

# Android Click to Call

## Пример Click to Call для Android

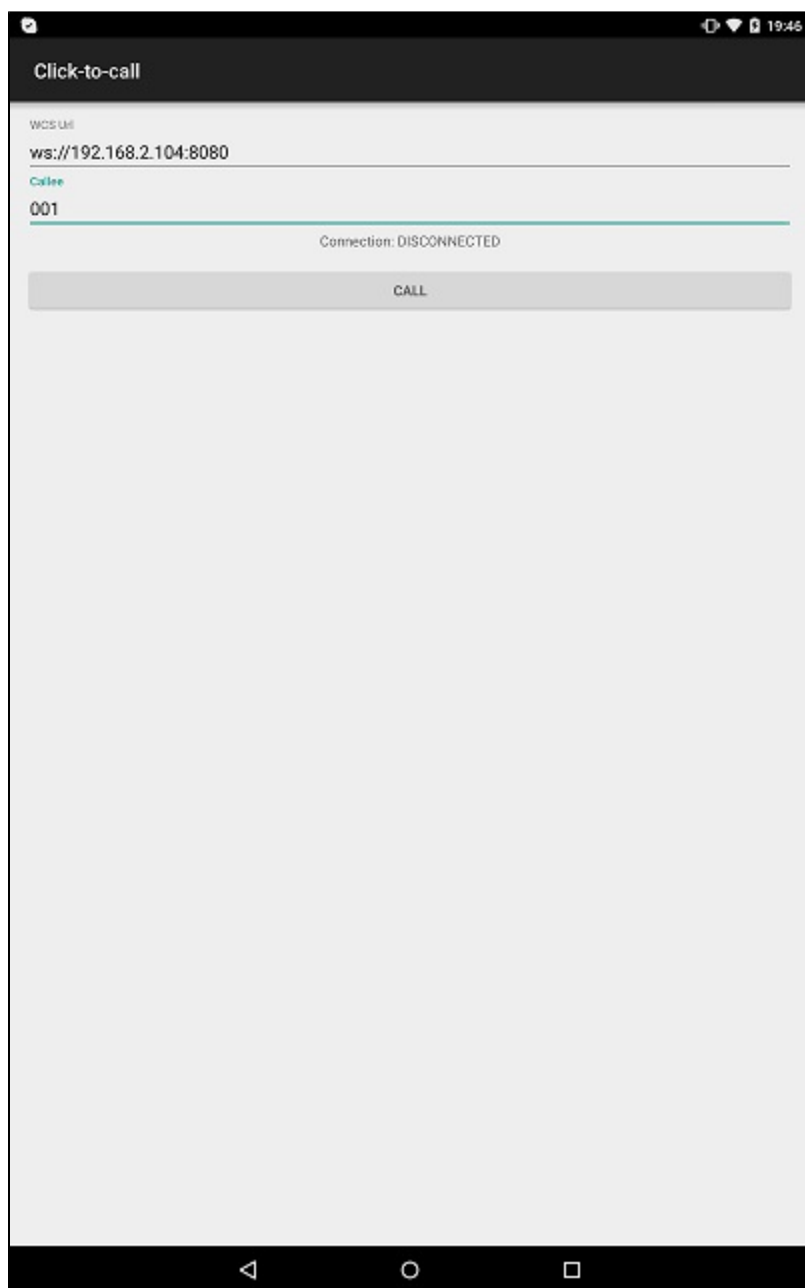
Данный пример позволяет сделать аудиозвонок одним кликом кнопки, используя аккаунт заданный в файле настроек сервера

*/usr/local/FlashphonerWebCallServer/conf/apps/click-to-call/accounts.xml*

На скриншоте ниже представлен пример после завершения звонка и закрытия соединения с сервером.

Поля ввода, необходимые для установления соединения и звонка

- 'WCS URL', где 192.168.2.104 - адрес WCS-сервера
- 'Callee', где 001 - SIP логин вызываемого пользователя



## Работа с кодом примера

Для разбора кода возьмем класс [ClickToCallActivity.java](#) примера click-to-call, который доступен для скачивания в соответствующей сборке [1.0.1.38](#).

### 1. Инициализация API.

Flashphoner.init() [код](#)

```
Flashphoner.init(this);
```

При инициализации методу init() передается объект Context.

### 2. Создание сессии.

Flashphoner.createSession() [код](#)

Методу передается объект SessionOptions с URL WCS-сервера.

```
SessionOptions sessionOptions = new SessionOptions(mwcsUrlView.getText().toString());  
session = Flashphoner.createSession(sessionOptions);
```

### 3. Подключение к серверу.

Session.connect() [код](#)

Методу передается объект Connection с параметрами:

- appKey внутреннего серверного приложения 'clickToCallApp'.

```
Connection connection = new Connection();  
connection.setAppKey("clickToCallApp");  
/**  
 * Connect to WCS server  
 */  
session.connect(connection);
```

### 4. Получение от сервера события, подтверждающего успешное соединение.

Session.onConnected(), Session.createCall() [код](#)

При получении данного события создается звонок методом Session.createCall(). Методу передается SIP-номер вызываемого абонента.

```

@Override
public void onConnected(final Connection connection) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            mCallButton.setText(R.string.action_hangup);
            mCallButton.setTag(R.string.action_hangup);
            mCallButton.setEnabled(true);
            mCallStatus.setText("Connection: " + connection.getStatus());

            /**
             * Pass 'callee' to the callOptions and create a new call object
             */
            CallOptions callOptions = new CallOptions(mCalleeView.getText().toString());
            call = session.createCall(callOptions);
            call.on(new CallStatusEvent() {
                ...
            });

            ActivityCompat.requestPermissions(ClickToCallActivity.this,
                new String[]{Manifest.permission.RECORD_AUDIO},
                CALL_REQUEST_CODE);
            ...
        }
    });
}

```

#### 5. Совершение исходящего звонка.

Call.call() [код](#)

```

case CALL_REQUEST_CODE: {
    if (grantResults.length == 0 ||
        grantResults[0] != PackageManager.PERMISSION_GRANTED) {
        mCallButton.setEnabled(false);
        session.disconnect();
        Log.i(TAG, "Permission has been denied by user");
    } else {
        /**
         * Make the outgoing call
         */
        call.call();
        Log.i(TAG, "Permission has been granted by user");
    }
}

```

#### 6. Закрытие соединения.

Session.disconnect() [код](#)

```

mCallButton.setEnabled(false);
session.disconnect();

```

#### 7. Получение события, подтверждающего разъединение.

session.onDisconnection() [код](#)

```
@Override
public void onDisconnection(final Connection connection) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            mCallButton.setText(R.string.action_call);
            mCallButton.setTag(R.string.action_call);
            mCallButton.setEnabled(true);
            mCallStatus.setText("Connection: " + connection.getStatus());
        }
    });
}
```