iOS Phone

Example of iOS application for audio calls

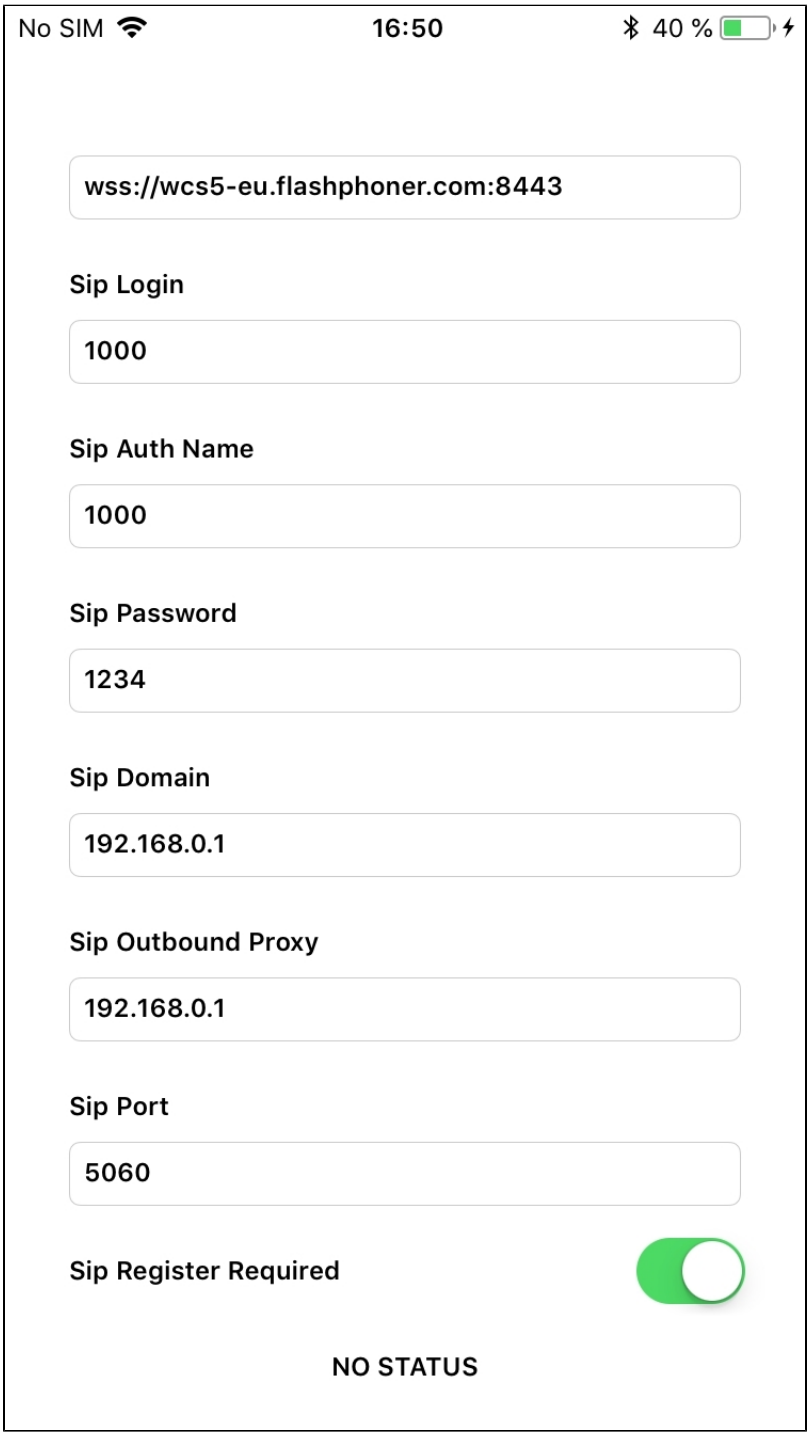
On the screenshot below the example is displayed before a call will be established.

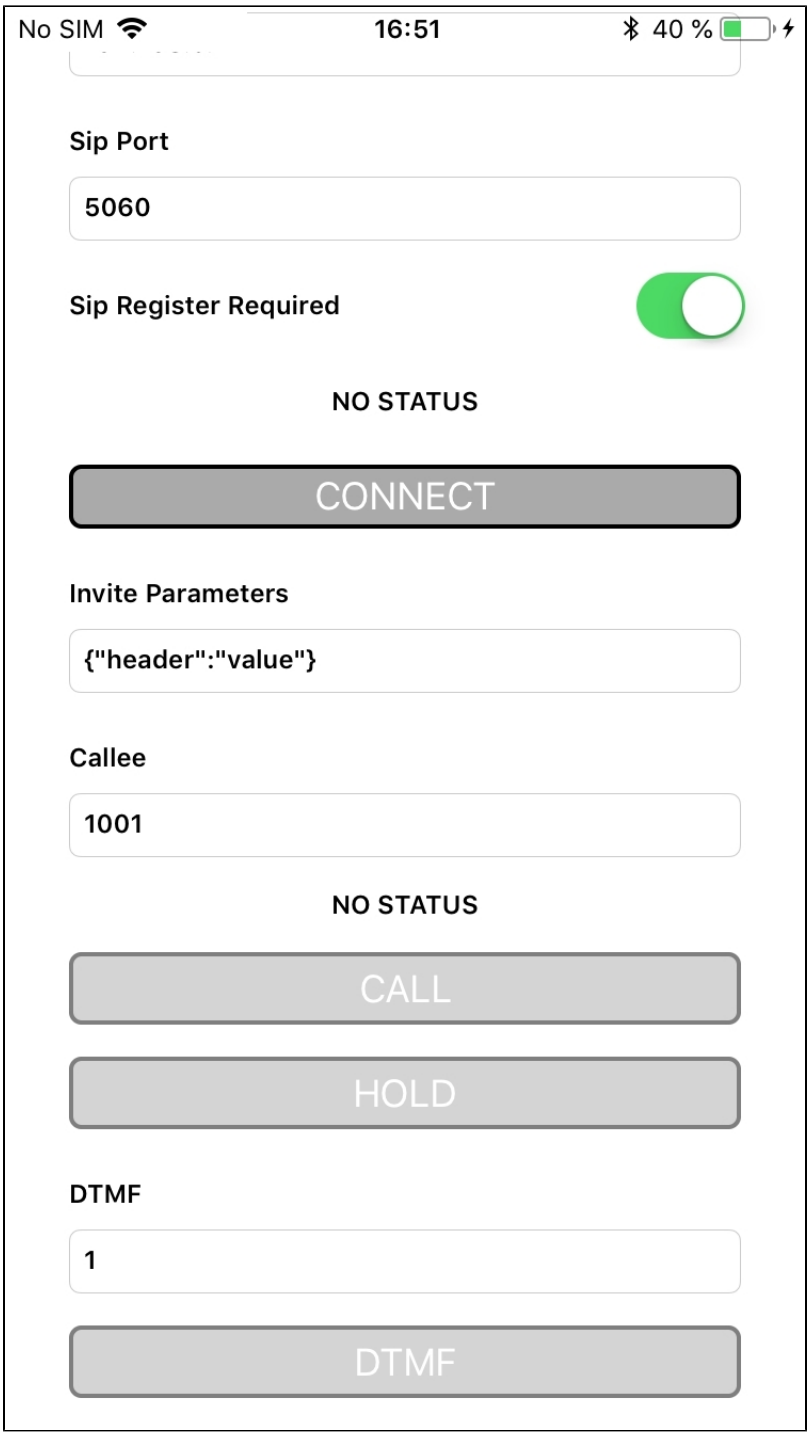
In the input field 'WCS URL', wcs5-eu.flashphoner.com is the address of the WCS server.

In 'SIP' fileds, SIP parameters to register on SIP server must be entered.
In the input field 'Callee', 1001 is the SIP username of the callee.

'Invite parameters' is field to enter additionat SIP INVITE message parameters.

SIP connection is established/closed when Connect/Disconnect button is clicked.
Call is placed/terminated when Call/Hangup button is clicked, and put on hold/retrieve when Hold/Unhold button is clicked.





Analyzing the code of the example

To analyze the code, let's takePhoneMin example, which can be downloaded with corresponding build2.5.2.

View class for the main view of the application: ViewController (header fileViewController.h; implementation fileViewController.m).

1. Import of API.code

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

2. Connection to the server.

FPWCSApi2 createSession, FPWCSApi2Session connectcode

FPWCSApi2SessionOptions object with the following parameters is passed to createSession() method

- URL of WCS server
- SIP parameters to make outgoing call and to receive incoming calls
- appKey of internal server-side application(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. Outgoing call.

FPWCSApi2Session createCall, FPWCSApi2Call callcode

The next parameters are passed to createCall() method:

- callee SIP username
- additional SIP INVITE parameters from string set by user

```
- (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;
    //used for only recv audio
//    options.localConstraints = [[FPWCSApi2MediaConstraints alloc] initWithAudio:NO video:NO];
//    options.remoteConstraints = [[FPWCSApi2MediaConstraints alloc] initWithAudio:YES video:NO];
    NSError *error;
    call = [session createCall:options error:&error];
    ...
    [call call];
    return call;
}
```

4.Receiving the event on incoming call

FPWCSApi2Session onIncomingCallCallbackcode

```
[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. Answering incoming call.

FPWCSApi2Call answercode

```
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. Call hold and retrieve.

FPWCSApi2Call hold, unholdcode

```
- (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.DTMF sending

FPWCSApi2Call sendDTMFcode

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

8. Outgoing call hangup.

FPWCSApi2Call hangupcode

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

9. Incoming call hangup.

FPWCSApi2Call hangupcode

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

[alert addAction:hangupButton];
```

10. Disconnection.

FPWCSApi2Session disconnectcode

```
- (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];
        }
        ...
    }
}
```