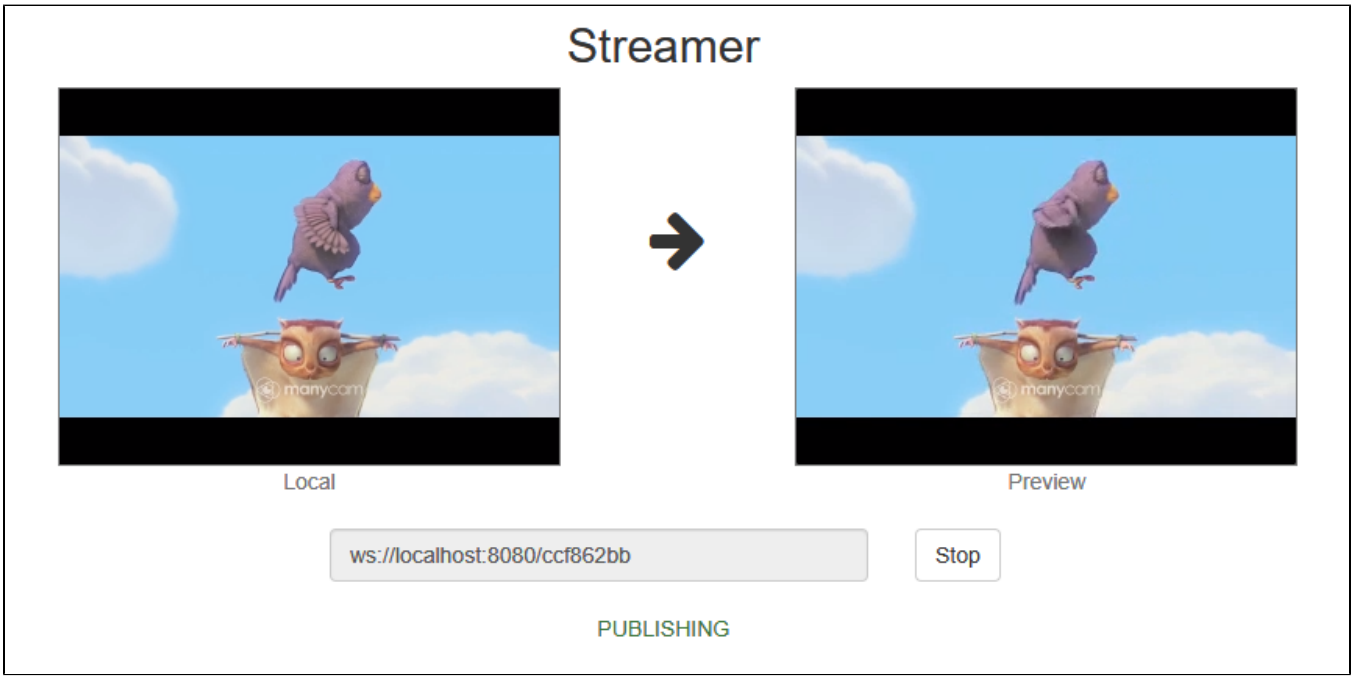
Streamer

Пример стримера

Данный стример может использоваться для публикации WebRTC потоков.

На скриншоте ниже представлен пример во время публикации потока:



В URL в поле ввода на скриншоте

- localhost:8080 - адрес и Websocket порт WCS-сервера
- ccf862bb - имя потока

На странице вопроизводятся два видео:

- 'Local' - видео с камеры
- 'Preview' - видео, которое приходит с сервера

Код примера

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

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

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

Тестировать данный пример можно по следующему адресу:

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

Здесь host - адрес WCS-сервера.

Работа с кодом примера

Для разбора кода возьмем версию файла streamer.js с хешем ecbadc3, которая находится здесь и доступна для скачивания в соответствующей сборке 2.0.212.

1. Инициализация API.

Flashphoner.init() code

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

2. Подключение к серверу.

Flashphoner.createSession() code

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

3. Получение от сервера события, подтверждающего успешное соединение.

ConnectionStatusEvent ESTABLISHEDcode

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
    //session connected, start streaming
    startStreaming(session);
}).on(SESSION_STATUS.DISCONNECTED, function(){
    ...
}).on(SESSION_STATUS.FAILED, function(){
    ...
});
```

4. Публикация видеопотока.

session.createStream(), publish() code

При создании передаются параметры:

- streamName - имя видеопотока;
- localVideo - div-элемент, в котором будет отображаться видео с камеры.

```
session.createStream({
    name: streamName,
    display: localVideo,
    cacheLocalResources: true
    ...
}).publish();
```

5. Получение от сервера события, подтверждающего успешную публикацию потока

StreamStatusEvent PUBLISHING code

При получении данного события создается превью-видеопоток при помощи createStream() и вызывается play() для его воспроизведения.

```
session.createStream({
    name: streamName,
    display: localVideo,
    cacheLocalResources: true
    ...
}).on(STREAM_STATUS.PUBLISHING, function(publishStream){
    setStatus(STREAM_STATUS.PUBLISHING);
    //play preview
    session.createStream({
        name: streamName,
        display: remoteVideo
        ...
    }).play();
    ...
}).publish();
```

6. Остановка воспроизведения видеопотока.

previewStream.stop() code

```
function onStarted(publishStream, previewStream) {
    $("#publishBtn").text("Stop").off('click').click(function(){
        $(this).prop('disabled', true);
        previewStream.stop();
    }).prop('disabled', false);
}
```

7. Получение от сервера события, подтверждающего остановку воспроизведения

StreamStatusEvent STOPPED code

```
    session.createStream({
        name: streamName,
        display: remoteVideo
    }).on(STREAM_STATUS.PLAYING, function(previewStream){
        ...
    }).on(STREAM_STATUS.STOPPED, function(){
        publishStream.stop();
    }).on(STREAM_STATUS.FAILED, function(){
        ...
    }).play();
```

8. Остановка публикации видеопотока после остановки воспроизведения превью-потока

publishStream.stop() code

```
    session.createStream({
        name: streamName,
        display: remoteVideo
    }).on(STREAM_STATUS.PLAYING, function(previewStream){
        ...
    }).on(STREAM_STATUS.STOPPED, function(){
        publishStream.stop();
    }).on(STREAM_STATUS.FAILED, function(){
        ...
    }).play();
```

9. Получение от сервера события, подтверждающего остановку публикации потока

StreamStatusEvent UNPUBLISHED code

```
session.createStream({
    name: streamName,
    display: localVideo,
    cacheLocalResources: true,
    ...
}).on(STREAM_STATUS.PUBLISHING, function(publishStream){
    ...
}).on(STREAM_STATUS.UNPUBLISHED, function(){
    setStatus(STREAM_STATUS.UNPUBLISHED);
    //enable start button
    onStopped();
}).on(STREAM_STATUS.FAILED, function(){
    ...
}).publish();
```