endInMs": "37336",
        "type": "media",
        "startInMs": "0"
      }
    ],
    "trackType": "VIDEO",
    "trackId": "0",
    "width": "320",
    "timescale": "90000",
    "streamName": "room-09f012-user1-e6ff",
    "trackCodec": "avc1",
    "mediaSessionId": "e6ff54e0-1c2a-11ec-90e8-79a2a32f3d9d",
    "height": "240"
  },
  {
    "durationInMS": "39274",
    "trackEdits": [
      {
        "endInMs": "100",
        "type": "pause",
        "startInMs": "0"
      },
      {
        "endInMs": "21534",
        "type": "pause",
        "startInMs": "100"
      },
      {
        "endInMs": "39274",
        "type": "media",
        "startInMs": "21534"
      }
    ],
    "channels": "2",
    "trackType": "AUDIO",
    "trackId": "3",
    "timescale": "44100",
    "streamName": "room-09f012-user2-f746",
    "trackCodec": "mp4a",
    "sampleRate": "44100",
    "mediaSessionId": "f74633a1-1c2a-11ec-bba5-af8cf43275a8"
  },
  {
    "durationInMS": "39303",
    "trackEdits": [
      {
        "endInMs": "21434",
        "type": "pause",
        "startInMs": "0"
      },
      {
        "endInMs": "39303",
        "type": "media",
        "startInMs": "21434"
      }
    ],
    "trackType": "VIDEO",
    "trackId": "2",
    "width": "320",
    "timescale": "90000",
    "streamName": "room-09f012-user2-f746",
    "trackCodec": "avc1",
    "mediaSessionId": "f74633a1-1c2a-11ec-bba5-af8cf43275a8",
    "height": "240"
  }
]
```

Where:

- durationInMS - track duration in milliseconds

- trackType - track type: AUDIO or VIDEO
- trackId - track Id
- streamName - stream name containing this track
- mediaSessionId - stream media session Id
- timescale - track samples per second quantity
- trackCodec - track codec
- width, height - video track picture size by first key frame
- channels - audio track channels count
- sampleRate - audio track sample rate (usually equal to timescale parameter)
- trackEdits - track timeline description

Track timeline is described as a set of segments according to MP4 'edit lists` atom content, with the following parameters:

- startInMs - segment start time in milliseconds relative to file beginning
- endInMs - segment end time in milliseconds relative to file beginning
- type - segment type: mediadata (media) or pause (pause)

Any single track may be extracted from multiple recording file using ffmpeg or other MP4 editing tool.

Note that if a stream with the same name was added to multiple recorder, then removed from recorder and then added again, this stream will be repersented in the file as different tracks with subsequent track Ids, for example:

Re-added stream track information example example

```
[
  {
    "durationInMS": "78978",
    "trackEdits": [
      {
        "endInMs": "63050",
        "type": "pause",
        "startInMs": "0"
      },
      {
        "endInMs": "78978",
        "type": "media",
        "startInMs": "63050"
      }
    ],
    "channels": "2",
    "trackType": "AUDIO",
    "trackId": "3",
    "timescale": "44100",
    "streamName": "test",
    "trackCodec": "mp4a",
    "sampleRate": "44100",
    "mediaSessionId": "fbbf5b50-20ee-11ec-bf06-ef6ec6048b2c"
  },
  {
    "durationInMS": "39708",
    "trackEdits": [
      {
        "endInMs": "23150",
        "type": "media",
        "startInMs": "0"
      }
    ],
    "channels": "2",
    "trackType": "AUDIO",
    "trackId": "1",
    "timescale": "44100",
    "streamName": "test",
    "trackCodec": "mp4a",
    "sampleRate": "44100",
    "mediaSessionId": "c7bc1460-20ee-11ec-bf06-ef6ec6048b2c"
  },
  {
    "durationInMS": "39791",
    "trackEdits": [
      {
        "endInMs": "23233",
        "type": "media",
        "startInMs": "0"
      }
    ],
    "trackType": "VIDEO",
    "trackId": "0",
    "width": "640",
    "timescale": "90000",
    "streamName": "test",
    "trackCodec": "avc1",
    "mediaSessionId": "c7bc1460-20ee-11ec-bf06-ef6ec6048b2c",
    "height": "360"
  },
  {
    "durationInMS": "63050",
    "trackEdits": [
      {
        "endInMs": "39791",
        "type": "pause",
        "startInMs": "0"
      },
      {
        "endInMs": "50191",
        "type": "media",
        "startInMs": "39791"
      }
    ],
    "trackType": "VIDEO",
    "trackId": "2",
    "width": "640",
    "timescale": "90000",
    "streamName": "test",
    "trackCodec": "avc1",
    "mediaSessionId": "ed3ebda0-20ee-11ec-bf06-ef6ec6048b2c",
    "height": "360"
  }
]
```

Multiple recording file handling script

When multiple stream recording is finished, a specila handling script is automatically launched as set by the following parameter

```
on_multiple_record_hook_script=on_multiple_record_hook.sh
```

By default, script will launch offline_mixer_tool.sh, passing multiple streams recording file name to it.

Since build5.2.1023, on_multiple_record_hook.sh script writes only file mixing result to log file /usr/local/FlashphonerWebCallServer/logs/multi-record.log, to decrease hard disk I/O load while offline mixing tool is working. A detailed logging mat=y be enabled if necessary by setting the following variable in the script

```
LOGGER_ENABLED=true
```

Multithreaded encoding while mixing multiple stream recordings

Since build 5.2.1089 multithreaded encoding can be enabled while mixing multiple stream recordings. To enable this feature, add the following parameter to /usr/local/FlashphonerWebCallServer/conf/offline_mixer.json file

```
{
  ...,
  "multithreading": true
}
```

Multiple stream recordings are mixed a twice faster when multithreaded encoding is enabled comparing to singe-threaded one.

Multiple recording data callback

Since build 5.2.1123 it is possible to send PORT query to a certain URL after multiple recording and mixing is finished. This allows to notify backend about file name to which chat room recording is mixed.

Callback URL should be set in /usr/local/FlashphonerWebCallServer/conf/offline_mixer.json file

```
{
  ...,
  "callbackUrl": "http://backend.url/multiRecorderCallback"
}
```

A recording data to send are passed via /usr/local/FlashphonerWebCallServer/bin/on_multiple_record_hook.sh script when launching offline_mixer_tool.sh. Therefore, if WCS build 5.2.1123 is installed over a previous build, or if custom on_multiple_record_hook.sh is used, the script should be modified as follows:

on_multiple_record_hook.sh

```
# This script copies a recorded stream to client/records
FILE_NAME=$1
CREATION_MODIFICATION_TIME=$2
DURATION_IN_MS=$3
RECORDER_URI=$4
WCS_HOME=/usr/local/FlashphonerWebCallServer
LOG_FILE=$WCS_HOME/logs/multi-record.log
MIXER_TOOL=$WCS_HOME/tools/offline_mixer_tool.sh
# Set LOGGER_ENABLED to true to enable mixing debug logging
LOGGER_ENABLED=false

echo "[$(date '+%Y-%m-%d %H:%M:%S')] Start mixing multiple recording file $FILE_NAME" >> $LOG_FILE

if $LOGGER_ENABLED; then
    bash $MIXER_TOOL $FILE_NAME $CREATION_MODIFICATION_TIME $DURATION_IN_MS $RECORDER_URI >> $LOG_FILE 2>&1
else
    bash $MIXER_TOOL $FILE_NAME $CREATION_MODIFICATION_TIME $DURATION_IN_MS $RECORDER_URI > /dev/null 2>&1
fi

CODE=$?
if [ "$CODE" -ne "0" ]; then
    if [ "$CODE" -eq "64" ]; then
        echo "ERROR: File to mix not found" >> $LOG_FILE
    elif [ "$CODE" -eq "65" ]; then
        echo "ERROR: Offline mixer config not found" >> $LOG_FILE
    else
            echo "ERROR: Offline mixer tool error code: $CODE" >> $LOG_FILE
        fi
        exit $CODE
fi
echo "[$(date '+%Y-%m-%d %H:%M:%S')] Multiple recording file $FILE_NAME is mixed successfully" >> $LOG_FILE
exit 0
```

POST query contains a following JSON data:

```
POST /multiRecorderCallback HTTP/1.1
Content-Type: application/json
Content-Length: 463
Host: localhost
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.3.5 (java 1.5)
Accept-Encoding: gzip,deflate

{
  "multiRecorderCreationModificationTime":3724973476,
  "multiRecorderDurationInMS":44061,
  "multiRecorderFilePath":"/usr/local/FlashphonerWebCallServer/multirecords/stream-32c7edd7-37bf-4bf2-a58d-
955679c5287e-mockLogin.mp4",
  "recorderUri":"multi-recorder://room-bace1f",
  "mixerParams":
  [
    {
      "path":"/usr/local/FlashphonerWebCallServer/multirecords/stream-32c7edd7-37bf-4bf2-a58d-955679c5287e-
mockLogin_mixed.mp4",
      "durationInMs":44000,
      "creationModificationTime":3724973524
    }
  ]
}
```

Multiple recording file parameters:

- multiRecorderCreationModificationTime - multiple recording file creation date and time
- multiRecorderDurationInMS - multiple recording file duration in milliseconds
- multiRecorderFilePath - multiple recording file path
- recorderUri - multiple recording identifier, contains room name if RoomApi is used

Mixed file parameters:

- path - mixed file path
- durationInMs - mixed file duration in milliseconds
- creationModificationTime - mixed file creation date and time

Minimal available disk space checking for multiple recordings mixing

Since build 5.2.1317 minimal available disk space checking is enabled for multiple recordings mixing. The mixing will not start or will be stopped when disk space becomes less then defined threshold value. The threshold is set by the following parameter in `/usr/local/FlashphonerWebCallServer/conf/offline_mixer.json` file

```
{
  ...,
  "minAvailableSpace": "1G"
}
```

By default, disk space threshold is set to 1 Gb (same as for single stream recordings). If the treshold is reached while mixing is already working, the mixing will be stopped with keeping data already written to destination file. The file can be correctly played after that.

Known issues

1. Maximum length of file name in all actual Linux file systems is limited to 255 characters. When record file is created, its name will be trimmed to this limit including extension and part number if rotation is enabled.

2. When stream published in chat room is recorded, file rotation will be automatically disabled, otherwise record files will not be merged.

3. Creation time will be set to MP4 recording metadata only.

4. In Amazon WCS instance, record files hook script requires sudo to execute any file operation.

Symptoms: record hook script does not perform any operation on record files

Solution: in Amazon WCS instance use sudo to make any file operation or call external script from record hook script, for example

```
sudo cp $SRC_FILE $DST_FILE
```

5. CPU load may grow and simultaneous streams recording may delay to finish on low powered servers when two channels audio is recording

Symptoms: all the CPU cores are loaded to 100% while a number of streams are recording simultaneously, and the recordings are finished with a big delay when streams stop.

Solution: disable two channels audio recording

```
record_audio_codec_channels=1
```

6. Stream is playing normally via WebRTC, but recording may be corrupted when publishing H264 stream from Android Firefox on some devices

Symptoms: recording file has a small size and cannot be played or picture seems corrupted when stream is published from Android Firefox

Solution:

a) use VP8 to publish stream from Android Firefox

b) use Chrome or other browser to publish stream from this device

7. Some Android devices may publish WebRTC H264 stream with High profile even if there is no such profile in SDP while establishing WebRTC connection

Symptoms: High profile is displayed in MP4 recording file data

Solution: if there are any problems playing stream recording files with High profile, re-encode those files using ffmpeg for example, running it after recording is finished by on_record_hook.sh script.

8. The first record after the server start may be corrupted if Java machine does not initialize all the necessary modules in time

Symptoms: a long freeze in the first recording file after server start

Solution:

a) update WCS to build 5.2.1105

b) if the build 5.2.1105 and newer is already used, make sure WebRTC stack modules pre-initilization on server startup is enabled

```
webrtc_pre_init=true
```

9. webm recording files cannot be played in iOS Safari

Symptoms: file download starts instead of playback when clicking webm recording file link

Solution: download recording file to device and play it in local player application