iOS Phone

Описание

Пример демонстрирует возможности совершения SIP аудио звонков при помощи iOS SDK

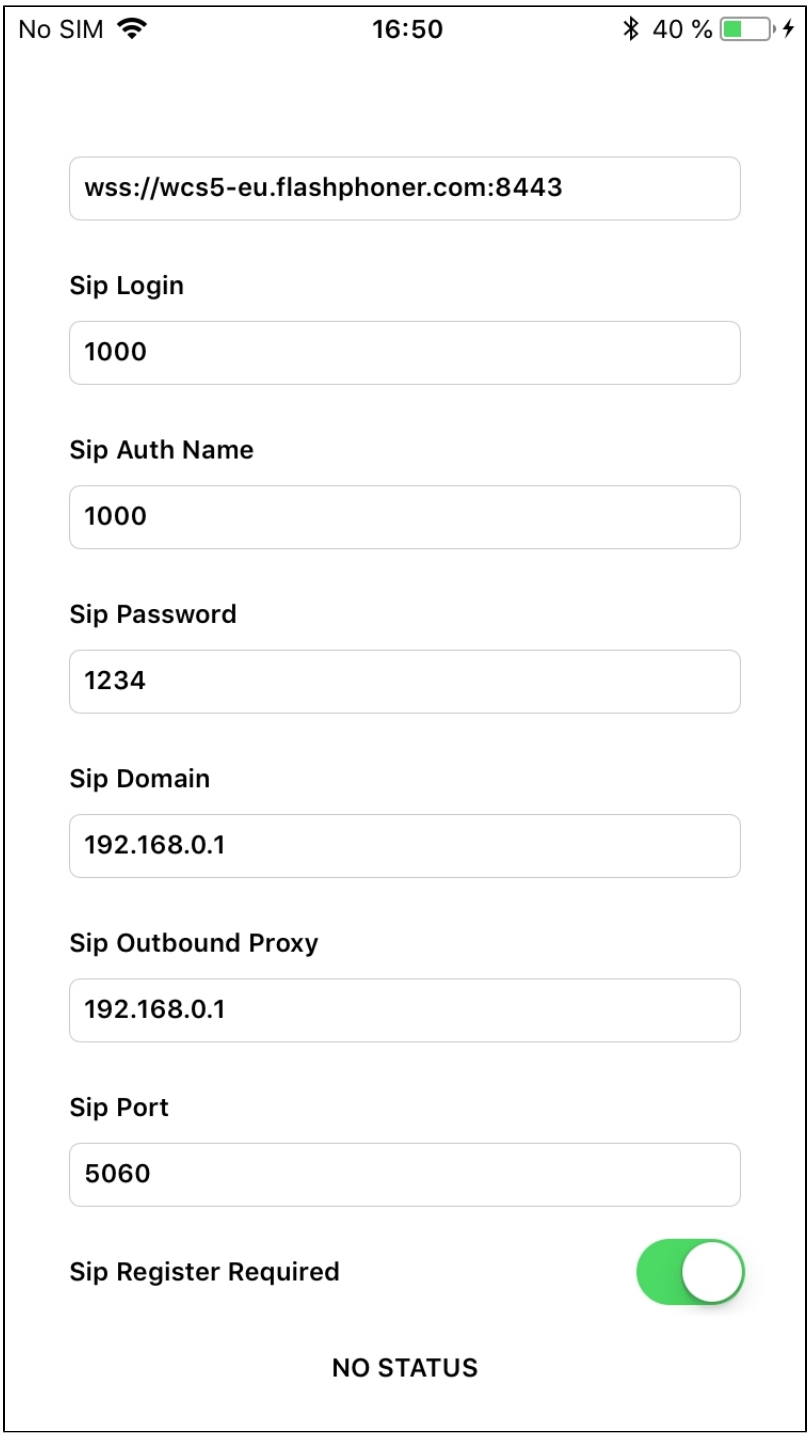
На скриншотах, приведенных ниже, отображается интерфейс приложения перед совершением звонка.

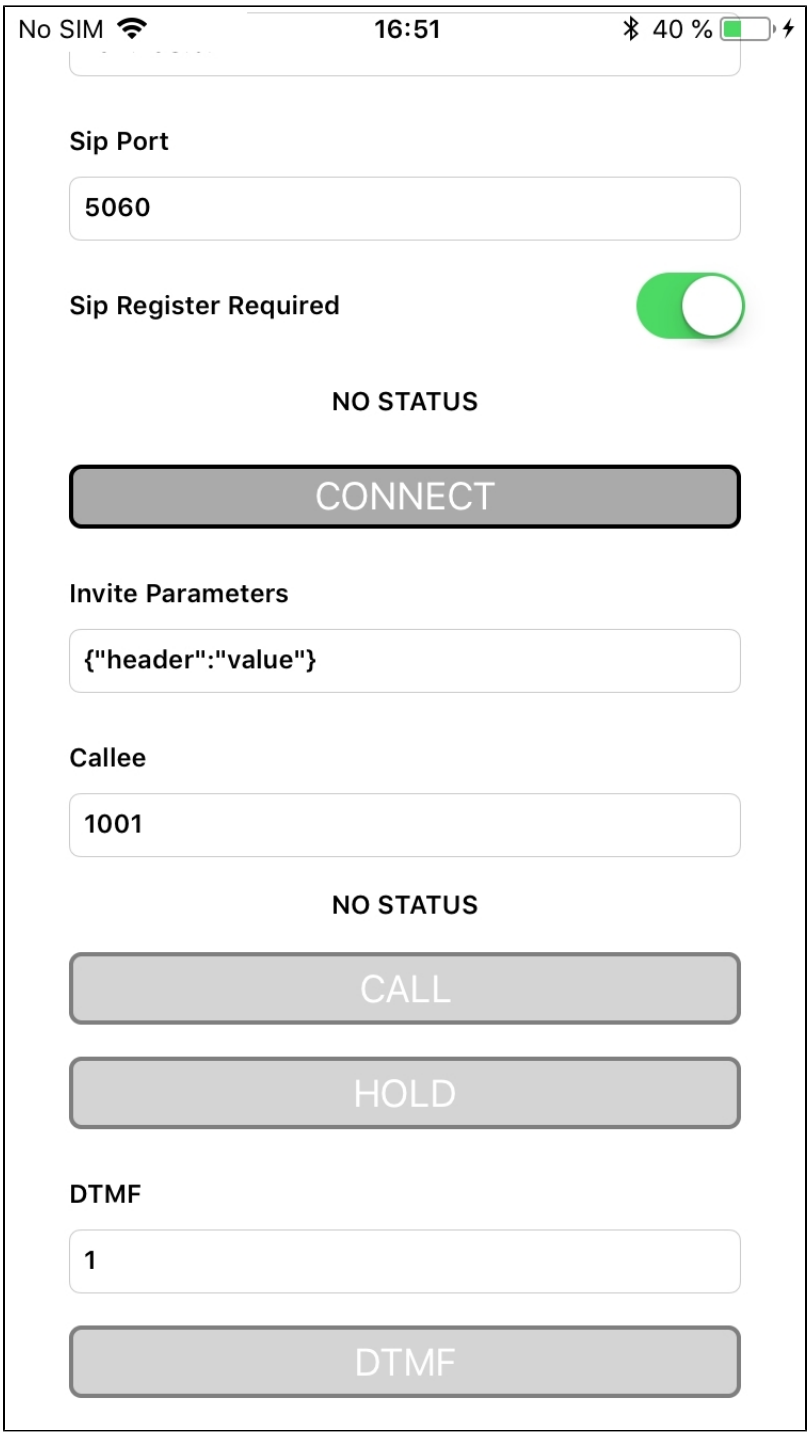
В поле ввода 'WCS URL' указан адрес демонстрационного WCS-серверawcs5-eu.flashphoner.com

В полях ввода 'SIP...' указываются параметры регистрации на SIP-сервере.
В поле 'Callee', указано имя вызываемого абонента.

Поле 'Invite parameters' предназначено для ввода дополнительных параметров сообщения SIP INVITE.

Соединение с сервером устанавливается при нажатии на кнопку 'Connect'.Звонок устанавливается/завершается по нажатию Call/Hangup, вводится в режим удержания либо выводится из него кнопкой Hold/Unhold.


No SIM
16:50
40 %
wss://wcs5-eu.flashphoner.com:8443
Sip Login
1000
Sip Auth Name
1000
Sip Password
1234
Sip Domain
192.168.0.1
Sip Outbound Proxy
192.168.0.1
Sip Port
5060
Sip Register Required
NO STATUS



Работа с кодом примера

Для разбора кода возьмем версию примера PhoneMin, которая доступна на GitHub.

Класс для основного вида приложения: ViewController (заголовочный файл ViewController.h; файл имплементации ViewController.m).

1. Импорт API code

```
#import <FPWCSApi2/FPWCSApi2.h>
```

2. Подключение к серверу.

FPWCSApi2.createSession, FPWCSApi2Session.connect code
В параметрах сессии указываются:

- URL WCS-сервера
- параметры SIP-аккаунта для совершения исходящих и приема входящих звонков
- имя серверного приложения defaultApp

```
FPWCSApi2SessionOptions *options = [[FPWCSApi2SessionOptions alloc] init];
options.urlServer = _connectUrl.text;
options.sipRegisterRequired = _sipRegRequired.control.isOn;
options.sipLogin = _sipLogin.input.text;
options.sipAuthenticationName = _sipAuthName.input.text;
options.sipPassword = _sipPassword.input.text;
options.sipDomain = _sipDomain.input.text;
options.sipOutboundProxy = _sipOutboundProxy.input.text;
options.sipPort = [NSNumber numberWithInteger: [_sipPort.input.text integerValue]];
options.appKey = @"defaultApp";
NSError *error;
...
session = [FPWCSApi2 createSession:options error:&error];
...
[session connect];
```

3. Исходящий звонок.

FPWCSApi2Session.createCall, FPWCSApi2Call.call code

При создании звонка в метод createCall передаются параметры:

- имя вызываемого SIP-аккаунта
- дополнительные параметры SIP INVITE запроса, введенные пользователем

```
- (FPWCSApi2Call *)call {
    FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
    FPWCSApi2CallOptions *options = [[FPWCSApi2CallOptions alloc] init];
    NSString *parameters = _inviteParameters.input.text;
    if (parameters && [parameters length] > 0) {
        NSError* err = nil;
        parameters = [parameters stringByReplacingOccurrencesOfString:@""" withString:@"\""];
        NSMutableDictionary *dictionary = [NSJSONSerialization JSONObjectWithData:[parameters dataUsingEncoding:
NSUTF8StringEncoding] options:0 error:&err];
        if (err) {
            NSLog(@"Error converting JSON Invite parameters to dictionary %@, JSON %@", err, parameters);
        } else {
            options.inviteParameters = dictionary;
        }
    }
    options.callee = _callee.input.text;
    ...
    NSError *error;
    call = [session createCall:options error:&error];
    ...
    [call call];
    return call;
}
```

4. Получение от сервера события, сигнализирующего о входящем звонке

FPWCSApi2Session.onIncomingCallCallback code

```
[session onIncomingCallCallback:^(FPWCSApi2Call *rCall) {
    call = rCall;

    [call on:kFPWCSCallStatusBusy callback:^(FPWCSApi2Call *call){
        [self changeCallStatus:call];
        [self toCallState];
    }];

    [call on:kFPWCSCallStatusFailed callback:^(FPWCSApi2Call *call){
        [self changeCallStatus:call];
        [self toCallState];
    }];

    [call on:kFPWCSCallStatusRing callback:^(FPWCSApi2Call *call){
        [self changeCallStatus:call];
        [self toHangupState];
    }];

    [call on:kFPWCSCallStatusHold callback:^(FPWCSApi2Call *call){
        [self changeCallStatus:call];
        [self changeViewState:_holdButton enabled:YES];
    }];

    [call on:kFPWCSCallStatusEstablished callback:^(FPWCSApi2Call *call){
        [self changeCallStatus:call];
        [self toHangupState];
        [self changeViewState:_holdButton enabled:YES];
    }];

    [call on:kFPWCSCallStatusFinish callback:^(FPWCSApi2Call *call){
        [self changeCallStatus:call];
        [self toCallState];
        [self dismissViewControllerAnimated:YES completion:nil];
    }];
    ...
}];
```

5. Ответ на входящий звонок.

FPWCSApi2Call.answer code

```
alert = [UIAlertController
                             alertControllerWithTitle:[NSString stringWithFormat:@"Incoming call from '%@'",
[rCall getCallee]]
                             message:error.localizedDescription
                             preferredStyle:UIAlertControllerStyleAlert];

UIAlertAction* answerButton = [UIAlertAction
                               actionWithTitle:@"Answer"
                               style:UIAlertActionStyleDefault
                               handler:^(UIAlertAction * action) {
                                   [call answer];
                               }];

[alert addAction:answerButton];
UIAlertAction* hangupButton = [UIAlertAction
                               actionWithTitle:@"Hangup"
                               style:UIAlertActionStyleDefault
                               handler:^(UIAlertAction * action) {
                                   [call hangup];
                               }];

[alert addAction:hangupButton];
[self presentViewController:alert animated:YES completion:nil];
```

6. Удержание звонка.

FPWCSApi2Call.hold, unhold code

```
- (void)holdButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"UNHOLD"]) {
        if (call) {
            [call unhold];
            [_holdButton setTitle:@"HOLD" forState:UIControlStateNormal];
        }
    } else {
        if (call) {
            [call hold];
            [_holdButton setTitle:@"UNHOLD" forState:UIControlStateNormal];
        }
    }
}
```

7. Отправка тонального сигнала

FPWCSApi2Call.sendDTMF code

```
- (void)dtmfButton:(UIButton *)button {
    if (call) {
        [call sendDTMF:_dtmf.input.text type:kFPWCSCallDTMFRFC2833];
    }
}
```

8. Переключение с голосового динамика на динамик громкой связи

FPWCSApi2Call.setLoudspeakerStatus code

```
- (void)useLoudSpeakerValueChanged:(id)sender {
    if (call) {
        [call setLoudspeakerStatus:_useLoudSpeaker.control.isOn withError:nil];
    }
}
```

9. Завершение исходящего звонка.

FPWCSApi2Call.hangup code

```
- (void)callButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"HANGUP"]) {
        if ([FPWCSApi2 getSessions].count) {
            [call hangup];
        } else {
            [self toCallState];
        }
        ...
    }
}
```

10. Завершение входящего звонка.

FPWCSApi2Call.hangup code

```
UIAlertAction* hangupButton = [UIAlertAction
                               actionWithTitle:@"Hangup"
                               style:UIAlertActionStyleDefault
                               handler:^(UIAlertAction * action) {
                                   [call hangup];
                               }];

[alert addAction:hangupButton];
```

11. Закрытие соединения.

FPWCSApi2Session.disconnect code

```
- (void)connectButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"DISCONNECT"]) {
        if ([FPWCSApi2 getSessions].count) {
            FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
            NSLog(@"Disconnect session with server %@", [session getServerUrl]);
            [session disconnect];
        } else {
            NSLog(@"Nothing to disconnect");
            [self onDisconnected];
        }
        ...
    }
}
```