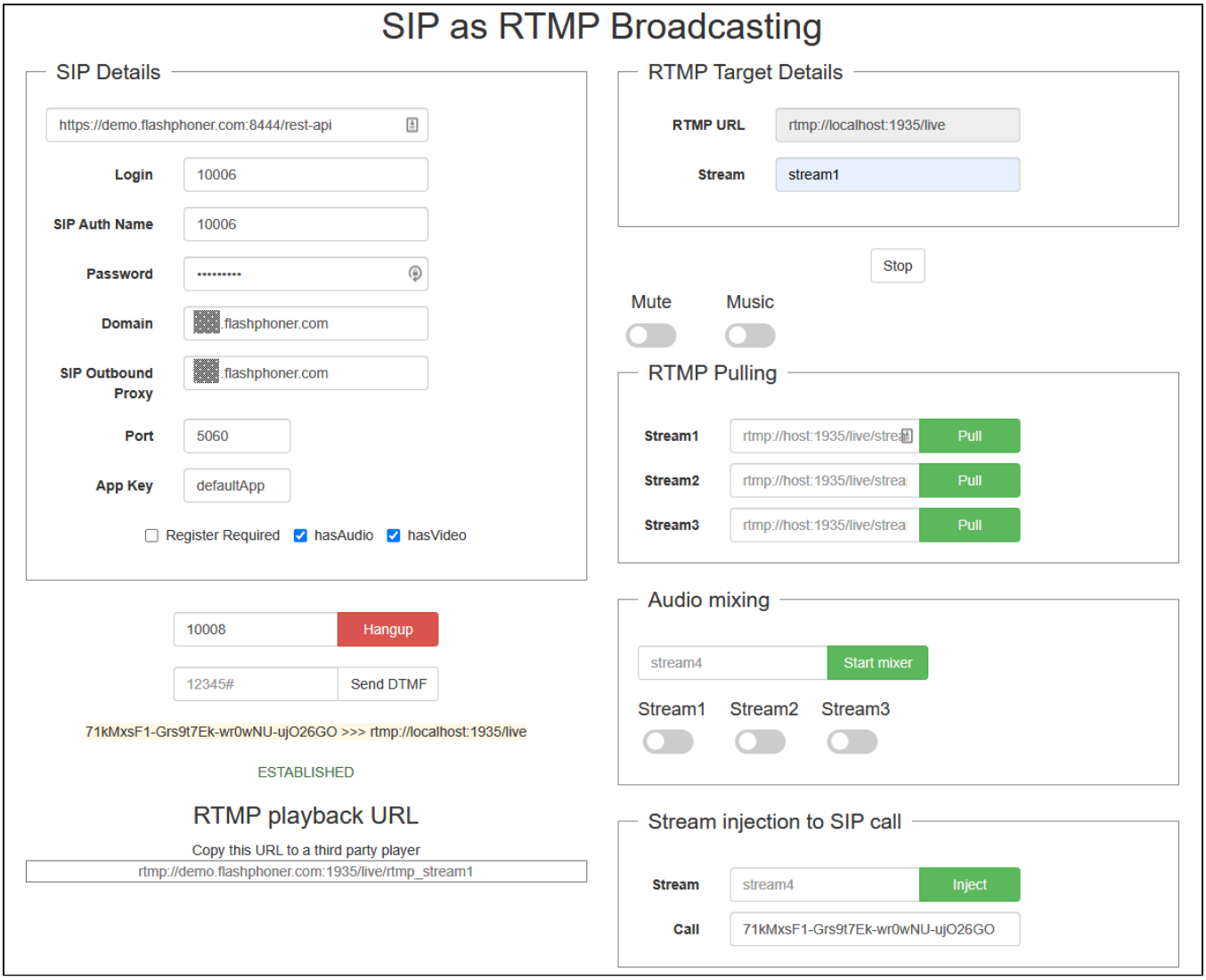
SIP as RTMP 4

- Пример захвата потока с SIP, захвата RTMP-потока с другого сервера, микширования потоков и ретрансляции результата на RTMP-сервер
- Код примера
- Работа с кодом примера

Пример захвата потока с SIP, захвата RTMP-потока с другого сервера, микширования потоков и ретрансляции результата на RTMP-сервер

Данный пример показывает, как можно сделать вызов на SIP, получить от SIP стороны аудио и видео трафик, захватить RTMP-поток с другого сервера, смикшировать аудиопотоки, добавить их к звонку и затем перенаправить полученный поток на RTMP-сервер



Код примера

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

/usr/local/FlashphonerWebCallServer/client2/examples/demo/sip/sip-as-rtmp-4

sip-as-rtmp-4.js - скрипт, обеспечивающий REST вызовы на WCS-сервер
sip-as-rtmp-4.html - страница примера

Тестировать данный пример можно по следующему адресу:

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

Здесь host - адрес WCS-сервера.

Работа с кодом примера

Для разбора кода возьмем версию файла sip-as-rtmp-4.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();
    $("#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. Захват RTMP-потока с другого сервера запросом /pull/rtmp/pull

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. Остановка захвата RTMP-потока с другого сервера запросом /pull/rtmp/terminate

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. Запуск микшера запросом /mixer/startup

code

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

6. Остановка микшера запросом /mixer/terminate

code

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

7. Добавление/удаление потоков в микшер запросами /mixer/add и /mixer/remove

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. Добавление выходного потока микшера в звонок запросом /call/inject

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. Ретрансляция звонка на RTMP-сервер 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;
        console.log("Start rtmp");
        sendREST(url, JSON.stringify(RESTObj), startupRtmpSuccessHandler, startupRtmpErrorHandler);
        sendDataToPlayer();
        startCheckTransponderStatus();
    }
}
```

10. Включение/отключение звука ретранслируемого 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);
    }
}
```

11. Включение/отключение дополнительной звуковой дорожки из файла в ретранслируемом RTMP-потоке.

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);
    }
}
```

12. 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);
}
```

13. 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);
}
```