SIP calls in a WebRTC-compatible browser

Web Call Server supports audio and video calls from a browser to SIP devices, PBX servers, SIP-GSM gates, VoIP conferences and other devices supporting the SIP protocol. Therefore, a web application can work in a browser as a software phone with the support for the SIP protocol, receive and initiate voice and video calls.

Overview

Supported platforms and browsers

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

Supported protocols

- WebRTC
- RTP
- SIP

Supported codecs

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

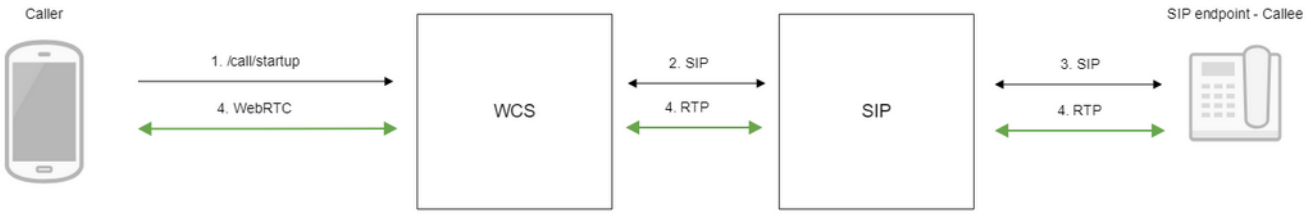
Supported SIP functions

- DTMF
- Holding a call
- Transferring a call

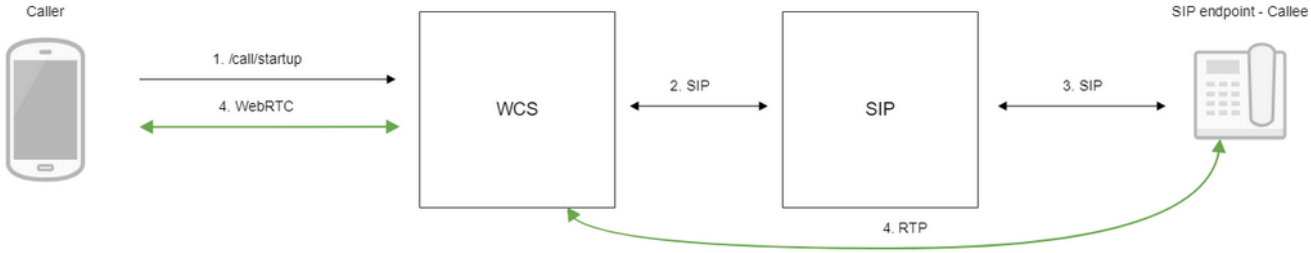
Management of SIP functions is performed with theREST API.

Operation flowchart

1: SIP server as a proxy server to transfer calls and RTP media



2: SIP server as a server to transfer calls only



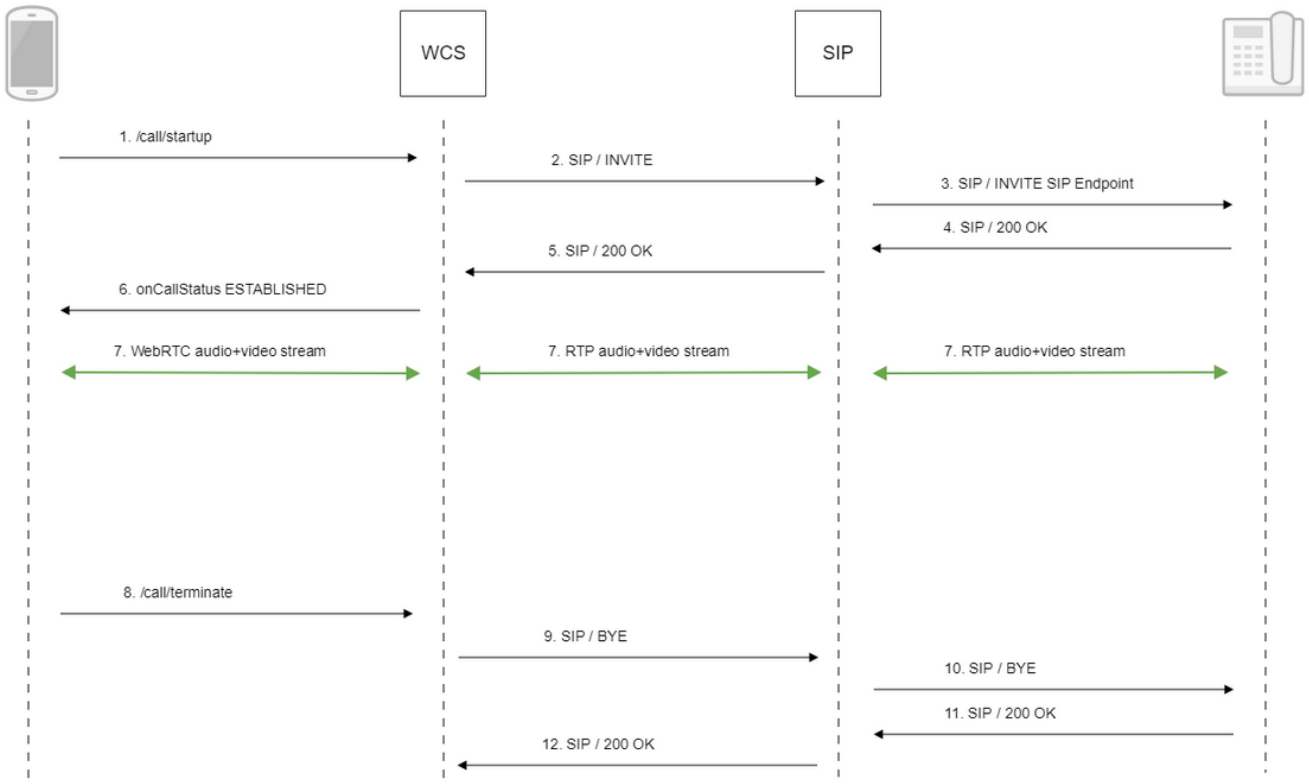
1. The browsers initiates a call using the /call/startup REST query.
2. WCS connects to the SIP server.
3. The SIP server connects to the SIP device receiving the call.
4. The browser and the SIP device exchange audio and video streams.

Call flow

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

phone.html

phone.js



1. Sending the /call/startup REST query using 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. Establishing a connection to the SIP server

3. Establishing a connection to the callee

4. Receiving a confirmation from the SIP device

5. Receiving a confirmation from the SIP server

6. Receiving from the server an event confirming successful connection.

CallStatusEvent ESTABLISHEDcode

```
    var outCall = session.createCall({
                callee: $("#callee").val(),
        visibleName: $("#sipLogin").val(),
                localVideoDisplay: localDisplay,
                remoteVideoDisplay: remoteDisplay,
                constraints: constraints,
                receiveAudio: true,
        receiveVideo: false
        }).on(CALL_STATUS.RING, function(){
        ...
    }).on(CALL_STATUS.ESTABLISHED, function(){
                setStatus("#callStatus", CALL_STATUS.ESTABLISHED);
        $("#holdBtn").prop('disabled',false);
        onAnswerOutgoing();
    }).on(CALL_STATUS.HOLD, function() {
        ...
    }).on(CALL_STATUS.FINISH, function(){
        ...
    }).on(CALL_STATUS.FAILED, function(){
        ...
    });

        outCall.call();
```

7. The caller and the callee exchange audio and video streams

8. Terminating the call

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);
}
```

9. Sending the command to the SIP server

10. Sending the command to the SIP device

11. Receiving a confirmation from the SIP device

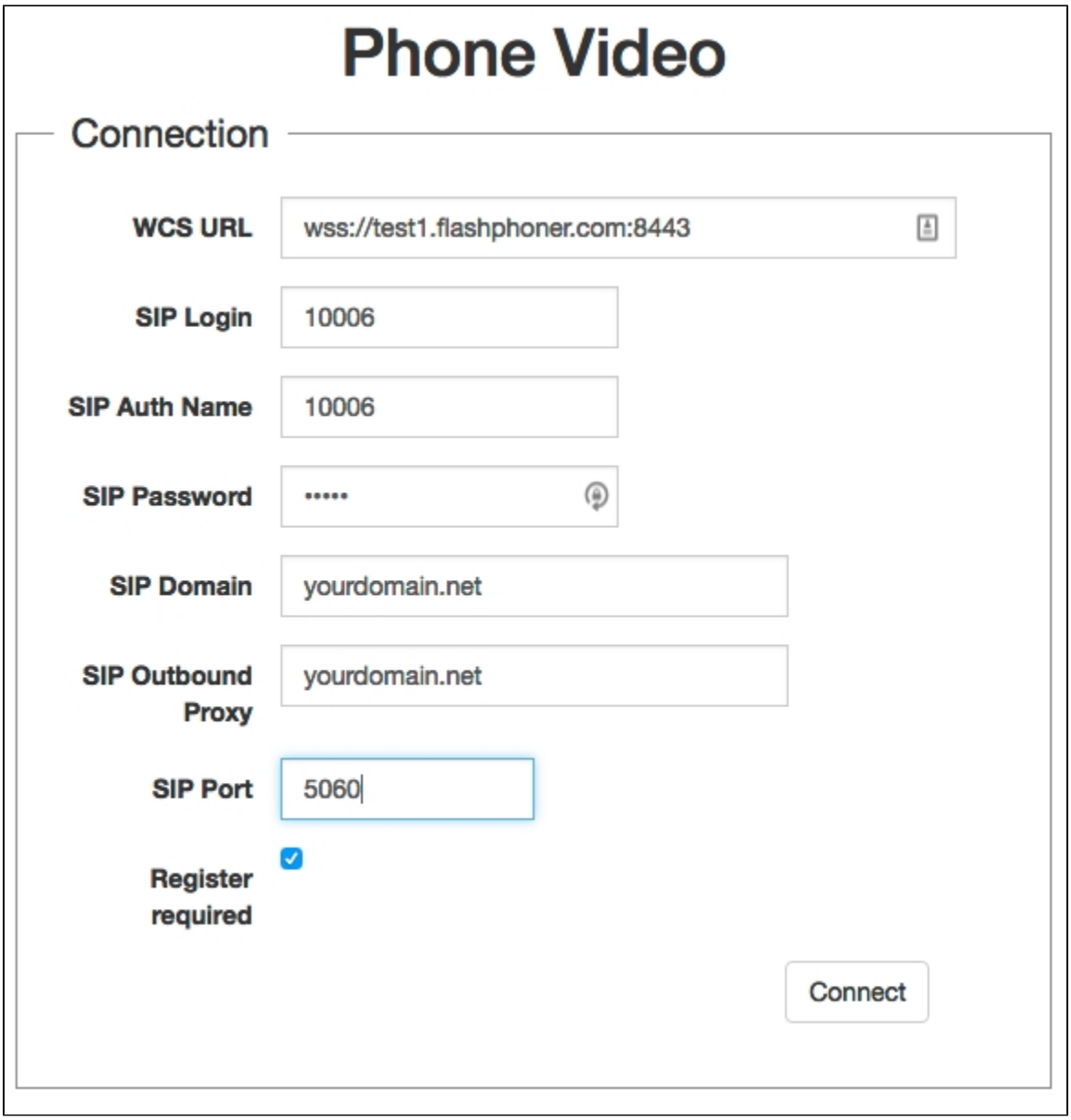
12. Receiving a confirmation from the SIP server

Making an outgoing call from a browser to a SIP device

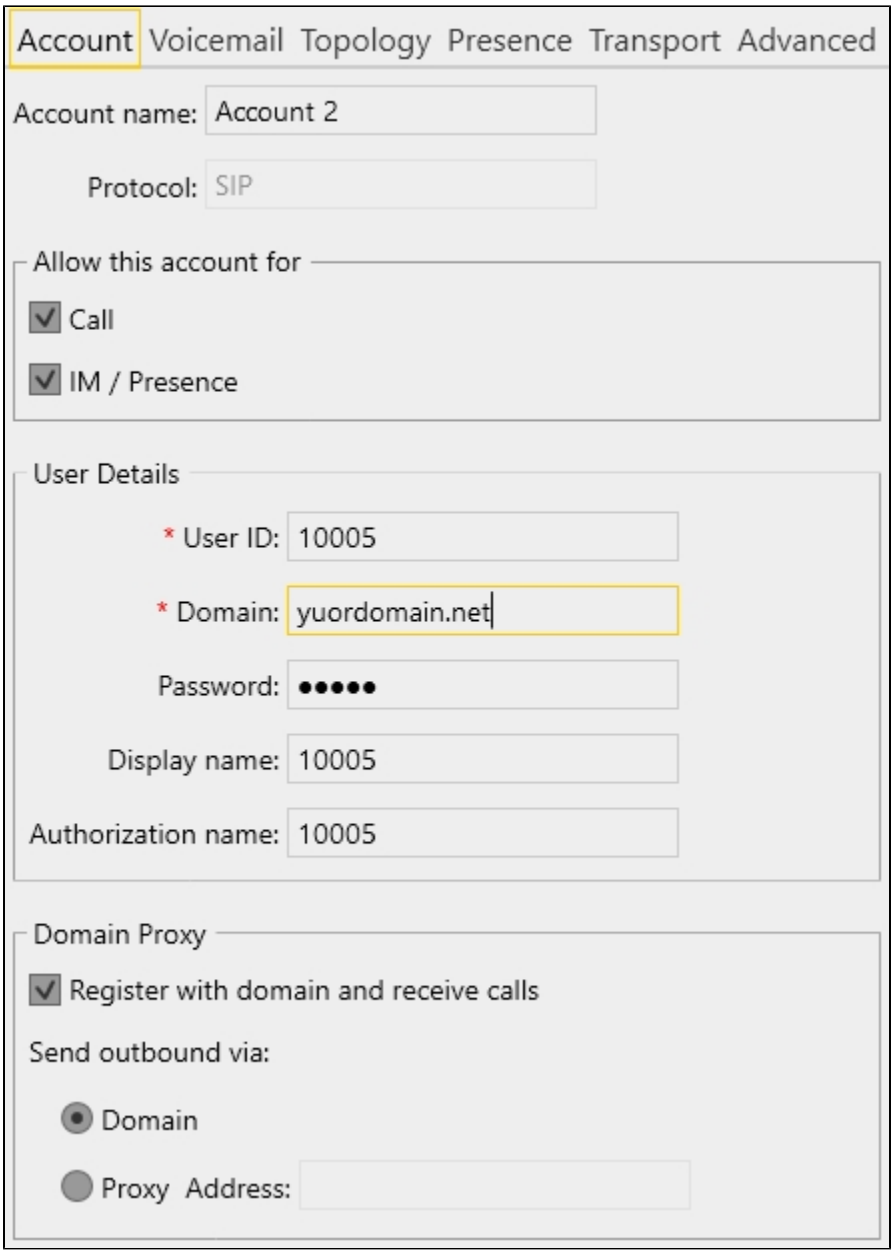
1. For the test we use:

- two SIP accounts;
- thePhone Videoweb application to make a call;
- a software phone to answer the call.

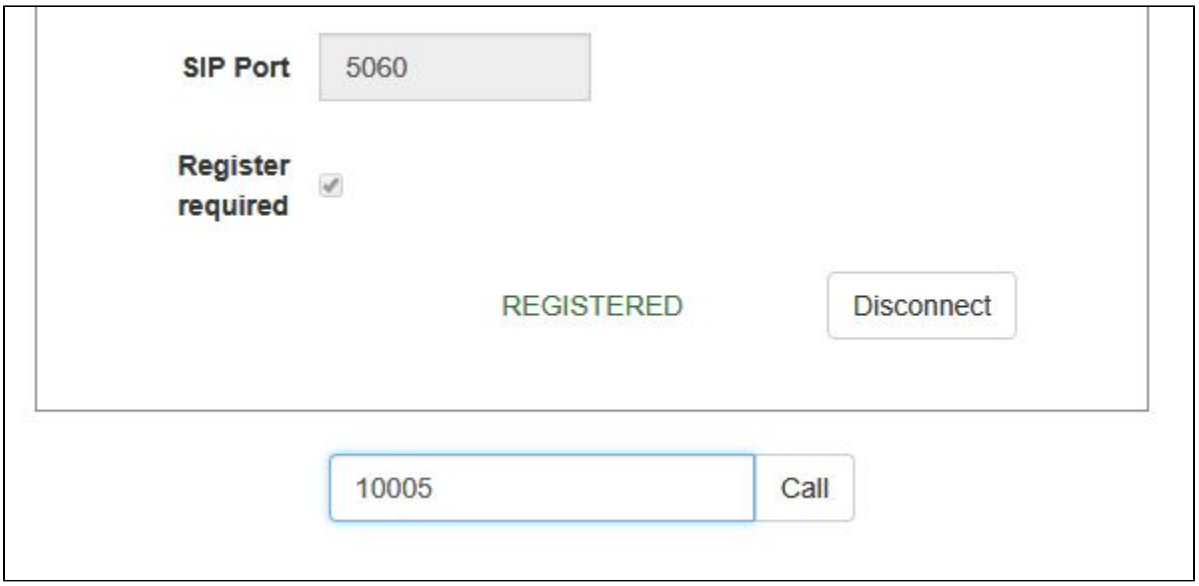
2. Open the Phone Video web application. Enter the data of the SIP account making the call from a browser:



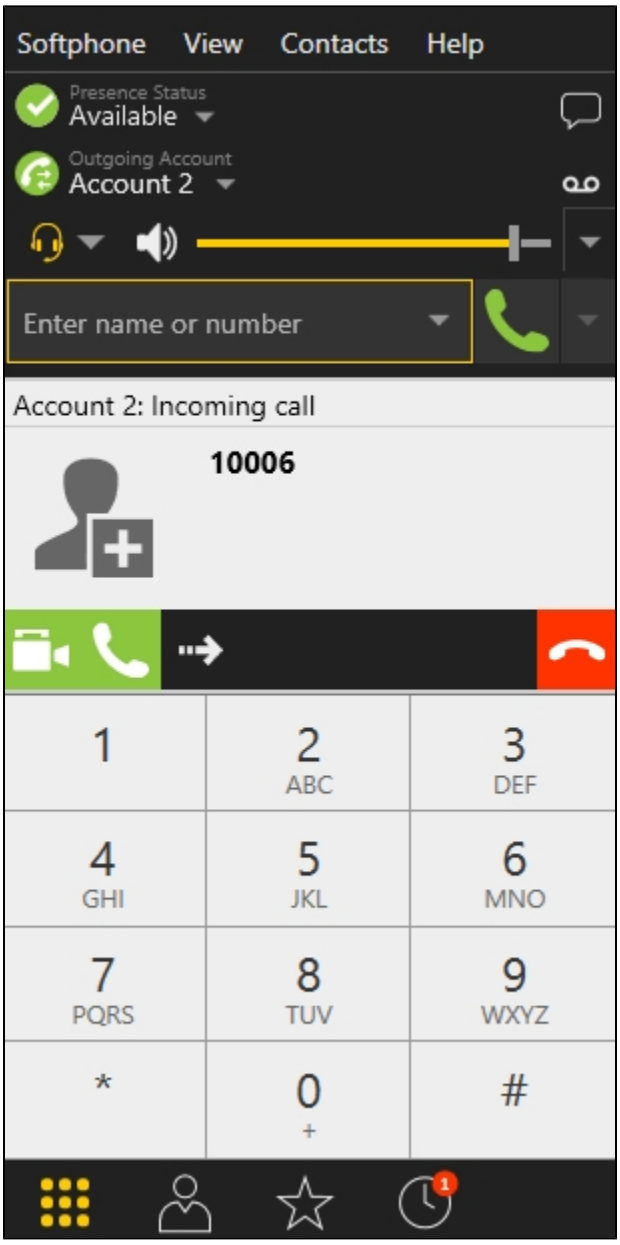
3. Run the software phone, enter the data of the SIP account receiving the call:

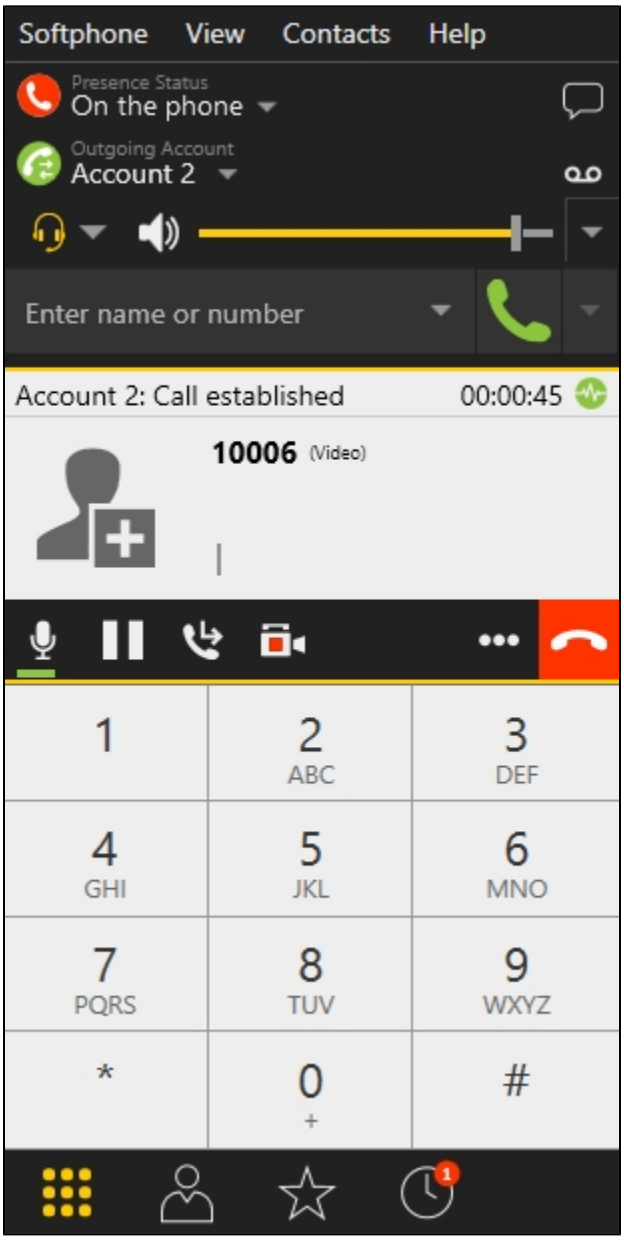


4. Click the Connect button in the browser. Then enter the identifier of the SIP account that receives the call and click the Call button:

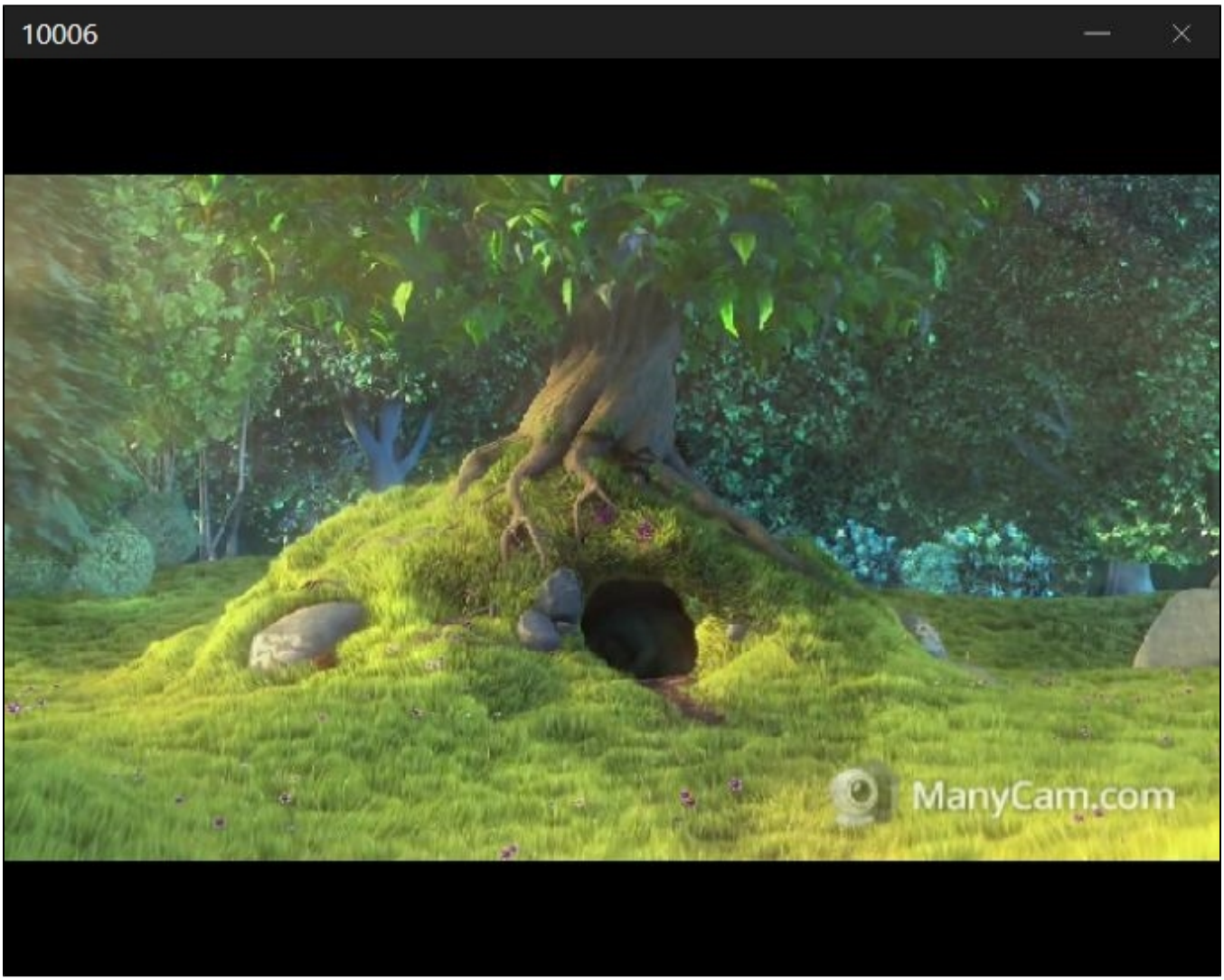


5. Answer the call in the softphone by clicking the answer a video call button:


Softphone
View
Contacts
Help
Presence Status
Available
Outgoing Account
Account 2
Enter name or number
Account 2: Incoming call
10006
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
*
0 +
#



In a separate video, the video broadcast from the browser is shown:



6. The browser also displays the video:

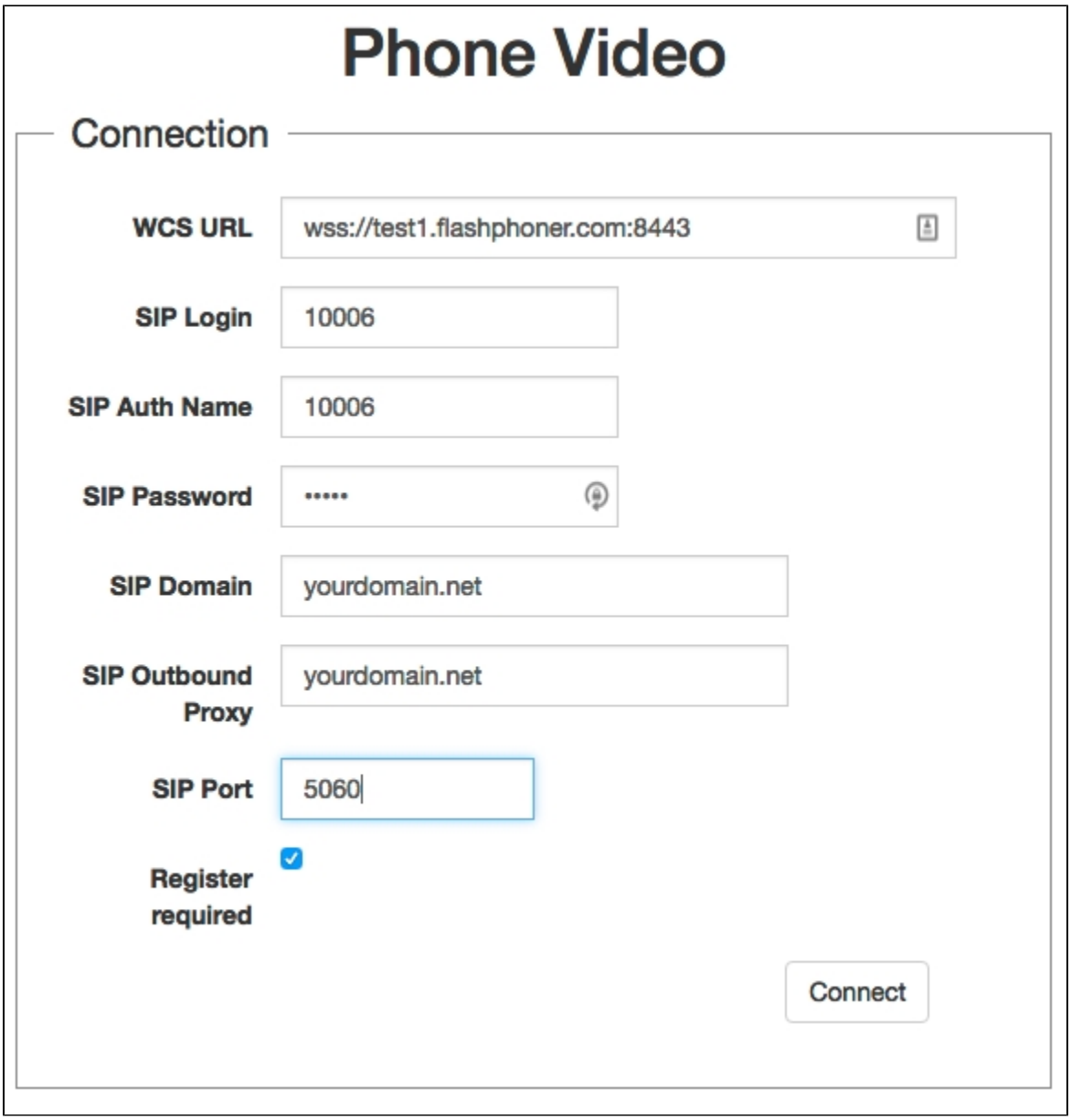
7. To terminate the call, click the Hangup button in the browser or in the softphone.

Receiving an incoming call from a SIP device in a browser

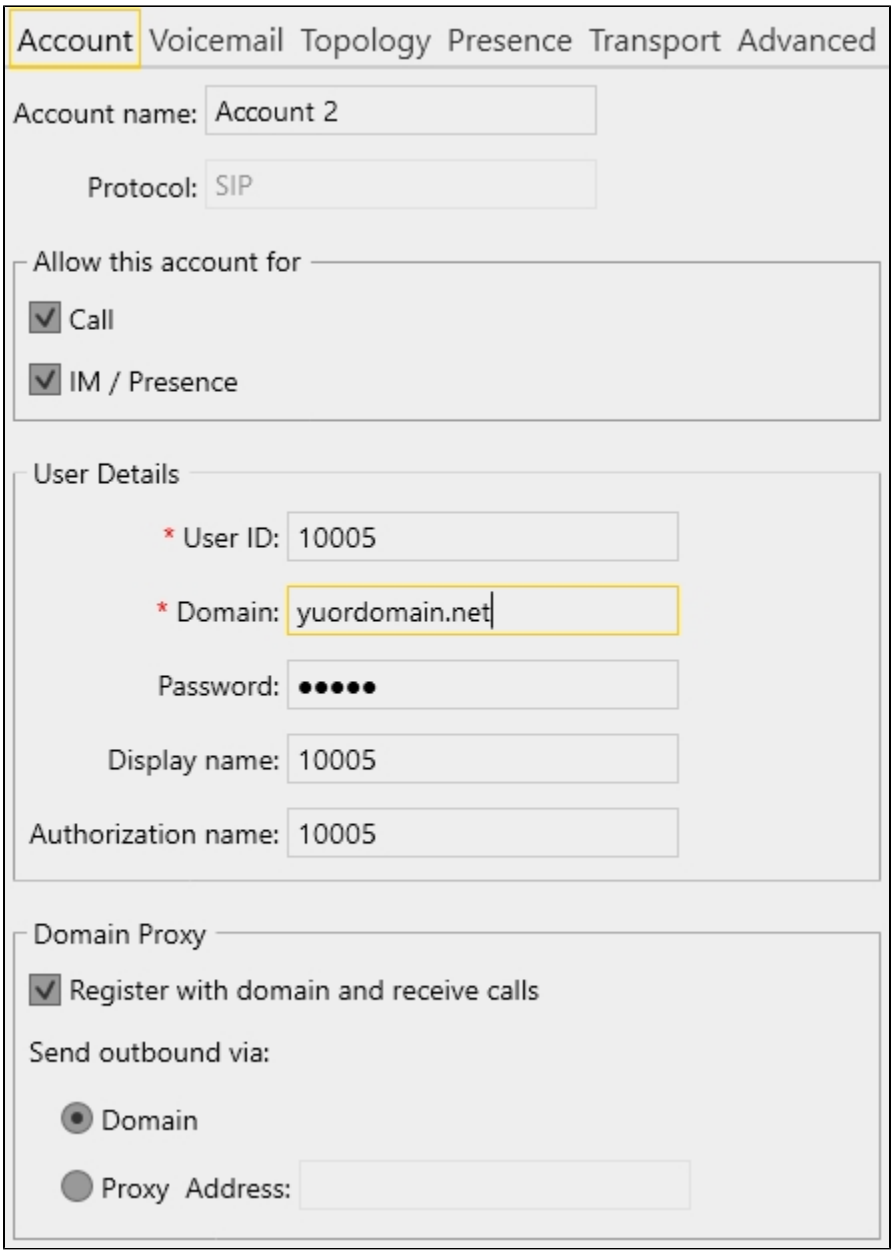
1. For the test we use:

- two SIP accounts;
- a software phone to make the call.
- thePhone Videoweb application to answer the call.

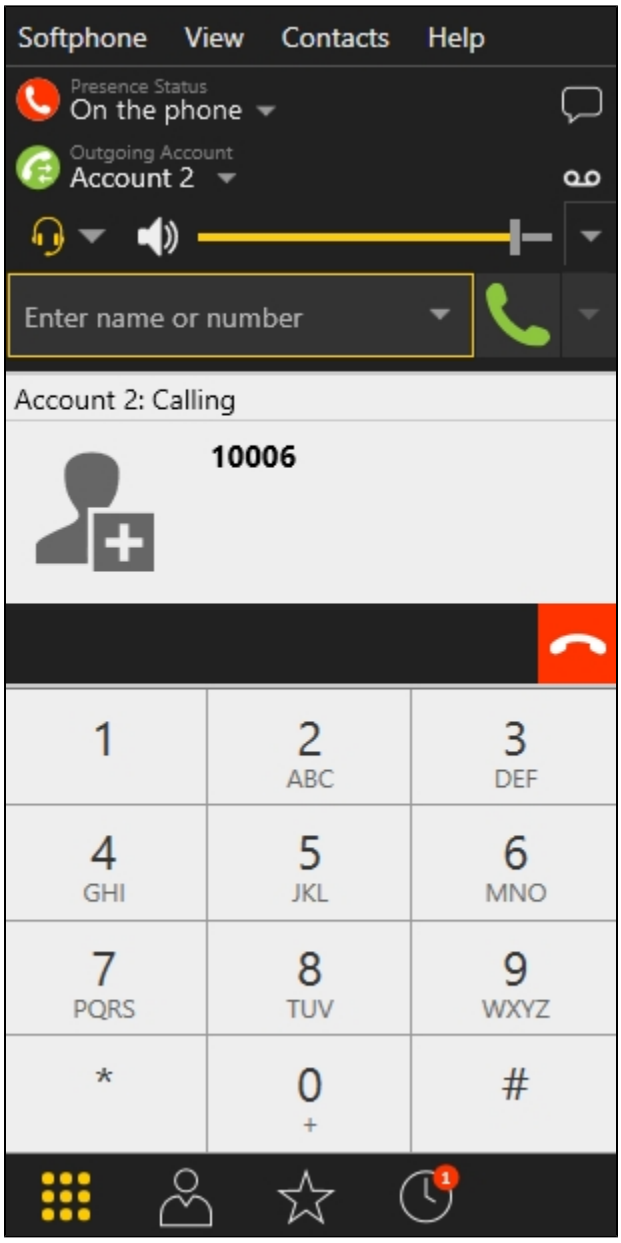
2. Open the Phone Video web application. Enter the data of the SIP account receiving the call in a browser:



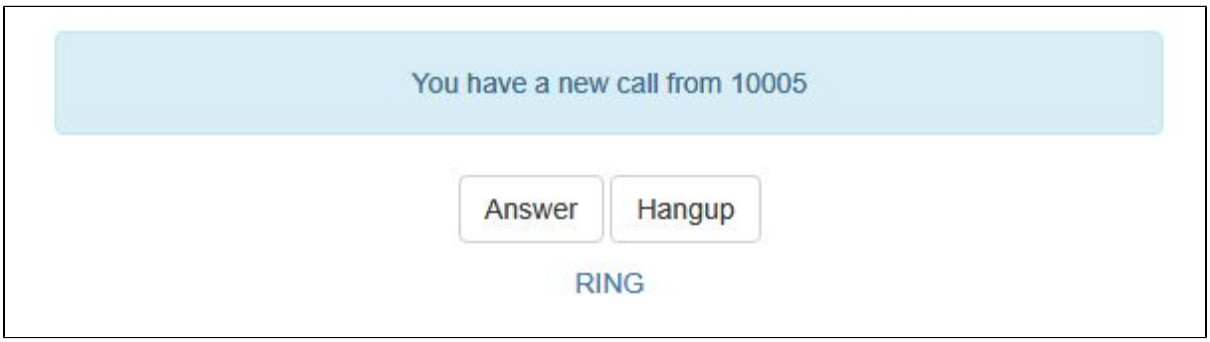
3. Run the software phone, enter the data of the SIP account making the call:



4. Click the Connect button in the browser, a connection to the server is established. In the softphone enter the identifier of the SIP account that receives the call and click the Call button:

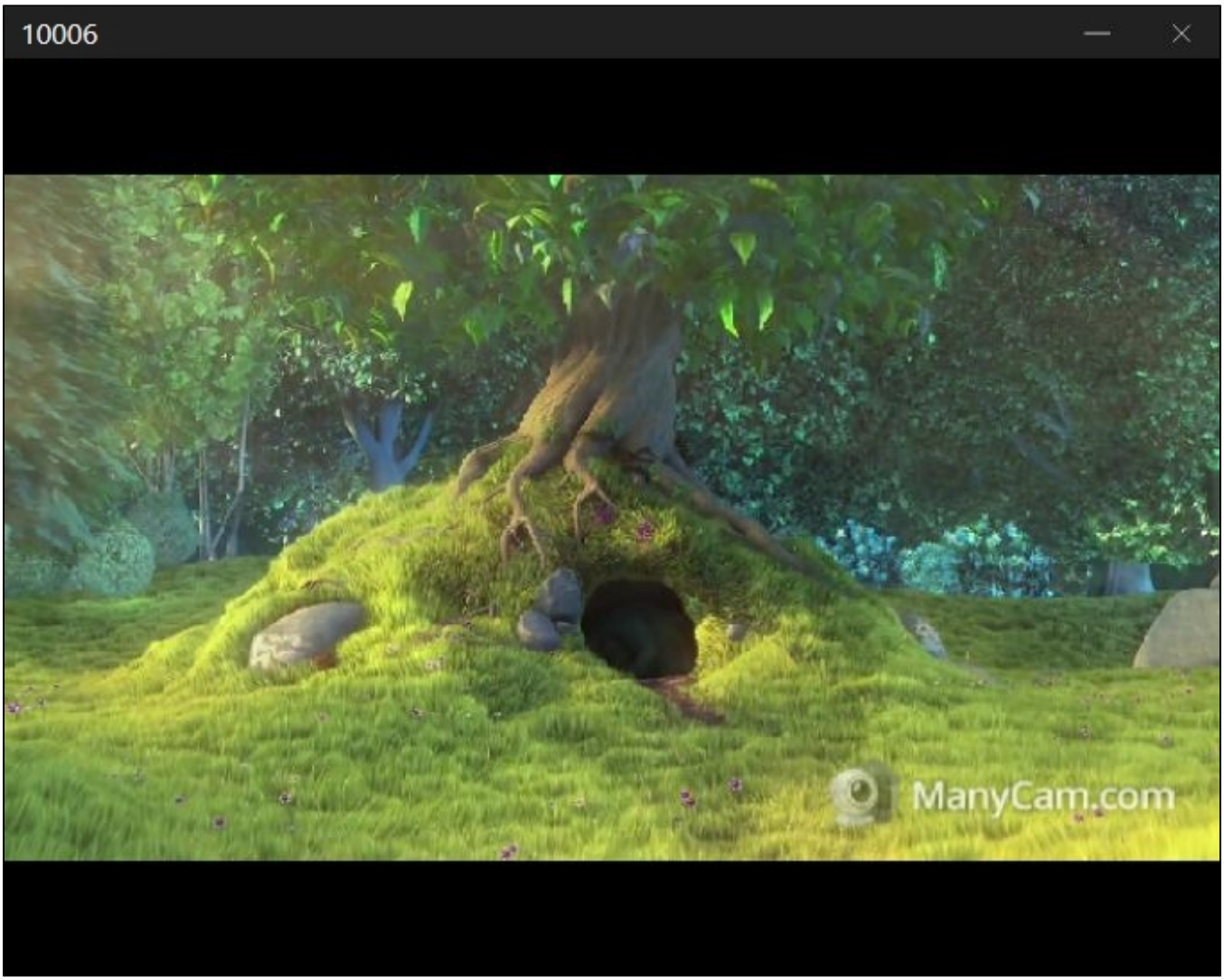


5. Answer the call in the browser by clicking the Answer button:



6. The browser displays the video:

7. The video broadcast from a browser also displays in a separate window of the softphone:

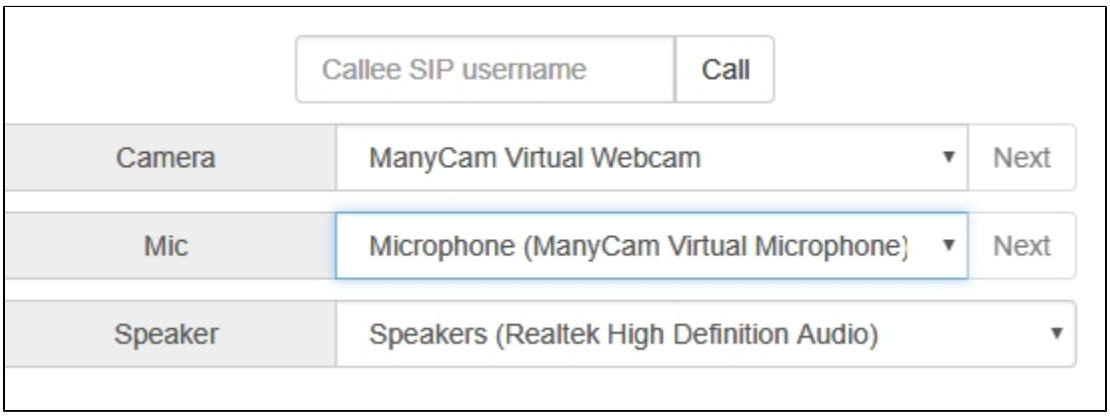


8. To terminate the call, click the Hangup button in the browser or the end call button in the softphone.

Camera, microphone and sound output devices management

Selection and switching input and output devices

Like a video stream capture, camera, microphone and (in Chrome browser only) sound output device can be selected while making a SIP call from browser. Besides, devices can be switched during a call.



1.Choosing camera, microphone and sound output devicecode:

```
Flashphoner.getMediaDevices(null, true, MEDIA_DEVICE_KIND.ALL).then(function (list) {
    for (var type in list) {
        if (list.hasOwnProperty(type)) {
            list[type].forEach(function(device) {
                if (device.type == "mic") {
                    ...
                } else if (device.type == "speaker") {
                    ...
                } else if (device.type == "camera") {
                    ...
                }
            });
        }

    }
    ...
}).catch(function (error) {

    $("#notifyFlash").text("Failed to get media devices "+error);
});
```

2. Switching sound output device during a callcode:

```
    $( "#speakerList" ).change(function() {
        if (currentCall) {
            currentCall.setAudioOutputId($(this).val());
        }
    });
```

3. Swithching microphone during a callcode:

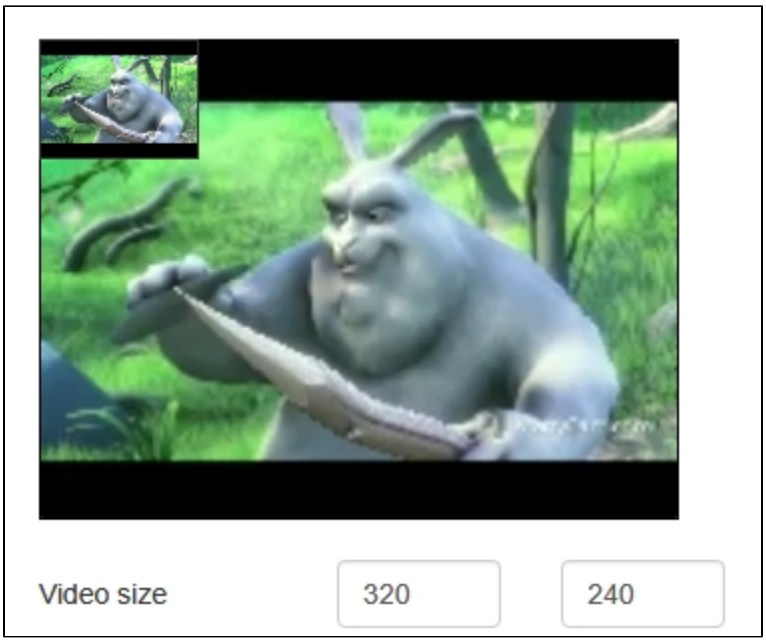
```
$("#switchMicBtn").click(function() {
    if (currentCall) {
        currentCall.switchMic().then(function(id) {
            $('#micList option:selected').prop('selected', false);
            $("#micList option[value='"+ id +"']").prop('selected', true);
        }).catch(function(e) {
            console.log("Error " + e);
        });
    }
}).prop('disabled', true);
```

4. Switching camera during a callcode:

```
$("#switchCamBtn").click(function() {
   if (currentCall) {
       currentCall.switchCam().then(function(id) {
           $('#cameraList option:selected').prop('selected', false);
           $("#cameraList option[value='"+ id +"']").prop('selected', true);
       }).catch(function(e) {
           console.log("Error " + e);
       });
   }
}).prop('disabled', true);
```

Video size setting

An outgoing video size can be specified while making a call



code:

```
function getConstraints() {
    var constraints = {
        ...
        video: {
            deviceId: {exact: $('#cameraList').find(":selected").val()},
            width: parseInt($('#sendWidth').val()),
            height: parseInt($('#sendHeight').val())
        }
    };
    if (Browser.isSafariWebRTC() && Browser.isiOS() && Flashphoner.getMediaProviders()[0] === "WebRTC") {
        constraints.video.width = {min: parseInt($('#sendWidth').val()), max: 640};
        constraints.video.height = {min: parseInt($('#sendHeight').val()), max: 480};
    }
    return constraints;
}
```

Making a call without microphone and camera

In some cases, when a call supposes no two-way communication, e.g. when calling to voice menu, it is possible to make a call without using microphone and camera.

To do this RTP activity timer should be disabled with following parameter in flashphoner.propertiesfile

```
rtp_activity_detecting=false
```

and turn off audio and video in outgoing call constraints for Chrome, Safari and MS Edge browsers

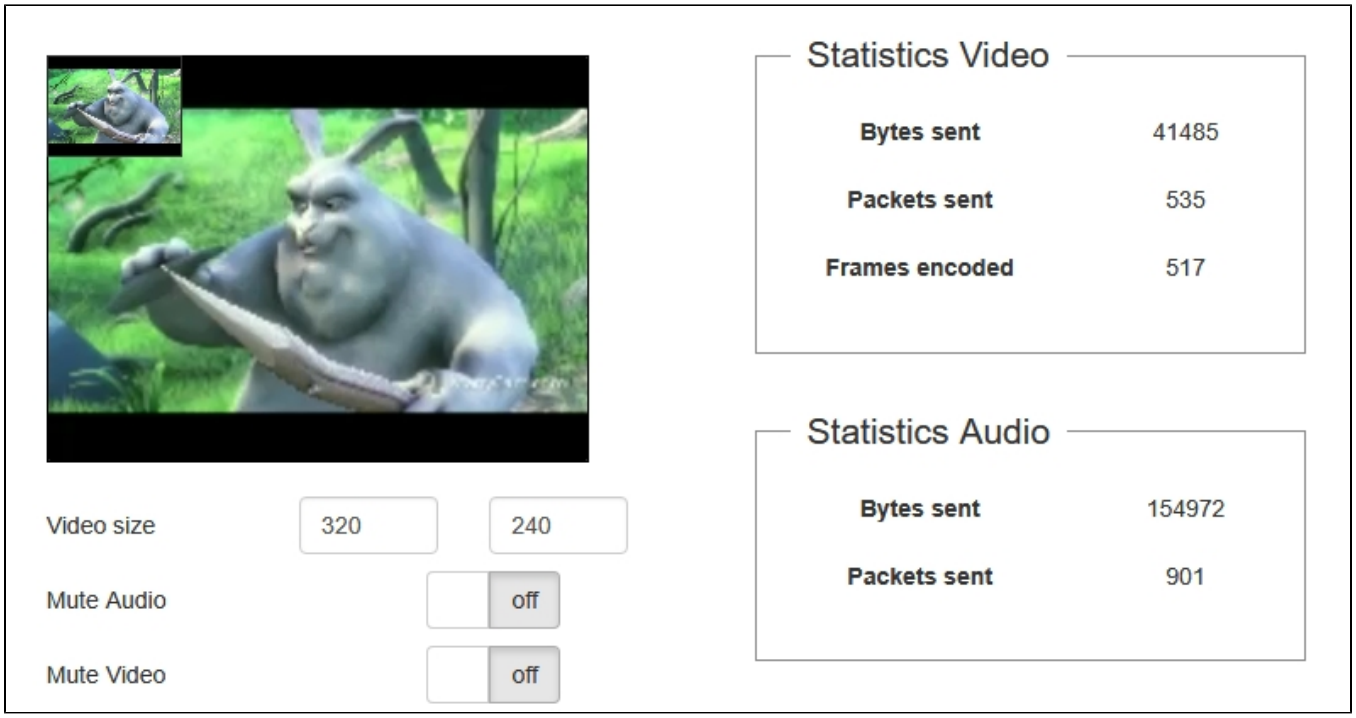
```
var constraints = {
    audio: false,
    video: false
};

var outCall = session.createCall({
            callee: $("#callee").val(),
    visibleName: $("#sipLogin").val(),
    constraints: constraints,
    ...
    })
```

In addition to it, an empty audio stream should be created for Firefox browser:

```
var constraints = {
    audio: false,
    video: false
};
if(Browser.isFirefox()) {
    var audioContext = new AudioContext();
    var emptyAudioStream = audioContext.createMediaStreamDestination().stream;
    constraints.customStream = emptyAudioStream;
}
var outCall = session.createCall({
            callee: $("#callee").val(),
    visibleName: $("#sipLogin").val(),
    constraints: constraints,
    ...
    })
```

WebRTC statistics displaying

A client application can get WebRTC statistics according to thestandardduring a SIP call. The statistics can be displayed in browser, for example:



Note that in Safari browser audio only statistics can be displayed.

1. Statistics displaying during a SIP call

call.getStats()code:

```
        currentCall.getStats(function (stats) {
            if (stats && stats.outboundStream) {
                if (stats.outboundStream.videoStats) {
                    $('#videoStatBytesSent').text(stats.outboundStream.videoStats.bytesSent);
                    $('#videoStatPacketsSent').text(stats.outboundStream.videoStats.packetsSent);
                    $('#videoStatFramesEncoded').text(stats.outboundStream.videoStats.framesEncoded);
                } else {
                    ...
                }

                if (stats.outboundStream.audioStats) {
                    $('#audioStatBytesSent').text(stats.outboundStream.audioStats.bytesSent);
                    $('#audioStatPacketsSent').text(stats.outboundStream.audioStats.packetsSent);
                } else {
                    ...
                }
            }
        });
```

Supported codecs setting

WCS sets the codecs supported to INVITE SDP according to the following parameters inflashphoner.propertiesfile

1. The codecs specified with `codecs` parameter are included to INVITE SDP, by default

```
codecs=opus,alaw,ulaw,g729,speex16,g722,mpeg4-generic,telephone-event,h264,vp8,flv,mpv
```

2. The codecs specified with `codecs_exclude_sip`parameter are excluded from INVITE SDP, by default

```
codecs_exclude_sip=mpeg4-generic,flv,mpv
```

3.The codecs specified by browserare excluded from INVITE SDP if this parameter is set

```
allow_outside_codecs=false
```

4.The codecs specified with`stripCodecs`parameter in client application are excluded from INVITE SDP, for example:

```
var outCall = session.createCall({
    callee: $("#callee").val(),
    ...
    stripCodecs: "SILK,G722"
    ...
});

outCall.call();
```

Additional SDP parameters passing in INVITE request and 200 OK response

When call is made with JavaScript API, an additional parameters can be passed to control bandwith via SDP, for outgoing calls (to INVITE request)

```
var sdpAttributes = ["b=AS:3000","b=TIAS:2500000","b=RS:1000","b=RR:3000"];
var outCall = session.createCall({
        sipSDP: sdpAttributes,
        ...
    });
```

and incoming calls (to 200 OK response)

```
        var sdpAttributes = ["b=AS:3000","b=TIAS:2500000","b=RS:1000","b=RR:3000"];
            inCall.answer({
            sipSDP: sdpAttributes,
            ...
        });
```

Those parameters will be added to SDP after connection information ("c=IN IP4 <WCS IP>") and before time description ("t=0 0"):

```
v=0
o=Flashphoner 0 1541068898263 IN IP4 192.168.1.5
s=Flashphoner/1.0
c=IN IP4 192.168.1.5
b=AS:3000
b=TIAS:2500000
b=RS:1000
b=RR:3000
t=0 0
m=audio
```

Known issues

1. It's impossible to make a SIP call if 'SIP Login' and 'SIP Authentification name' fields are incorrect

Symptoms: SIP call stucks in PENDING state.

Solution: according to thestandard, 'SIP Login' and 'SIP Authentification name' should not contain any of unescaped spaces and special symbols and should not be enclosed in angle brackets '<>'.

For example, this is not allowedby the standard

```
sipLogin='Ralf C12441@host.com'
sipAuthenticationName='Ralf C'
sipPassword='demo'
sipVisibleName='null'
```

and this is allowed

```
sipLogin='Ralf_C12441'
sipAuthenticationName='Ralf_C'
sipPassword='demo'
sipVisibleName='Ralf C'
```

2. There are some problems with sound while SIP calls from Edge browser.

Symptoms:

a) The outgoing sound is sometimes abruptly muffled, then it goes normally.

b) The incoming sound is heard only if you speak into the microphone.

Solution:

Switch SILK and G.722 codecs usage off in SIP calls for Edge browserwith stripCodecs option:

```
var outCall = session.createCall({
    callee: $("#callee").val(),
    visibleName: $("#sipLogin").val(),
    localVideoDisplay: localDisplay,
    remoteVideoDisplay: remoteDisplay,
    constraints: constraints,
    receiveAudio: true,
    receiveVideo: false,
    stripCodecs: "silk,g722"
    ...
});

outCall.call();
```

or with server setting

```
codecs_exclude_sip=g722,mpeg4-generic,flv,mpv
```

3. Microphone swithing does not work in Safari browser.

Symptoms: microphone does not switch usingswitchMic() WCS WebSDK method.

Solution: use another browser, because Safari always uses sound input microphone, that is chosen in system sound menu (hold down the option (alt) button and click on the sound icon in the menu bar). When microphone is chosen in sound menu, Mac reboot is required.

If Logitech USB camers microphone does not work (when it is chosen in sound menu), format / sample rate changing in Audio MIDI Setup and rebooting can help.

4. Outgoing video SIP call cannot be established if INVITE SDP size exceeds MTU

Symptoms: SIP server return 408 Reques timeout when trying to establish video SIP call, audio calls can be established successfully through the same server.

Solution: reduce the number of codecs in the INVITE SDP so that the SDP fit into the packet size defined by MTU (usually 1500 bytes)using the following settings

```
codecs_exclude_sip=mpeg4-generic,flv,mpv,opus,ulaw,h264,g722,g729
allow_outside_codecs=false
```

Only codecs supported by both sides of the call should be left, in this case it is VP8 andPCMA (alaw).