

SIP as RTMP 2

Пример доставки видеопотока из SIP звонка на RTMP-сервер с добавлением звуковой дорожки к потоку

Данный пример показывает, как можно сделать вызов на SIP, получить от SIP стороны аудио и видео трафик, добавить к потоку звуковую дорожку и затем перенаправить полученный видеопоток на RTMP-сервер

SIP as RTMP Broadcasting

SIP Details

Login:

SIP Auth Name:

Password:

Domain:

SIP Outbound Proxy:

Port:

App Key:

Register Required hasAudio hasVideo

RTMP Target Details

RTMP URL:

Stream:

Mute: Music:

RTMP playback URL

Copy this URL to a third party player

SL8ERz-QC03TM1gM-zhefDih-ULE1V36 >>> rtmp://localhost:1935/live

ESTABLISHED

Код примера

Пример представляет собой простого REST-клиента, написанного на JavaScript и находится по следующему пути:

`/usr/local/FlashphonerWebCallServer/client2/examples/demo/sip/sip-as-rtmp-2`

- sip-as-rtmp-2.js - скрипт, обеспечивающий REST вызовы на WCS-сервер
- sip-as-rtmp-2.html - страница примера

Тестировать данный пример можно по следующему адресу:

`https://host:8888/client2/examples/demo/sip/sip-as-rtmp-2/sip-as-rtmp-2.html`

Здесь host - адрес WCS-сервера.

Работа с кодом примера

Для разбора кода возьмем версию файла `sip-as-rtmp-2.js` с хешем `ecbadc3`, которая находится [здесь](#) и доступна для скачивания в соответствующей сборке [2.0.212](#).

1. Отправка REST / HTTP - запросов

[code](#)

Отправка происходит методом POST с `ContentType: application/json` AJAX запросом с использованием фреймворка jquery.

```
function sendREST(url, data, successHandler, errorHandler) {
  console.info("url: " + url);
  console.info("data: " + data);
  $.ajax({
    url: url,
    beforeSend: function ( xhr ) {
      xhr.overrideMimeType( "text/plain;" );
    },
    type: 'POST',
    contentType: 'application/json',
    data: data,
    success: (successHandler === undefined) ? handleAjaxSuccess :
successHandler,
    error: (errorHandler === undefined) ? handleAjaxError : errorHandler
  });
}
```

2. Создание исходящего звонка при помощи REST-запроса

`/call/startup`

[code](#)

Из текстовых форм собираются данные для установки соединения и звонка

`(RESTCall)`

```
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();

```

3. Получение статуса звонка запросом `/call/find`

code

```

function getStatus() {
  var url = field("restUrl") + "/call/find";
  currentCallId = { callId: callId };
  $("#callTrace").text(callId + " >>> " + field("rtmpUrl"));
  var data = JSON.stringify(currentCallId);
  sendREST(url, data);
}

```

4. Отправка DTMF сигнала запросом `/call/send_dtmf`

code

```

function sendDTMF(value) {
  var url = field("restUrl") + "/call/send_dtmf";
  var data = {};
  data.callId = callId;
  data.dtmf = value;
  data.type = "RFC2833";
  data = JSON.stringify(data);
  sendREST(url, data);
}

```

5. Ретрансляция звонка на RTMP-сервер с добавлением звукового файла в поток запросом `/push/startup`

code

```

function startRtmpStream() {
  if (!rtmpStreamStarted) {
    rtmpStreamStarted = true;
    var url = field("restUrl") + "/push/startup";
    var RESTObj = {};
    var options = {};
    if ($("#mute").is(':checked')) {
      options.action = "mute";
    } else if ($("#music").is(':checked')) {
      options.action = "sound_on";
      options.soundFile = "sample.wav";
    }
    RESTObj.streamName = field("rtmpStream");
    RESTObj.rtmpUrl = field("rtmpUrl");
    RESTObj.options = options;
    sendREST(url, JSON.stringify(RESTObj), startupRtmpSuccessHandler,
startupRtmpErrorHandler);
    sendDataToPlayer();
    startCheckTransponderStatus();
  }
}

```

6. Получение статуса RTMP-потока запросом `/push/find`

code

```

function getTransponderStatus() {
  var url = field("restUrl") + "/push/find";
  var RESTObj = {};
  // By default transponder's stream name will contain prefix "rtmp_"
  RESTObj.streamName = "rtmp_" + field("rtmpStream");
  RESTObj.rtmpUrl = field("rtmpUrl");
  sendREST(url, JSON.stringify(RESTObj), transponderStatusSuccessHandler,
transponderStatusErrorHandler);
}

```

7. Включение/отключение звука RTMP-потока

Отключение звука `/push/mute` code

```

function mute() {
  if (rtmpStreamStarted) {
    $("#mute").prop('disabled', true);
    var RESTObj = {};
    RESTObj.mediaSessionId = rtmpMediaSessionId;
    var url = field("restUrl") + "/push/mute";
    sendREST(url, JSON.stringify(RESTObj), muteSuccessHandler,
muteErrorHandler);
  }
}

```

Включение звука `/push/unmute` code

```
function unmute() {
  if (rtmpStreamStarted) {
    $("#mute").prop('disabled', true);
    var RESTObj = {};
    RESTObj.mediaSessionId = rtmpMediaSessionId;
    var url = field("restUrl") + "/push/unmute";
    sendREST(url, JSON.stringify(RESTObj), muteSuccessHandler,
muteErrorHandler);
  }
}
```

8. Включение/отключение дополнительной звуковой дорожки RTMP-потока

Включение звуковой дорожки из файла `/push/sound_on` [code](#)

```
function soundOn() {
  if (rtmpStreamStarted) {
    $("#music").prop('disabled', true);
    var RESTObj = {};
    RESTObj.mediaSessionId = rtmpMediaSessionId;
    RESTObj.soundFile = "sample.wav";
    RESTObj.loop = false;
    var url = field("restUrl") + "/push/sound_on";
    sendREST(url, JSON.stringify(RESTObj), injectSoundSuccessHandler,
injectSoundErrorHandler);
  }
}
```

Отключение звуковой дорожки `/push/sound_off` [code](#)

```
function soundOff() {
  if (rtmpStreamStarted) {
    $("#music").prop('disabled', true);
    var RESTObj = {};
    RESTObj.mediaSessionId = rtmpMediaSessionId;
    var url = field("restUrl") + "/push/sound_off";
    sendREST(url, JSON.stringify(RESTObj), injectSoundSuccessHandler,
injectSoundErrorHandler);
  }
}
```

9. Завершение звонка запросом `/call/terminate`

[code](#)

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

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

10. Отображение RTMP URL на странице для копирования в сторонний плеер

code

```
function sendDataToPlayer() {  
    var host = field("rtmpUrl")  
        .replace("localhost", window.location.hostname)  
        .replace("127.0.0.1", window.location.hostname);  
  
    var rtmpStreamPrefix = "rtmp_";  
    var url = host + "/" + rtmpStreamPrefix + field("rtmpStream");  
    $("#player").text(url);  
}
```