

Перенаправление SIP-звонка в поток (функция SIP as Stream)

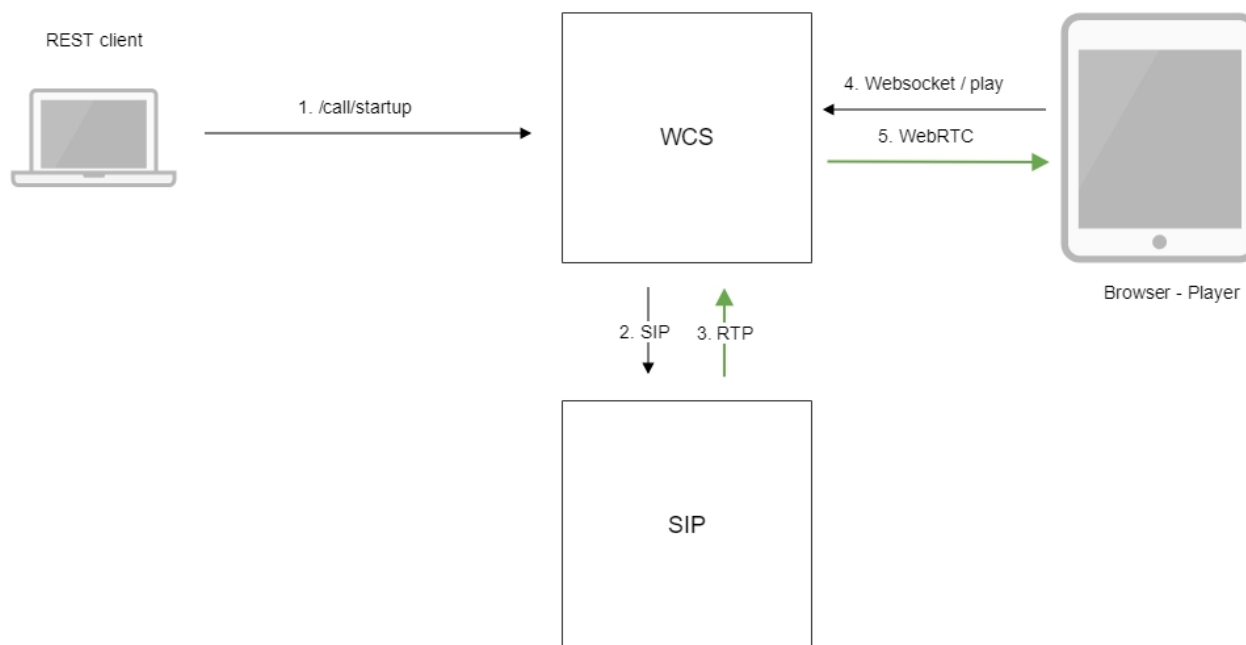
- [Описание](#)
 - [Схема работы](#)
- [Краткое руководство по тестированию](#)
- [Последовательность выполнения операций \(Call Flow\)](#)
- [Запись потоков SIP-звонков](#)
- [Известные проблемы](#)

Описание

SIP-звонок, произведенный через WCS-сервер, может быть захвачен в поток на сервере при создании звонка. Затем этот поток можно воспроизвести в браузере [любым способом из поддерживаемых WCS](#).

Поток, захваченный из SIP-звонка, может быть [ретранслирован на RTMP-сервер](#) при помощи REST-запроса `/push/startup`, как любой медиапоток на WCS-сервере.

Схема работы



1. Браузер начинает звонок с помощью REST-вызова `/call/startup`
2. WCS соединяется с SIP-сервером
3. SIP-сервер передает RTP-поток звонка на WCS
4. Второй браузер запрашивает воспроизведение потока звонка
5. Второй браузер получает WebRTC-поток

Краткое руководство по тестированию

1. Для тестирования используем:

- два SIP-аккаунта;
- программный телефон для ответа на звонок;
- [REST-клиент](#) в браузере Chrome;
- веб-приложение [Player](#) для воспроизведения потока.

2. Откройте REST-клиент. Отправьте запрос `/call/startup` на WCS-сервер, указав в параметрах запроса:

- параметры Вашего SIP-аккаунта, с которого будет совершен звонок
- имя потока для ретрансляции звонка (параметр toStream), например, call_stream1
- имя Вашего второго SIP-аккаунта, на который будет совершаться звонок.

Method: POST, Request URL: http://p11.flashphoner.com:9091/rest-api/call/startup

Parameters: Headers, Body, Variables

Body content type: application/json, Editor view: Raw input

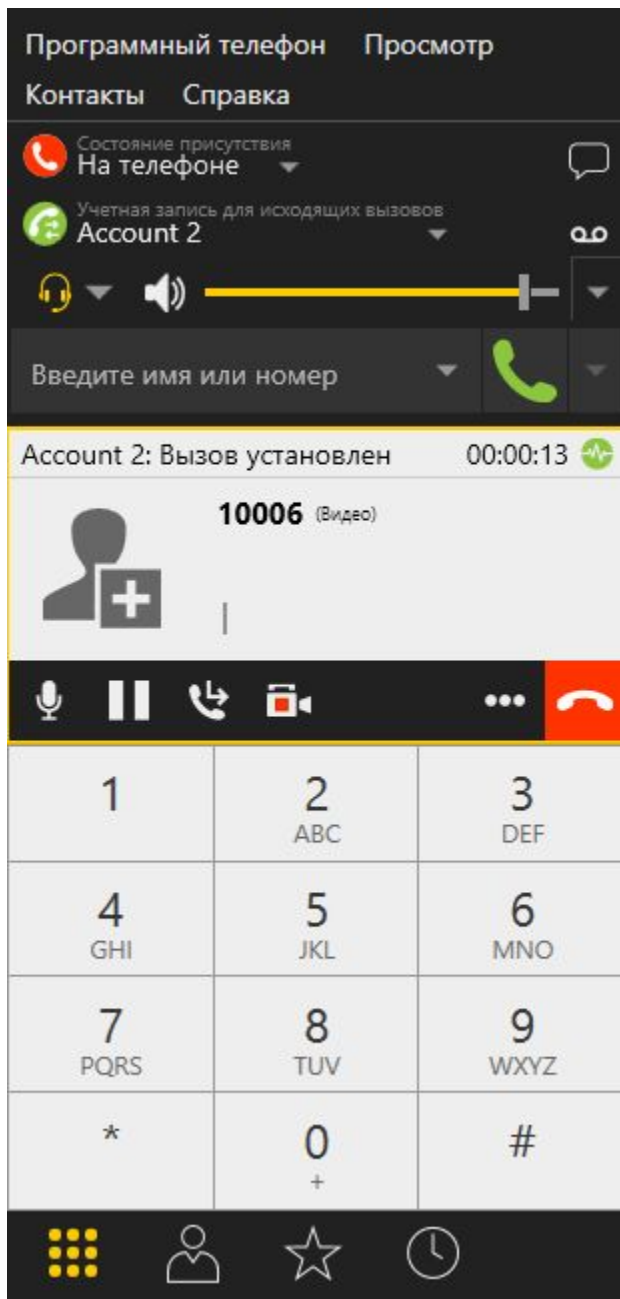
FORMAT JSON, MINIFY JSON

```
{
  "callId": "12345678910",
  "callee": "10005",
  "toStream": "call_stream1",
  "hasAudio": "true",
  "hasVideo": "true",
  "sipLogin": "10006",
  "sipAuthenticationName": "10006",
  "sipPassword": "*****",
  "sipDomain": "domain.net",
  "sipOutboundProxy": "domain.net",
  "sipPort": "5060",
  "appKey": "defaultApp",
  "sipRegisterRequired": "true"
}
```

200 OK 90.70 ms

DETAILS

3. Примите входящий звонок на программном телефоне:



4. Откройте веб-приложение Player, укажите в поле "Stream" имя потока, в который перенаправлен звонок (в нашем примере call_stream1):

WCS URL


Stream

Volume

Full Screen

5. Нажмите "Play". Начнется воспроизведение потока:

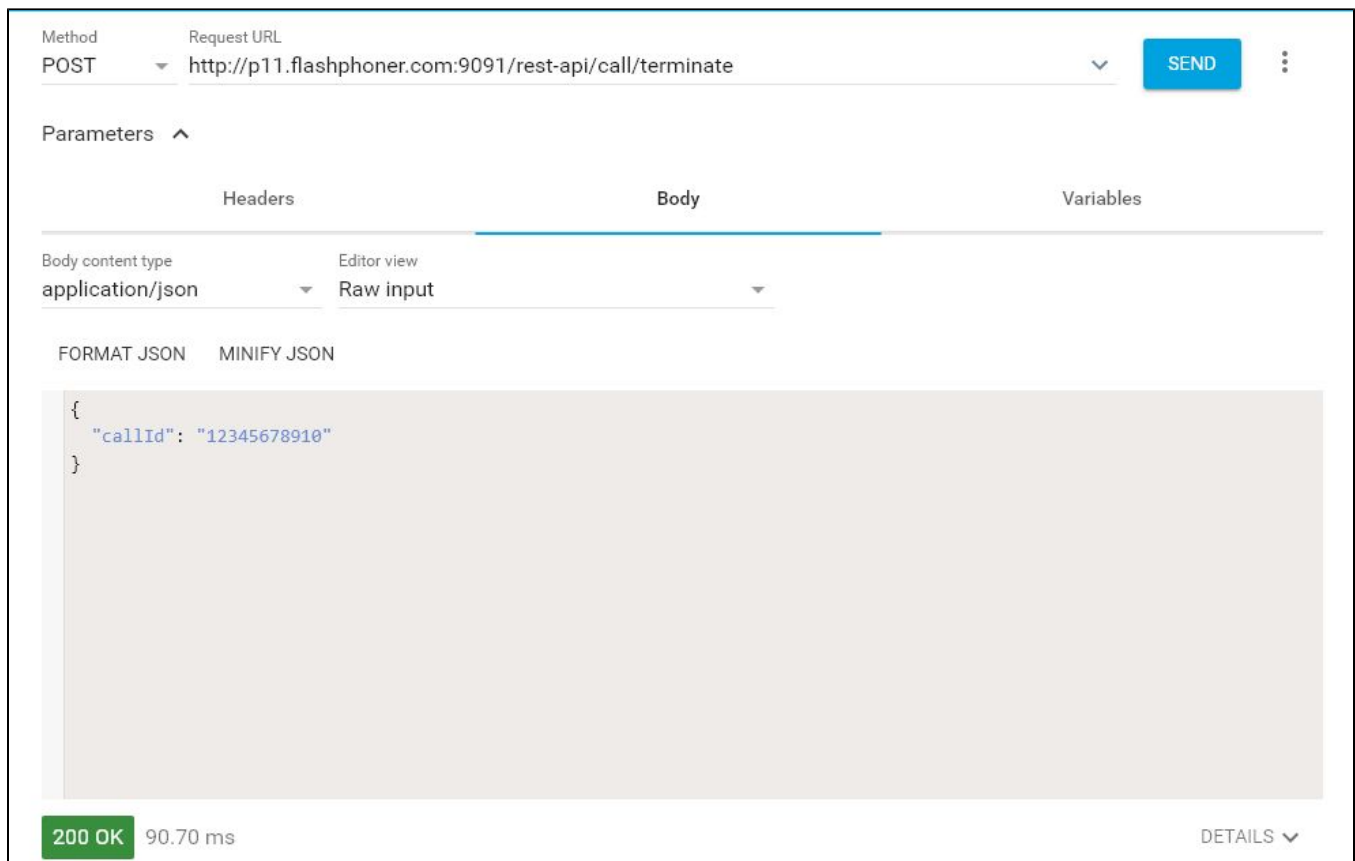
Player



WCS URL

Stream

6. Для завершения звонка отправьте из REST-клиента запрос /call/terminate на WCS-сервер, указав в параметрах запроса идентификатор звонка:



Последовательность выполнения операций (Call Flow)

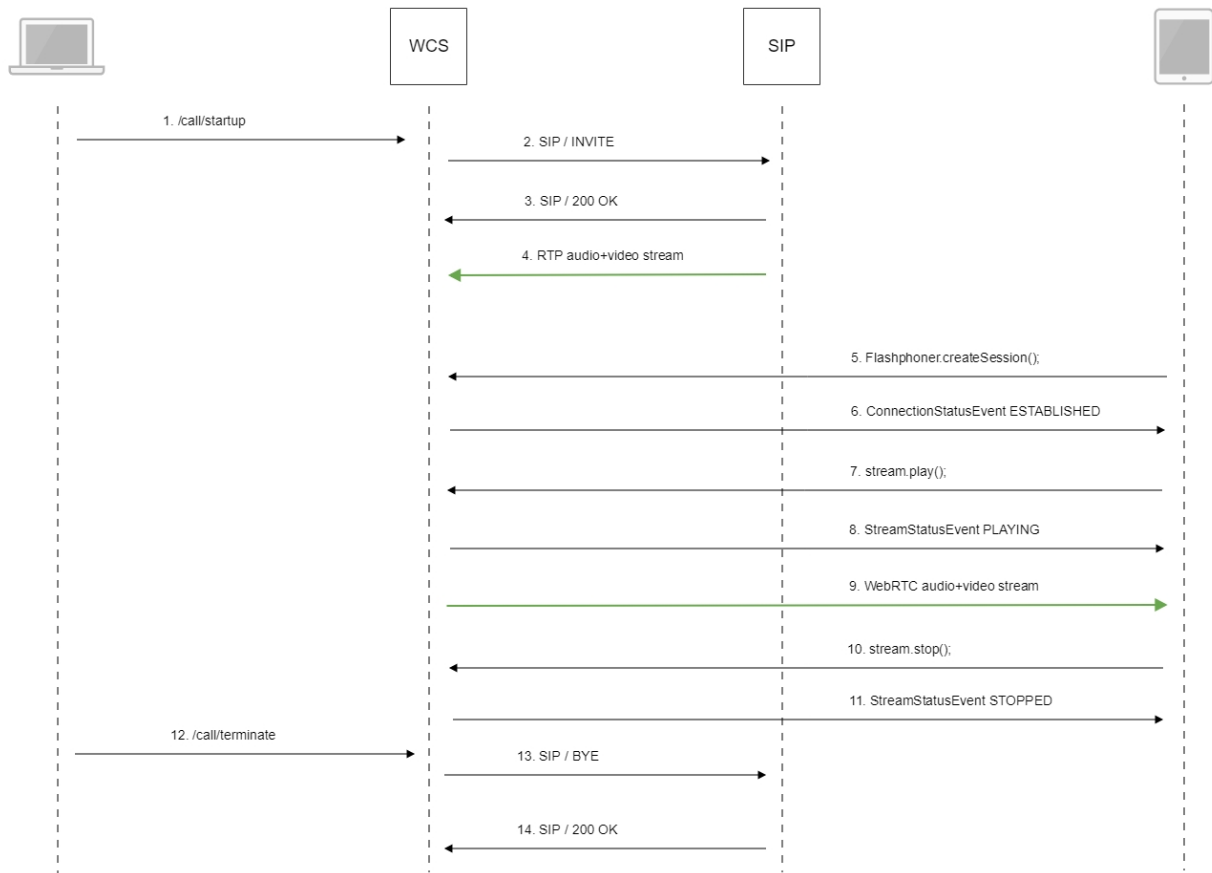
Ниже описана последовательность вызовов при использовании примера SIP as RTMP для создания звонка и примера Player для его воспроизведения

[sip-as-rtmp-4.html](#)

[sip-as-rtmp-4.js](#)

[player.html](#)

[player.js](#)



1. Отправка REST-запроса `/call/startup`:

`sendREST()`[code](#)

```

function startCall() {
    ...
    var url = field("restUrl") + "/call/startup";
    callId = generateCallID();
    ...
    var RESTCall = {};
    RESTCall.toStream = field("rtmpStream");
    RESTCall.hasAudio = field("hasAudio");
    RESTCall.hasVideo = field("hasVideo");
    RESTCall.callId = callId;
    RESTCall.sipLogin = field("sipLogin");
    RESTCall.sipAuthenticationName = field("sipAuthenticationName");
    RESTCall.sipPassword = field("sipPassword");
    RESTCall.sipPort = field("sipPort");
    RESTCall.sipDomain = field("sipDomain");
    RESTCall.sipOutboundProxy = field("sipOutboundProxy");
    RESTCall.appKey = field("appKey");
    RESTCall.sipRegisterRequired = field("sipRegisterRequired");

    for (var key in RESTCall) {
        setCookie(key, RESTCall[key]);
    }

    RESTCall.callee = field("callee");

    var data = JSON.stringify(RESTCall);

    sendREST(url, data);
    startCheckCallStatus();
}

```

2. Установка соединения с SIP-сервером

3. Получение подтверждения от SIP-сервера

4. RTP-поток звонка передается на WCS-сервер

5. Установка соединения браузера с сервером.

Flashphoner.createSession();[code](#)

```

Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
    setStatus(session.status());
    //session connected, start playback
    playStream(session);
}).on(SESSION_STATUS.DISCONNECTED, function(){
    setStatus(SESSION_STATUS.DISCONNECTED);
    onStopped();
}).on(SESSION_STATUS.FAILED, function(){
    setStatus(SESSION_STATUS.FAILED);
    onStopped();
});

```

6. Получение от сервера события, подтверждающего успешное соединение.

ConnectionStatusEvent ESTABLISHED[code](#)

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
    setStatus(session.status());
    //session connected, start playback
    playStream(session);
}).on(SESSION_STATUS.DISCONNECTED, function(){
    ...
}).on(SESSION_STATUS.FAILED, function(){
    ...
});
```

7. Запрос на воспроизведение потока.

`stream.play();code`

```
stream = session.createStream(options).on(STREAM_STATUS.PENDING, function(stream) {
    var video = document.getElementById(stream.id());
    if (!video.hasListeners) {
        video.hasListeners = true;
        video.addEventListener('playing', function () {
            $("#preloader").hide();
        });
        video.addEventListener('resize', function (event) {
            var streamResolution = stream.videoResolution();
            if (Object.keys(streamResolution).length === 0) {
                resizeVideo(event.target);
            } else {
                // Change aspect ratio to prevent video stretching
                var ratio = streamResolution.width / streamResolution.height;
                var newHeight = Math.floor(options.playWidth / ratio);
                resizeVideo(event.target, options.playWidth, newHeight);
            }
        });
    }
    ...
});
stream.play();
```

8. Получение от сервера события, подтверждающего успешное проигрывание потока.

`StreamStatusEvent, статус PLAYINGcode`

```
stream = session.createStream(options).on(STREAM_STATUS.PENDING, function(stream) {
    ...
}).on(STREAM_STATUS.PLAYING, function(stream) {
    $("#preloader").show();
    setStatus(stream.status());
    onStarteds(stream);
    ...
});
stream.play();
```

9. Отправка аудио-видео потока по WebRTC

10. Остановка воспроизведения потока.

`stream.stop();code`


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

11. Получение от сервера события, подтверждающего остановку воспроизведения потока.

StreamStatusEvent, статус STOPPED [code](#)

```
stream = session.createStream(options).on(STREAM_STATUS.PENDING, function(stream) {
    ...
}).on(STREAM_STATUS.PLAYING, function(stream) {
    ...
}).on(STREAM_STATUS.STOPPED, function() {
    setStatus(STREAM_STATUS.STOPPED);
    onStopped();
}).on(STREAM_STATUS.FAILED, function(stream) {
    ...
}).on(STREAM_STATUS.NOT_ENOUGH_BANDWIDTH, function(stream){
    ...
});
stream.play();
```

12. Отправка REST-запроса /call/terminate:

sendREST() [code](#)

```
function hangup() {
    var url = field("restUrl") + "/call/terminate";
    var currentCallId = { callId: callId };
    var data = JSON.stringify(currentCallId);
    sendREST(url, data);
}
```

13. Отправка команды на SIP-сервер

14. Получение подтверждения от SIP-сервера

Запись потоков SIP-звонков

Потоки, полученные из SIP-звонков, могут быть записаны на сервере. Для этого необходимо указать следующие настройки в файле [flashphoner.properties](#):

```
sip_single_route_only=true
sip_record_stream=true
```

При этом поддерживаются следующие кодеки:

- Видео: H264
- Аудио: opus, PCMA (alaw), PCMU (ulaw)

Запись потоков на сервере подробно описана в [соответствующем разделе](#).

Известные проблемы

1. Поток, захваченный из звонка, не проигрывается, если не инициализирована RTP-сессия для этого потока

Симптомы: поток звонка создается на сервере, но не воспроизводится.

Решение: включить принудительную инициализацию RTP-сессии при помощи настройки

```
rtplib_session_init_always=true
```