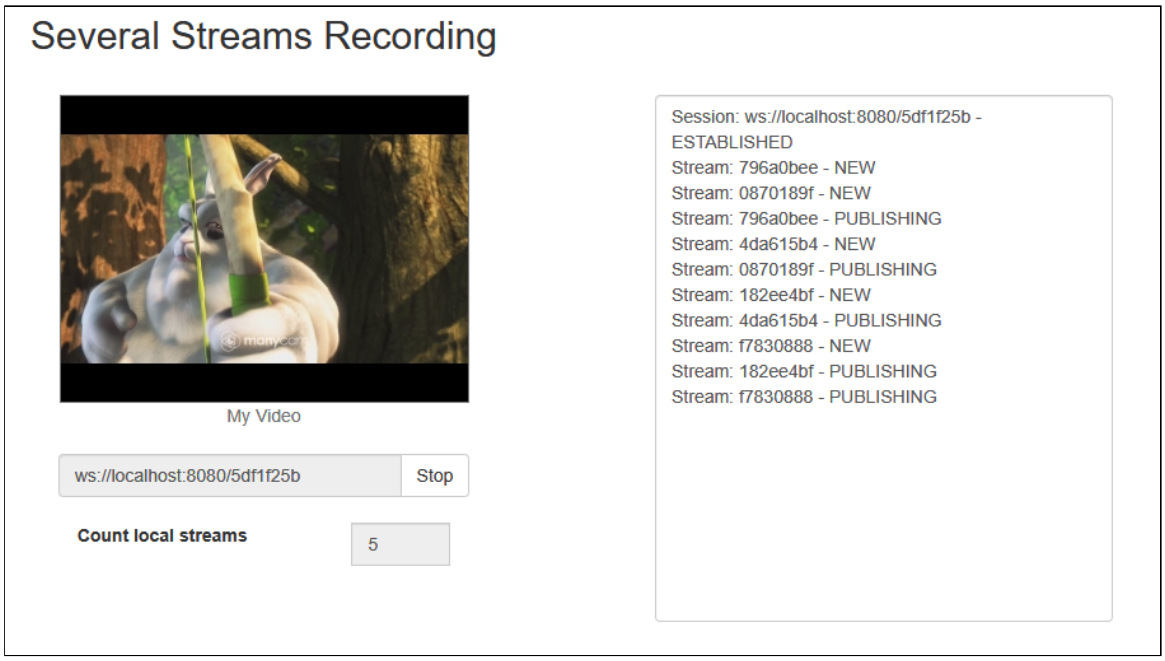
Several Streams Recording

Example of streamer with recording of several published video streams in one session

This streamer can be used to publish and record several WebRTC streams simutaneously to Web Call Server in one session.

This example does not work in iOS Safari. It is recommended to use desktop PC/Mac browser.

On the screenshot below 5 streams are publishing and recording



Code of the example

The path to the source code of the example on WCS server is:

/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/several_streams_recording

- several_streams_recording.css - file with styles
- several_streams_recording.html - page of the streamer
- several_streams_recording.js - script providing functionality for the streamer

This example can be tested using the following address:

https://host:8888/client2/examples/demo/streaming/several_streams_recording/several_streams_recording.html

Here host is the address of the WCS server.

Analyzing the code

To analyze the code, let's take the version of file `several_streams_recording.js` with hash `ecbadc3` which is available here and can be downloaded in build 2.0.212.

1. Initialization of the API

`Flashphoner.init()` code

```
Flashphoner.init();
```

2. Connection to server.

`Flashphoner.createSession()` code

```
testSession = Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED,
function (session) {
  ...
}).on(SESSION_STATUS.DISCONNECTED, function (session) {
  ...
}).on(SESSION_STATUS.FAILED, function (session) {
  ...
});
```

3. Receiving the event confirming successful connection

`ConnectionStatusEvent ESTABLISHED` code

```
testSession = Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED,
function (session) {
  addSessionStatusLog(session);
  //session connected, start playback
  publishStreams(session);
}).on(SESSION_STATUS.DISCONNECTED, function (session) {
  ...
});
```

4. Video streaming

`Session.createStream()`, `Stream.publish()` code

When stream is created, the following parameters are passed

- `streamName` - name of the stream
- `localVideo` - `div` element, in which video from camera will be displayed
- `record: true` - to enable stream recording

Every stream is added to `streams` array

```
var stream = session.createStream({
    name: streamName,
    display: localVideo,
    record: true,
    receiveVideo: false,
    receiveAudio: false
    ...
});
addStatusLog(stream);
stream.publish();
streams.push(stream);
```

5. Receiving the event confirming successful streaming

`StreamStatusEvent PUBLISHING` code

```
var stream = session.createStream({
    ...
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
    checkCountStreams();
    addStatusLog(stream);
    ...
});
```

6. Streams count checking and publishing new streams until desired value is reached

code

```
function checkCountStreams() {
    var $publishBtn = $("#publishBtn");
    if ($publishBtn.text() === "Start" && $publishBtn.prop('disabled') ) {
        if (streams.length < $("#countStreams").val()) {
            publishStreams(session);
        } else {
            toRecordedState();
        }
    }
}
```

7. Streaming stop

`Stream.stop()` code

```
function toRecordedState() {
    $("#publishBtn").text("Stop").off('click').click(function () {
        for (var i in streams) {
            streams[i].stop();
        }
        streams = [];
        toInitialState();
    }).prop('disabled', false);
}
```

8. Receiving the event confirming successful streaming stop

`StreamStatusEvent UNPUBLISHED` code

```
var stream = session.createStream({
    ...
}).on(STREAM_STATUS.UNPUBLISHED, function (stream) {
    checkCountStreams();
    addStatusLog(stream);
}).on(STREAM_STATUS.FAILED, function (stream) {
    ...
});
```