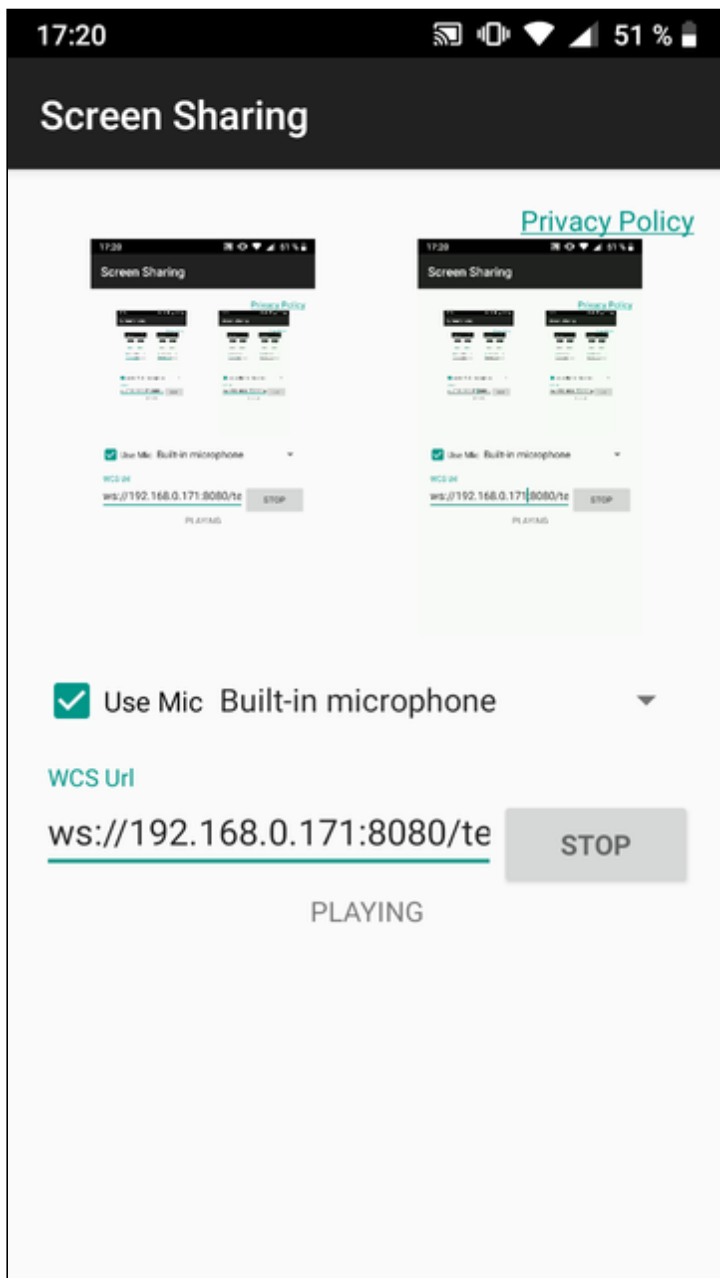


# Android Screen sharing

Example of Android application to publish device screen



## Analyzing the example code

To analyze the code let's take the class `ScreenSharingActivity.java` of the `screen-sharing` example, which can be downloaded with build `1.1.0.55`.

## 1. Initialization of the API

`Flashphoner.init()` code

`Context` object is passed to method `init()` for initialization.

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

## 2. Request the permission to use microphone

code

```
mMicCheckBox.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (mMicCheckBox.isChecked()) {
            ActivityCompat.requestPermissions(ScreenSharingActivity.this,
                new String[]{Manifest.permission.RECORD_AUDIO},
                PUBLISH_REQUEST_CODE);
        }
    }
});
```

## 3. Choose the microphone

code

```
mMicSpinner = (Spinner) findViewById(R.id.spinner_mic);
ArrayAdapter<MediaDevice> arrayAdapter = new ArrayAdapter<MediaDevice>(this,
    android.R.layout.simple_spinner_item,
    Flashphoner.getMediaDevices().getAudioList());
arrayAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it

mMicSpinner.setAdapter(arrayAdapter);
```

## 4. Session creation

`Flashphoner.createSession()` code

`SessionOptions` object is passed to the method with the following parameters

- WCS server URL
- `SurfaceViewRenderer localRender` to display local video
- `SurfaceViewRenderer remoteRender` to display video published

```
SessionOptions sessionOptions = new SessionOptions(url);
sessionOptions.setLocalRenderer(localRender);
sessionOptions.setRemoteRenderer(remoteRender);
```

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

## 5. Connection to the server

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

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

## 6. Receiving the event confirming successful connection

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

```
@Override
public void onConnected(final Connection connection) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            mStartButton.setText(R.string.action_stop);
            mStartButton.setTag(R.string.action_stop);
            mStartButton.setEnabled(true);
            mStatusView.setText(connection.getStatus());
            ...
        }
    });
}
```

## 7. Video stream creation

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

```
StreamOptions streamOptions = new StreamOptions(streamName);
VideoConstraints videoConstraints = new VideoConstraints();
DisplayMetrics metrics = getResources().getDisplayMetrics();
videoConstraints.setResolution(metrics.widthPixels, metrics.heightPixels);
videoConstraints.setVideoFps(metrics.densityDpi);
streamOptions.getConstraints().setVideoConstraints(videoConstraints);
streamOptions.getConstraints().updateAudio(mMicCheckBox.isChecked());

/**
 * Stream is created with method Session.createStream().
 */
publishStream = session.createStream(streamOptions);
...
startScreenCapture();
```

## 8. Prepare to capture device screen

code

```
private void startScreenCapture() {
    mMediaProjectionManager = (MediaProjectionManager) getSystemService(
        Context.MEDIA_PROJECTION_SERVICE);
    Intent permissionIntent =
mMediaProjectionManager.createScreenCaptureIntent();
    startActivityForResult(permissionIntent, REQUEST_CODE_CAPTURE_PERM);
}
```

## 9. Start foreground service, capture the screen and publish the stream

`startService()`, `setVideoCapturer()`, `Stream.publish()` code

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
    if (REQUEST_CODE_CAPTURE_PERM == requestCode && resultCode == RESULT_OK)
    {
        serviceIntent = new Intent(this, ScreenSharingService.class);
        startService(serviceIntent);
        videoCapturer = new ScreenCapturerAndroid(data, new
MediaProjection.Callback() {
            @Override
            public void onStop() {
                super.onStop();
            }
        });
        WebRTCMediaProvider.getInstance().setVideoCapturer(videoCapturer);

        /**
         * Method Stream.publish() is called to publish stream.
         */
        publishStream.publish();
        Log.i(TAG, "Permission has been granted by user");
        ...
    }
}
```

## 10. Receiving the event confirming the successful stream publishing

`StreamStatusEvent.PUBLISHING` code

On receiving this event preview stream is created with `Session.createStream()` and `Stream.play()` is invoked to play it.

```
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.
                 */
                StreamOptions streamOptions = new
                StreamOptions(streamName);

                streamOptions.getConstraints().updateAudio(mMicCheckBox.isChecked());

                /**
                 * Stream is created with method Session.createStream().
                 */
                playStream = session.createStream(streamOptions);
                ...
                playStream.play();
            } else {
                Log.e(TAG, "Can not publish stream " + stream.getName() +
                " " + streamStatus);
            }
            mView.setText(streamStatus.toString());
        }
    });
}
});

```

## 11. Session disconnection

`Session.disconnect()` code

```

mStartButton.setEnabled(false);

/**
 * Connection to WCS server is closed with method Session.disconnect().
 */
session.disconnect();

```

## 12. Starting foreground service

`Service.onCreate()`, `startForeground()` code

```

@Override
public void onCreate() {
    super.onCreate();

    NotificationChannel chan =
        new NotificationChannel(

```

```

        CHANNEL_ID, CHANNEL_NAME,
        NotificationManager.IMPORTANCE_NONE);
        NotificationManager manager =
            (NotificationManager)
            getSystemService(Context.NOTIFICATION_SERVICE);
        chan.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);
        manager.createNotificationChannel(chan);

        final int notificationId = (int) System.currentTimeMillis();
        NotificationCompat.Builder notificationBuilder =
            new NotificationCompat.Builder(this, CHANNEL_ID);
        Notification notification =
            notificationBuilder
                .setSmallIcon(R.drawable.service_icon)
                .setOngoing(true)
                .setShowWhen(true)
                .setContentTitle("ScreenSharingService is running in the
foreground")
                .setPriority(NotificationManager.IMPORTANCE_MIN)
                .setCategory(Notification.CATEGORY_SERVICE)
                .build();
        NotificationManager notificationManager
            = (NotificationManager)
            getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.notify(NOTIFICATION_ID, notification);

        startForeground(notificationId, notification);
    }

```

### 13. Stopping foreground service

`Service.onDestroy()`, `stopForeground()` code

```

@Override
public void onDestroy() {
    stopForeground(true);
    super.onDestroy();
}

```