

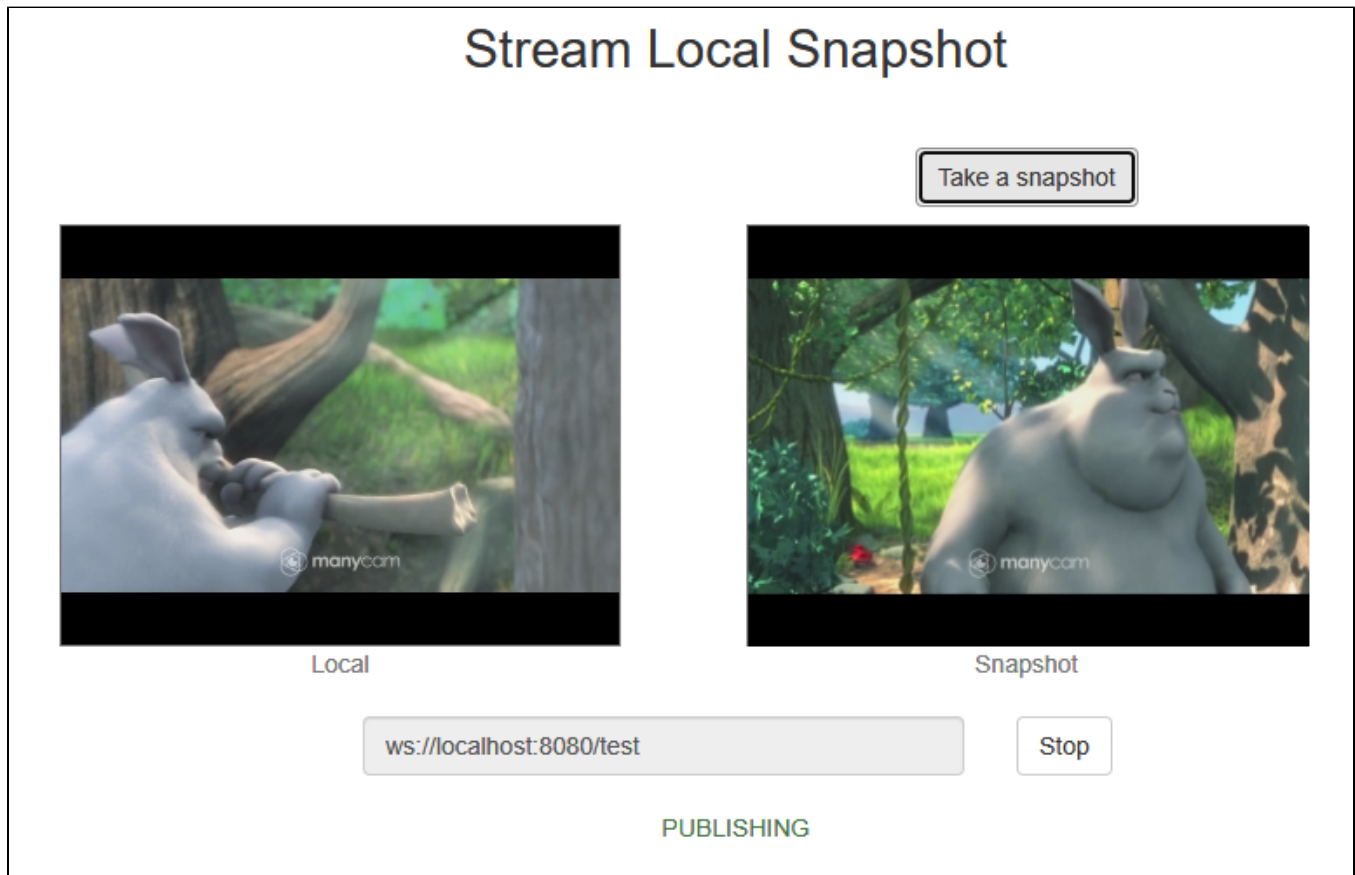
Stream Local Snapshot

- [The example to show stream snapshot capturing in local browser](#)
- [The code of the example](#)
- [Analyzing the code](#)

The example to show stream snapshot capturing in local browser

The example shows how to capture published stream snapshot locally in browser.

On the screenshot below, stream snapshot is already captured



When publishing starts, video is playin in 'Local' element at left side.

After clicking 'Take a snapshot' button snapshot is capturing from HTML5 video element, when is shown in 'Snapshot' element at right side.

The code of the example

The code of the example is on WCS server by the following path:

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

stream-local-snapshot.css - styles file
stream-local-snapshot.html - example HTML page
stream-local-snapshot.js - main example script

The example can be tetsted by the fiollowing address:

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

where host is WCS server address.

Analyzing the code

To analyze the code let's take a stream-local-snapshot.js version with hash ecbadc3, which is available [here](#) and can be downloaded in build 2.0.212.

1. API initialization.

Flashphoner.init() [code](#)

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

2. HTML page elements initialization and storing snapshot preview picture size

[code](#)

```
localVideo = document.getElementById("localVideo");
snapshotImg = document.getElementById("snapshotImg");
canvas = document.getElementById("canvas");

//preview size
snapshotImgSize = {
  w: snapshotImg.width,
  h: snapshotImg.height
};
```

Where

- localVideo - div to create vide element to capture a stream
- snapshotImg - img element to preview snapshot picture
- canvas - element to draw a picture captured from video stream and convert to PNG
- snapshotImgSize - a structure to store snapshot preview size

3. Connection establishing to the server.

Flashphoner.createSession() [code](#)

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

4. Receiving the event confirming successful connection.

ConnectionStatusEvent 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(){
  ...
});
```

5. Stream publishin.

session.createStream(), stream.publish() [code](#)

The following parameters are passed to createStream():

- streamName - stream name
- localVideo - div element to display a video from web camera

```

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

```

6. Receiving the event confirming successful publishing.

StreamStatusEvent 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();

```

7. Snapshot capture function invocation by button click

[code](#)

```

$("#snapshotBtn").off('click').click(function(){
  snapshot(stream);
}).prop('disabled', false);

```

8. Frame captured drawing on the canvas and converting to PNG

[code](#)

```

function snapshot(stream) {
  let video = document.getElementById(stream.id());
  let canvasContext = canvas.getContext("2d");
  if (video === undefined) {
    console.log("Failed to get video item for stream " + stream.name);
  } else {
    let videoSize = {
      w: video.videoWidth,
      h: video.videoHeight
    };
    // Draw snapshot on hidden canvas in full video size
    canvas.width = videoSize.w;
    canvas.height = videoSize.h;
    canvasContext.drawImage(video, 0, 0, canvas.width, canvas.height);
    let data = canvas.toDataURL('image/png');
    if (data === undefined) {
      console.log("Failed to get image data from canvas");
    } else {
      ...
    }
  }
}

```

9. Snapshot picture scaling to display preview and adding data to img element

[code](#)

```
function snapshot(stream) {
  let video = document.getElementById(stream.id());
  let canvasContext = canvas.getContext("2d");
  if (video === undefined) {
    console.log("Failed to get video item for stream " + stream.name);
  } else {
    ...
    if (data === undefined) {
      console.log("Failed to get image data from canvas");
    } else {
      // Downscale snapshot preview keeping video aspect ratio
      let previewSize;
      previewSize = downScaleToFitSize(videoSize.w, videoSize.h, snapshotImgSize.w, snapshotImgSize.h);
      console.log("previewSize: " + previewSize.w + "x" + previewSize.h);
      snapshotImg.style.width = previewSize.w + "px";
      snapshotImg.style.height = previewSize.h + "px";

      // Snapshot preview vertical align
      let margin = 0;
      if (snapshotImgSize.h - previewSize.h > 1) {
        margin = Math.floor((snapshotImgSize.h - previewSize.h) / 2);
      }
      snapshotImg.style.margin = margin + "px auto";

      // Set image data to snapshot page item. "Open image in new tab" or "Save image as" will open full
      size snapshot
      snapshotImg.setAttribute('src', data);
    }
  }
}
```

10. Helper function to scale the picture

[code](#)

```
function downScaleToFitSize(videoWidth, videoHeight, dstWidth, dstHeight) {
  let newWidth, newHeight;
  let videoRatio = videoWidth / videoHeight;
  let dstRatio = dstWidth / dstHeight;
  if (dstRatio > videoRatio) {
    newHeight = dstHeight;
    newWidth = Math.floor(videoRatio * dstHeight);
  } else {
    newWidth = dstWidth;
    newHeight = Math.floor(dstWidth / videoRatio);
  }
  return {
    w: newWidth,
    h: newHeight
  };
}
```

11. Stream stopping.

stream.stop() [code](#)

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

12. Receiving the event confirming successful stopping.

StreamStatusEvent 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();
```