

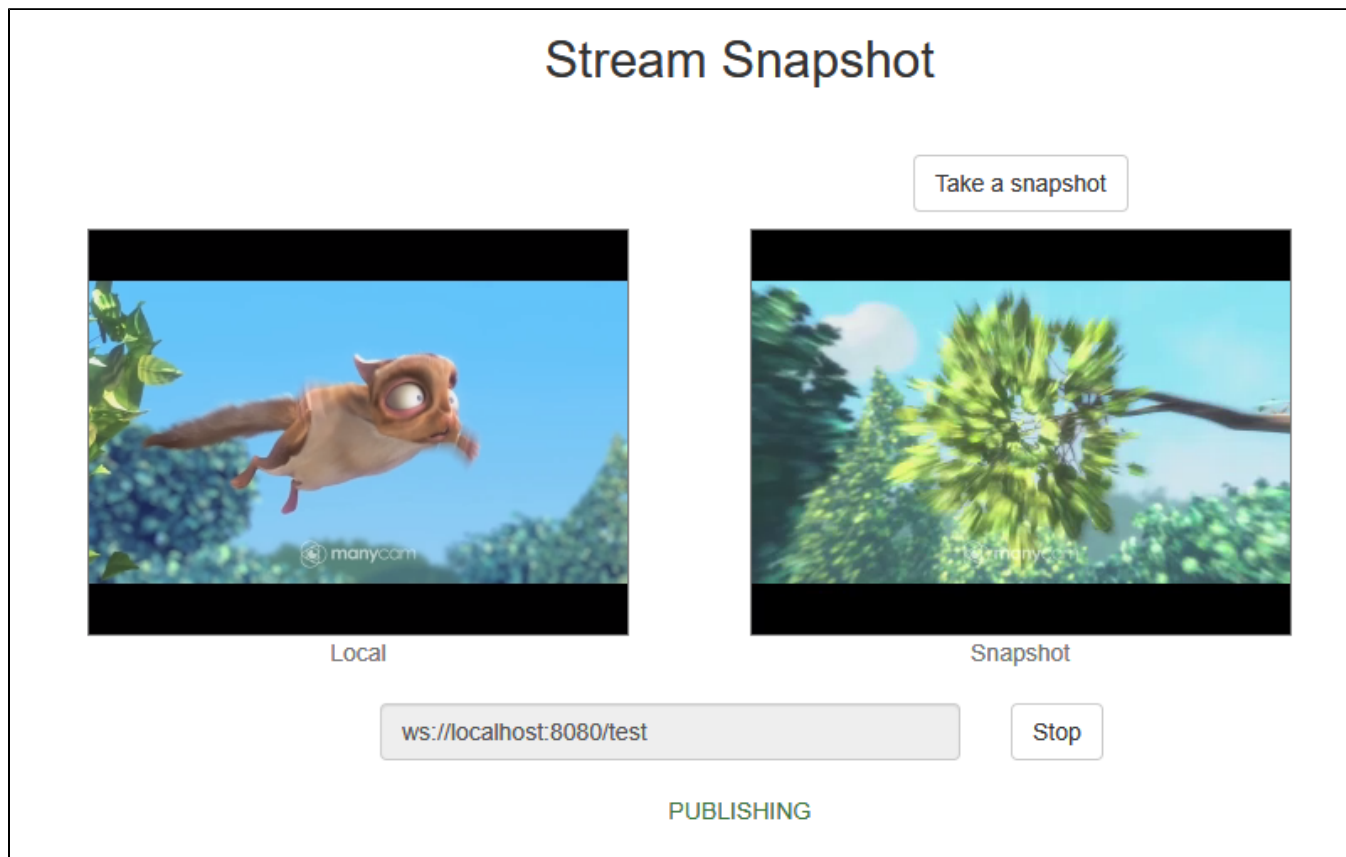
Stream Snapshot

- [Пример, демонстрирующий получение снимка опубликованного потока на стороне сервера](#)
- [Код примера](#)
- [Работа с кодом примера](#)

Пример, демонстрирующий получение снимка опубликованного потока на стороне сервера

В данном примере показано, как получить снимок потока, опубликованного на Web Call Server, снятый на стороне сервера

На скриншоте ниже был получен снимок публикуемого потока.



После начала публикации видео с камеры воспроизводится в 'Local' элементе слева.

После клика на кнопке 'Take a snapshot' запрашивается снимок, который в случае успешного получения будет отображен в 'Snapshot' элементе справа.

Код примера

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

/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/stream-snapshot

stream-snapshot.css - файл стилей

stream-snapshot.html - страница примера

stream-snapshot.js - скрипт, обеспечивающий работу примера

Тестировать данный пример можно по следующему адресу:

<https://host:8888/client2/examples/demo/streaming/stream-snapshot/stream-snapshot.html>

Здесь host - адрес WCS-сервера.

Работа с кодом примера

Для разбора кода возьмем версию файла stream-snapshot.js с хешем 7fff01f, которая находится [здесь](#) и доступна для скачивания в соответствующей сборке [2.0.219](#).

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

SESSION_STATUS.ESTABLISHED [code](#)

```
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(), Stream.publish() [code](#)

При создании потока передаются параметры:

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

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

5. Получение от сервера события, подтверждающего успешную публикацию

STREAM_STATUS.PUBLISHING [code](#)

```

session.createStream({
    ...
}).on(STREAM_STATUS.PUBLISHING, function(publishStream){
    setStatus(STREAM_STATUS.PUBLISHING);
    onPublishing(publishStream);
}).on(STREAM_STATUS.UNPUBLISHED, function(){
    ...
}).on(STREAM_STATUS.FAILED, function(){
    ...
}).publish();

```

6. Получение снимка

Session.createStream(), Stream.snapshot() [code](#)

```

session.createStream({name: name}).on(STREAM_EVENT, function(streamEvent){
    ...
}).snapshot();

```

7. Получение от сервера события, подтверждающего успешное снятие снимка

STREAM_EVENT_TYPE.SNAPSHOT_COMPLETE [code](#)

При получении данного события, элемент streamEvent.payload.snapshot содержит снимок в кодировке base64. После этого поток, созданный для снятия снимка, останавливается.

```

session.createStream({name: name}).on(STREAM_EVENT, function(streamEvent){
    if (STREAM_EVENT_TYPE.SNAPSHOT_COMPLETED === streamEvent.type) {
        console.log("Snapshot complete");
        setSnapshotStatus(STREAM_STATUS.SNAPSHOT_COMPLETE);
        snapshotImg.src = "data:image/png;base64,"+streamEvent.payload.snapshot;
    } else if (STREAM_EVENT_TYPE.SNAPSHOT_FAILED === streamEvent.type) {
        setSnapshotStatus(STREAM_STATUS.FAILED);
        console.log("Snapshot failed, info: " + streamEvent.payload.info);
    }
}).snapshot();

```

8. Остановка публикации видеопотока

Stream.stop() [code](#)

```

function onPublishing(stream) {
    $("#publishBtn").text("Stop").off('click').click(function(){
        $(this).prop('disabled', true);
        stream.stop();
    }).prop('disabled', false);
    ...
}

```

9. Получение от сервера события, подтверждающего успешную остановку публикации

STREAM_STATUS.UNPUBLISHED [code](#)

```

session.createStream({
    ...
}).on(STREAM_STATUS.PUBLISHING, function(publishStream){
    ...
}).on(STREAM_STATUS.UNPUBLISHED, function(){
    setStatus(STREAM_STATUS.UNPUBLISHED);
    //enable start button
    onUnpublished();
}).on(STREAM_STATUS.FAILED, function(){
    ...
}).publish();

```