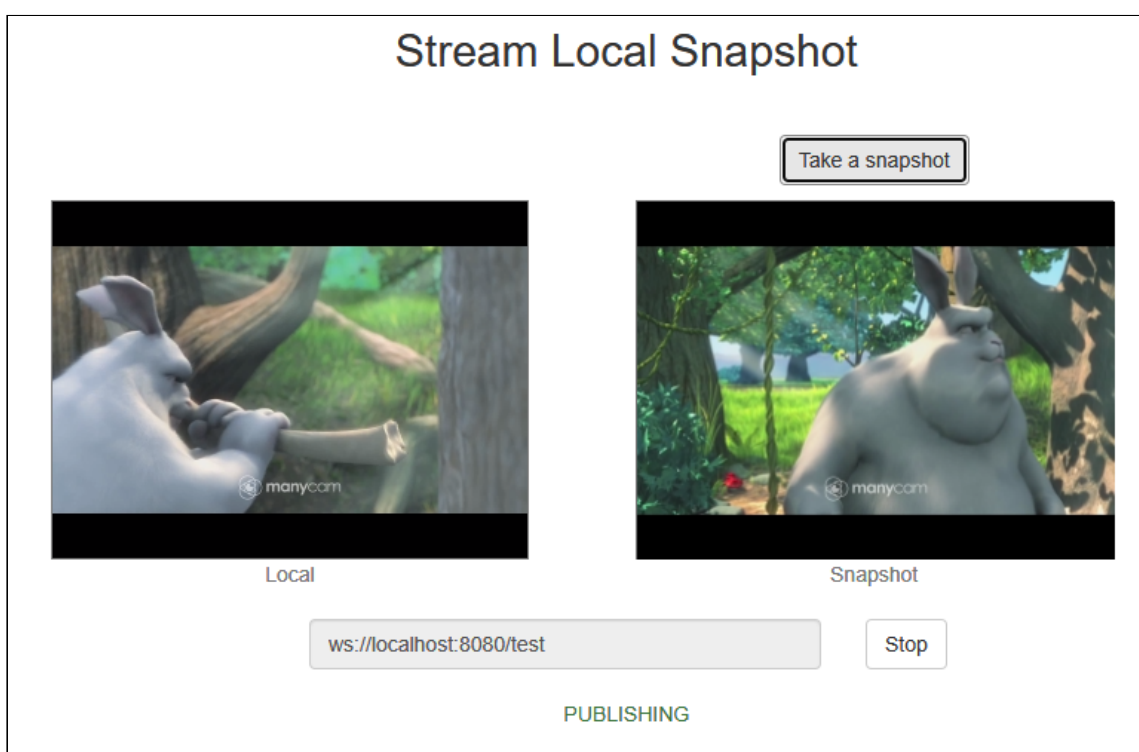


# Stream Local Snapshot

## Пример, демонстрирующий получение снимка опубликованного потока в браузере

В данном примере показано, как получить снимок потока, опубликованного на Web Call Server, локально в браузере.

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** с HTML5 видео элемента захватывается 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-local-snapshot
```

- stream-local-snapshot.css - файл стилей
- stream-local-snapshot.html - страница примера
- stream-local-snapshot.js - скрипт, обеспечивающий работу примера

Тестировать данный пример можно по следующему адресу:

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

Здесь host - адрес WCS-сервера.

## Работа с кодом примера

Для разбора кода возьмем версию файла `stream-local-snapshot.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. Инициализация элементов страницы и запоминание размеров картинки просмотра снимота

[code](#)

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

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

Здесь

- `localVideo` - `div`, в котором создается видеоэлемент для захвата потока
- `snapshotImg` - `img` элемент для просмотра картинки снимота
- `canvas` - элемент для отрисовки захваченного изображения и преобразования в PNG

- `snapshotImgSize` - структура для хранения размеров картинки просмотра снимка

### 3. 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(){
    ...
});
```

### 4. Получение от сервера события, подтверждающего успешное соединение

`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. 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();
```

## 6. Получение от сервера события, подтверждающего успешную публикацию

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. Вызов функции захвата снимка по нажатию кнопки

code

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

## 8. Отрисовка кадра, захваченного из видео, на канвасе, и преобразование в 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. Масштабирование картинки снимка для предварительного просмотра и добавление данных в `img` элемент

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. Служебная функция для масштабирования изображения

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. 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);
  ...
}
```

## 12. Получение от сервера события, подтверждающего успешную остановку публикации

`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();
```