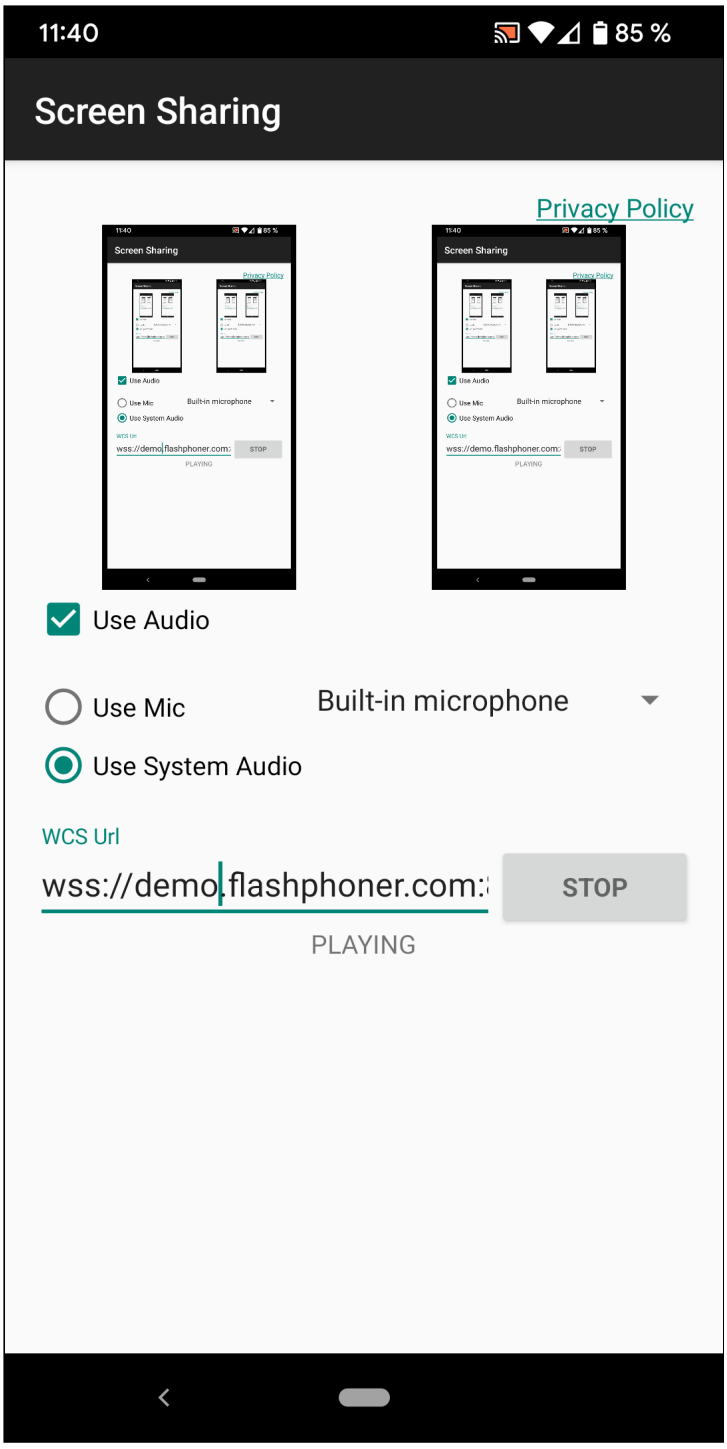
Android Screen sharing

Пример Android приложения для демонстрации экрана устройства

Пример демонстрирует возможность трансляции экрана устройства. К видео может быть добавлено аудио с микрофона устройства или (для Android 10 и выше) системный звук.



Работа с кодом примера

Для разбора кода возьмем класс ScreenSharingActivity.java примера `screen-sharing`, который доступен для скачивания в соответствующей сборке 1.1.0.64.

1. Инициализация API

`Flashphoner.init()` code

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

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

2. Скрытие или отображение захвата системного звука в зависимости от версии Android

code

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

}
```

3. Разрешение на захват звука

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);
        }
    }
});
```

4. Выбор микрофона

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. Создание сессии

`Flashphoner.createSession()` code

Методу передается объект `SessionOptions` со следующими параметрами

- URL WCS-сервера
- `SurfaceViewRenderer localRender`, который будет использоваться для отображения видео с экрана
- `SurfaceViewRenderer remoteRender`, который будет использоваться для воспроизведения опубликованного видеопотока

```
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. Подключение к серверу

`Session.connect()` code

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

7. Получение от сервера события, подтверждающего успешное соединение

`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. Создание потока и подготовка к публикации

`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. Подготовка захвата экрана

code

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

10. Запуск сервиса

`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. Захват экрана и публикация потока

`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. Получение от сервера события, подтверждающего успешную публикацию потока

`StreamStatusEvent.PUBLISHING` code

При получении данного события создается превью-видеопоток при помощи `Session.createStream()`, и вызывается `Stream.play()` для его воспроизведения.

```
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. Закрытие соединени

`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. Создание сервиса

`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. Остановка сервиса

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

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