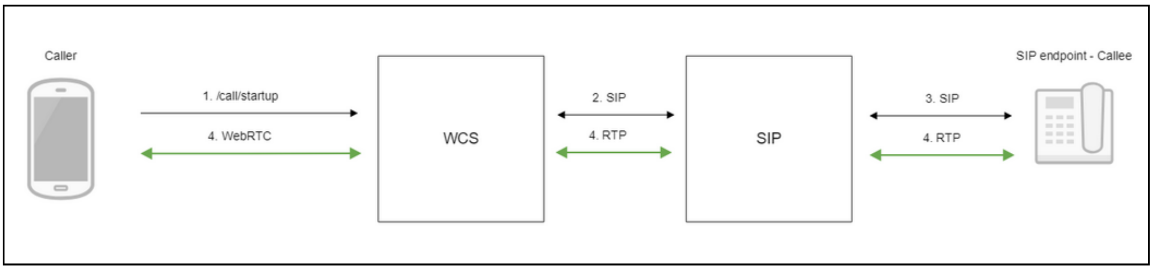
SIP calls using iOS SDK

Overview

SIP call on iOS devices can be made both from a browser and using the iOS SDK.

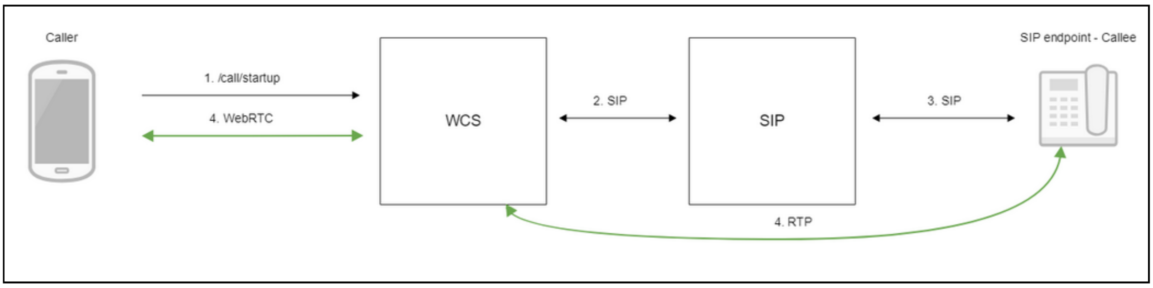
Operation flowchart

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



1. The iOS device begins a call
2. WCS connects to the SIP server
3. The SIP server connects to the SIP device that receives the call
4. The iOS device and the SIP device exchange audio and video streams

2. SIP server as a server to transfer calls only

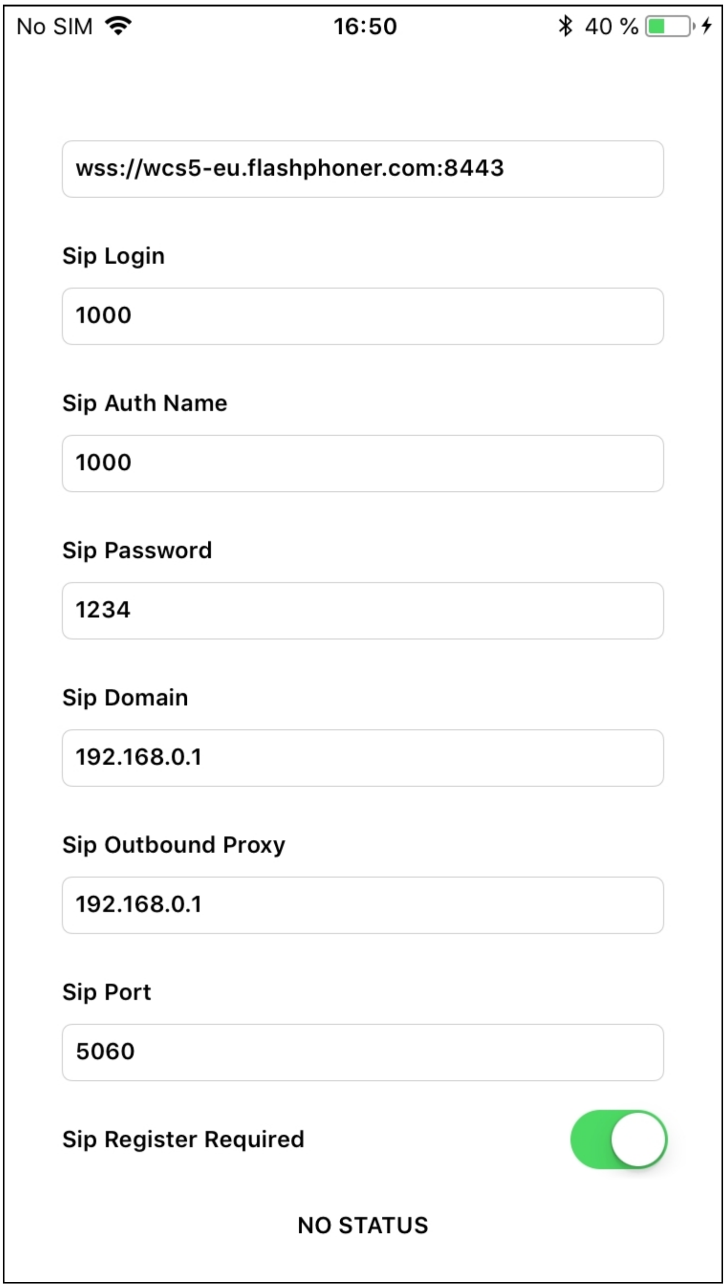


1. The iOS device begins a call
2. WCS connects to the SIP server
3. The SIP server connects to the SIP device that receives the call
4. The iOS device and the SIP device exchange audio and video streams

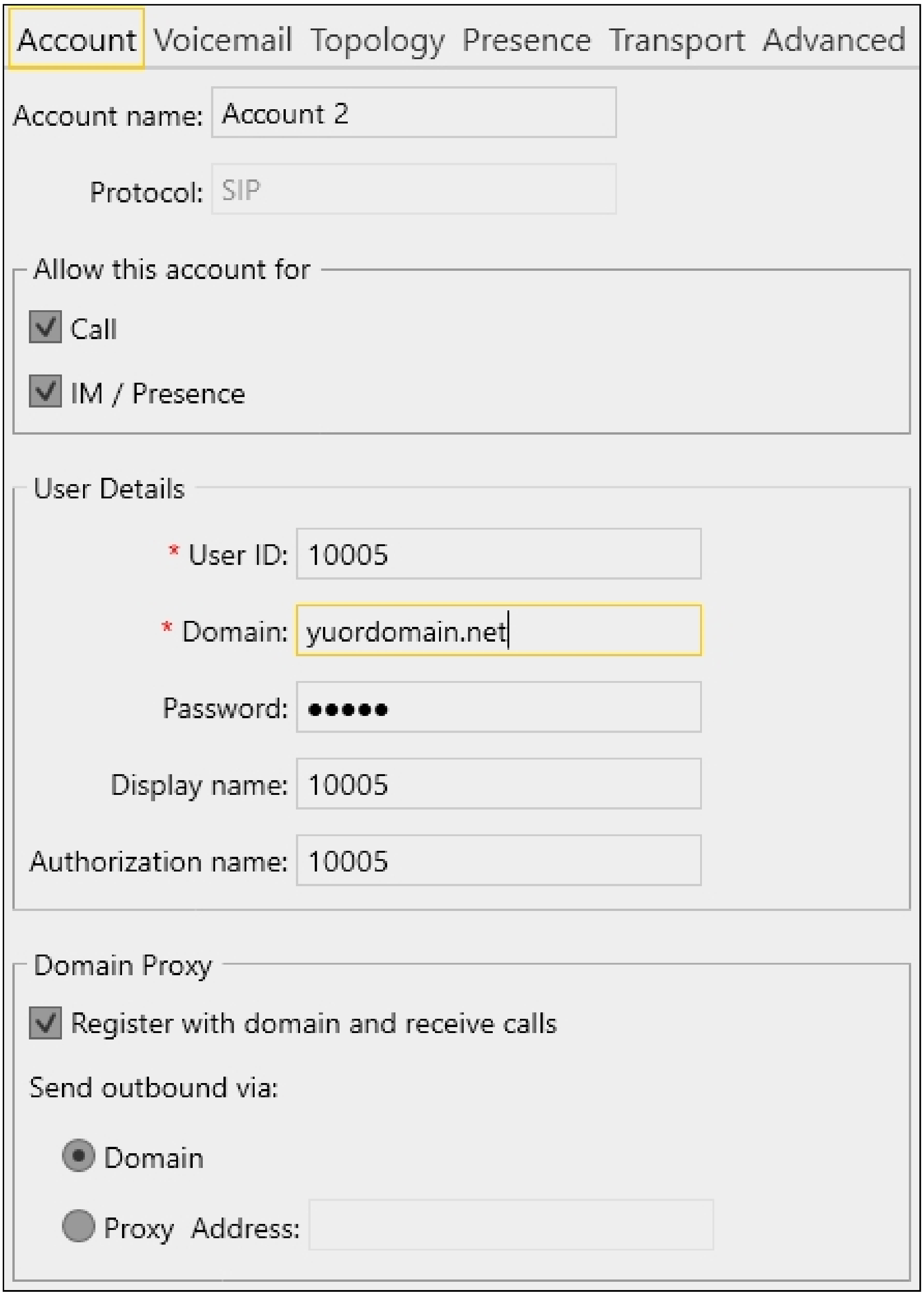
Testing

Making an outgoing call from iOS to a SIP device

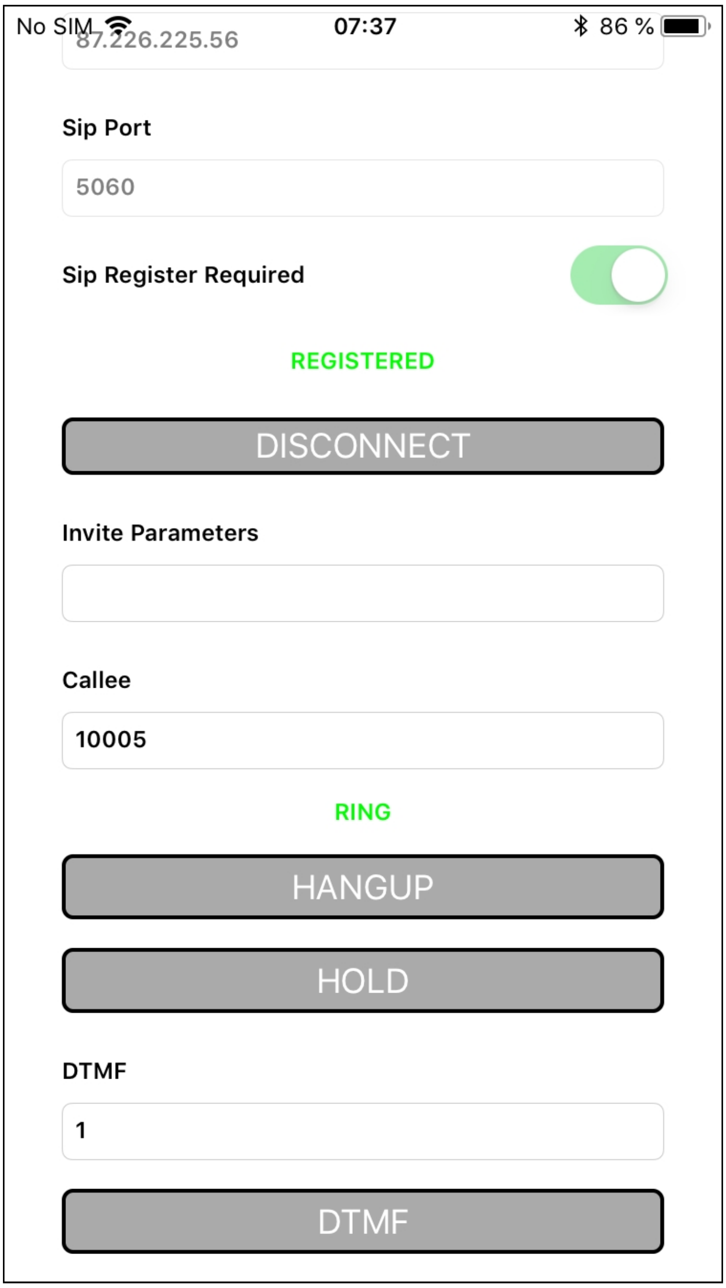
1. For the test we use:
2. two SIP accounts;
3. the Phone application to make a call;
4. a software phone to answer the call.
5. Build and install the Phone app to the iOS device. Start the app, enter the URL of the WCS server to connect to it via Websocket and the data of the SIP account making a call:



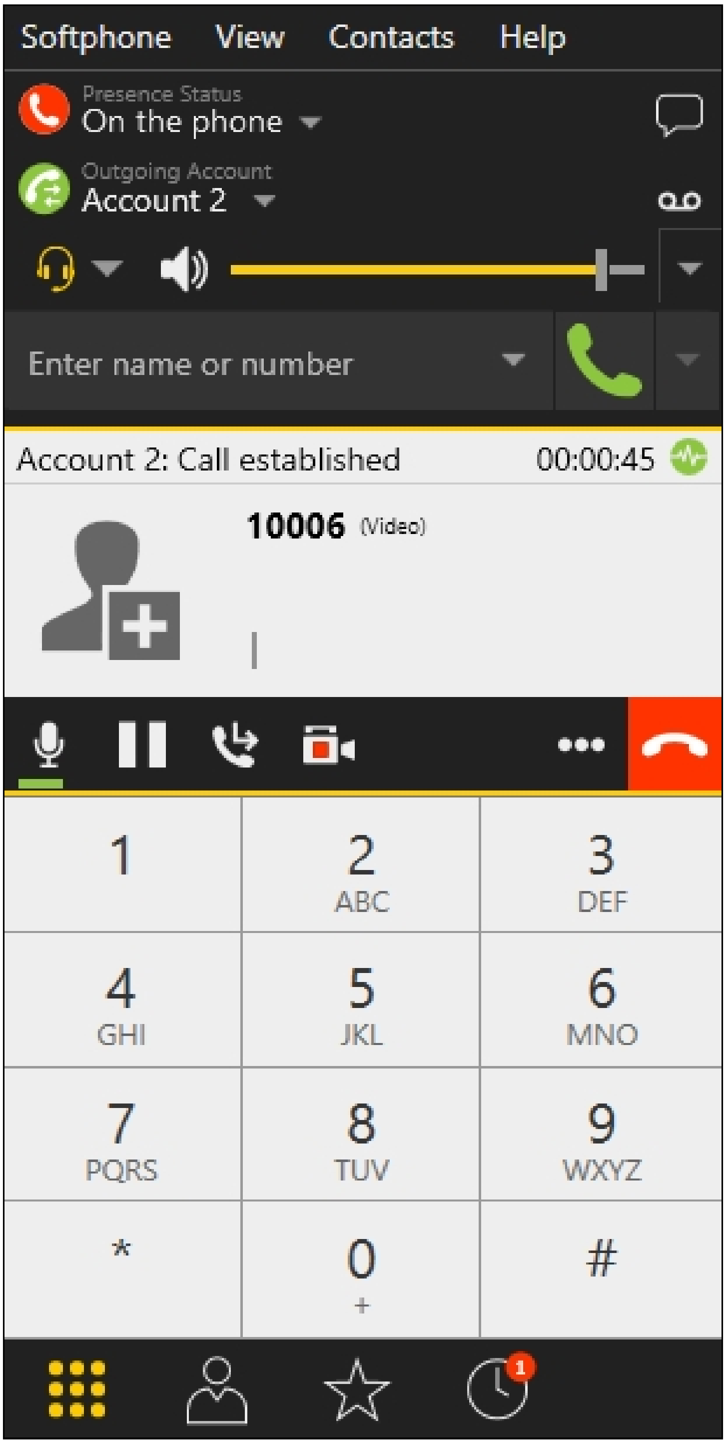
6. Run the softphone, enter the data of the SIP account that receives the call:



7. Tap the `Connect` button in the app, a connection will be established to the server. Then enter the identifier of the SIP account that receives the call and click the `Call` button:


No SIM
07:37
86 %
87.226.225.56
Sip Port
5060
Sip Register Required
REGISTERED
DISCONNECT
Invite Parameters
Callee
10005
RING
HANGUP
HOLD
DTMF
1
DTMF

8. Answer the call in the softphone by clicking the answer button:

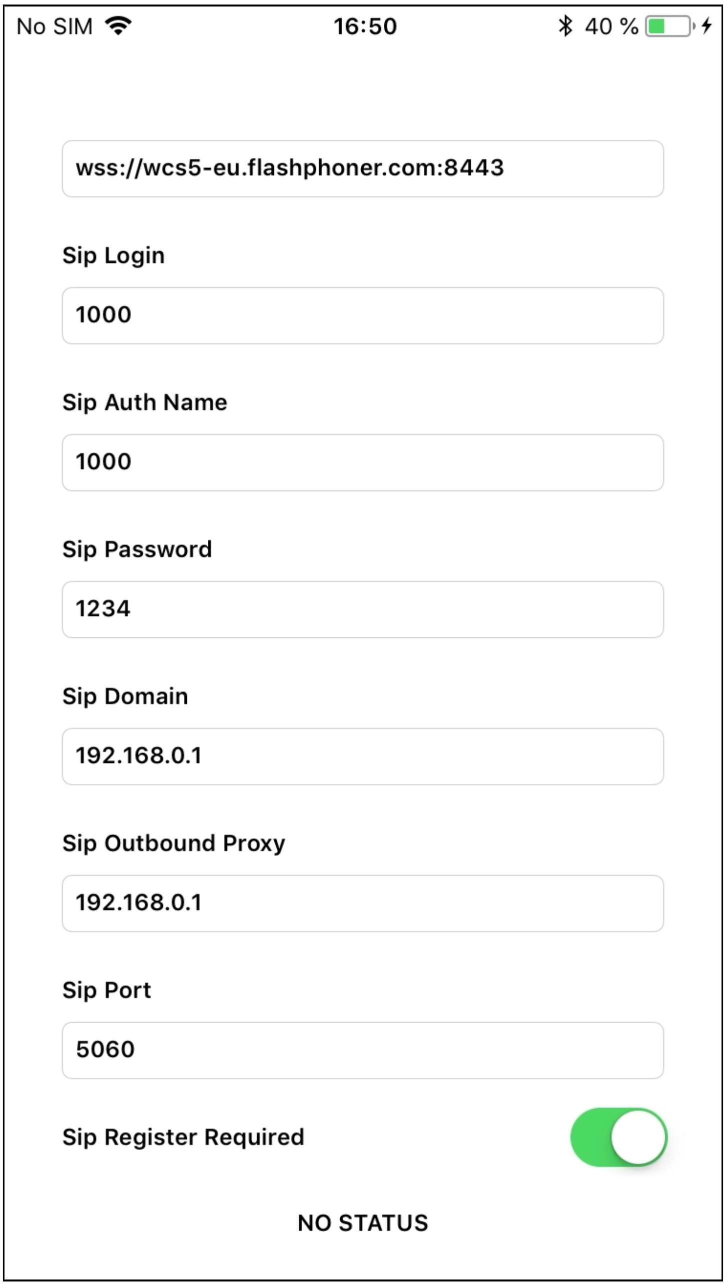


9. To terminate the call, tap the `Hangup` button in the application, or click the end call button in the softphone.

Receving an outgoing call from a SIP device to iOS

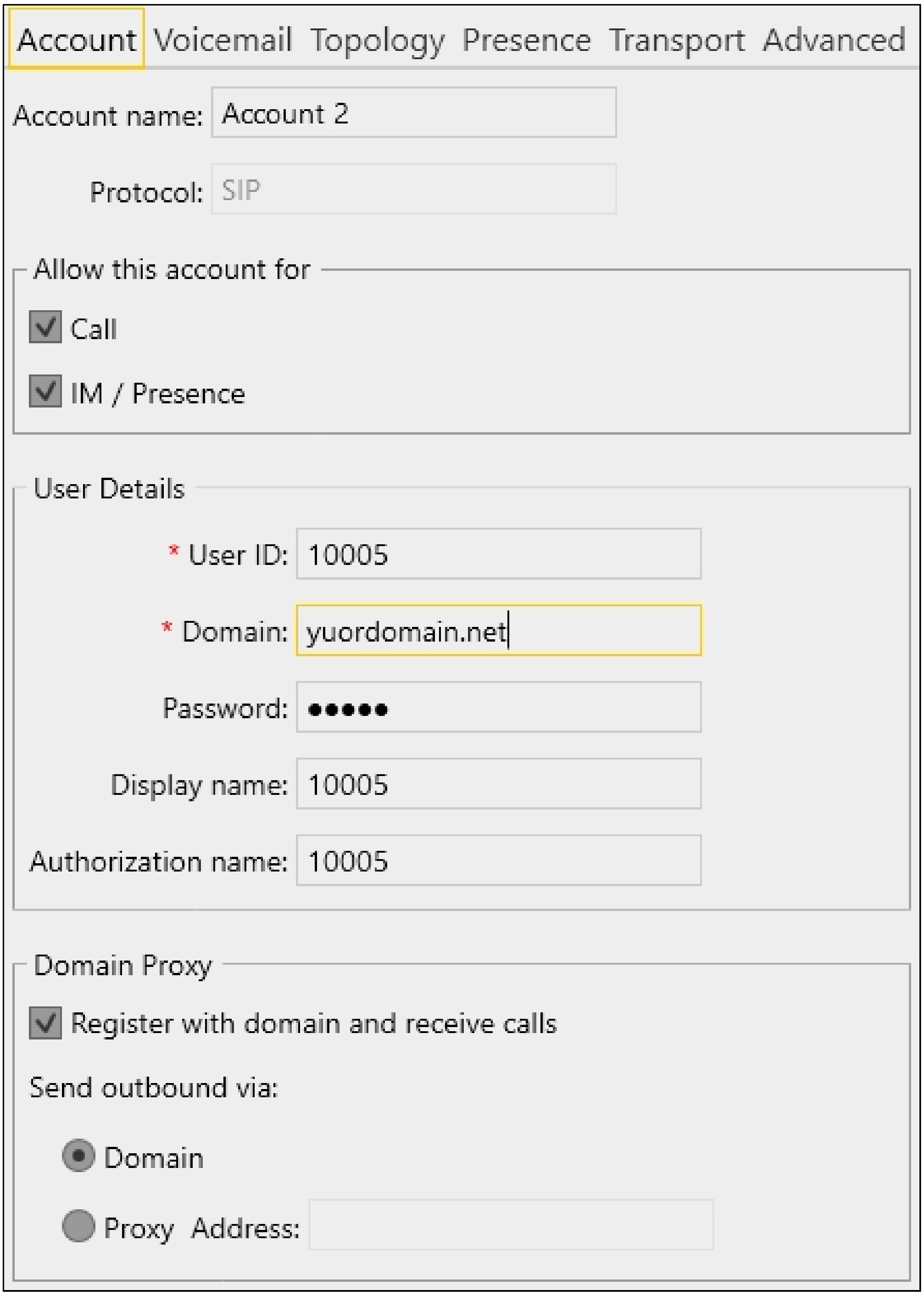
1. For the test we use:
2. two SIP accounts;

3. a softphone to make a call;
4. the Phone application to answer the call.
5. Build and install the Phone app to the iOS device. Start the app, enter the URL of the WCS server to connect via Secure Websocket and the data of the SIP account that receives the call:

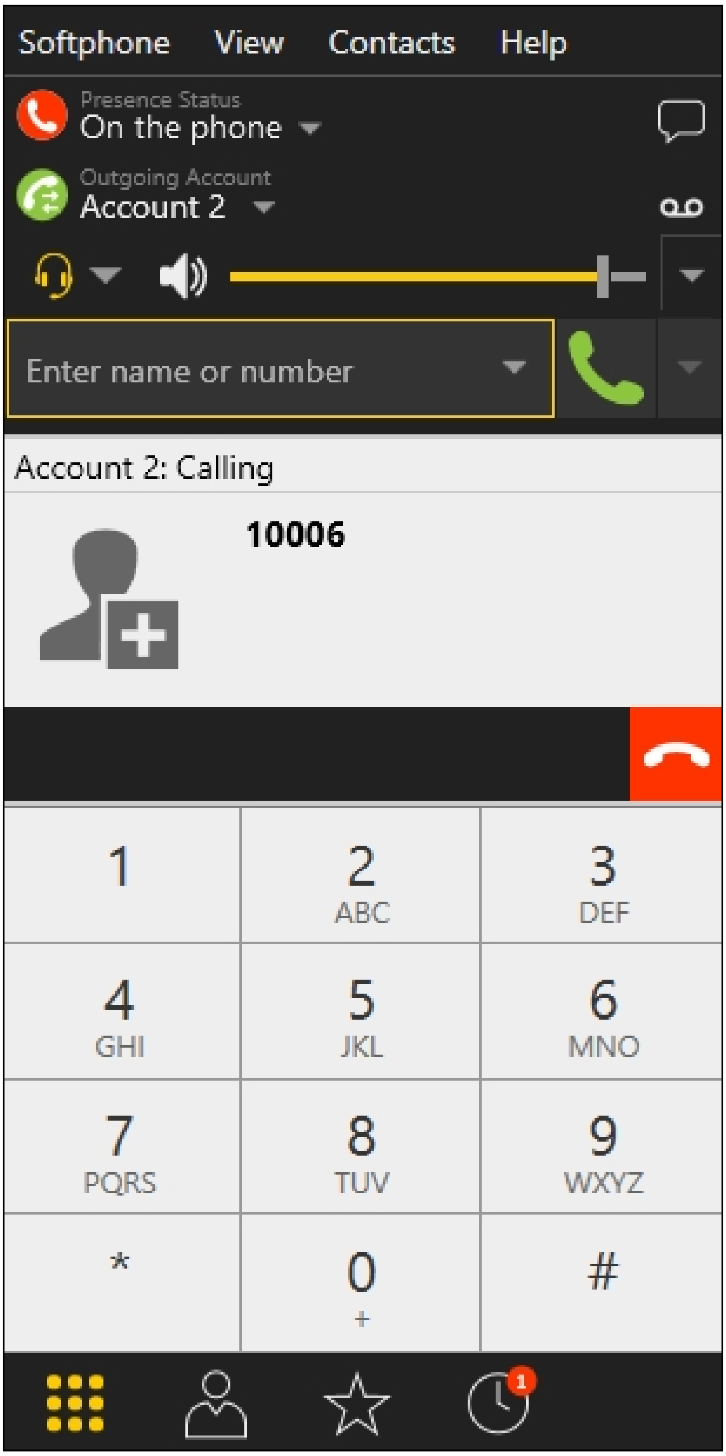


Tap the `Connect` button in the app to establish a connection to the WCS server

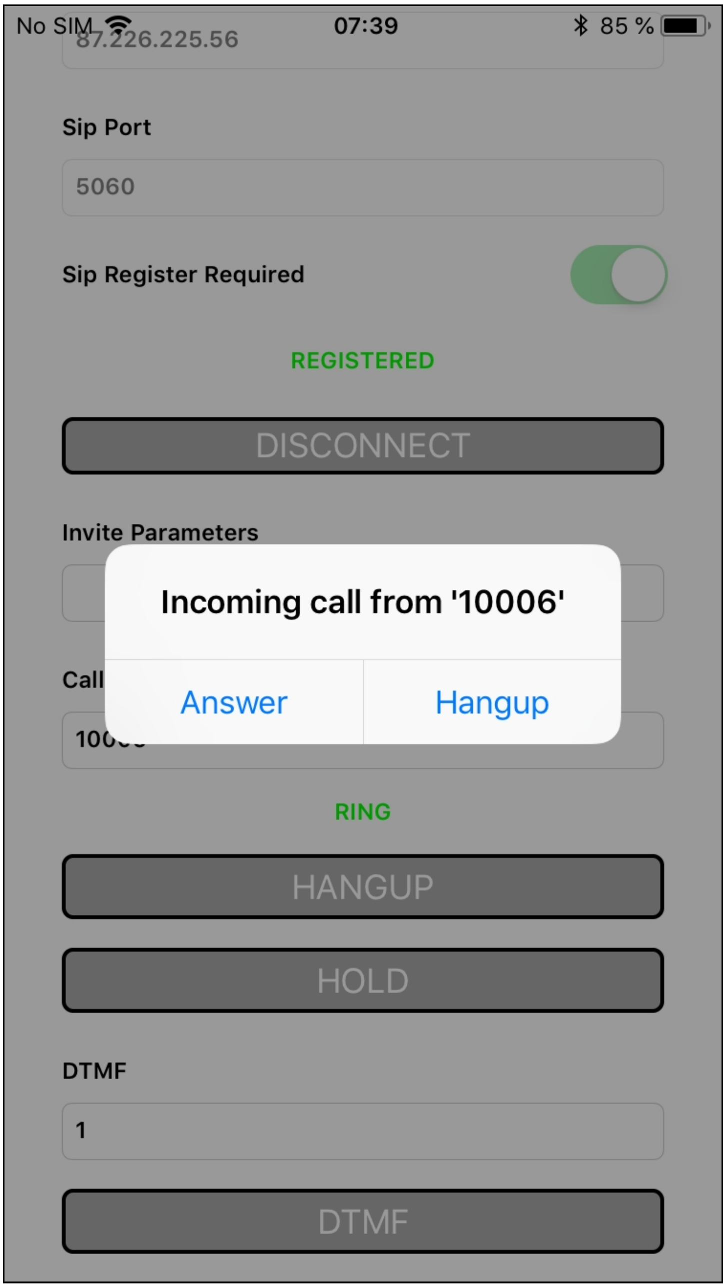
6. Run the software phone and enter the data of the SIP account making the call:



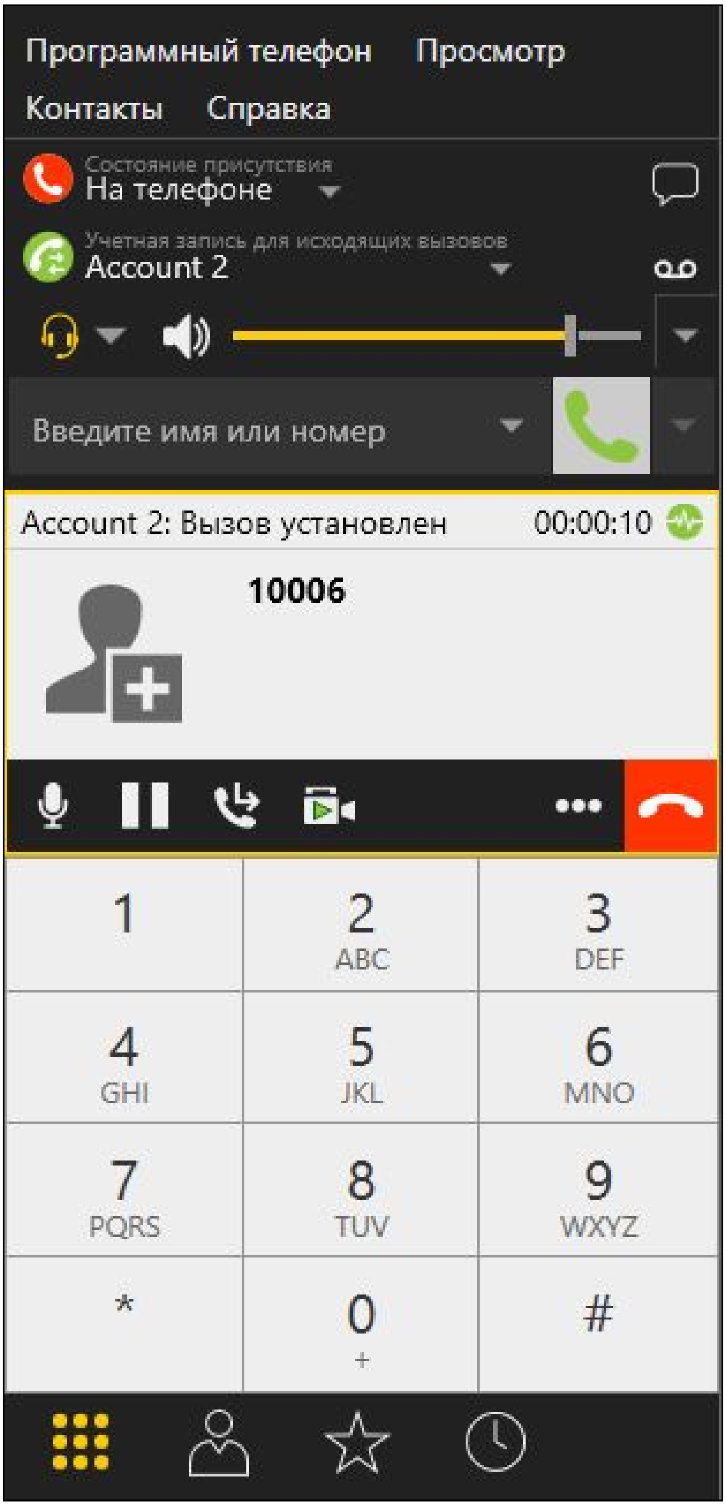
7. In the softphone enter the identifier of the SIP account that receives the call and click the call button:


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

8. Answer the call in the application by tapping `Answer`:



9. In the softphone make sure the call has started:

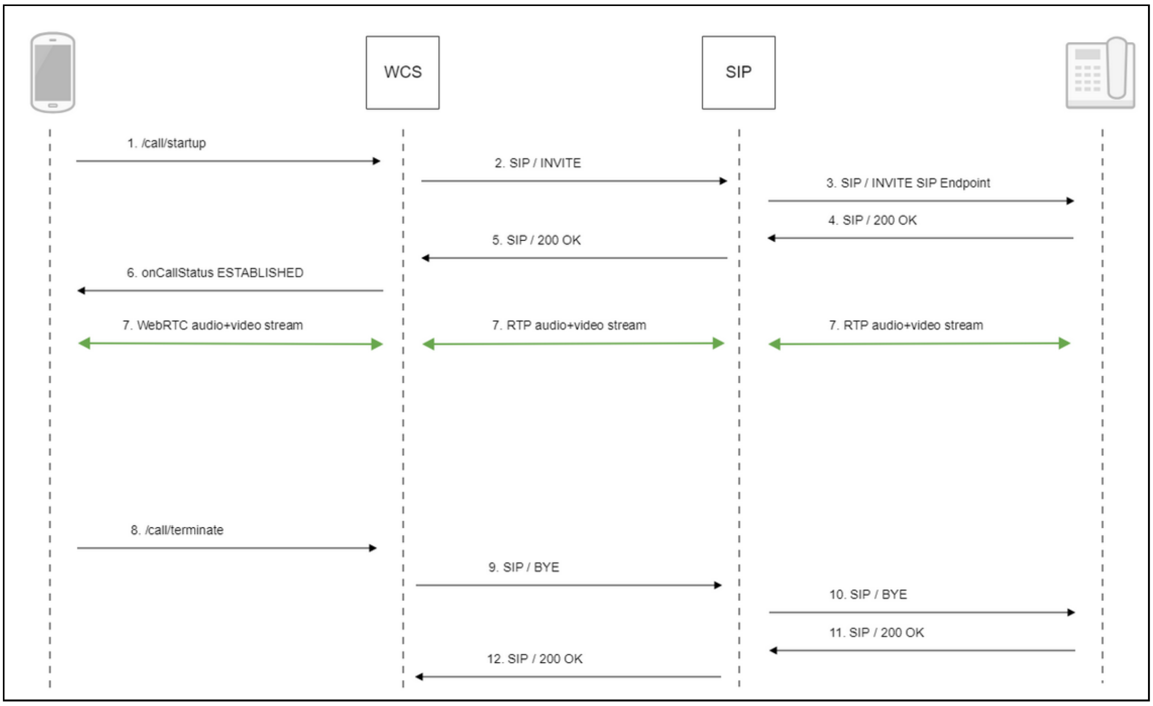


10. To terminate the call, tap the Hangup button in the app, or click the end call button in the softphone.

Call flow

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

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



1. Creating a call
   `FPWCSApi2Session.createCall()`, `FPWCSApi2Call.call()` code

```
- (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;
}
```

2. Sending `SIP INVITE` to the SIP server

3. Sending `SIP INVITE` to the SIP device

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

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

8. Terminating the call
   `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];
        }
        ...
    }
}
```

9. Sending `SIP BYE` to the SIP server

10. Sending `SIP BYE` to the SIP device

11. Receiving a confirmation from the SIP device

12. Receiving a confirmation from the SIP server

Known issues

1. It's impossible to make a SIP call if `SIP Login` and `SIP Authentification name` fields contain unappropriate characters

Symptoms

SIP call stucks in `PENDING` state

Solution

According to RFC3261, `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 allowed by the specification

```
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'
```