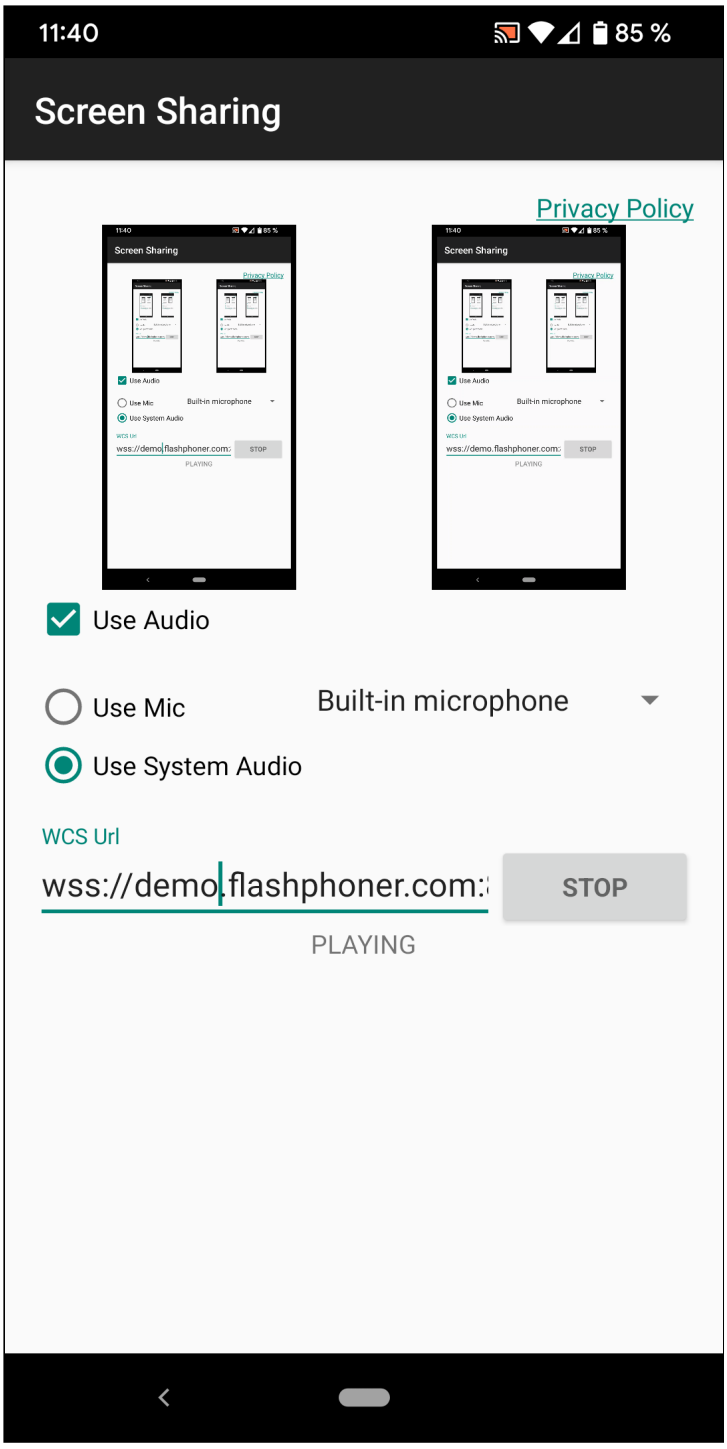
Android Screen sharing

Example of Android application to publish device screen

The example shows how to share device screen. Device microphone or (in Android 10 or above) system audio may also be captured.



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 1.1.0.64.

1. Initialization of the API

`Flashphoner.init()` code

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

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

2. Display or hide a system sound capture button depending on Android version

code

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
    mAudioRadioGroup.setVisibility(View.VISIBLE);
} else {
    mAudioRadioGroup.setVisibility(View.GONE);

}
```

3. Request the permission to capture audio

code

```
mUseAudioCheckBox.setOnClickListener(v -> {
    if (mUseAudioCheckBox.isChecked()) {
        ActivityCompat.requestPermissions(ScreenSharingActivity.this,
                new String[]{Manifest.permission.RECORD_AUDIO},
                PUBLISH_REQUEST_CODE);
    }
});
```

4. 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);
```

5. 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);
```

6. Connection to the server

`Session.connect()` code

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

7. Receiving the event confirming successful connection

`Session.onConnected()` code

```
@Override
public void onConnected(final Connection connection) {
    runOnUiThread(() -> {
        mStartButton.setText(R.string.action_stop);
        mStartButton.setTag(R.string.action_stop);
        mStatusView.setText(connection.getStatus());
    });
    ...
}
```

8. 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(mUseAudioCheckBox.isChecked());

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

```
...
startScreenCapture();
```

9. 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);
}
```

10. Start foreground service

`context.startForegroundService()` code

```
protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (REQUEST_CODE_CAPTURE_PERM == requestCode && resultCode == RESULT_OK)
{

        this.mediaProjectionData = data;

        Context context = getApplicationContext();
        this.serviceIntent = new Intent(context, ScreenSharingService.class);
        context.startForegroundService(serviceIntent);
    } else {
        runOnUiThread(() -> mStartButton.setEnabled(false));
        stop();
        Log.i(TAG, "Permission has been denied by user");
    }
}
```

11. Capture device screen and publish a stream

`ScreenCapturerAndroid()`, `Stream.publish()` code

```
private final BroadcastReceiver mMessageReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent != null) {
            if (ScreenSharingService.ACTION_START.equals(intent.getAction()))
{
                MediaProjection mediaProjection = null;
                if (mUseAudioCheckBox.isChecked() &&
!mUseMicRadioButton.isChecked() && Build.VERSION.SDK_INT >=
Build.VERSION_CODES.Q) {
```

```
                    mediaProjection =
mediaProjectionManager.getMediaProjection(Activity.RESULT_OK,
mediaProjectionData);
                }


WebRTCMediaProvider.getInstance().setMediaProjection(mediaProjection);
                videoCapturer = new ScreenCapturerAndroid(mediaProjection,
mediaProjectionData, new MediaProjection.Callback() {
                    @Override
                    public void onStop() {
                        super.onStop();
                        handler.post(ScreenSharingActivity.this::stop);
                    }
                });

WebRTCMediaProvider.getInstance().setVideoCapturer(videoCapturer);

                publishStream.publish();
            } else if
(ScreenSharingService.ACTION_STOP.equals(intent.getAction())) {
                handler.post(ScreenSharingActivity.this::stop);
            }
        }
    }
};
```

12. 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(mUseAudioCheckBox.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);
                }
                mStatusView.setText(streamStatus.toString());
            }
        });
    }
});
```

13. Session disconnection

`Session.disconnect()` code

```
private synchronized void stop() {
    if (session != null) {
        session.disconnect();
        session = null;
    }

    WebRTCMediaProvider.getInstance().releaseLocalMediaAccess();

    if (serviceIntent != null) {
        stopService(serviceIntent);
        this.serviceIntent = null;
    }

    ...
}
```

14. Foreground service creation

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

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

    NotificationChannel chan = new NotificationChannel(CHANNEL_ID,
CHANNEL_NAME, NotificationManager.IMPORTANCE_NONE);
    chan.setImportance(NotificationManager.IMPORTANCE_MIN);

    NotificationManager manager =
            (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
    chan.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);
    manager.createNotificationChannel(chan);

    final int notificationId = (int) System.currentTimeMillis();
```

```
    Notification.Builder notificationBuilder = new Notification.Builder(this,
CHANNEL_ID);
    Notification notification =
            notificationBuilder
                    .setSmallIcon(R.drawable.service_icon)
                    .setOngoing(true)
                    .setShowWhen(true)
                    .setContentTitle("ScreenSharingService is running in the
foreground")
                    .setCategory(Notification.CATEGORY_SERVICE)
                    .addAction(createStopAction())
                    .build();

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
        startForeground(notificationId, notification,
ServiceInfo.FOREGROUND_SERVICE_TYPE_MEDIA_PROJECTION);
    } else {
        startForeground(notificationId, notification);
    }
}
```

15. Stopping foreground service

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

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