Звонок на мобильный телефон через SIP сервер

SIP звонок на мобильный телефон является частным случаем звонков между браузером и SIP-устройством, при этом SIP-сервер либо сам предоставляет услуги GSM/PSTN шлюза, либо соединяется с таковым в процессе звонка.

Описание

Поддерживаемые платформы и браузеры

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

Поддерживаемые протоколы

- WebRTC
- RTP
- SIP

Поддерживаемые кодеки

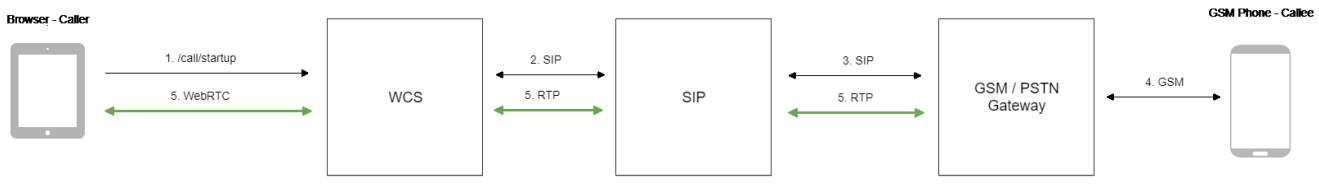
- H.264
- VP8
- G.711
- Speex
- G.729
- Opus

Поддерживаемые SIP функции

- DTMF
- Удержание звонка
- Перевод звонка

SIP функции управляются при помощиREST API.

Схема работы



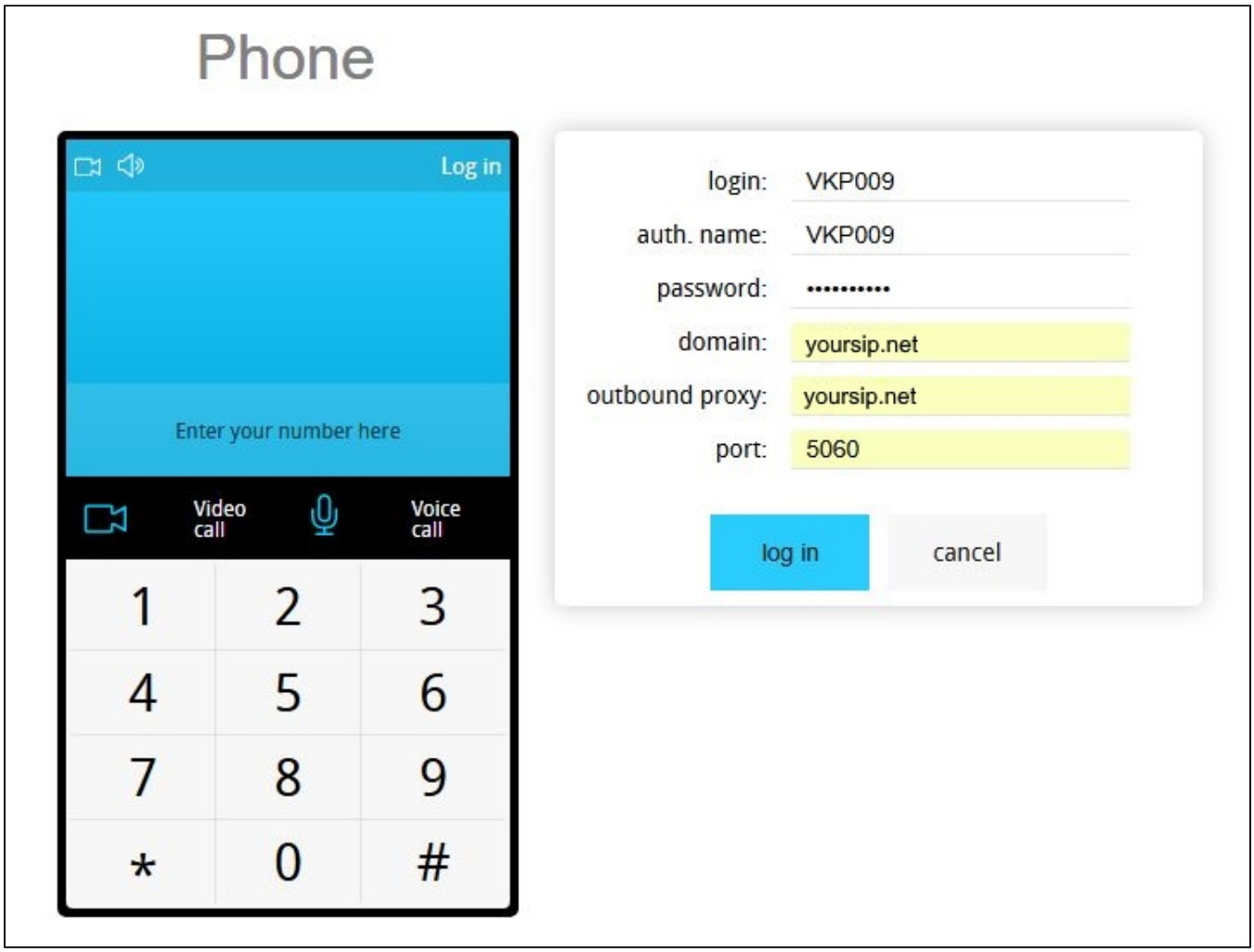
1. Браузер начинает звонок с помощью REST-вызова /call/startup
2. WCS соединяется с SIP-сервером
3. SIP-сервер соединяется с GSM/PSTN шлюзом
4. GSM/PSTN шлюз соединяется с телефоном
5. Браузер и телефон обмениваются аудиопотоками

Краткое руководство по тестированию

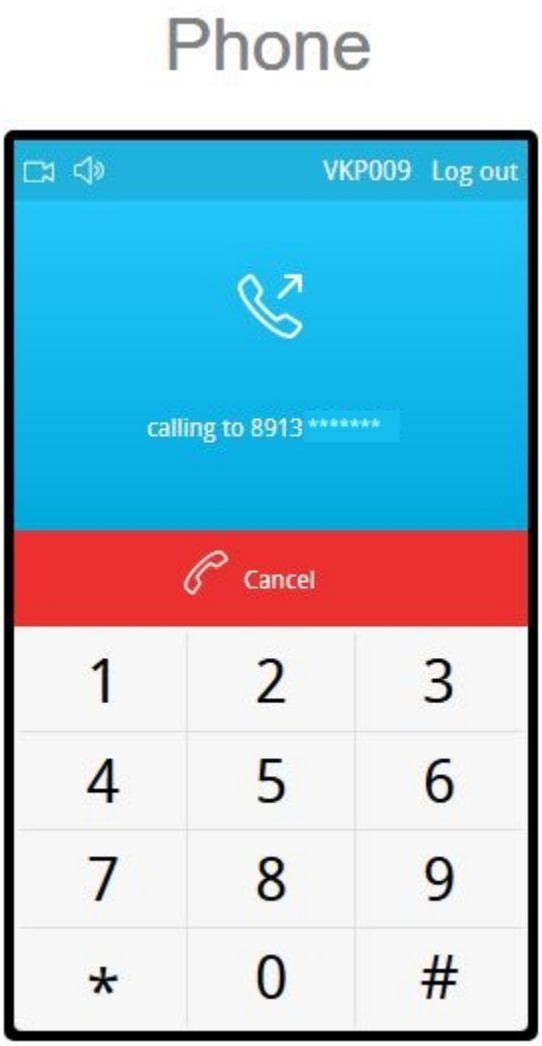
1. Для тестирования используем:

- два SIP-аккаунта;
- веб-приложениеPhone UIдля совершения звонка;
- мобильный телефон для ответа на звонок.

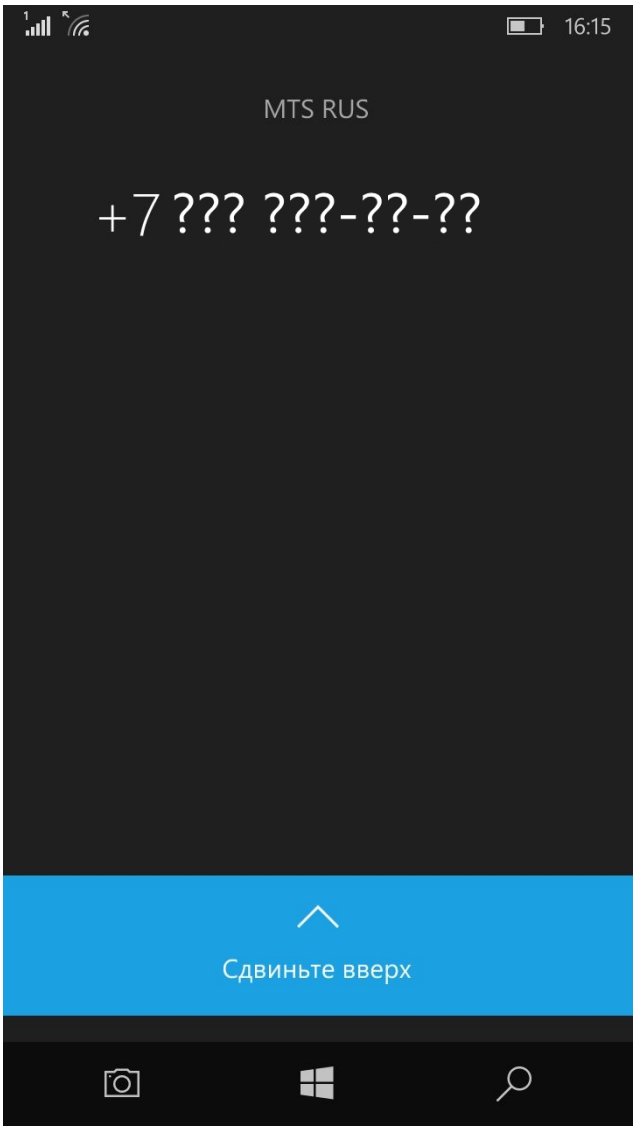
2. Откройте веб-приложение Phone UI. Нажмите Log in и введите данные SIP-аккаунта:



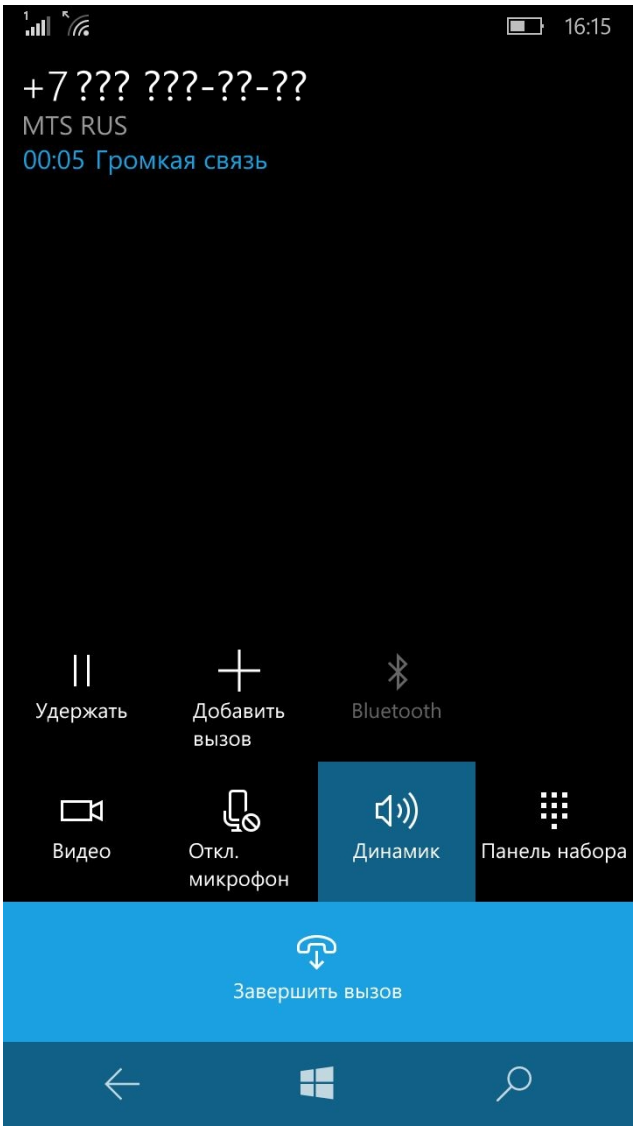
3. Введите номер мобильного телефона и нажмите Voice call. Начнется дозвон:



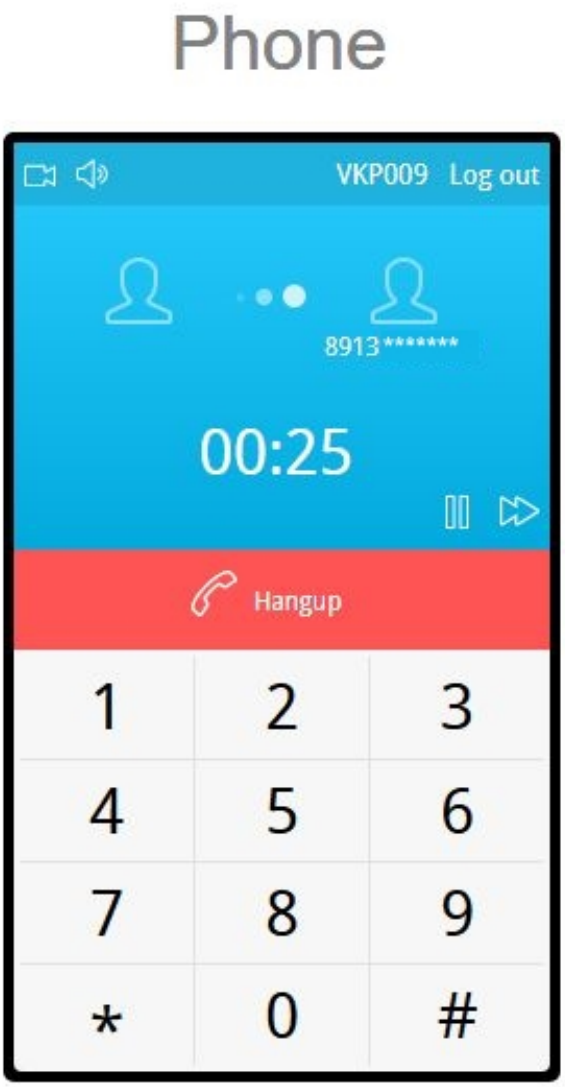
4. На экране мобильного телефона отображается входящий вызов:



5. Примите звонок на мобильном телефоне:



6. В браузере также отображается установленное соединение



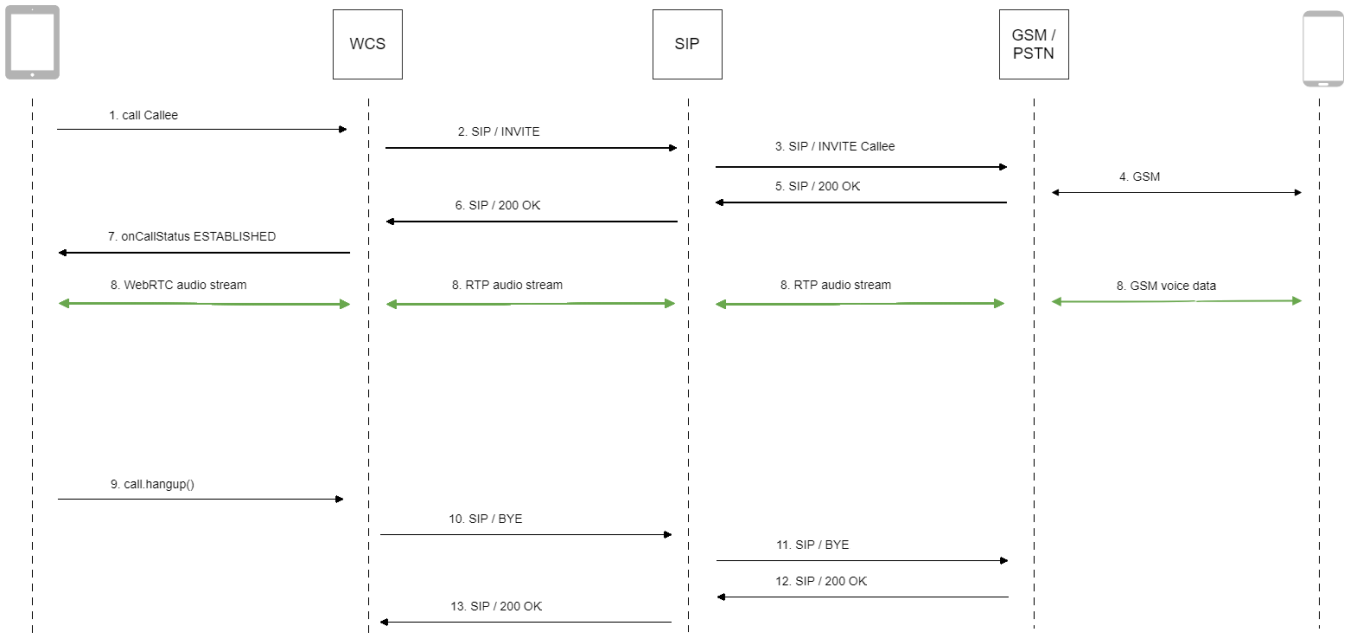
7. Для завершения звонка нажмите кнопку "Hangup".

Последовательность выполнения операций (Call Flow)

Ниже описана последовательность вызовов при использовании примера Phone для создания звонка

Phone.html

Phone.js



1. Создание звонка:

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

```
    var outCall = this.session.createCall({
        callee: callee,
        visibleName: this.sipOptions.login,
        localVideoDisplay: this.localVideo,
        remoteVideoDisplay: this.remoteVideo,
        constraints: constraints
        ...
    });

    outCall.call();
```

2. Установка соединения с SIP-сервером

3. Установка соединения с GSM/PSTN шлюзом

4. Установка соединения с мобильным терминалом

5. Получение подтверждения от GSM/PSTN шлюза

6. Получение подтверждения от SIP-сервера

7. Получение от сервера события, подтверждающего успешное соединение.

CallStatusEvent ESTABLISHEDcode

```
    var outCall = this.session.createCall({
        callee: callee,
        visibleName: this.sipOptions.login,
        localVideoDisplay: this.localVideo,
        remoteVideoDisplay: this.remoteVideo,
        constraints: constraints
        ...
    }).on(CALL_STATUS.ESTABLISHED, function(call){
        me.callStatusListener(call);
        ...
    });

    outCall.call();
```

8. Стороны звонка обмениваются аудиопотоком

9. Завершение звонка

call.hangup()code

```
Phone.prototype.hangup = function () {
    trace("Phone - hangup " + this.currentCall.id() + " status " + this.currentCall.status());
    this.hideFlashAccess();
    if (this.currentCall.status() == CALL_STATUS.PENDING) {
        this.callStatusListener(this.currentCall);
    } else {
        this.currentCall.hangup();
    }
    this.flashphonerListener.onHangup();
};
```

10. Отправка команды на SIP-сервер

11. Отправка команды на GSM/PSTN шлюз

12. Получение подтверждения от GSM/PSTN шлюза

13. Получение подтверждения от SIP-сервера