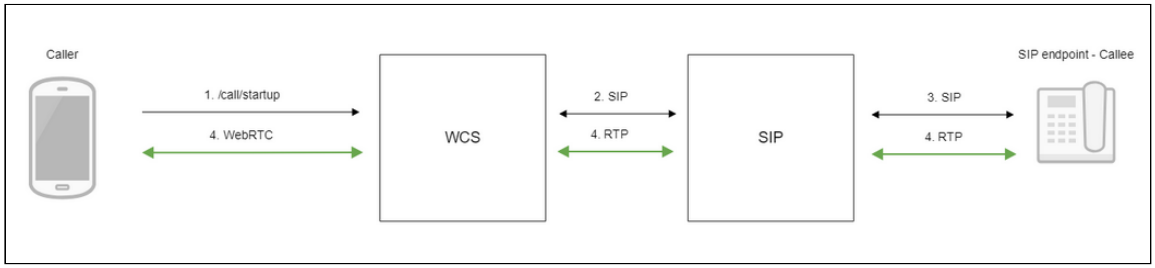
SIP calls using Android SDK

Overview

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

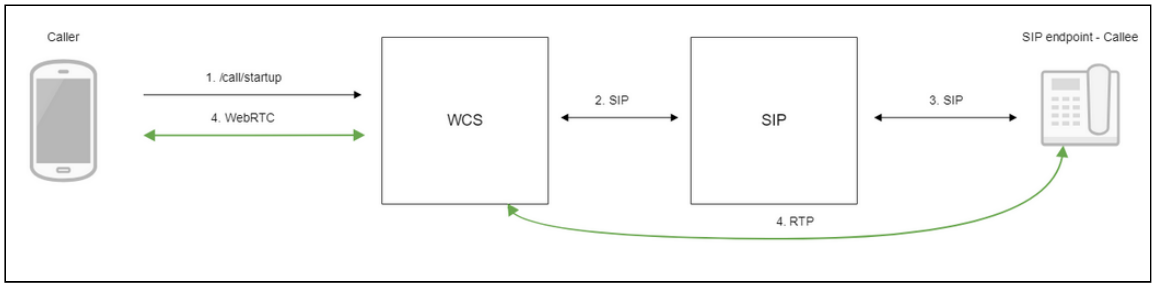
Operation flowchart

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



1. The Android 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 Android device and the SIP device exchange audio and video streams

2. SIP server as a server to transfer calls only

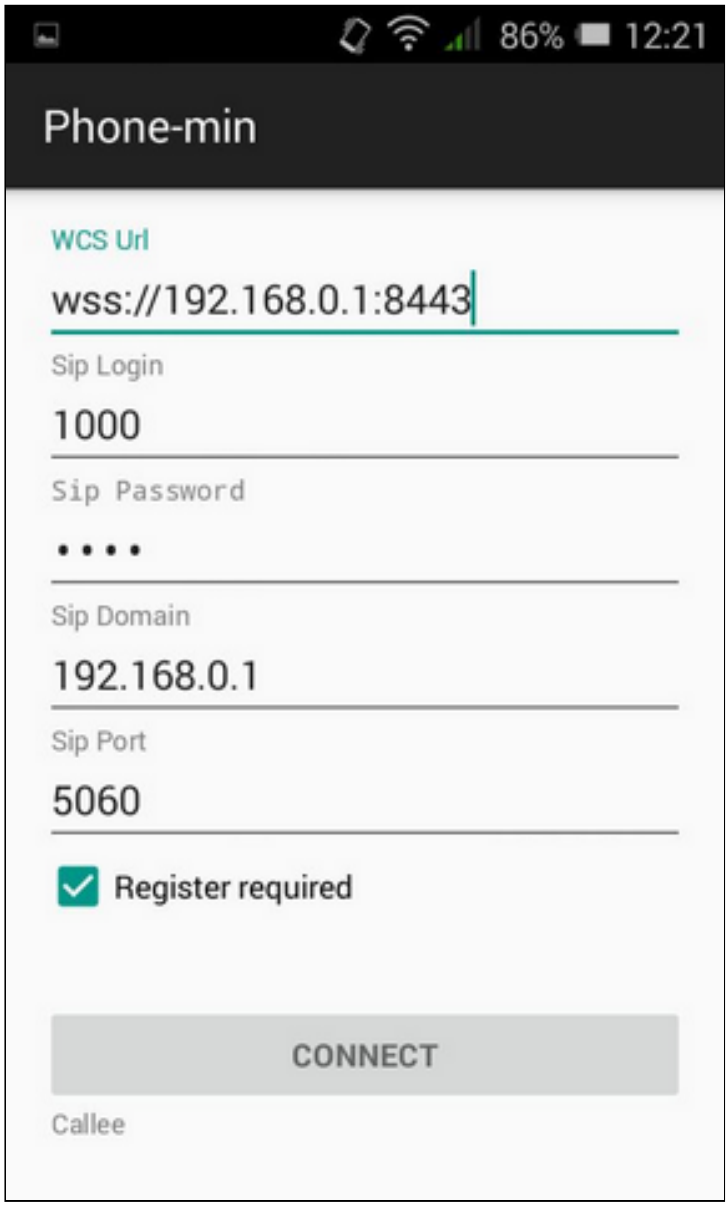


1. The Android device begins a call
2. WCS connects to the SIP server
3. The SIP server connects to the SIP device that receives the call
4. The Android device and the SIP device exchange audio and video streams

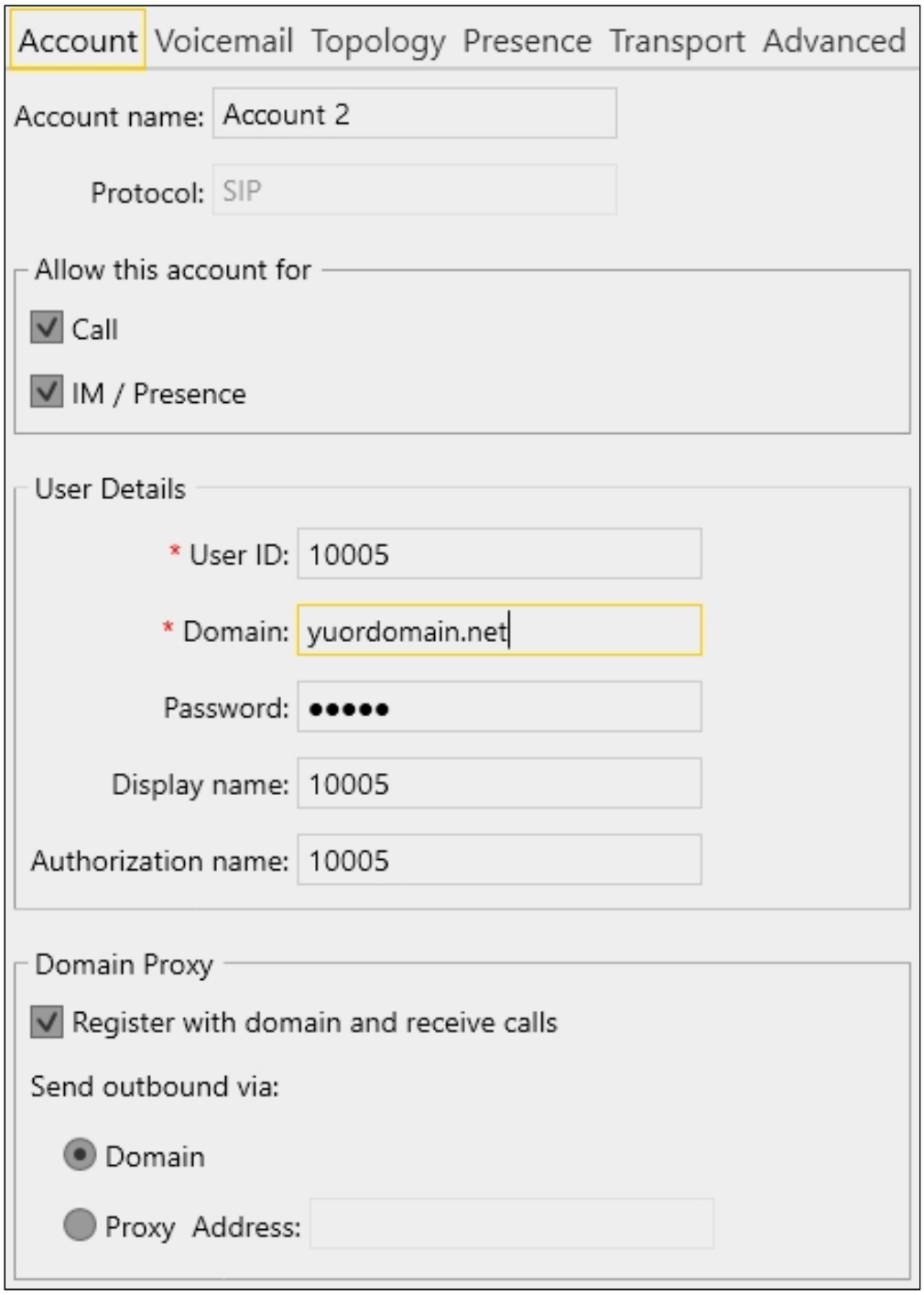
Testing

Making an outgoing call from Android to a SIP device

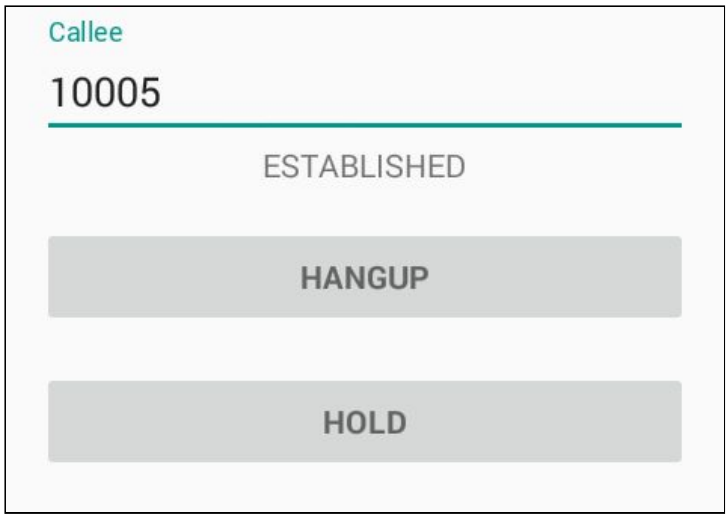
1. For the test we use:
2. two SIP accounts;
3. the Phone application to make a call;
4. a software phone to answer the call.
5. Install the Phone app to the Android device. Start the app, enter the URL of the WCS server to connect to it via Websocket and the data of the SIP account making a call:



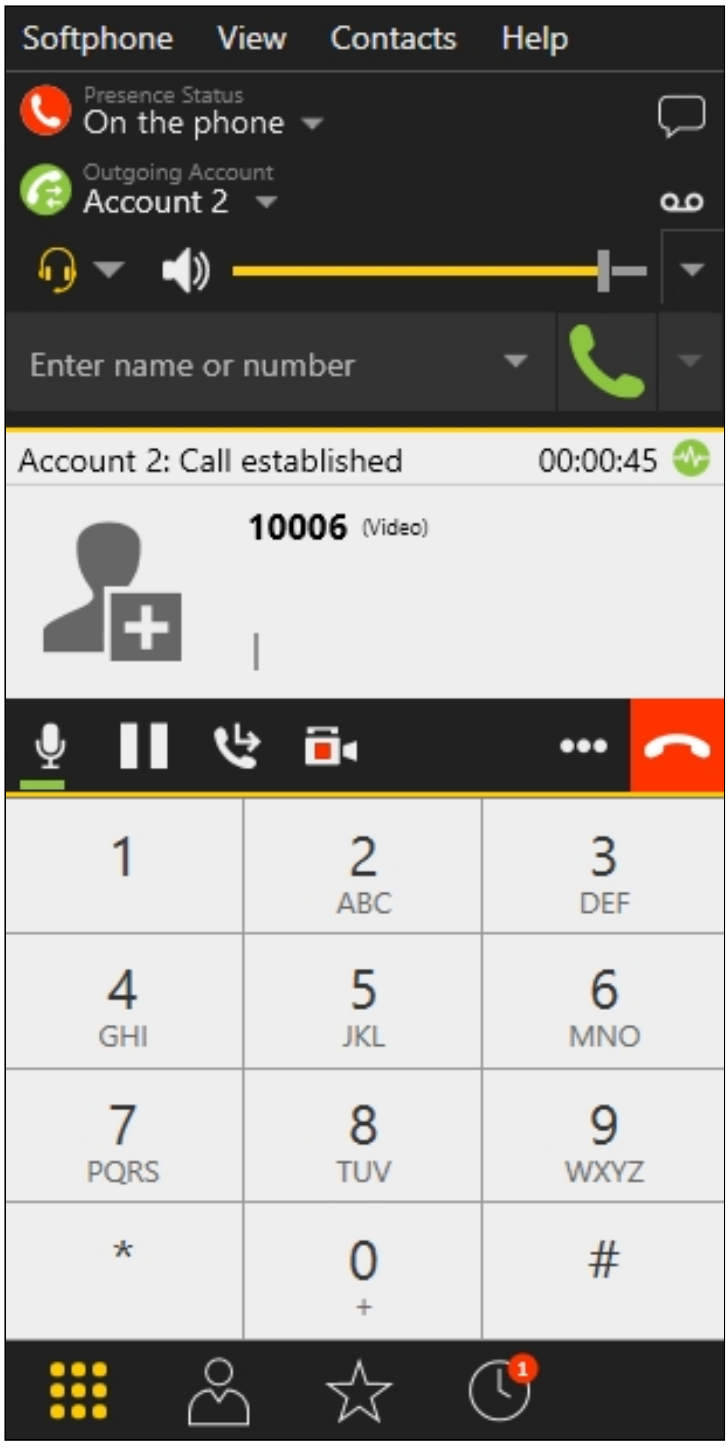
6. Run the softphone, enter the data of the SIP account that receives the call:



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:


Callee
10005
ESTABLISHED
HANGUP
HOLD

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

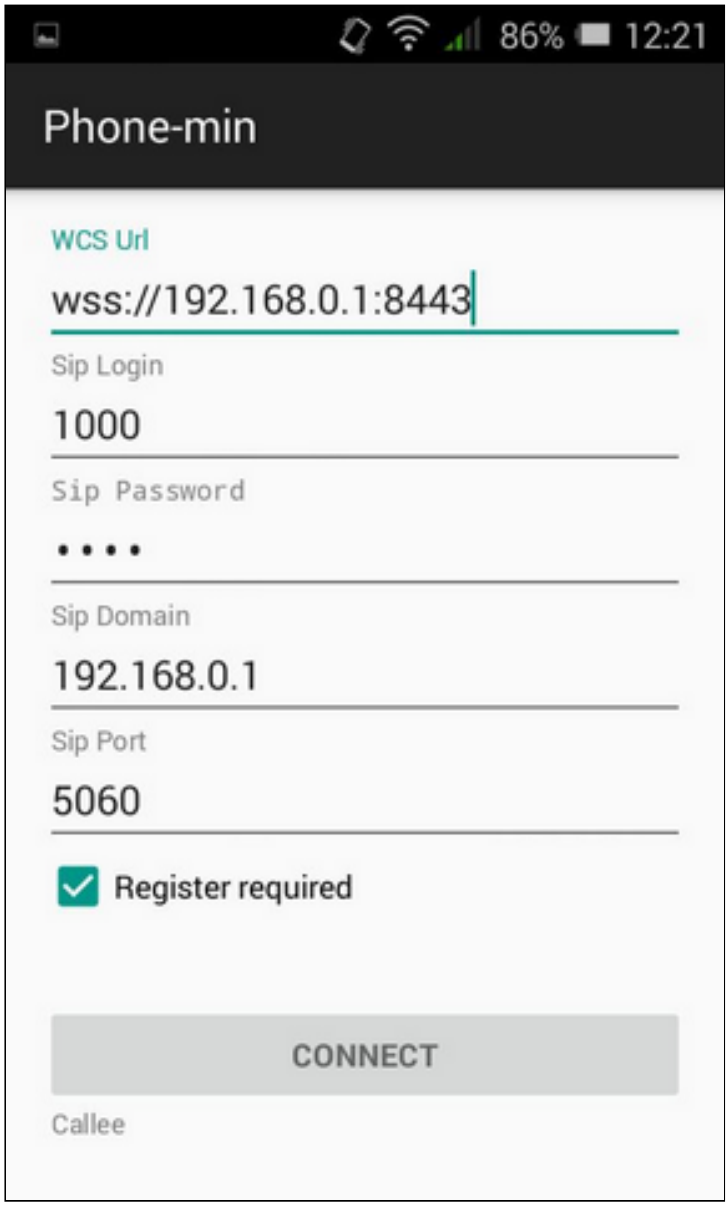


9. To terminate the call, tap the Hangup button in the application, or click the end call button in the softphone.

Receving an incoming call from a SIP device to Android

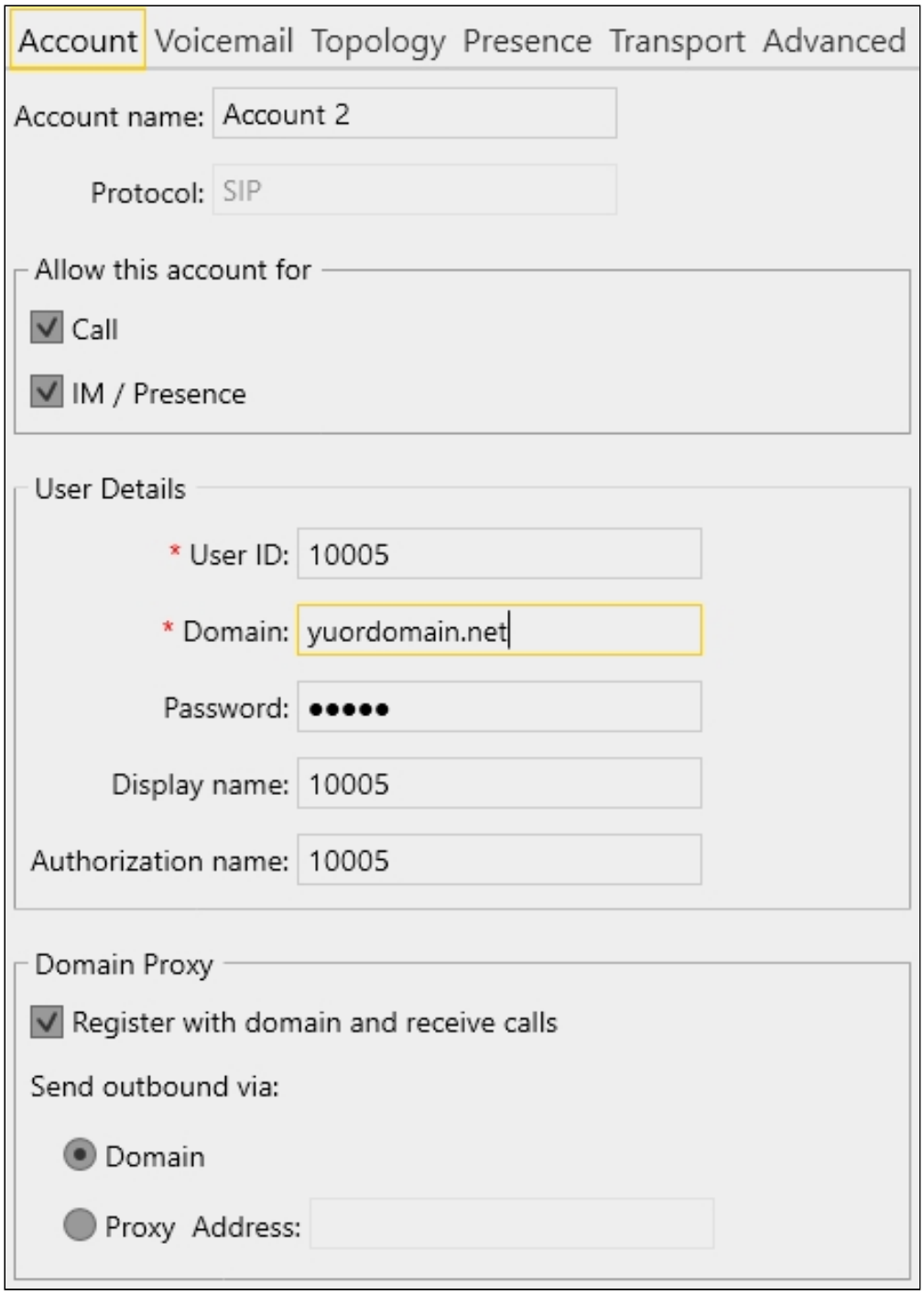
1. For the test we use:
2. two SIP accounts;

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

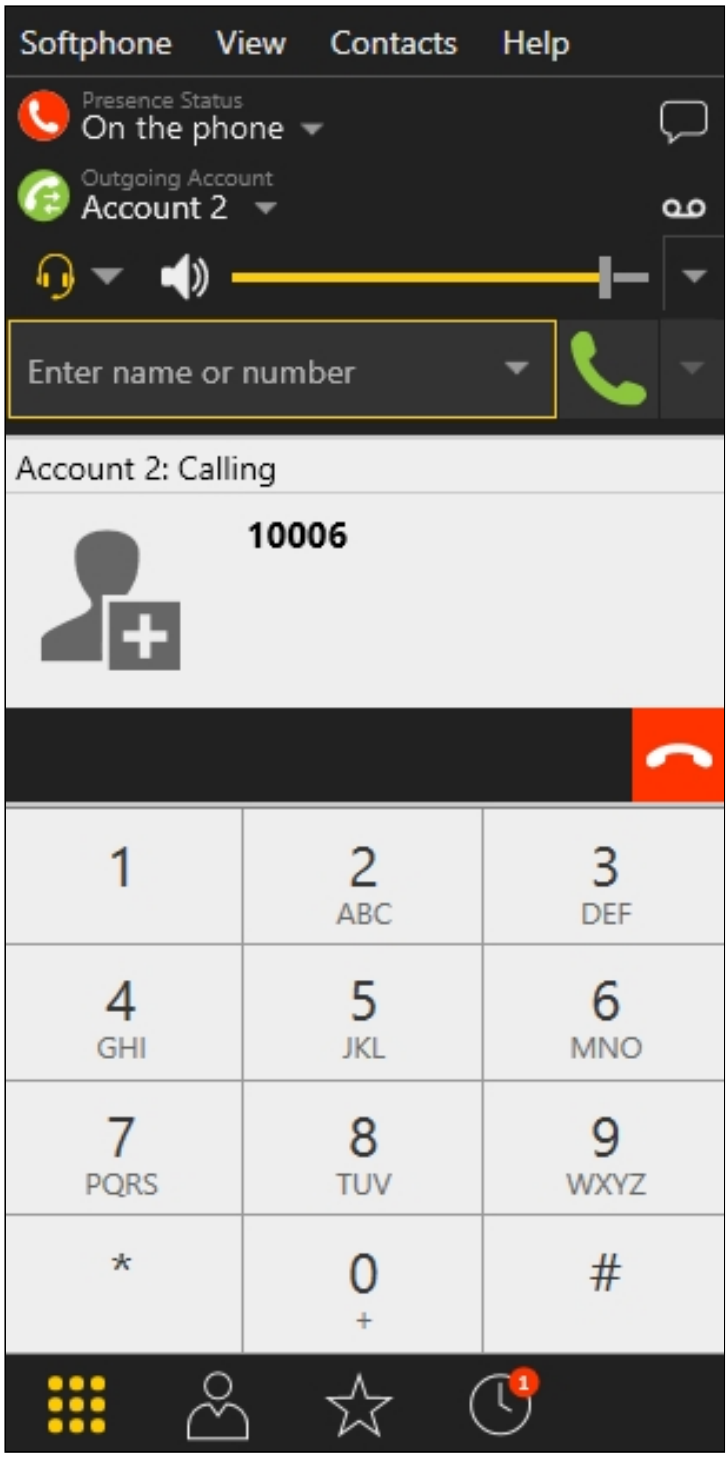


Tap the `Connect"` button in the app to establish a connection to the WCS server.

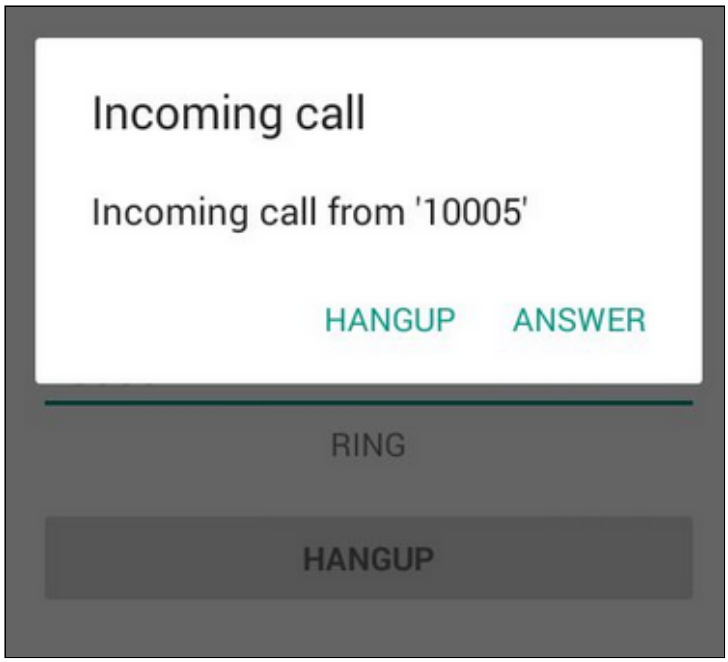
6. Run the software phone and enter the data of the SIP account making the call:

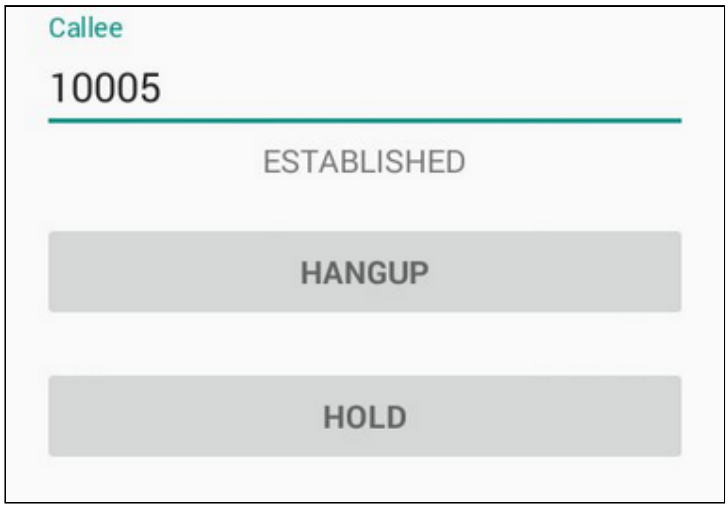


7. In the softphone enter the identifier of the SIP account that receives the call and click the call button:

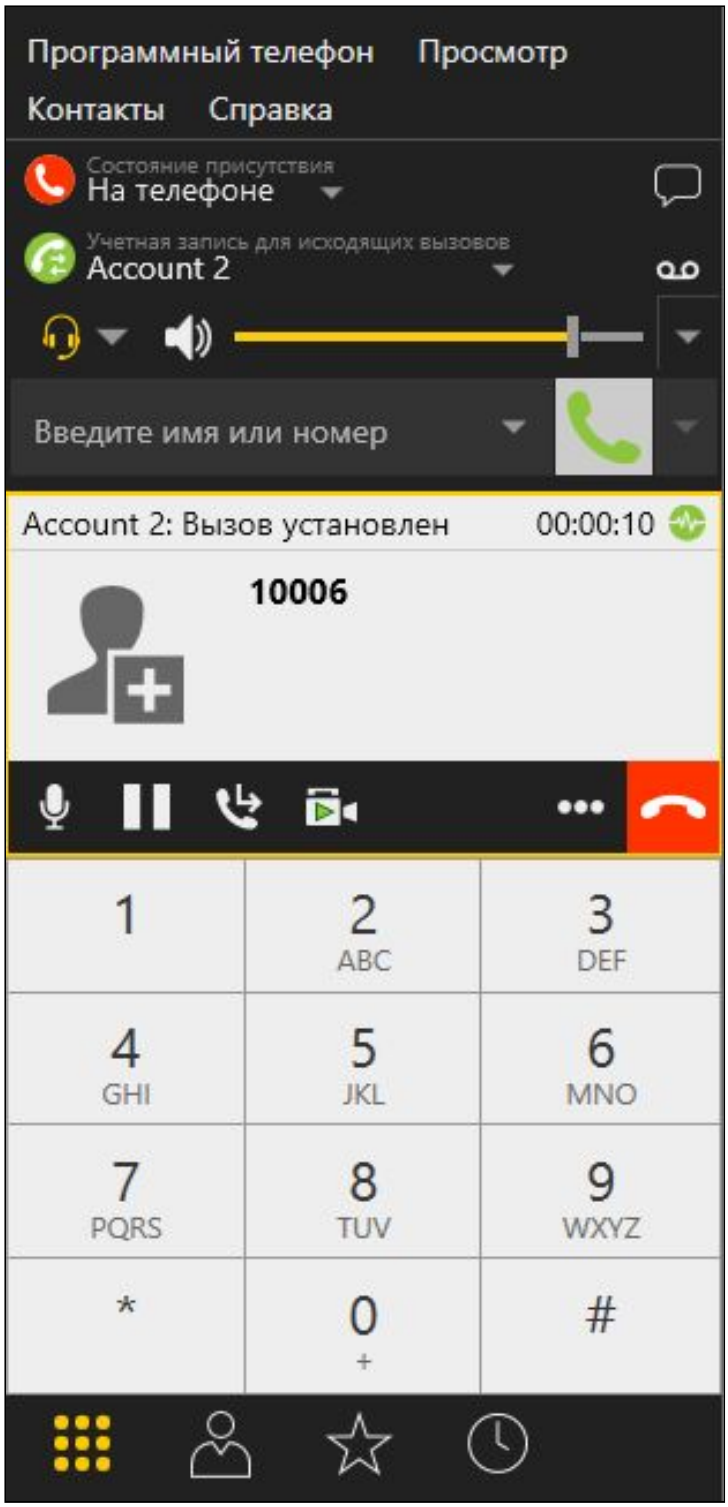

Softphone
View
Contacts
Help
Presence Status
On the phone
Outgoing Account
Account 2
Enter name or number
Account 2: Calling
10006
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
*
0 +
#

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





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

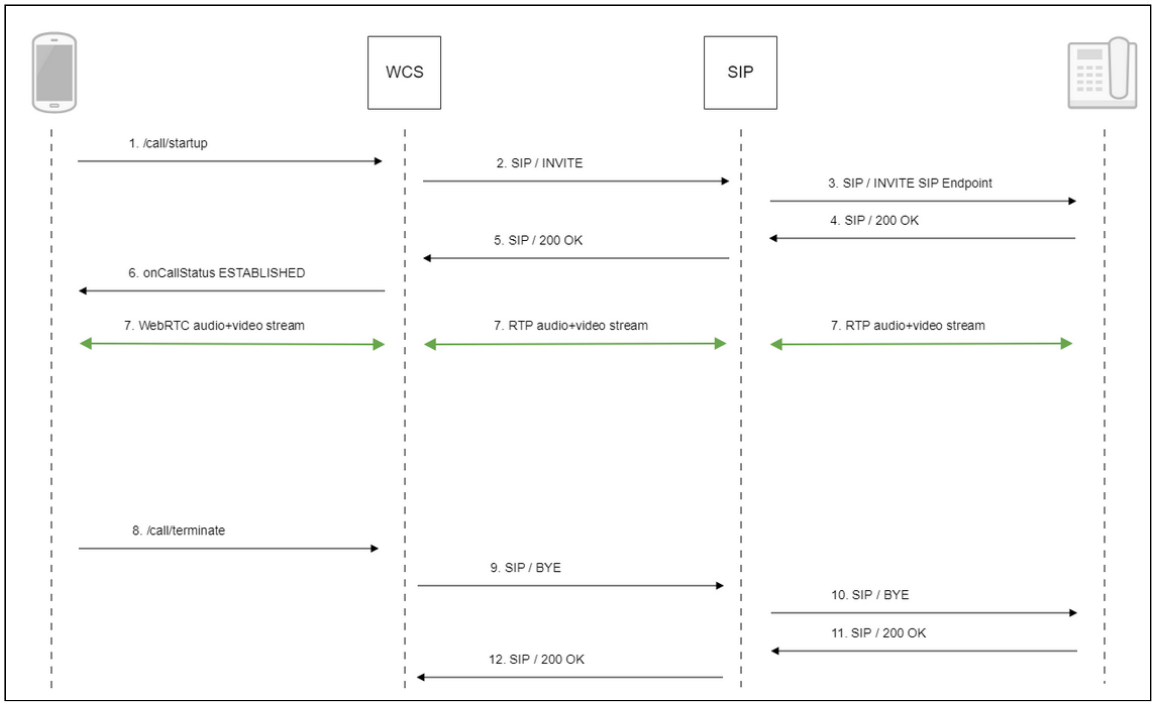


10. To terminate the call, tap the Hangup button in the app, or click the end call button in the softphone.

Call flow

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

PhoneMinActivity.java



1. Creating a call
   `Session.createCall()`, `Call.call()` code

```
CallOptions callOptions = new
CallOptions(mCalleeView.getText().toString());
AudioConstraints audioConstraints =
callOptions.getConstraints().getAudioConstraints();
MediaConstraints mediaConstraints =
audioConstraints.getMediaConstraints();
...
call = session.createCall(callOptions);
call.on(callStatusEvent);
/**
 * Make the outgoing call
 */
call.call();
Log.i(TAG, "Permission has been granted by user");
```

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
   `Call.hangup()` code

```
if (mCallButton.getTag() == null ||
Integer.valueOf(R.string.action_call).equals(mCallButton.getTag())) {
    if ("".equals(mCalleeView.getText().toString())) {
        return;
    }
    ...
} else {
    mCallButton.setEnabled(false);
    call.hangup();
    call = null;
}
```

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'
```