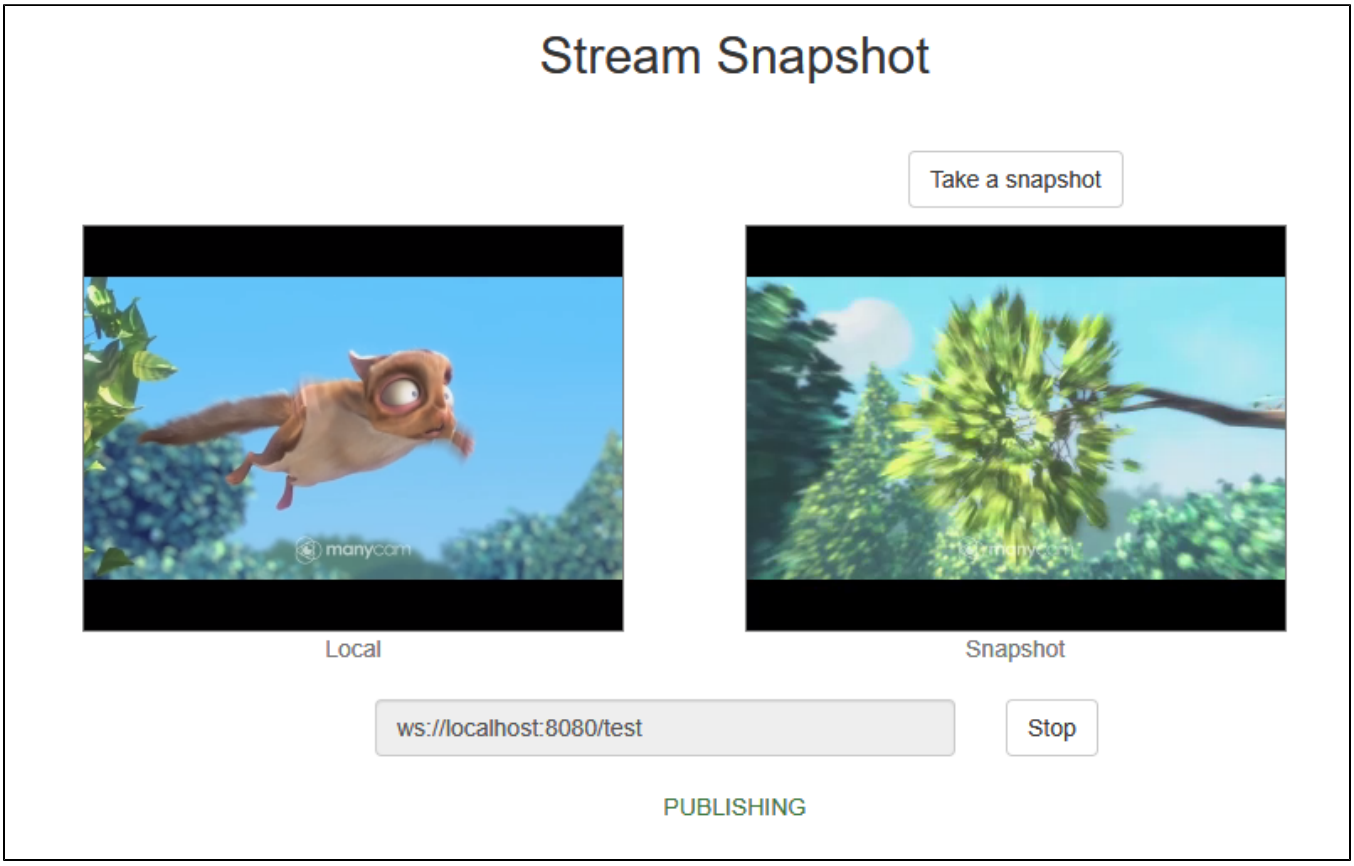
Stream Snapshot

Example demonstrating how to take snapshot of a published stream at server side

This example demonstrates taking snapshot of a stream published on Web Call Server at server side

On the screenshot below a snapshot of the stream being published has been taken.



When publishing is started, video from the camera is played in 'Local' element on the left side of the page.
When 'Take a snapshot' button is clicked, snapshot is taken and displayed in 'Snapshot' element on the right side of the page.

Code of the example

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

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

stream-snapshot.css - file with styles
stream-snapshot.html - page of the example
stream-snapshot.js - script providing functionality for the example

This example can be tested using the following address:

https://host:8888/client/examples/demo/streaming/stream-snapshot/stream-snapshot.html

Here host is the address of the WCS server.

Analyzing the code

To analyze the code, let's take the version of file stream-snapshot.js with hash 7fff01f, which is available here and can be downloaded with corresponding build 2.0.219.

1. Initialization of the API.

Flashphoner.init() code

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

2. Connection to server.

Flashphoner.createSession() code

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

3.Receiving the event confirming successful connection

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. Video streaming

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

When stream is created, the following parameters are passed

- streamName - name of the stream
- localVideo - `div` element to display video from camera

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

5.Receiving the event confirming successful streaming

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. Taking snapshot

Session.createStream(), Stream.snapshot() code

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

7.Receiving the event confirming successful snapshot taking

STREAM_EVENT_TYPE.SNAPSHOT_COMPLETE code

On receivng the event, `streamEvent.payload.snapshot` member contains snapshot in base64 encoding. Then stream created to take snapshot is stopped.

```
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 = ""+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. Stop streaming

Stream.stop() code

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

9.Receiving the event confirming successful streaming stop

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();
```