

SIP as RTMP 2

- [Example of delivery of video stream from SIP to RTMP server with sound injection to the stream](#)
- [The code of the example](#)
- [Analyzing the code](#)

Example of delivery of video stream from SIP to RTMP server with sound injection to the stream

This example demonstrates how to make a call to SIP, receive audio and video traffic from SIP in response, inject sound from file to the received videostream and then redirect the stream to a third-party RTMP server for further broadcasting

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

rtmp://demo.flashphoner.com:1935/live/rtmp_stream1

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

ESTABLISHED

The code of the example

This example is a simple REST client written on JavaScript, available at:

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

`sip-as-rtmp-2.js` - a script dealing with REST queries to the WCS server
`sip-as-rtmp-2.html` - example page

The example may be tested at this URL:

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

where host is WCS server address.

Analyzing the code

To analyze the code get `sip-as-rtmp-2.js` file version with hash `ecbadc3` that can be found [here](#) and is available to download in build [2.0.212](#).

1. REST / HTTP queries sending.

[code](#)

Sending is done using POST method with ContentType application/json by AJAX query using jQuery framework.

```
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. Making outgoing call with REST-request /call/startup

[code](#)

Call data (RESTCall) are collected from the boxes on page

```
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. Getting the SIP call status with /call/find REST query.

[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. Sending DTMF signal with /call/send_dtmf REST query.

[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. Re-publishing the SIP call stream to an RTMP server with sound file injecting to the stream by /push/startup REST query

[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. Getting RTMP stream status with /push/find REST query.

[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. Mute/unmute stream sound.

Mute sound with /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);
  }
}
```

Unmute sound with /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. Injecting additional sound to RTMP stream.

Injecting sound from file with /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);
  }
}
```

Stop injecting sound from file with /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. Hangup the SIP call with /call/terminate REST query.

[code](#)

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

10. RTMP URL displaying on the page to copy to a third party player

[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);
}
```

