Звонок между двумя браузерами через SIP сервер

- Описание
  - Поддерживаемые платформы и браузеры
  - Поддерживаемые протоколы
  - Поддерживаемые кодеки
  - Поддерживаемые SIP функции
  - Схема работы
- Краткое руководство по тестированию
- Последовательность выполнения операций (Call Flow)

SIP звонок между браузерами через WCS является частным случаем звонков между браузером и SIP-устройством, при этом веб-приложение в браузере исполняет роль программного телефона с обоих сторон звонка.

Описание

Поддерживаемые платформы и браузеры

	Chrome	Firefox	Safari 11	Edge
Windows	+	+		+
Mac OS	+	+	+	
Android	+	+		
iOS	-	-	+	

Поддерживаемые протоколы

- WebRTC
- RTP
- SIP

Поддерживаемые кодеки

- H.264
- VP8
- G.711
- Speex
- G.729
- Opus

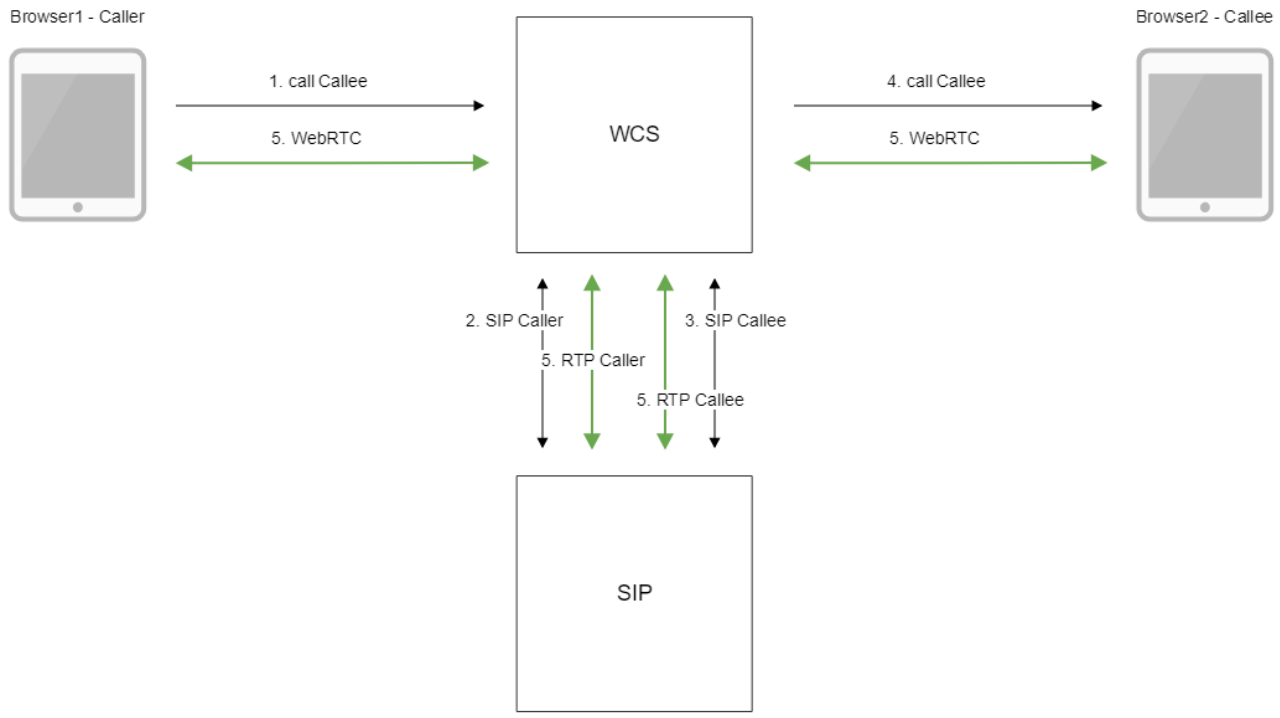
Поддерживаемые SIP функции

- DTMF
- Удержание звонка
- Перевод звонка

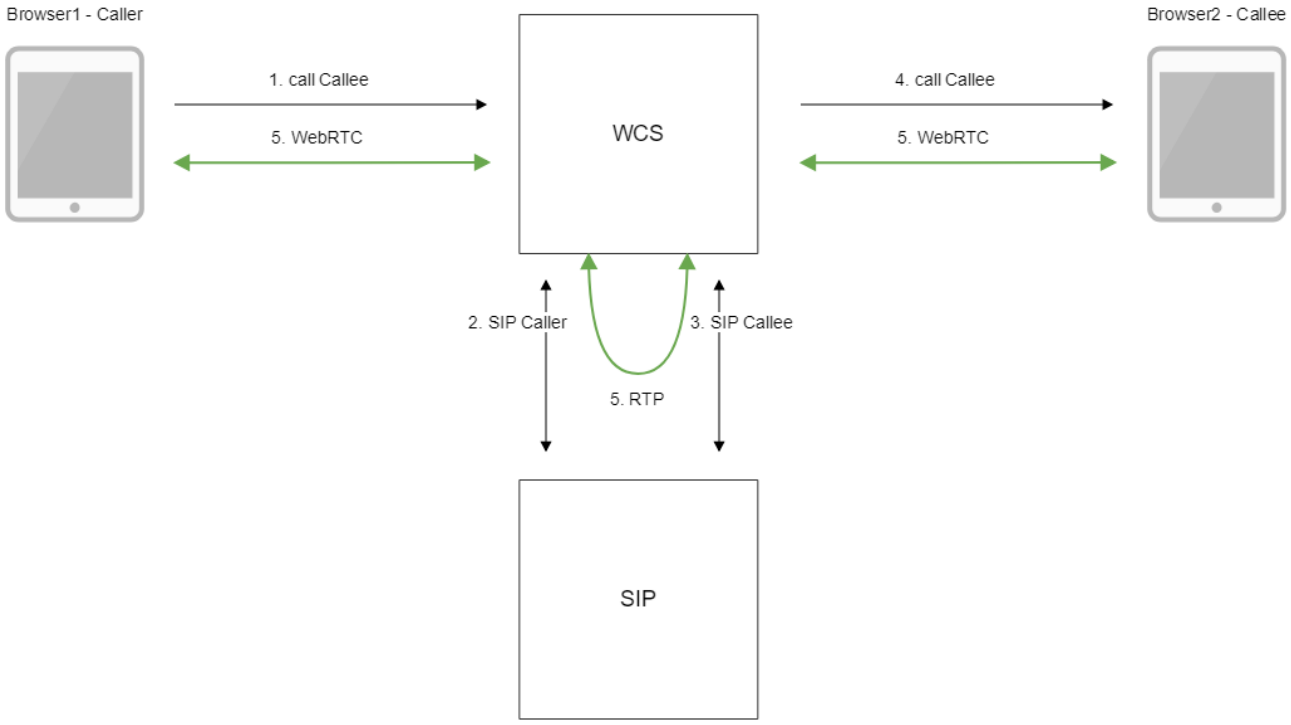
SIP функции управляются при помощиREST API.

Схема работы

1: SIP-сервер как прокси-сервер для передачи вызовов и RTP медиа

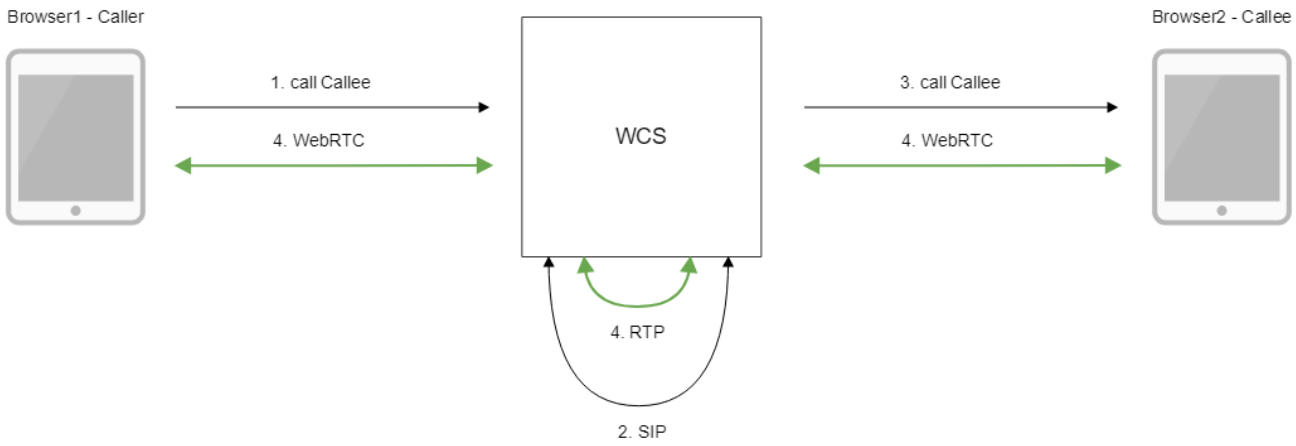


2: SIP-сервер только как сервер для передачи вызовов



1. Браузер 1 начинает звонок с аккаунта Caller на аккаунт Callee
2. WCS соединяется с SIP-сервером
3. SIP-сервер передает WCS входящий вызов на аккаунт Callee
4. WCS передает браузеру 2 событие о поступлении звонка
5. Браузеры обмениваются аудио- и видеопотоками

3: Без внешнего SIP-сервера. SIP и RTP медиа обрабатываются на WCS.



1. Браузер 1 начинает звонок с аккаунта Caller на аккаунт Callee
2. WCS устанавливает SIP-соединение между аккаунтами
3. WCS передает браузеру 2 событие о поступлении звонка
4. Браузеры обмениваются аудио- и видеопотоками

Краткое руководство по тестированию

1. Для тестирования используем:

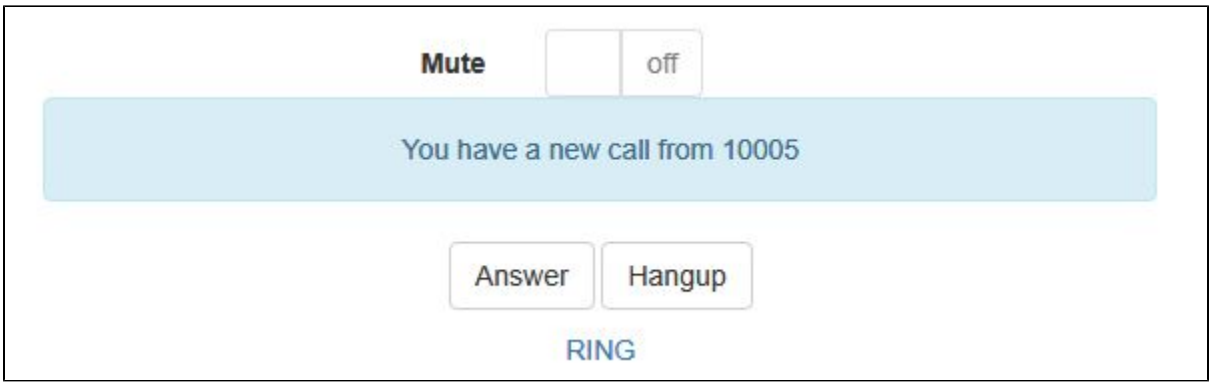
- два SIP-аккаунта;
- веб-приложение Phone для совершения звонка

2. Откройте веб-приложение Phone. Введите данные SIP-аккаунта и нажмите кнопку Connect для установки соединения с сервером:

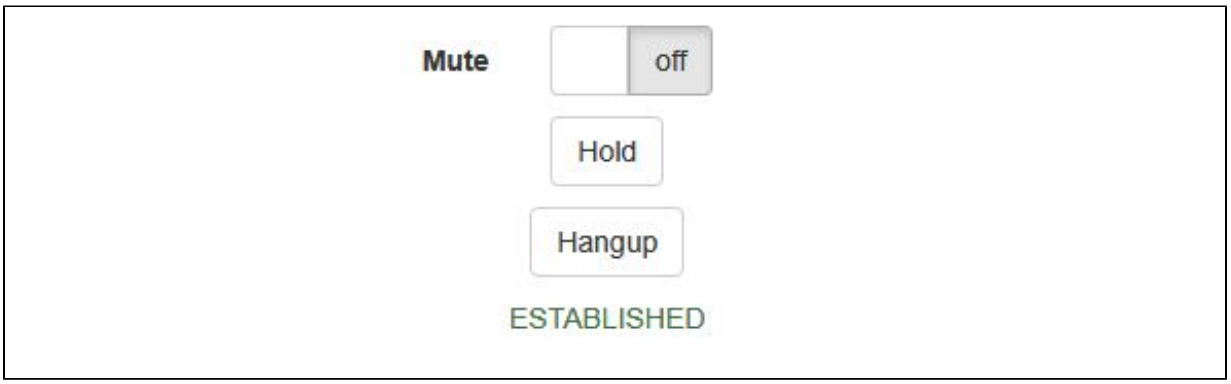
3. Откройте веб-приложение Phone в другом окне браузера. Введите данные второго SIP-аккаунта и нажмите кнопку Connect:

4. Введите идентификатор SIP-аккаунта, принимающего звонок, и нажмите кнопку Call:

5. Примите звонок, нажав кнопку Answer:



Звонок начался:



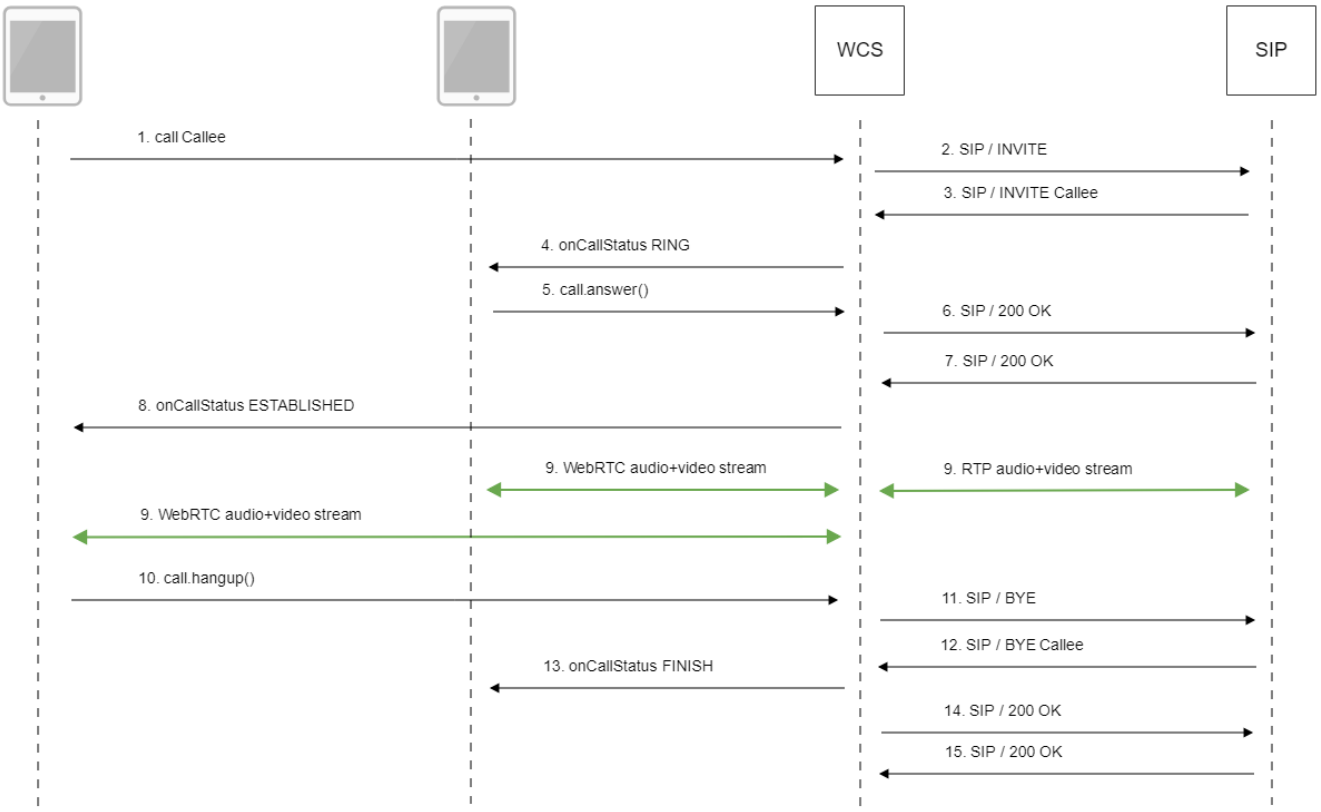
6. Для завершения звонка нажмите кнопку "Hangup".

Последовательность выполнения операций (Call Flow)

Ниже описана последовательность вызовов при использовании примера Phone для создания звонка. SIP-сервер используется как прокси-вервер для передачи команд и медиа

phone.html

phone.js



1. Отправка REST-запроса /call/startup при помощи JavaScript API:

session.createCall(), call.call()code

```
    var outCall = session.createCall({
                callee: $("#callee").val(),
        visibleName: $("#sipLogin").val(),
                localVideoDisplay: localDisplay,
                remoteVideoDisplay: remoteDisplay,
                constraints: constraints,
                receiveAudio: true,
        receiveVideo: false
        ...
    });

        outCall.call();
```

2. Установка соединения с SIP-сервером

3. SIP-сервер устанавливает соединение с WCS

4. Отправка второму браузеру события, оповещающего о входящем звонке

CallStatusEvent RINGcode

```
    Flashphoner.createSession(connectionOptions).on(SESSION_STATUS.ESTABLISHED, function(session, connection){
        ...
    }).on(SESSION_STATUS.INCOMING_CALL, function(call){
        call.on(CALL_STATUS.RING, function(){
                    setStatus("#callStatus", CALL_STATUS.RING);
            ...
        });
```

5. Второй браузер отвечает на звонок

call.answer()code

```
function onIncomingCall(inCall) {
        currentCall = inCall;

        showIncoming(inCall.caller());

    $("#answerBtn").off('click').click(function(){
                $(this).prop('disabled', true);
        var constraints = {
            audio: true,
            video: false
        };
                inCall.answer({
            localVideoDisplay: localDisplay,
            remoteVideoDisplay: remoteDisplay,
            receiveVideo: false,
            constraints: constraints
        });
                showAnswered();
    }).prop('disabled', false);
    ...
}
```

6. Передача подтверждения SIP-серверу

7. Получение подтверждения от SIP-сервера

8. Первый браузер получает от сервера событие, подтверждающего успешное соединение.

CallStatusEvent ESTABLISHEDcode

```
    var outCall = session.createCall({
        ...
    }).on(CALL_STATUS.ESTABLISHED, function(){
                setStatus("#callStatus", CALL_STATUS.ESTABLISHED);
        $("#holdBtn").prop('disabled',false);
        onAnswerOutgoing();
        ...
    });

        outCall.call();
```

9. Стороны звонка обмениваются аудио- и видеопотоками

10. Завершение звонка

call.hangup()code

```
function onConnected(session) {
    $("#connectBtn, #connectTokenBtn").text("Disconnect").off('click').click(function(){
        $(this).prop('disabled', true);
                if (currentCall) {
                        showOutgoing();
                        disableOutgoing(true);
                        setStatus("#callStatus", "");
                        currentCall.hangup();
                }
        session.disconnect();
    }).prop('disabled', false);
}
```

11. Отправка команды на SIP-сервер

12. Получение команды от SIP-сервера

13. Отправка второму браузеру события, оповещающего о завершении звонка

CallStatusEvent FINISHcode

```
    Flashphoner.createSession(connectionOptions).on(SESSION_STATUS.ESTABLISHED, function(session, connection){
        ...
    }).on(SESSION_STATUS.INCOMING_CALL, function(call){
        call.on(CALL_STATUS.RING, function(){
                    ...
        }).on(CALL_STATUS.FINISH, function(){
                    setStatus("#callStatus", CALL_STATUS.FINISH);
                        onHangupIncoming();
                    currentCall = null;
            ...
        });
```

14. Отправка подтверждения на SIP-сервер

15. Получение подтверждения от SIP-сервера