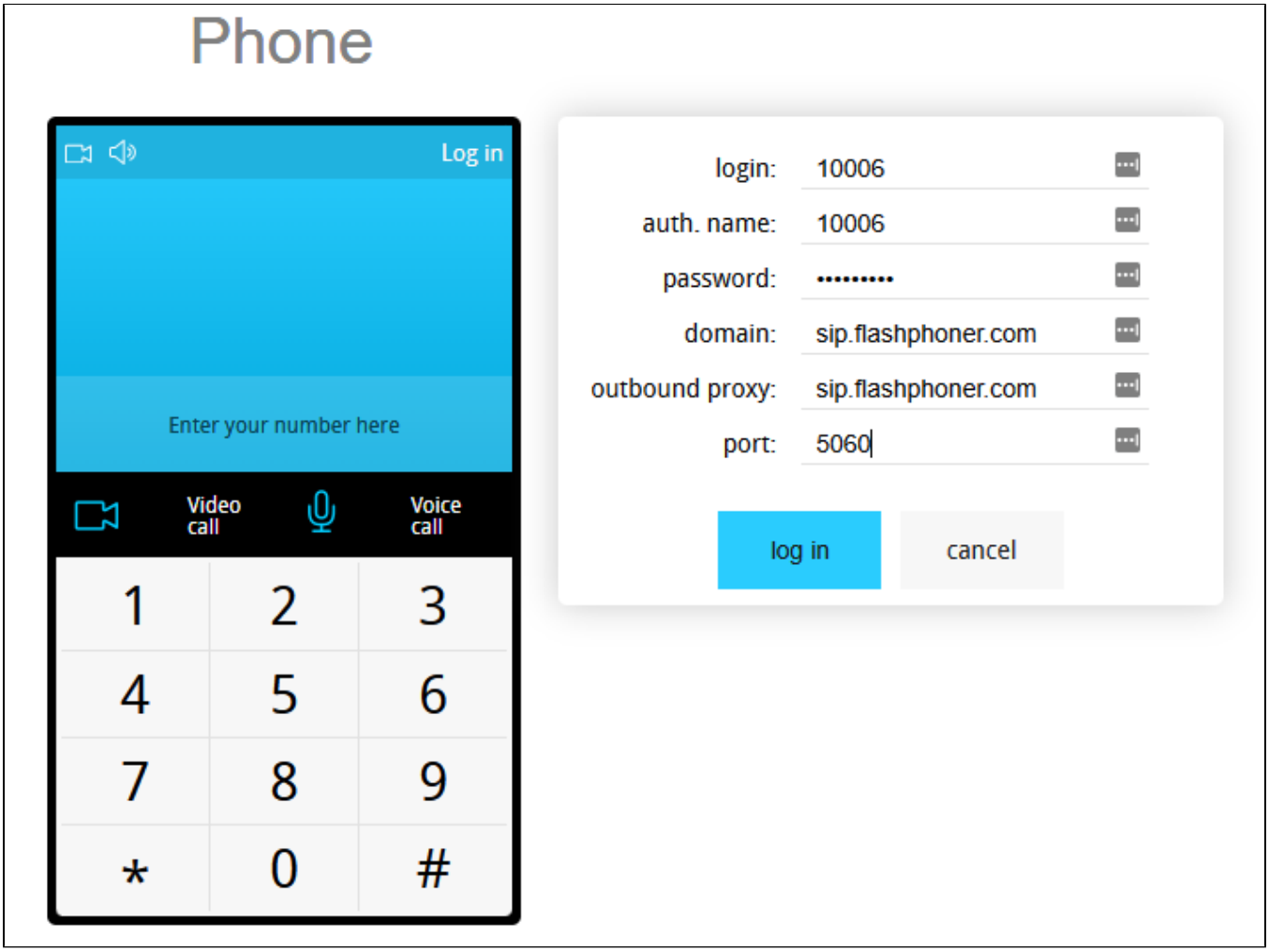
Phone UI

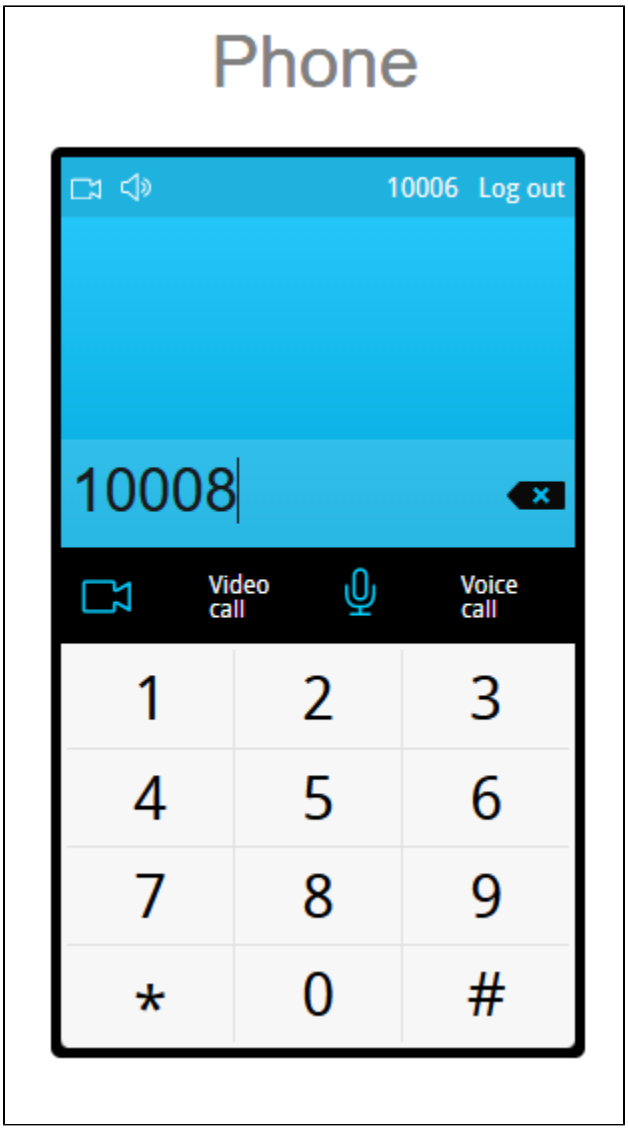
Пример вебфона с интерфейсом телефона

На скриншотах ниже

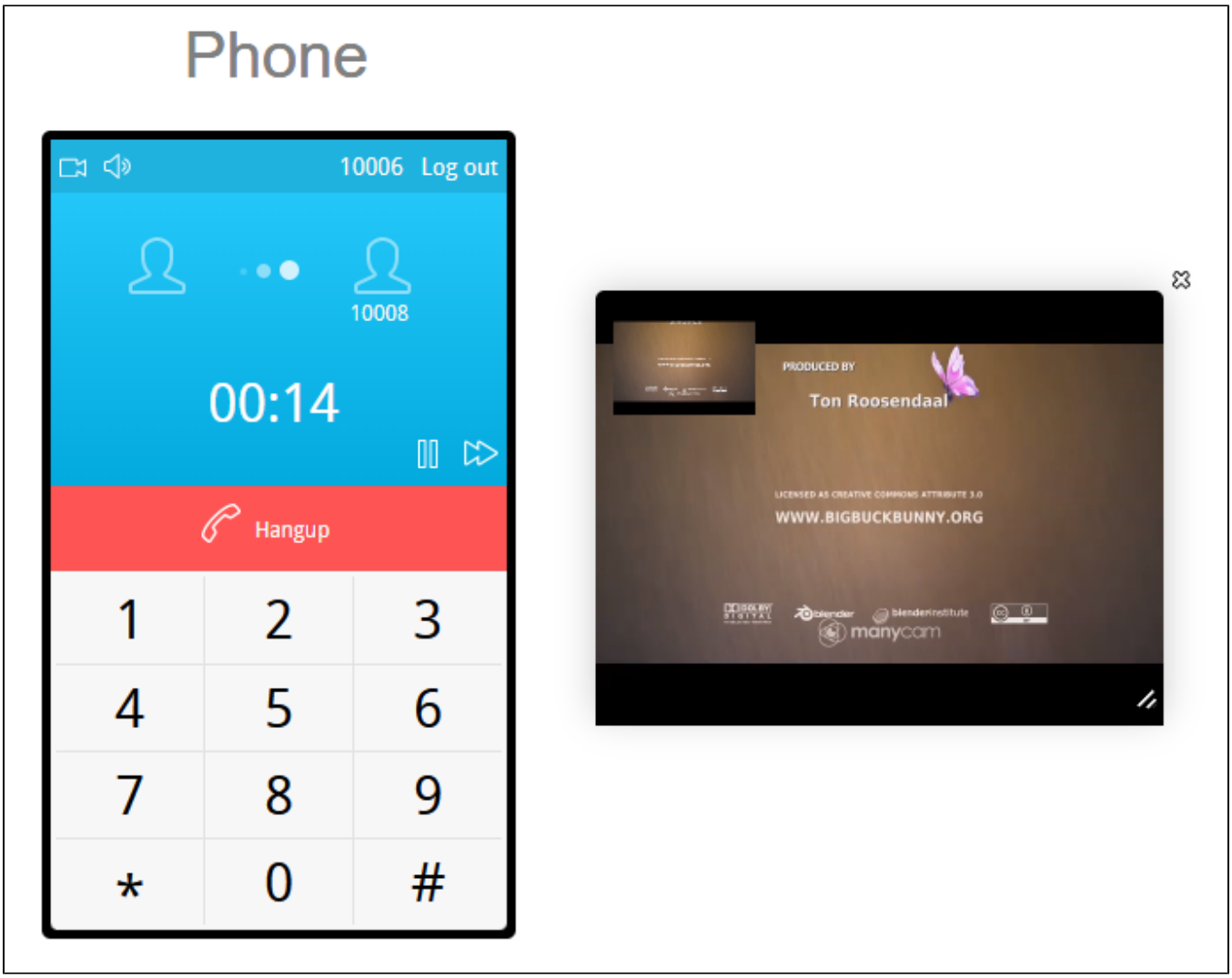
- параметры для соединения введены в форме входа



- введено SIP имя вызываемого пользователя



- установлен видеозвонок



Видео или аудиозвонок инициируется при нажатии на соответствующую кнопку и завершается при нажатии на кнопку Hangup.
Для удержания (hold) звонка используется иконка 'пауза'.
Для регулировки громкости используется ползунок, который появляется при клике на иконке 'динамик' в левом верхнем углу.

Код примера

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

/usr/local/FlashphonerWebCallServer/client2/examples/demo/sip/phone-ui

gui - директория с файлами необходимыми для графического интерфейса: файлы стилей, фонты, изображения
listener - директория, содержащая скрипты с функциями для обработки событий (event listeners)
sounds - директория, содержащая звуковые файлы для событий
SoundControl.js - скрипт для воспроизведения звуков
Phone.js - скрипт, обеспечивающий работу вебфона
Phone.html - страница вебфона

Тестировать данный пример можно по следующему адресу:

https://host:8888/client2/examples/demo/sip/phone-ui/Phone.html

Здесь host - адрес WCS-сервера.

Работа с кодом примера

Для разбора кода возьмем версию файла Phone.js с хешем ecbadc3, которая находится здесь и доступна для скачивания в соответствующей сборке 2.0.212.

В этом скрипте методы Flashphoner API вызываются из соответствующих методов объекта Phone.

Например, метод для установления соединения с сервером createSession() вызывается из метода connect()

code

```
Phone.prototype.connect = function () {
   var me = this;
   ...
   Flashphoner.createSession(connectionOptions).on(SESSION_STATUS.ESTABLISHED, function(session){
   ...
}
```

Объект Phone создается и инициализируется после инициализации API

code

```
var phone = new Phone();

phone.init();
```

Кроме методов, необходимых для звонков, объект предоставляет методы для изменения элементов интерфейса в зависимости от статуса соединения и звонка (line 194 - line 416).

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 = setURL();

    this.sipOptions = {};
    this.sipOptions.login = $('#sipLogin').val();
    this.sipOptions.password = $('#sipPassword').val();
    this.sipOptions.authenticationName = $('#sipAuthenticationName').val();
    this.sipOptions.domain = $('#sipDomain').val();
    this.sipOptions.outboundProxy = $('#sipOutboundProxy').val();
    this.sipOptions.port = $('#sipPort').val();
    this.sipOptions.useProxy = true;
    this.sipOptions.registerRequired = true;

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

    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){
    me.session = session;
    me.connectionStatusListener(SESSION_STATUS.ESTABLISHED);
}).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){
    me.registrationStatusListener(SESSION_STATUS.REGISTERED);
}).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(){
        ...
    });
    me.onCallListener(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
- constraints - объект с параметрами, указывающими на наличие аудио и видео

```
    var constraints = {
        audio: true,
        video: hasVideo
    };

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

    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

```
Phone.prototype.answer = function () {
    trace("Phone - answer " + this.currentCall.id());
    this.flashphonerListener.onAnswer(this.currentCall.id());
    this.currentCall.answer({
        localVideoDisplay: this.localVideo,
        remoteVideoDisplay: this.remoteVideo
    });
};
```

8. Удержание звонка.

- удержание: call.hold() code

```
Phone.prototype.hold = function () {
    trace("Phone - hold callId: " + this.currentCall.id());
    this.currentCall.hold();
};
```

- возобновление: call.unhold() code

```
Phone.prototype.unhold = function () {
    trace("Phone - hold callId: " + this.currentCall.id());
    this.currentCall.unhold();
};
```

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. Закрытие соединения.

session.disconnect() code

```
Phone.prototype.disconnect = function () {
    trace("Phone - disconnect");
    this.session.disconnect();
};
```