Снятие превью трансляции в виде PNG

Описание

WCS предоставляет возможность снятия превью публикуемого потока при помощи REST-вызовов, а также при помощи JavaScript API.

Поддерживаемые протоколы

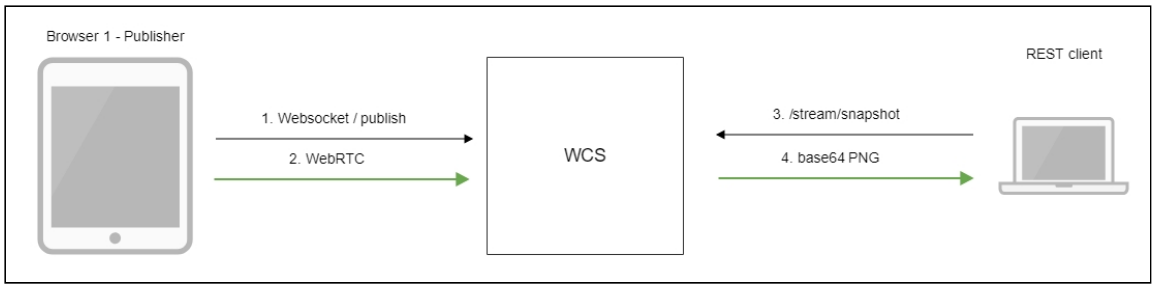
- WebRTC
- RTMP
- RTSP

Поддерживаемые форматы превью

- PNG

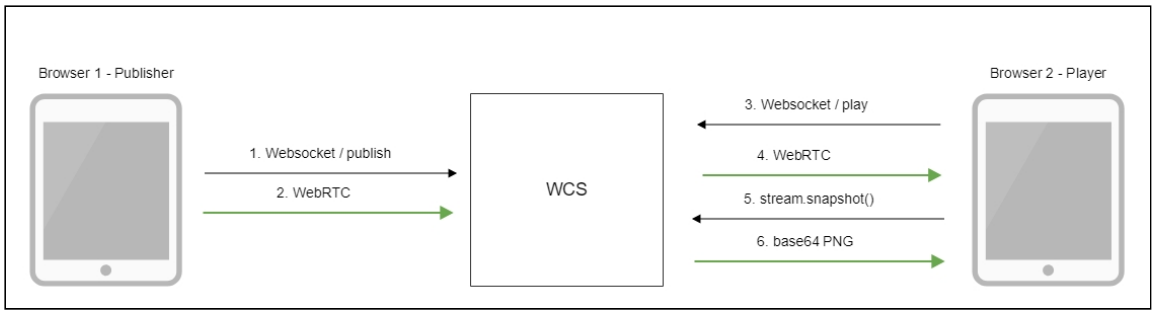
Схема работы

1: С использованием REST-запроса



1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду `publishStream`.
2. Браузер захватывает микрофон и камеру и отправляет WebRTC поток на сервер.
3. REST-клиент отправляет WCS REST-запрос `/stream/snapshot`.
4. REST-клиент получает ответ с превью потока, закодированным в base64.

2: С использованием JavaScript API**



1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду `publishStream`.
2. Браузер захватывает микрофон и камеру и отправляет WebRTC поток на сервер.
3. Второй браузер устанавливает соединение также по Websocket и отправляет команду `playStream`.
4. Второй браузер получает WebRTC поток и воспроизводит этот поток на странице.
5. Второй браузер вызывает `stream.snapshot()` для снятия превью.
6. Второй браузер получает ответ с превью потока, закодированным в base64.

REST-вызовы

WCS-сервер поддерживает REST-метод `/stream/snapshot` для снятия превью.

REST-запрос должен быть HTTP/HTTPS POST запросом в таком виде:

- HTTP: `http://streaming.flashphoner.com:8081/rest-api/stream/snapshot`
- HTTPS: `https://streaming.flashphoner.com:8444/rest-api/stream/snapshot`

Здесь:

- `streaming.flashphoner.com` - адрес WCS-сервера
- `8081` - стандартный REST / HTTP порт WCS-сервера
- `8444` - стандартный HTTPS порт
- `rest-api` - обязательная часть URL
- `/stream/snapshot` - используемый REST-метод

REST-методы и статусы ответа

REST метод	Тело запроса	Тело ответа	Статусы ответа

REST метод	Тело запроса	Тело ответа	Статусы ответа
`/stream/snapshot`	{ "streamName": "64966f33" }	{ "data": "iVBORw0KGgoAAAANSUhEUgAAAUAAAADwCAYAAAB...." }	200 OK 404 Stream not found 500 Internal server error

Параметры

Параметр	Описание	Пример
streamName	Уникальное имя потока	`64966f33`
data	Файл превью в base64-кодировке	`iVBORw0KGgoAAAANSUhEUgAAAUAAAADwCAYAAAABxLb1rAAAACXBIWXMAAAAAAAAAAQCEeRdzAAAQA...`

Отправка REST-запроса к WCS-серверу

Для отправки REST-запроса к WCS-серверу необходимо использовать REST-клиент.

Настройка

Начиная со сборки 5.2.1116, при получении превью трансляции при помощи REST API можно настроить максимальную длительность фиксации превью, включая возможную задержку при записи на диск сервера. По умолчанию, максимальная длительность установлена в 3000 мс, за это время предпринимается 30 попыток проверить, готов ли файл превью

```
snapshot_taking_interval_ms=3000
snapshot_taking_attempts=30
```

Если файл превью не готов по истечении указанного интервала, запрос `/stream/snapshot` возвращает ошибку

```
{
  "exception":
"com.flashphoner.rest.server.exception.InternalErrorException",
  "reason": "com.flashphoner.rest.server.exception.InternalErrorException,
```

```
Internal Server Error, Snapshot response timeout, ts: 1640836780816, path:
/rest-api/stream/snapshot",
  "path": "/rest-api/stream/snapshot",
  "error": "Internal Server Error",
  "message": "Snapshot response timeout",
  "timestamp": 1640836780816,
  "status": 500
}
```

JavaScript API

Для снятия превью трансляции при помощи WebSDK предназначен метод `snapshot` объекта `Stream`. Пример использования метода приведен в веб-приложении Stream Snapshot для публикации потока и снятия превью.

stream-snapshot.html

stream-snapshot.js

1. Из опубликованного потока создается новый поток
   code:

   ```
   function snapshot(name) {
       setSnapshotStatus();
       var session = Flashphoner.getSessions()[0];
       session.createStream({name: name}).on(STREAM_STATUS.SNAPSHOT_COMPLETE,
   function(stream){
       ...
   }
   ```

2. Вызывается метод `snapshot()`
   code:

   ```
   function snapshot(name) {
       setSnapshotStatus();
       var session = Flashphoner.getSessions()[0];
       session.createStream({name: name}).on(STREAM_STATUS.SNAPSHOT_COMPLETE,
   function(stream){
           ...
       }).snapshot();
   }
   ```

3. При получении события `SNAPSHOT_COMPLETE`, функция `stream.getInfo()` возвращает превью, закодированный в base64
   code:

   ```
   function snapshot(name) {
       setSnapshotStatus();
       var session = Flashphoner.getSessions()[0];
       session.createStream({name: name}).on(STREAM_STATUS.SNAPSHOT_COMPLETE,
   ```

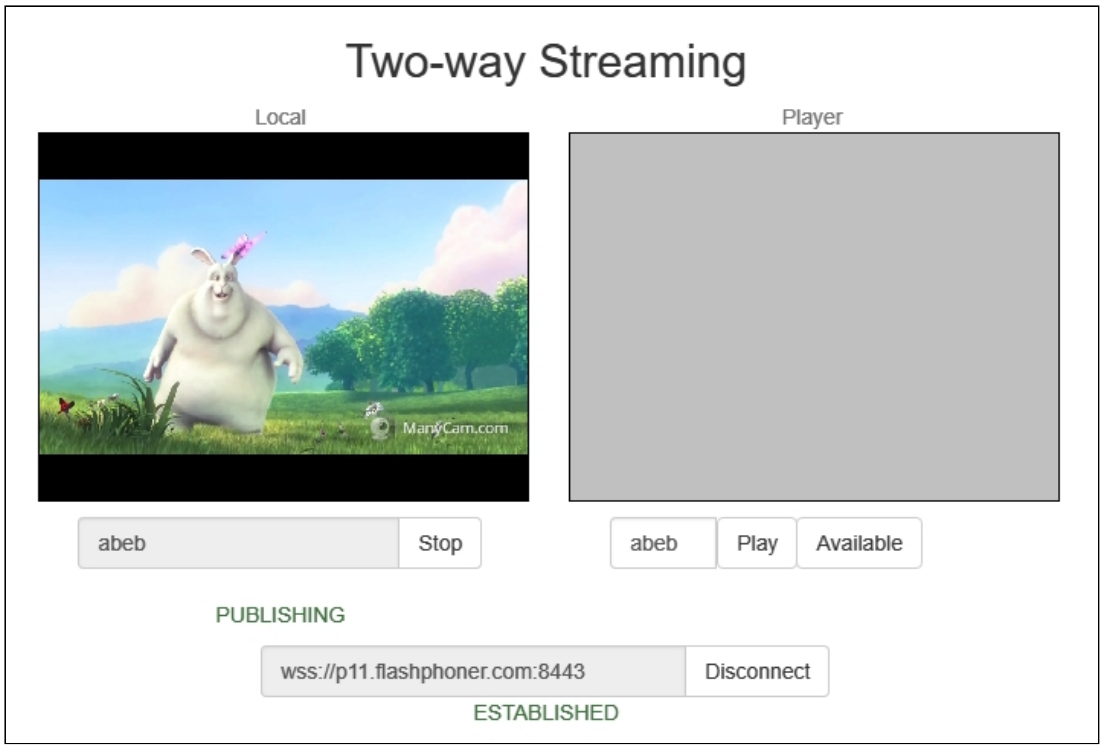
```
function(stream){
        console.log("Snapshot complete");
        setSnapshotStatus(STREAM_STATUS.SNAPSHOT_COMPLETE);
        snapshotImg.src = ""+stream.getInfo();
        ...
}
```

4. Поток останавливается
   code:

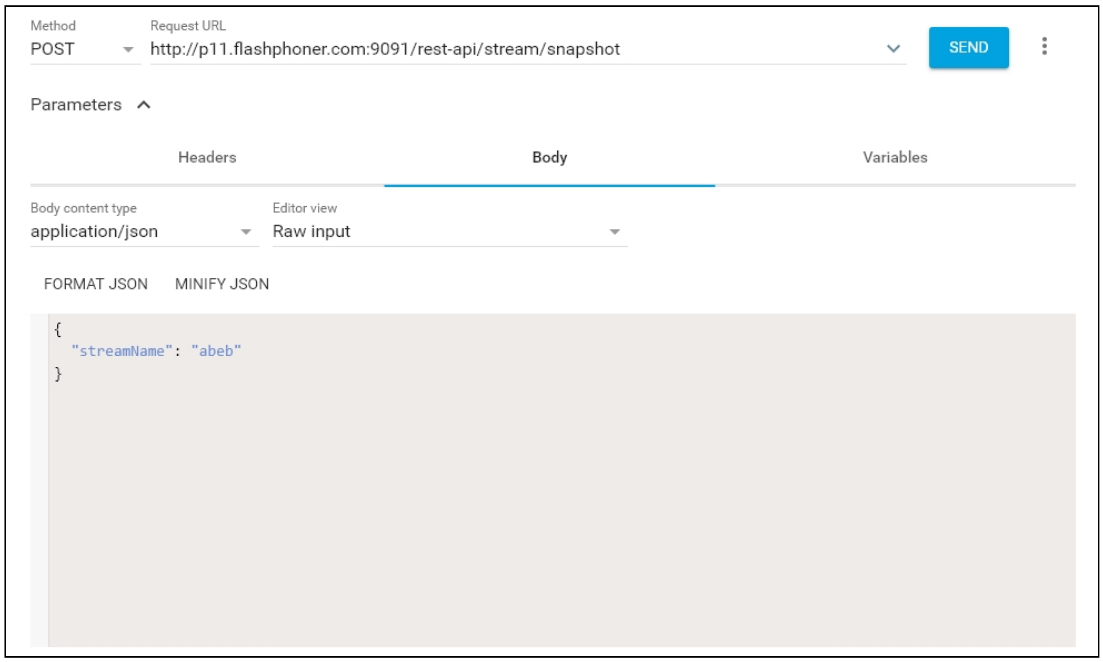
```
function snapshot(name) {
    setSnapshotStatus();
    var session = Flashphoner.getSessions()[0];
    session.createStream({name: name}).on(STREAM_STATUS.SNAPSHOT_COMPLETE,
function(stream){
        ...
        stream.stop();
    }).on(STREAM_STATUS.FAILED, function(stream){
        setSnapshotStatus(STREAM_STATUS.FAILED);
        console.log("Snapshot failed, info: " + stream.getInfo());
    }).snapshot();
}
```

Краткое руководство по тестированию

1. Для теста используем:
2. демо-сервер `demo.flashphoner.com`;
3. браузер Chrome и REST-клиент для отправки запросов на сервер;
4. веб-приложение Two Way Streaming для публикации потока;
5. сервис https://www.motobit.com/util/base64-decoder-encoder.asp для декодирования превью.
6. Откройте страницу веб-приложения Two Way Streaming. Нажмите `Connect`, затем нажмите `Publish` для публикации потока:

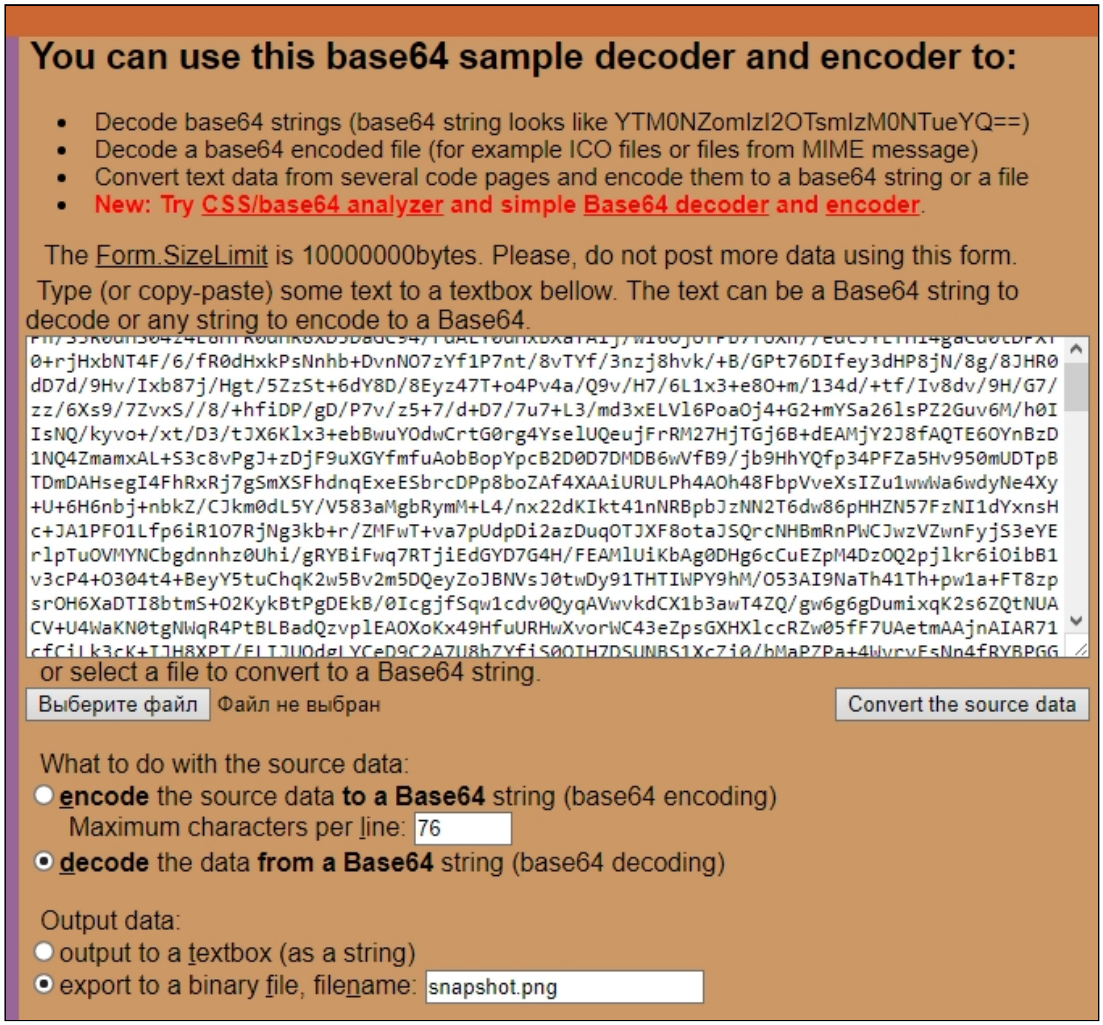


7. Откройте REST-клиент. Отправьте запрос `/stream/snapshot`, указав в параметрах идентификатор опубликованного потока:



8. Убедитесь, что ответ получен:

9. Откройте страницу онлайн-декодера, скопируйте в форму содержание ответа и нажмите `Convert the source data`:



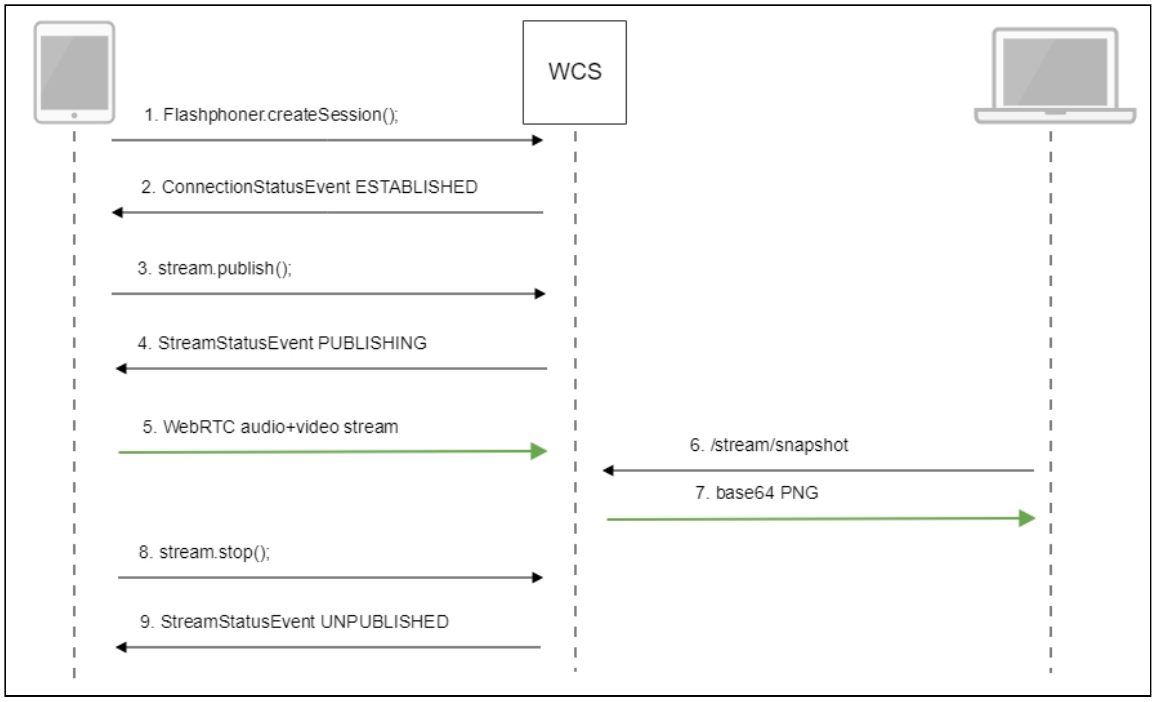
10. Полученный файл превью:

Последовательность выполнения операций

Ниже описана последовательность вызовов при использовании примера Stream Snapshot для публикации потока и снятия превью

stream-snapshot.html

stream-snapshot.js



1. Установка соединения с сервером
   `Flashphoner.createSession()` code

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

2. 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(){
            ...
        });
```

3. Публикация потока `stream.publish()` code

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

4. Получение от сервера события, подтверждающего успешную публикацию потока `STREAM_STATUS.PUBLISHING` code

```
     session.createStream({
        name: streamName,
        display: localVideo,
        cacheLocalResources: true,
        receiveVideo: false,
        receiveAudio: false
    }).on(STREAM_STATUS.PUBLISHING, function(publishStream){
        setStatus(STREAM_STATUS.PUBLISHING);
        onPublishing(publishStream);
    }).on(STREAM_STATUS.UNPUBLISHED, function(){
        ...
    }).on(STREAM_STATUS.FAILED, function(stream){
        ...
    }).publish();
```

5. Отправка аудио-видео потока по WebRTC
6. Снятие превью трансляции. Создается новый поток из опубликованного, специально для снятия превью

`stream.snapshot()` code

```
function snapshot(name) {
    setSnapshotStatus();
    var session = Flashphoner.getSessions()[0];
    session.createStream({name: name}).on(STREAM_STATUS.SNAPSHOT_COMPLETE,
function(stream){
        console.log("Snapshot complete");
        setSnapshotStatus(STREAM_STATUS.SNAPSHOT_COMPLETE);
        snapshotImg.src = ""+stream.getInfo();
        //remove failed callback
        stream.on(STREAM_STATUS.FAILED, function(){});
        //release stream object
        stream.stop();
    }).on(STREAM_STATUS.FAILED, function(stream){
        setSnapshotStatus(STREAM_STATUS.FAILED);
        console.log("Snapshot failed, info: " + stream.getInfo());
    }).snapshot();
}
```

7. Остановка публикации потока
   `stream.stop()` code

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

8. Получение от сервера события, подтверждающего остановку публикации потока
   `STREAM_STATUS.UNPUBLISHED` code

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

Автоматическое создание превью опубликованного потока

При необходимости, превью каждого потока, поддерживаемого формата, опубликованного на сервере, могут создаваться автоматически. Эта возможность включается при помощи настройки в файле flashphoner.properties

```
snapshot_auto_enabled=true
```

Расположение кадров превью задается настройкой

```
snapshot_auto_dir=/usr/local/FlashphonerWebCallServer/snapshots
```

В указанном каталоге для опубликованного потока создается подкаталог с именем, соответствующим идентификатору медиасессии (по умолчанию)

```
snapshot_auto_naming=mediaSessionId
```

или имени потока

```
snapshot_auto_naming=streamName
```

Кадры превью в каталоге нумеруются последовательно и создаются с периодичностью, заданной при помощи настройки

```
snapshot_auto_rate=30
```

В этом случае будет создано превью каждого 30 кадра.

Для экономии дискового пространства, может быть задано ограничение на количество хранимых кадров превью при помощи настройки

```
snapshot_auto_retention=20
```

В этом случае в каталоге для потока будут сохранены последние 20 кадров превью.

Если поток с таким же именем публикуется повторно, нумерация кадров превью будет продолжена.

Attachments:

. stream_snapshot_call_flow.jpg (image/jpeg)
. stream_snapshot-decode.jpg (image/jpeg)
. stream_snapshot-snapshot.png (image/png)

. stream_snapshot-publish.jpg (image/jpeg)
. stream_snapshot-request.jpg (image/jpeg)
. stream_snapshot-response.jpg (image/jpeg)
. stream_snapshot_deployment.jpg (image/jpeg)
. stream_snapshot_deployment2.jpg (image/jpeg)