A call to a mobile phone via the SIP server

Overview

A SIP call to a mobile phone is a special case of calls between a browser and a SIP device, when the SIP server either operates as a GSM/PSTN gateway itself or connects to one during the call.

Supported platforms and browsers

	Chrome	Firefox	Safari	Edge
Windows	✅	✅	❌	✅
Mac OS	✅	✅	✅	✅
Android	✅	✅	❌	✅
iOS	✅	✅	✅	✅

Supported protocols

- WebRTC
- RTP
- SIP

Supported codecs

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

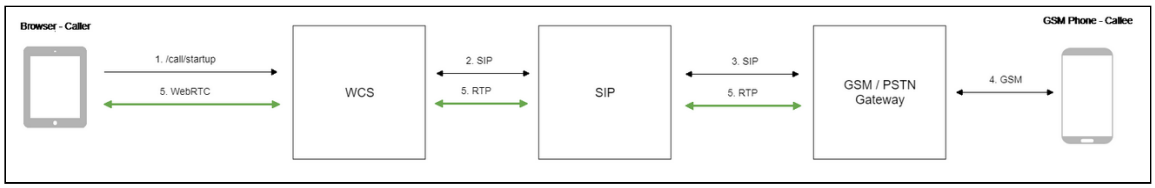
Supported SIP functions

- DTMF

- Holding a call
- Transferring a call

SIP functions are managed using the WebSDK.

Operation flowchart

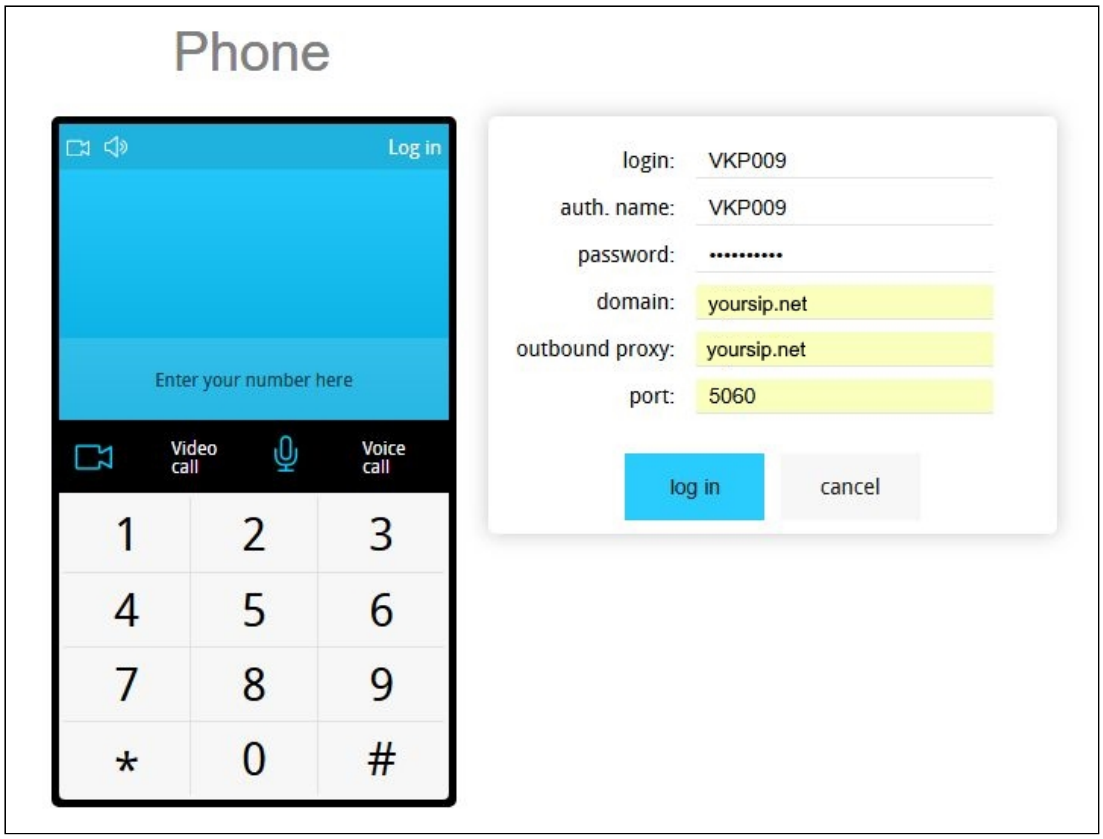


1. The browser begins a call
2. WCS connects to the SIP server
3. The SIP server connects to the GSM/PSTN gateway
4. The GSM/PSTN gateway connects to the mobile phone
5. The browser and the phone exchange audio streams

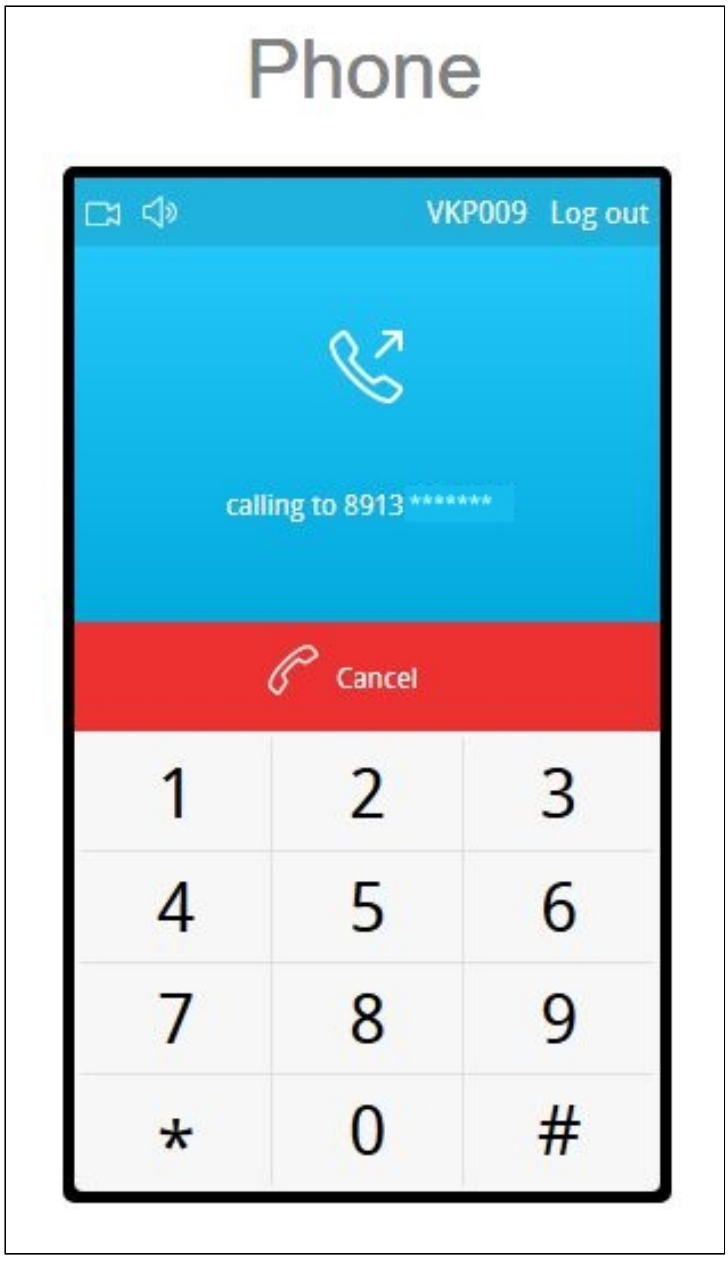
Quick manual on testing

1. For the test we use:
2. a SIP account;
3. the Phone UI web application to make a call;
4. a mobile phone to answer the call.

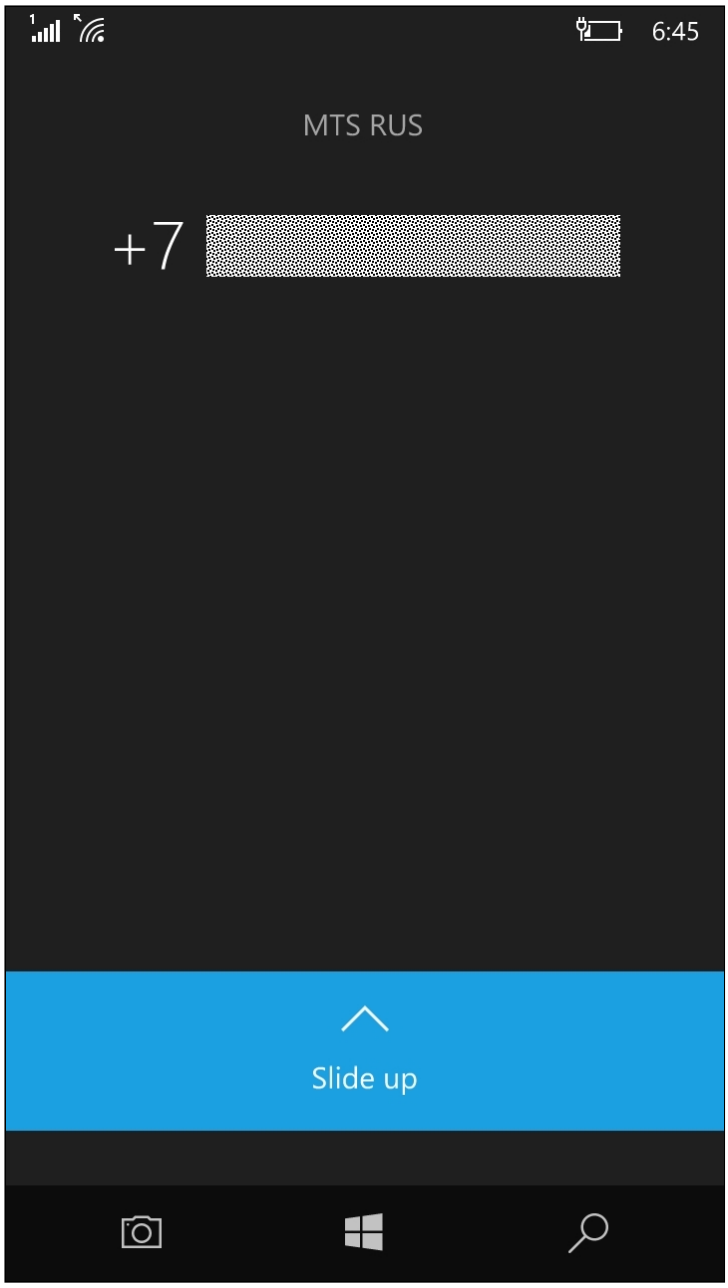
5. Open the Phone UI web application. Click `Log in` and enter the SIP account credentials:



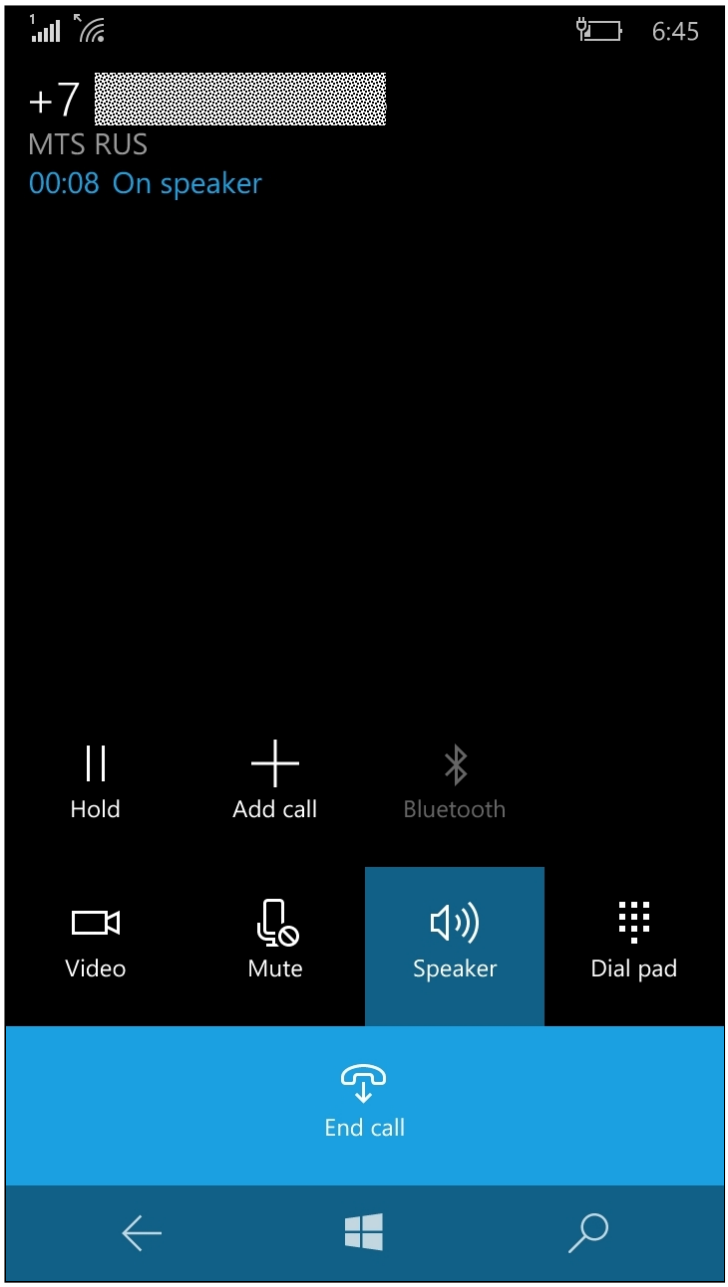
6. Enter the mobile phone number and click `Voice call`. Dialing starts:



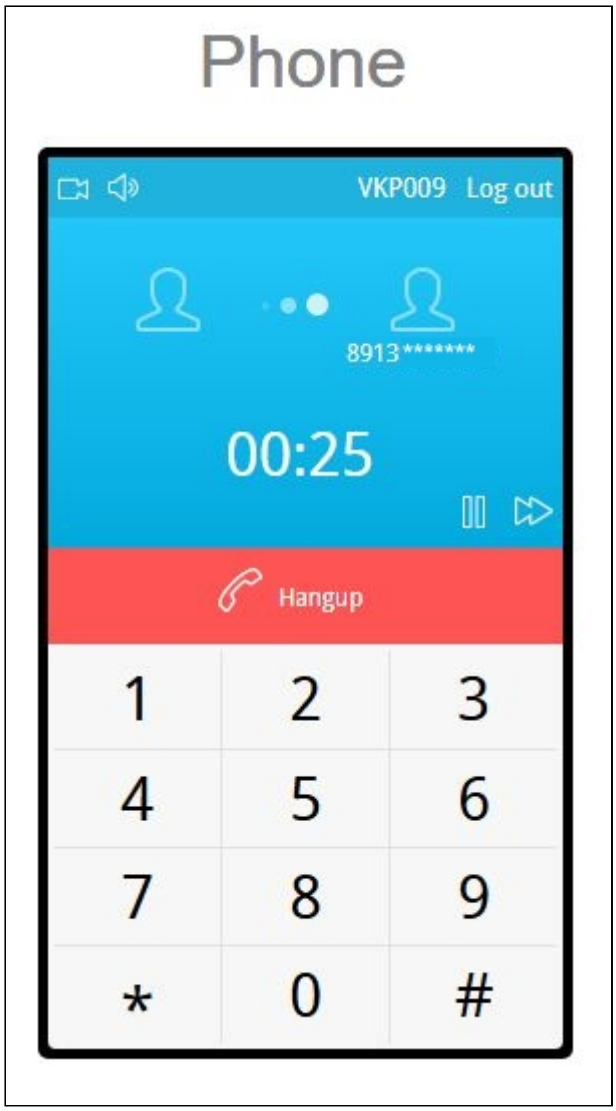
7. The mobile phone displays an incoming call on the screen:



8. Answer the call on the mobile phone:



9. The browser also shows that the connection is established:



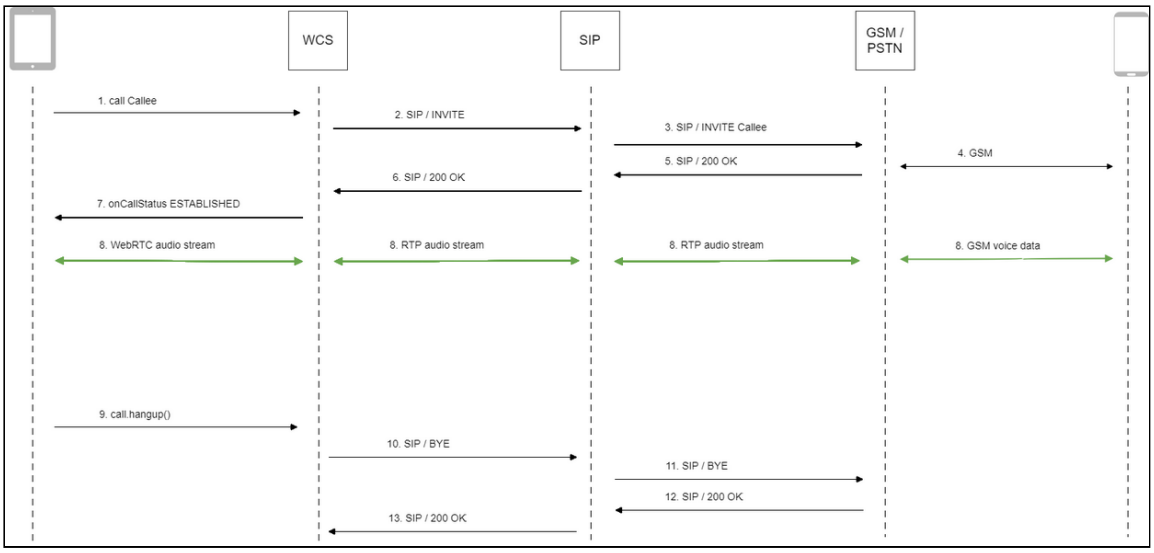
10. To terminate the call, click the `Hangup` button.

Call flow

Below is the call flow when using the Phone example to create a call.

Phone.html

Phone.js



1. Creating a call
   `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. Sending `SIP INVITE` to the SIP server
3. Sending `SIP INVITE` to the GSM/PSTN gateway
4. Establishing a connection to the mobile terminal
5. Receiving a confirmation from the GSM/PSTN gateway
6. Receiving a confirmation from the SIP server
7. Receiving from the server an event confirming successful connection
   `CALL_STATUS.ESTABLISHED` code

```
var outCall = this.session.createCall({
    ...
}).on(CALL_STATUS.ESTABLISHED, function(call){
    me.callStatusListener(call);
    ...
});
```

8. Participants of the call exchange audio streams

9. Terminating the call
   `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. Sending `SIP BYE` to the SIP server
11. Sending `SIP BYE` to the GSM/PSTN gateway
12. Receiving a confirmation from the GSM/PSTN gateway
13. Receiving a confirmation from the SIP server