Several Streams Recording

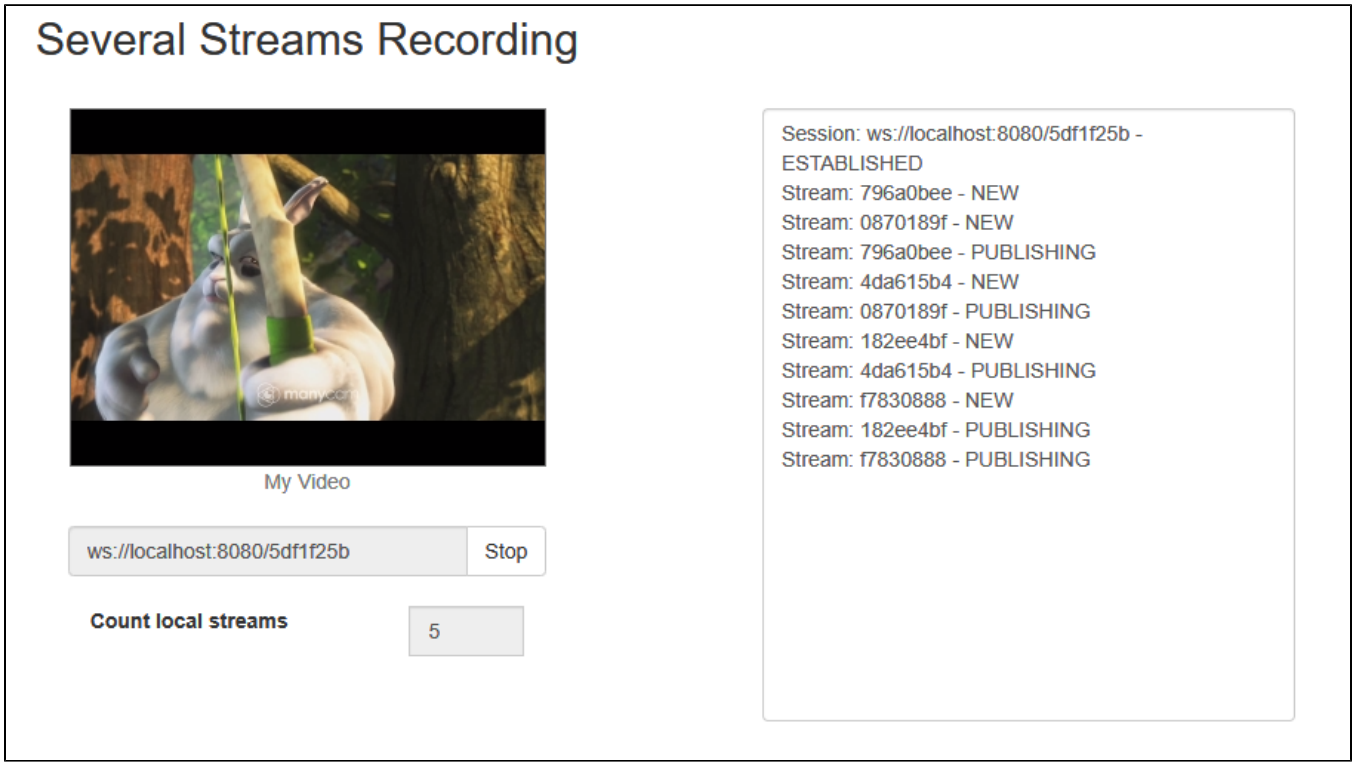
- Пример стримера с публикацией и записью нескольких видеопотоков в одной сессии
- Код примера
- Работа с кодом примера

Пример стримера с публикацией и записью нескольких видеопотоков в одной сессии

Данный стример может использоваться с Web Call Server для публикации и записи нескольких WebRTC потоковв одной сессии

> Пример не работает в браузере iOS Safari. Рекомендуется использовать для тестирования записи браузеры на PC/Mac

На скриншоте ниже представлен пример публикации и записи 5 потоков



Код примера

Код данного примера находится на WCS-сервере по следующему пути:

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

recording.css - файл стилей
recording.html - страница стримера
recording.js - скрипт, обеспечивающий работу стримера

Тестировать данный пример можно по следующему адресу:

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

Здесь host - адрес WCS-сервера.

Работа с кодом примера

Для разбора кода возьмем версию файла several_streams_recording.js с хешем ecbadc3, которая находится здесь и доступна для скачивания в соответствующей сборке2.0.212.

1. Инициализация API.

Flashphoner.init() code

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

2. Подключение к серверу.

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. Получение от сервера события, подтверждающего успешное соединение.

ConnectionStatusEvent ESTABLISHEDcode

```
    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. Публикация видеопотока.

session.createStream(), publish() code

При создании передаются параметры

- streamName - имя видеопотока
- localVideo - div-элемент, в котором будет отображаться видео с камеры
- record: true - для того, чтобы была сделана запись потока

Все потоки добавляются в массив streams

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

5. Получение от сервера события, подтверждающего успешную публикацию.

StreamStatusEvent PUBLISHING code

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

6. Проверка количества потоков и публикация новых до достижения заданной величины

code

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

7. Остановка публикации потоков.

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. Получение от сервера события, подтверждающего успешную остановку публикации.

StreamStatusEvent UNPUBLISHED code

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