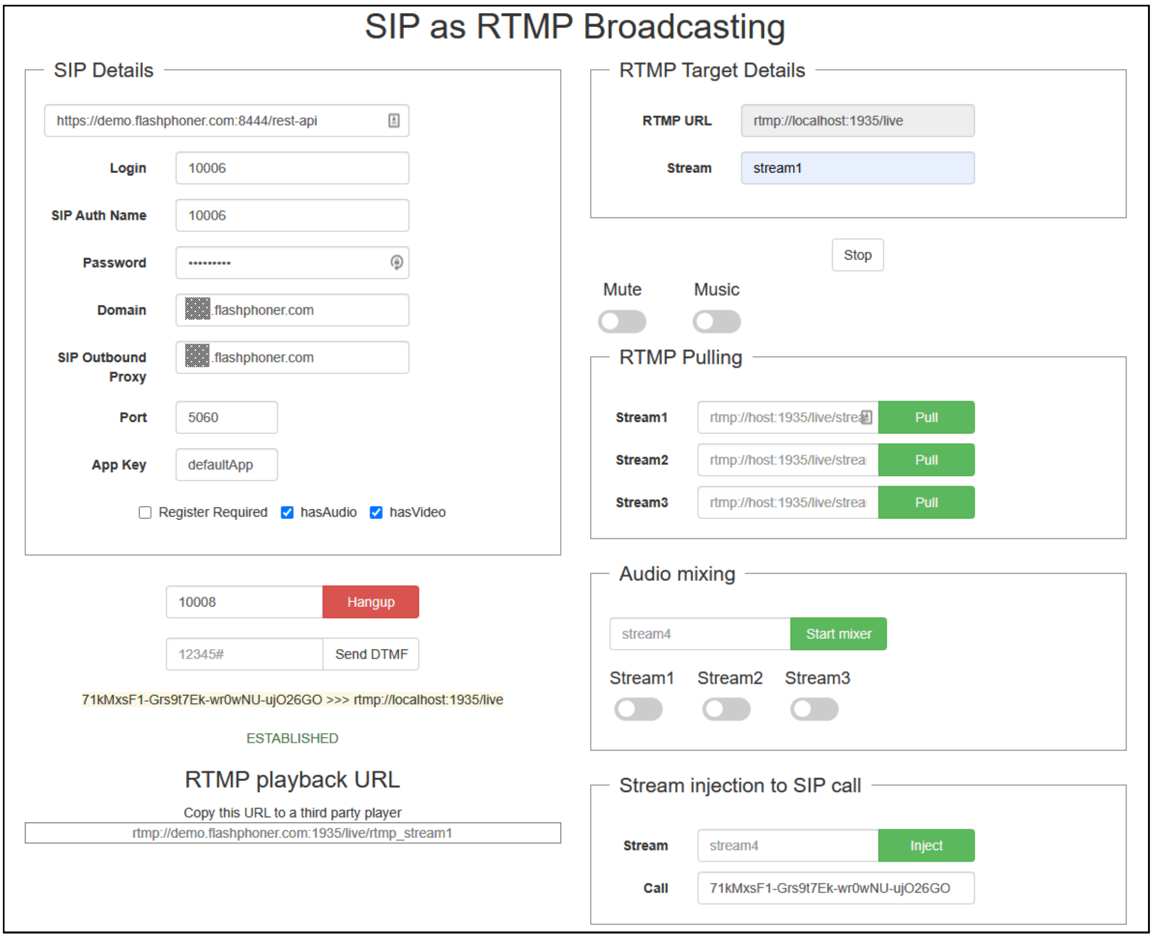
SIP as RTMP 4

Example of stream capturing from SIP call, RTMP stream pulling from other server, streams mixing and re-publishing to a third party RTMP server

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



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-4

- sip-as-rtmp-4.js - a script dealing with REST queries to the WCS server
- sip-as-rtmp-4.html - example page

The example may be tested at this URL:

https://host:8888/client2/examples/demo/sip/sip-as-rtmp-4/sip-as-rtmp-4.html

where host is WCS server address.

Analyzing the code

To analyze the code get `sip-as-rtmp-4.js` file version with hash `ecbadc3` that can be found here and is availabe 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

Connection and call data (`RESTCall`) are collected from the boxes on page

```
var url = field("restUrl") + "/call/startup";
callId = generateCallID();
$("#sipCallId").val(callId);
...
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. Capturing RTMP stream from other server with `/pull/rtmp/pull` REST query

code

```
var pullRtmp = function(uri, fn) {
    console.log("Pull rtmp " + uri);
    send(field("restUrl") + "/pull/rtmp/pull", {
        uri: uri
    }).then(
        fn(STREAM_STATUS.PENDING)
    ).catch(function(e){
        console.error(e);
        fn(STREAM_STATUS.FAILED);
    });
};
```

4. Stop capturing RTMP stream from other server with `/pull/rtmp/terminate` REST query

code

```
var terminateRtmp = function(uri, fn) {
    console.log("Terminate rtmp " + uri);
    send(field("restUrl") + "/pull/rtmp/terminate", {
        uri: uri
    }).then(
        fn(STREAM_STATUS.STOPPED)
```

```
    ).catch(function(e) {
        fn(STREAM_STATUS.FAILED);
        console.error(e);
    })
};
```

5. Mixer starting with `/mixer/startup` REST query

code

```
var startMixer = function(streamName) {
    console.log("Start mixer " + streamName);
    return send(field("restUrl") + "/mixer/startup", {
        uri: "mixer://" + streamName,
        localStreamName: streamName
    });
};
```

6. Mixer stopping with `/mixer/terminate` REST query

code

```
var stopMixer = function(streamName, fn) {
    console.log("Stop mixer " + streamName);
    return send(field("restUrl") + "/mixer/terminate", {
        uri: "mixer://" + streamName,
        localStreamName: streamName
    });
};
```

7. Adding/removing streams to mixer with `/mixer/add` and `/mixer/remove` REST queries

code

```
if ($(ctx).is(':checked')) {
        // Add stream to mixer
        send(field("restUrl") + "/mixer/add", {
            uri: "mixer://" + mixerStream,
            localStreamName: mixerStream,
            remoteStreamName: stream
        }).then(function(){
            console.log("added");
        });
    } else {
        // Remove stream from mixer
        send(field("restUrl") + "/mixer/remove", {
            uri: "mixer://"  + mixerStream,
            localStreamName: mixerStream,
            remoteStreamName: stream
```

```
        }).then(function(){
            console.log("removed");
        });
    }
```

8. Injecting mixer stream to the SIP call with `/call/inject` REST query

code

```
function injectStreamBtn(ctx) {
    var streamName = $("#injectStream").val();
    if (!streamName) {
        $("#injectStream").parent().addClass('has-error');
        return false;
    }
    var $that = $(ctx);
    send(field("restUrl") + "/call/inject_stream", {
        callId: $("#sipCallId").val(),
        streamName: streamName
    }).then(function(){
        $that.removeClass('btn-success').addClass('btn-danger');
        $that.parents().closest('.input-
group').children('input').attr('disabled', true);
    }).catch(function() {
        $that.removeClass('btn-danger').addClass('btn-success');
        $that.parents().closest('.input-
group').children('input').attr('disabled', false);
    });
}
```

9. Re-publishing the SIP call stream to an RTMP server with `/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;
        console.log("Start rtmp");
        sendREST(url, JSON.stringify(RESTObj), startupRtmpSuccessHandler,
startupRtmpErrorHandler);
```

```
        sendDataToPlayer();
        startCheckTransponderStatus();
    }
}
```

10. Mute/unmute RTMP stream re-published 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 `/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);
    }
}
```

11. Injecting additional sound to RTMP stream re-published.

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);
    }
}
```

12. 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);
}
```

13. 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);
}
```