

# Android MCU Client

## Пример клиента на Android для участника многоточечной конференции

Данный пример может использоваться для организации многоточечной видео конференции (MCU) на Web Call Server. Каждый участник такой конференции может публиковать WebRTC-поток и воспроизводить микшированный поток с аудио и видео других участников и собственным видео (без собственного аудио).

Для работы примера требуются следующие настройки в файле настроек [flashphoner.properties](#) WCS-сервера

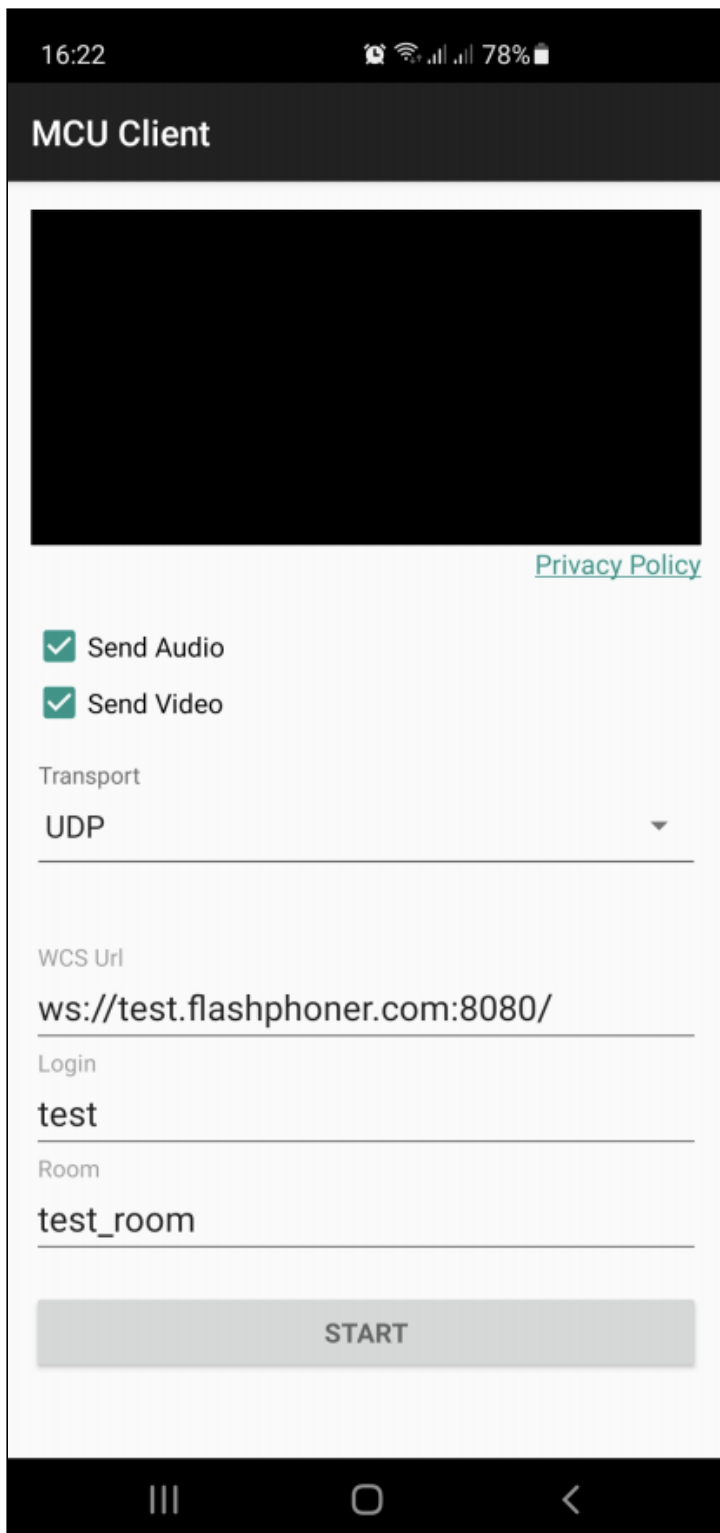
```
mixer_auto_start=true  
mixer_mcu_audio=true  
mixer_mcu_video=true
```

При подключении нового участника, использующего данный клиент, к конференции

- публикуется поток с видео участника и именем `participantName` + `#` + `roomName`, например `user#testroom`
- этот поток добавляется к микшеру с именем `roomName` (если такой микшер еще не существует, то он создается)
- публикуется выходной поток микшера с именем `roomName` + `-` + `participantName` + `roomName`, который содержит видео всех участников (включая данного) и аудио только от других участников, и начинается воспроизведение этого потока, например `testroom-user#testroom`

Поля ввода

- `WCS URL`, где `test.flashphoner.com` - адрес WCS-сервера
- `Login` - имя пользователя
- `Room` - имя комнаты
- `Transport` - выбор WebRTC транспорта
- `Send Audio` - переключатель, разрешающий/запрещающий публикацию аудио
- `Send Video` - переключатель, разрешающий/запрещающий публикацию видео



## Работа с кодом примера

Для разбора кода возьмем класс [McuClientActivity.java](#) примера `mcu-client`, который доступен для скачивания в соответствующей сборке [1.1.0.24](#).

## 1. Инициализация API

`Flashphoner.init()` [code](#)

При инициализации методу `init()` передается объект `Context`.

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

## 2. Создание сессии

`Flashphoner.createSession()` [code](#)

Методу передается объект `SessionOptions` со следующими параметрами

- URL WCS-сервера
- `SurfaceViewRenderer remoteRenderer`, который будет использоваться для отображения воспроизводимого потока

```
sessionOptions = new SessionOptions(mWcsUrlView.getText().toString());
sessionOptions.setRemoteRenderer(remoteRender);

/**
 * Session for connection to WCS server is created with method
 * createSession().
 */
session = Flashphoner.createSession(sessionOptions);
```

## 3. Подключение к серверу

`Session.connect()` [code](#)

```
session.connect(new Connection());
```

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

`session.onConnected()` [code](#)

```
@Override
public void onConnected(final Connection connection) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            mStatusView.setText(connection.getStatus());
            ...
        }
    });
}
```

```
});  
};
```

## 5. Создание потока

`Session.createStream()` [code](#)

```
StreamOptions streamOptions = new StreamOptions(publishStreamName);  
Constraints constraints = getConstraints();  
streamOptions.setConstraints(constraints);  
streamOptions.setTransport(Transport.valueOf(mTransportOutput.getSpinner().getSe  
  
/**  
 * Stream is created with method Session.createStream().  
 */  
publishStream = session.createStream(streamOptions);
```

## 6. Запрос прав на публикацию потока

`ActivityCompat.requestPermissions()` [code](#)

```
@Override  
public void onConnected(final Connection connection) {  
    runOnUiThread(new Runnable() {  
        @Override  
        public void run() {  
            ...  
            ActivityCompat.requestPermissions(StreamingMinActivity.this,  
                new String[]{Manifest.permission.RECORD_AUDIO,  
Manifest.permission.CAMERA},  
                PUBLISH_REQUEST_CODE);  
            ...  
        }  
    });  
};
```

## 7. Публикация потока после предоставления соответствующих прав

`Stream.publish()` [code](#)

```
@Override  
public void onRequestPermissionsResult(int requestCode,  
                                       @NonNull String permissions[],  
                                       @NonNull int[] grantResults) {  
    switch (requestCode) {  
        case PUBLISH_REQUEST_CODE: {  
            if (grantResults.length == 0 ||  
                grantResults[0] != PackageManager.PERMISSION_GRANTED ||  
                grantResults[1] != PackageManager.PERMISSION_GRANTED) {
```

```

        muteButton();
        session.disconnect();
        Log.i(TAG, "Permission has been denied by user");
    } else {
        /**
         * Method Stream.publish() is called to publish stream.
         */
        publishStream.publish();
        Log.i(TAG, "Permission has been granted by user");
    }
    break;
}
...
}
}
}

```

## 8. Воспроизведение потока микшера после успешной публикации

`Session.createStream()`, `Stream.play()` code

```

publishStream.on(new StreamStatusEvent() {
    @Override
    public void onStreamStatus(final Stream stream, final StreamStatus
streamStatus) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if (StreamStatus.PUBLISHING.equals(streamStatus)) {
                    /**
                     * The options for the stream to play are set.
                     * The stream name is passed when StreamOptions object is
created.
                     */
                    String playStreamName = roomName + "-" + login +
roomName;
                    StreamOptions streamOptions = new
StreamOptions(playStreamName);
                    streamOptions.setTransport(Transport.valueOf(mTransportOutput.getSpinner().getSe

                    playStream = session.createStream(streamOptions);
                    ...
                    /**
                     * Method Stream.play() is called to start playback of
the stream.
                     */
                    playStream.play();
                } else {
                    Log.e(TAG, "Can not publish stream " + stream.getName() +
" " + streamStatus);
                }
                mStatusView.setText(streamStatus.toString());
            }
        });
    }
});

```

```
}  
});
```

## 9. Закрытие соединения

`Session.disconnect()` code

```
mStartButton.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        muteButton();  
        if (mStartButton.getTag() == null ||  
Integer.valueOf(R.string.action_start).equals(mStartButton.getTag())) {  
            ...  
        } else {  
            /**  
             * Connection to WCS server is closed with method  
Session.disconnect().  
            */  
            session.disconnect();  
        }  
        ...  
    }  
});
```

## 10. Получение события, подтверждающего разъединение

`session.onDisconnection()` code

```
@Override  
public void onDisconnection(final Connection connection) {  
    runOnUiThread(new Runnable() {  
        @Override  
        public void run() {  
            mStatusView.setText(connection.getStatus());  
            mStatusView.setText(connection.getStatus());  
            onStoped();  
        }  
    });  
}
```

## 11. Настройки публикации/проигрывания аудио и видео

code

```
@NonNull  
private Constraints getConstraints() {  
    AudioConstraints audioConstraints = null;  
    if (mSendAudio.isChecked()) {  
        audioConstraints = new AudioConstraints();  
    }  
}
```

```
VideoConstraints videoConstraints = null;
if (mSendVideo.isChecked()) {
    videoConstraints = new VideoConstraints();
}
return new Constraints(audioConstraints, videoConstraints);
}
```