

Phone Video

Пример вебфона для видеозвонков

The screenshot displays the 'Phone Video' web interface. On the left, the 'Connection' section contains several input fields: 'WCS URL' (ws://localhost:8080), 'SIP Login' (10006), 'SIP Auth Name' (10006), 'SIP Password' (masked), 'SIP Domain' (sip.flashphoner.com), 'SIP Outbound Proxy' (sip.flashphoner.com), and 'SIP Port' (5060). A 'Register required' checkbox is checked. Below these fields, the status is 'REGISTERED' and a 'Disconnect' button is visible. At the bottom left, there is a '10006' field and a 'Hangup' button. In the center, a video call is shown with a green-tinted landscape scene. Below the video, there are controls for 'Video size' (320 and 240), 'Video FPS' (30), 'Mute Audio' (off), 'Mute Video' (off), and 'SDP replace' (0 with 0). On the right, two statistics boxes are present: 'Statistics Video' showing 'Bytes sent' (439775) and 'Packets sent' (545), and 'Statistics Audio' showing 'Bytes sent' (88193) and 'Packets sent' (1379).

Код примера

Код данного примера находится на WCS-сервере по следующему пути:

`/usr/local/FlashphonerWebCallServer/client2/examples/demo/sip/phone-video`

- phone-video.css - файл стилей
- phone-video.html - страница вебфона
- call-fieldset.html - форма с полями, необходимыми для установления соединения
- call-controls.html - элементы для управления звонками
- phone-video.js - скрипт, обеспечивающий работу вебфона

Тестировать данный пример можно по следующему адресу:

`https://host:8888/client2/examples/demo/sip/phone-video/phone-video.html`

Здесь host - адрес WCS-сервера.

Работа с кодом примера

Для разбора кода возьмем версию файла `phone-video.js` с хешем `ecbadc3`, которая находится [здесь](#) и доступна для скачивания в соответствующей сборке [2.0.212](#).

1. Инициализация API

`Flashphoner.init()` [code](#)

```
Flashphoner.init();
```

2. Подключение к серверу

`Flashphoner.createSession()` [code](#)

Методу `createSession()` передается объект с параметрами для подключения

- `urlServer` - URL для WebSocket-соединения с WCS-сервером
- `sipOptions` - объект с параметрами для SIP-соединения

```
var url = $('#urlServer').val();
var registerRequired = $("#sipRegisterRequired").is(':checked');

var sipOptions = {
  login: $("#sipLogin").val(),
  authenticationName: $("#sipAuthenticationName").val(),
  password: $("#sipPassword").val(),
  domain: $("#sipDomain").val(),
  outboundProxy: $("#sipOutboundProxy").val(),
  port: $("#sipPort").val(),
  registerRequired: registerRequired
};

var connectionOptions = {
  urlServer: url,
  sipOptions: sipOptions
};

//create session
console.log("Create new session with url " + url);
Flashphoner.createSession(connectionOptions).on(SESSION_STATUS.ESTABLISHED,
function(session){
  ...
});
```

3. Получение от сервера события, подтверждающего успешное соединение

`ConnectionStatusEvent ESTABLISHED` [code](#)

```
Flashphoner.createSession(connectionOptions).on(SESSION_STATUS.ESTABLISHED,
function(session){
    setStatus("#regStatus", SESSION_STATUS.ESTABLISHED);
    onConnected(session);
    if (!registerRequired) {
        disableOutgoing(false);
    }
}).on(SESSION_STATUS.REGISTERED, function(session){
    ...
}).on(SESSION_STATUS.DISCONNECTED, function(){
    ...
}).on(SESSION_STATUS.FAILED, function(){
    ...
}).on(SESSION_STATUS.INCOMING_CALL, function(call){
    ...
});
```

4. Получение от сервера события, подтверждающего успешную регистрацию на SIP-сервере

`ConnectionStatusEvent REGISTERED` [code](#)

```
Flashphoner.createSession(connectionOptions).on(SESSION_STATUS.ESTABLISHED,
function(session){
    ...
}).on(SESSION_STATUS.REGISTERED, function(session){
    setStatus("#regStatus", SESSION_STATUS.REGISTERED);
    onConnected(session);
    if (registerRequired) {
        disableOutgoing(false);
    }
}).on(SESSION_STATUS.DISCONNECTED, function(){
    ...
}).on(SESSION_STATUS.FAILED, function(){
    ...
}).on(SESSION_STATUS.INCOMING_CALL, function(call){
    ...
});
```

5. Получение от сервера события, сигнализирующего о входящем звонке

`ConnectionStatusEvent INCOMING_CALL` [code](#)

```
Flashphoner.createSession(connectionOptions).on(SESSION_STATUS.ESTABLISHED,
function(session){
    ...
}).on(SESSION_STATUS.REGISTERED, function(session){
    ...
}).on(SESSION_STATUS.DISCONNECTED, function(){
    ...
});
```

```

    }).on(SESSION_STATUS.FAILED, function(){
        ...
    }).on(SESSION_STATUS.INCOMING_CALL, function(call){
        call.on(CALL_STATUS.RING, function(){
            ...
        });
        onIncomingCall(call);
    });
});

```

6. Исходящий звонок

`Session.createCall()`, `Call.call()` [code](#)

При создании звонка в метод передаются следующие параметры:

- `callee` - SIP имя пользователя вызываемого
- `visibleName` - отображаемое имя
- `localVideoDisplay` - `div` элемент, в котором будет отображаться видео с камеры
- `remoteVideoDisplay` - `div` элемент, в котором будет отображаться видео другой стороны

```

var outCall = session.createCall({
    callee: $("#callee").val(),
    visibleName: $("#sipLogin").val(),
    localVideoDisplay: localVideo,
    remoteVideoDisplay: remoteVideo,
    localVideoDisplay: localVideo,
    constraints: constraints,
    sdpHook: rewriteSdp,
    stripCodecs: "SILK"
    ...
});

outCall.call();

```

7. Ответ на входящий звонок

`call.answer()` [code](#)

Методу передается объект с опциями для ответа

- `localVideoDisplay` - `div` элемент, в котором будет отображаться видео с камеры
- `remoteVideoDisplay` - `div` элемент, в котором будет отображаться видео другой стороны

```

$("#answerBtn").off('click').click(function(){
    $(this).prop('disabled', true);
    inCall.answer({
        localVideoDisplay: localVideo,

```

```
        remoteVideoDisplay: remoteVideo,  
        constraints: constraints,  
        sdpHook: rewriteSdp,  
        stripCodecs: "SILK"  
    });  
    showAnswered();  
}).prop('disabled', false);
```

8. Завершение исходящего звонка

`Call.hangup()` [code](#)

```
$("#callBtn").text("Hangup").off('click').click(function(){  
    $(this).prop('disabled', true);  
    outCall.hangup();  
}).prop('disabled', false);
```

9. Завершение входящего звонка

`Call.hangup()` [code](#)

```
$("#hangupBtn").off('click').click(function(){  
    $(this).prop('disabled', true);  
    $("#answerBtn").prop('disabled', true);  
    inCall.hangup();  
}).prop('disabled', false);
```

10. Завершение текущего звонка при закрытии соединения

`Call.hangup()` [code](#)

```
function onConnected(session) {  
    $("#connectBtn").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. Включение/выключение аудио и видео

`Call.muteAudio()`, `Call.unmuteAudio()`, `Call.muteVideo()`, `Call.unmuteVideo()` [code](#)

```

// Mute audio in the call
function mute() {
  if (currentCall) {
    currentCall.muteAudio();
  }
}

// Unmute audio in the call
function unmute() {
  if (currentCall) {
    currentCall.unmuteAudio();
  }
}

// Mute video in the call
function muteVideo() {
  if (currentCall) {
    currentCall.muteVideo();
  }
}

// Unmute video in the call
function unmuteVideo() {
  if (currentCall) {
    currentCall.unmuteVideo();
  }
}

```

12. Сбор WebRTC статистики во время звонка

`Call.getStats()` code

```

function loadStats() {
  if (currentCall) {
    // Stats should be collected for active calls only #WCS-3260
    let status = currentCall.status();
    if (status !== CALL_STATUS.ESTABLISHED && status !== CALL_STATUS.HOLD)
    {
      return;
    }
    currentCall.getStats(function (stats) {
      if (stats && stats.outboundStream) {
        if (stats.outboundStream.video) {
          $('#videoStatBytesSent').text(stats.outboundStream.video.bytesSent);
          $('#videoStatPacketsSent').text(stats.outboundStream.video.packetsSent);
        } else {
          $('#videoStatBytesSent').text(0);
          $('#videoStatPacketsSent').text(0);
        }
      }

      if (stats.outboundStream.audio) {
        $('#audioStatBytesSent').text(stats.outboundStream.audio.bytesSent);

```

```
$('#audioStatPacketsSent').text(stats.outboundStream.audio.packetsSent);
    } else {
        $('#audioStatBytesSent').text(0);
        $('#audioStatPacketsSent').text(0);
    }
}
});
}
}
```