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

Browser 1 - Publisher]	REST client
•	1. Websocket / publish 2. WebRTC	WCS	3. /stream/snapshot ← 4. base64 PNG	

- 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**

Browser 1 - Publisher	WCS	3. Websocket / play 4. WebRTC 5. stream.snapshot() 6. base64 PNG

- 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/snapsho t`	{ "streamName ": "64966f33" }	{	200 OK 404 Strea m not found 500 I nternal server erro r

Параметры

Параметр	Описание	Пример
streamName	Уникальное имя потока	`64966f33`
data	Файл превью в base64- кодировке	`iVBORw0KGgoAAAANSU hEUgAAAUAAAADwCAY AAABxLb1rAAAACXBIWX MAAAAAAAAAAAQCEeRd zAAAQA`

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



```
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 = "data:image/png;base64,"+stream.getInfo();
    ...
}
```

4. Поток останавливается

code:



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:

Two-way Streaming				
	Local		Pla	ayer
	ManýCarn.com			
abeb	Stop	abeb	Play	Available
PUE	BLISHING			
	wss://p11.flashphoner.com:84	43	Disconnect	t
ESTABLISHED				

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

Method Request URL POST Request URL http://p11.flashphoner.com:9091/rest-api/stream/snapshot			SEND :
Parameters 🔨			
Headers		Body	Variables
Body content type application/json	Editor view Raw input	Ψ	
<pre>FORMAT JSON MINIFY JSON { "streamName": "abeb" }</pre>	1		

8. Убедитесь, что ответ получен:

200	458.60 ms	DETAILS 🗸
ــــا ۲	ata": "iVBORw0KGgoAAAANSUhEUgAAAUAAAADwCAYAAABxLbirAAAACXBIWXMAAAAAAAAAAAAAACE hcmx1V5bvLMt3liXbkiWtrFlv105qVxu4icucwUyABAMTAg5IQOQcB1nzAADDIDJuadzijmH173 0jo6FismvdfoK0jo20+zPsv0MHR0TFf5v0K00jo63gw8/LdHR0d82Xef460jo6OULvP8CHR0dH RNV/m/Rfo60jomC/z/gt0dHR0zJd5/WU60jo65su8/wIdHR0d82Xef460jo6OULvP8CHR0dH RNV/m/Rfo60jomC/z/gt0dHR0zJd5/WU60jo65su8/wIdHR0d82Xef460jo6OULv9RCHR0dH RNV/m/Rfo60jomC/z/gt0dHR0zJd5/WU60jo65su8/wIdHR0d82Xef460jo6OULv9RCHR0dH RNV/m/Rfo60jomC/z/gt0dHR0zJd5/WJ60jo5su8/wIdHR0d82Xef460jo6OULv9RCHR0dH D/8Ey247T+o4Pv4a/Q9v/H7/6L1X3+e80+m/134d/+tf/Iv8dv/9H/G7/zz/6Xs9/7Zvx5//8/H 2+mY5a261sP22Guv6M/h0IIsNQ/kyvo+/xt/03/t3X6Xix3+ebBwuY0dwCrt60rg4YselUQeujfr 3c8vPg3+zDjf9uXGYfmfuAobBopYpc82D0D7DND66wYf89/jb9HhYQfp34PF2a5Hv959MUD7pBT 3c8vPg3+zDjf9uXGYfmfuAobBopYpc82D0D7DND66wYf89/jb9HhYQfp34PF2a5Hv959MUD7pBT 3klRULPHAAbA8FbpVvxSIIZu1wNw8dwdyle42V+U+6Hnobj+hb2/C3Km0dL5V/V583aHgBkgNM xnsHc+JA1PF01Lfp6iR107RjNg3kb+r/ZMFwT+va7pUdpD12azDuqOTJXF8otaJSQrcNHBmRnPWC 7RTjEdGYD764H/FEAN1UXkbAg0DHg6CcUZp4M2OQ2pj1kr6i01b81v3cP4+0304H45BeyY5v U9NB51k2i0/bMaPZpa+4WivvFSNn4fRYBPGG0jV82AZT4EY6fCmB/sECjvyCckXDDANbHL5MM gR4PELBadQzvp1EA0X6Xx49HfuURHwXvorWC43e2psGXHX1ccR2w05FF7UAetmAAjn1AR71cfC DSUNB51k2ci0/bMaPZpa+4WivvFSNn4fRYBPGG0jV82AZT4EY6fCmB/sECjvyCckXDDANbHL5MM HxYwGcJBE7CHU7APSJFgVX57PeFBKVv6ybUgcconNsB0ADSFFTUG2Ta+fxMpoz5HRa2Jp pLEzxiFxM8fz0j8kskYMomQUNGr1gngPKo+H0azDOC92jCHjCH1sWkEZZtHm/e1D+CUZqd8KWEXH xSwbHrWMX8GCNRU0fNWj85b5nAG8SgVukwZMWHXC45LER84bVStQDNQHMhrxy2wngF ZU2jUSDft1DHjtc1520ZXHTky14S87jsKqDbUX19N0X75tdC2Q3jVgE+fnzsAeLCN106ID0H xg/UagJBXAwny0PW6ANnF8c+eadz3wZWLLX4HL8r74KET8WNNOSEdJ6JFiNJS7vgU8BVFX7V04geVpAdgL /LydfH8LGbE51+YQ+Hv1587jsKqDbUX19N0X75tdC2Q3jVFE+fnzsAeLCN106ID0H xg/UagJSWA/mq0N766NF8c+eadg23UQUKLX4HzqHL8r7AtET8WNNSEdJ6JFiNS7v0dggeVpAdgL /LydfH8LGbE51+YQ+Hv159FXw118qA4rfAAJHgXqETBW1AVKNSC5dJ0FJNSPycdBU0VXXS zg/J3JXLCaU++1Jd19xpA86gENBWPR38qfsk0DJ7FA115TSx5BbL92BHjVST70dggeVpAdgL /LydfH8LGbE51+YQ+Hv159FXw118qA4rfAAJHgXqETBW1AVKACQ5dJ0FJWSFYcdBgVFVQFg oSHSAfnGqVKs6QUAXqegz51QQNGnu+UJ12EFZT0z0J	ERdzAAAQAElEQVR4nOzd95Pcd37feVjhrFKwXT6fr+r+g 3f74/30216g04GiCHIZtWjeqZnejAAu5/9+XzD57tgwYI fHN1an+Bjo6Ojvky779AR0dHx3yZ91+go6OjY77H+J(00 33K6Cjo6HjvsZrL9DR0dExXH9F+jo6OiYL/P+C3R0dHT ALV%dHxEXafAlj/w1E0jofPb7foXh//edJYEfH14gaC 3dHP3JN/8g/8JHR0dD7d/9Hv/Ixb87j/Hgt/5Z2St+6d hfDP/gD/P7v/z5+7/d+D7/Tv7+L3/md3xELV16Poa0j4 rM27HjTgi6B+dEAHjY23afAQTE6OYn8zD1NQ4ZmamxAL JmDAHseg14FhRx8fjZgSmX5FhdnqExeESbrcDPBboZAf4 HL4/nx22dK1kt41NRBpJJNN2T6dMs6pHH2DS7FzHI1 CJwzVZwnFyjS3eYEr1pTuOVMYWCbgdnnhz0Uhi/gRYBiF uChqK2w5bvZm5DQeyZoJBNv5J0twDy91THTIWFY9hV/05 Lib3awT42Q/gw6g6gDumixq23s6ZUNACVHUAKXN0tg ciLk3c+LJH3XPT/ELIJU0dgLYCeD02A7U8hzYfiS00I dwLYCWAngJ0AthbA+hi2+udyAA/pETxhqASQ06e07X5qy ESOVO3Xv10j/KU16Mnr5D4DbD20X2T6A632VcPyuuoFBXxF 46zcpD06zf4pzYH35nVNaQ5OVz5WnzfrsFFHMdNkm3bEn ff5MkHv68E80ADsB/B1HsF2NAshha2ZR517idKWwx1 PH40J1g022Q(qdwsh1A)/39KH4M06Z5b8x2v5yJSRy EBW140xc0hnyg1MgIR0ibw1THoF4UA1657QswE880AL/ SEHIV0gVgWRWUXV3J1/R7+kESPATgA7AW0gEdp5HfMQ 44sAWQwfrHvxswFnaCB111py11FE0X+paRd17aaMdW
	YSRUbe81y99564UL2B15gU/AXTNS0HS/HTM/wCmlpCu02/AUU0eynx/JSR5HX09ey6AUL2DUCUmbu e9titY4jdqm6Hb6m4M5ULyDfJklWIAtUIKd6mTxFXSIGIY/hc1qwd/PHNQvH90GeFYbKVDbikhhc ax2w17XSzmonM841CDR6caftvX=hum4zhmuHaKafdaK1hXYHvyEHvEFz1HaAHfwataDuB1Hc25m2Va4	wyBFV5Kaqq6HWZNAcocF2WChXCYzo+WPKM3X+ez8yg8N2 csjGCiKfKKGQyIISXFOkOTIrRGIiH47K7YIN9M7dDhXBd wwV+BallRW17nTR6T3Vh61770hw4h+UrMdv14C1Hx+Tz16

9. Откройте страницу онлайн-декодера, скопируйте в форму содержание ответа и

нажмите Convert the source data:



• export to a binary file, filename: snapshot.png

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



2. Получение от сервера события, подтверждающего успешное соединение SESSION_STATUS.ESTABLISHED code



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



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

stream.snapshot() code



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)