ДОДАТОК А
JAVA КОД ANDROID ДОДАТКА

MainActivity.java

```
import android.app.Activity;
import android.app.admin.DevicePolicyManager;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.method.ScrollingMovementMethod;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.samsung.android.knox.AppIdentity;
import com.samsung.android.knox.EnterpriseDeviceManager;
import com.samsung.android.knox.application.ApplicationPolicy;
import com.samsung.android.knox.license.EnterpriseLicenseManager;
import com.samsung.android.knox.license.KnoxEnterpriseLicenseManager;
import com.samsung.android.knox.restriction.RestrictionPolicy;

import java.util.ArrayList;
import java.util.List;

import static
com.samsung.android.knox.application.ApplicationPolicy.PERMISSION_POLICY
_STATE_GRANT;

public class MainActivity extends AppCompatActivity {

    public static final String TAG = "MainActivity";
    private static final int DEVICE_ADMIN_ADD_RESULT_ENABLE = 1;
    private Button mToggleAdminBtn;
    private DevicePolicyManager mDPM;
    private ComponentName mDeviceAdmin;
    private Utils mUtils;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //...called when the activity is starting. This is where most
```

```
initialization should go.
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TextView LogView = (TextView) findViewById(R.id.logview_id);
        LogView.setMovementMethod(new ScrollingMovementMethod());
        mUtils = new Utils(LogView, TAG);

        // Check if device supports Knox SDK
        mUtils.checkApiLevel(24, this);

        mDPM = (DevicePolicyManager)
getSystemService(Context.DEVICE_POLICY_SERVICE);
        mDeviceAdmin = new ComponentName(MainActivity.this,
SampleAdminReceiver.class);

        mToggleAdminBtn = (Button) findViewById(R.id.ToggleAdmin);
        mToggleAdminBtn.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {toggleAdmin();
            }
        });
        Button ActivateLicencebtn = (Button)
findViewById(R.id.ActivateLicencebtn);
        ActivateLicencebtn.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                activateLicence();
            }
        });
        Button DeactivateLicencebtn = (Button)
findViewById(R.id.DeactivateLicensebtn);
        DeactivateLicencebtn.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                deactivateLicense();
            }
        });
        Button ToggleCamerabtn = (Button)
findViewById(R.id.ToggleCamerabtn);
        ToggleCamerabtn.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View v){
                toggleCameraState();
            }
        });
    }

    @Override
    protected void onResume() {
        super.onResume();
        refreshButtons();
    }

    /** Handle result of device administrator activation request */
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        if (requestCode == DEVICE_ADMIN_ADD_RESULT_ENABLE) {
            switch (resultCode) {
                // End user cancels the request
                case Activity.RESULT_CANCELED:
                    mUtils.log(getString(R.string.admin_cancelled));
                    break;
                // End user accepts the request
                case Activity.RESULT_OK:
                    mUtils.log(getString(R.string.admin_activated));
                    refreshButtons();
                    break;
            }
        }
    }

    /** Present a dialog to activate device administrator for this
application */
    private void toggleAdmin() {
        boolean adminState = mDPM.isAdminActive(mDeviceAdmin);
        if (adminState) {
            mUtils.log(getString(R.string.deactivating_admin));
            // Deactivate application as device administrator
            mDPM.removeActiveAdmin(new ComponentName(this,
SampleAdminReceiver.class));
            mToggleAdminBtn.setText(getString(R.string.activate_admin));
        } else {
            try {
                mUtils.log(getString(R.string.activating_admin));
```

```
                // Ask the user to add a new device administrator to the
system
                Intent intent = new
Intent(DevicePolicyManager.ACTION_ADD_DEVICE_ADMIN);
                intent.putExtra(DevicePolicyManager.EXTRA_DEVICE_ADMIN,
mDeviceAdmin);
                // Start the add device administrator activity
                startActivityForResult(intent,
DEVICE_ADMIN_ADD_RESULT_ENABLE);

            } catch (Exception e) {
                mUtils.processException(e, TAG);
            }
        }
    }


    private void activateLicence() {

        KnoxEnterpriseLicenseManager licenseManager =
KnoxEnterpriseLicenseManager.getInstance(this);

        try {
            licenseManager.activateLicense(Constants.KPE_LICENSE_KEY);

mUtils.log(getResources().getString(R.string.license_progress));

        } catch (Exception e) {
            mUtils.processException(e,TAG);
        }
    }

    /**
     *  Deactivate license key.
     */
    private void deactivateLicense() {

        KnoxEnterpriseLicenseManager licenseManager =
KnoxEnterpriseLicenseManager.getInstance(this);

        try {
            // License deactivation
            licenseManager.deActivateLicense(Constants.KPE_LICENSE_KEY);
```

```
mUtils.log(getResources().getString(R.string.license_deactivation));

        } catch (Exception e) {
            mUtils.processException(e, TAG);
        }
    }

    /** Toggle the restriction of the device camera on/off. When set to
disabled, the end user will
    * be unable to use the device cameras.
    */
    private void toggleCameraState() {

        // Instantiate the EnterpriseDeviceManager class
        EnterpriseDeviceManager enterpriseDeviceManager =
EnterpriseDeviceManager.getInstance(this);
        // Get the RestrictionPolicy class where the setCameraState
method lives
        RestrictionPolicy restrictionPolicy =
enterpriseDeviceManager.getRestrictionPolicy();

        boolean cameraEnabled = restrictionPolicy.isCameraEnabled(true);
////


        try {
            // Toggle the camera state on/off
            restrictionPolicy.setCameraState(!cameraEnabled); //true
            mUtils.log(getResources().getString(R.string.camera_state,
!cameraEnabled));

        } catch (Exception e) {
            mUtils.processException(e,TAG);
        }


    }

    /** Update button state */
    private void refreshButtons() {
        boolean adminState = mDPM.isAdminActive(mDeviceAdmin);

        if (!adminState) {
            mToggleAdminBtn.setText(getString(R.string.activate_admin));
```

```
        } else {

mToggleAdminBtn.setText(getString(R.string.deactivate_admin));
        }
    }
}
```

SamleAdminReceiver.java

```
import android.app.admin.DeviceAdminReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class SampleAdminReceiver extends DeviceAdminReceiver {

    void showToast(Context context, CharSequence msg) {
        Toast.makeText(context, msg, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onEnabled(Context context, Intent intent) {
        showToast(context, "Device admin enabled");
    }

    @Override
    public void onDisabled(Context context, Intent intent) {
        showToast(context, "Device admin disabled");
    }
}
```

SampleLicenceReceiver.java

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.widget.Toast;
```

```
import com.samsung.android.knox.license.KnoxEnterpriseLicenseManager;

public class SampleLicenseReceiver extends BroadcastReceiver {

    private int DEFAULT_ERROR_CODE = -1;

    private void showToast(Context context, int msg_res) {
        Toast.makeText(context,
context.getResources().getString(msg_res), Toast.LENGTH_SHORT).show();
    }

    private void showToast(Context context, String msg) {
        Toast.makeText(context, msg, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onReceive(Context context, Intent intent) {

        int msg_res = -1;

        if (intent == null) {
            // No intent action is available
            showToast(context, R.string.no_intent);
            return;
        } else {
            String action = intent.getAction();
            if (action == null) {
                // No intent action is available
                showToast(context, R.string.no_intent_action);
                return;
            }  else if
(action.equals(KnoxEnterpriseLicenseManager.ACTION_LICENSE_STATUS)) {
                // Key activation result Intent is obtained
                int errorCode =
intent.getIntExtra(KnoxEnterpriseLicenseManager.EXTRA_LICENSE_ERROR_CODE
, DEFAULT_ERROR_CODE);

                if (errorCode ==
KnoxEnterpriseLicenseManager.ERROR_NONE) {
                    // Key activated or deactivated successfully
                    showToast(context, R.string.klm_action_successful);
                    Log.d("SampleLicenseReceiver",
context.getString(R.string
                            .klm_action_successful));
```

```
                    return;
                } else {
                    // activation failed
                    switch (errorCode) {
                        case
KnoxEnterpriseLicenseManager.ERROR_INTERNAL:
                            msg_res = R.string.err_klm_internal;
                            break;
                        case
KnoxEnterpriseLicenseManager.ERROR_INTERNAL_SERVER:
                            msg_res = R.string.err_klm_internal_server;
                            break;
                        case
KnoxEnterpriseLicenseManager.ERROR_INVALID_LICENSE:
                            msg_res =
R.string.err_klm_licence_invalid_license;
                            break;
                        case
KnoxEnterpriseLicenseManager.ERROR_INVALID_PACKAGE_NAME:
                            msg_res =
R.string.err_klm_invalid_package_name;
                            break;
                        case
KnoxEnterpriseLicenseManager.ERROR_LICENSE_TERMINATED:
                            msg_res =
R.string.err_klm_licence_terminated;
                            break;
                        case
KnoxEnterpriseLicenseManager.ERROR_NETWORK_DISCONNECTED:
                            msg_res =
R.string.err_klm_network_disconnected;
                            break;
                        case
KnoxEnterpriseLicenseManager.ERROR_NETWORK_GENERAL:
                            msg_res = R.string.err_klm_network_general;
                            break;
                        case
KnoxEnterpriseLicenseManager.ERROR_NOT_CURRENT_DATE:
                            msg_res = R.string.err_klm_not_current_date;
                            break;
                        case
KnoxEnterpriseLicenseManager.ERROR_NULL_PARAMS:
                            msg_res = R.string.err_klm_null_params;
                            break;
```

```
                        case KnoxEnterpriseLicenseManager.ERROR_UNKNOWN:
                            msg_res = R.string.err_klm_unknown;
                            break;
                        case
KnoxEnterpriseLicenseManager.ERROR_USER_DISAGREES_LICENSE_AGREEMENT:
                            msg_res =
R.string.err_klm_user_disagrees_license_agreement;
                            break;

                        default:
                            // Unknown error code
                            String errorStatus =
intent.getStringExtra(KnoxEnterpriseLicenseManager.EXTRA_LICENSE_STATUS)
;
                            String msg =
context.getResources().getString(R.string.err_klm_code_unknown,
Integer.toString(errorCode), errorStatus);
                            showToast(context, msg);
                            Log.d("SampleLicenseReceiver", msg);
                            return;
                    }

                    // Display KLM error message
                    showToast(context, msg_res);
                    Log.d("SampleLicenseReceiver",
context.getString(msg_res));
                    return;
                }
            }
        }
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.samsung.knox.knoxsdk">

    <!-- Permissions TODO List permissions used by application -->
    <uses-permission
android:name="com.samsung.android.knox.permission.KNOX_HW_CONTROL" />
    <permission
        android:name="com.example.supportlibclient.SUPPORT_PERMISSION"
```

```
        android:label="Support permission"
        android:protectionLevel="signature" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name="com.samsung.knox.knoxsdk.MainActivity"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <!--<meta-data
android:name="com.samsung.knoxlicense.permissions" android:value="true"
/>-->
        <!-- SampleAdminReceiver TODO Provide receiver for device admin
receiver class -->
        <receiver
            android:name="com.samsung.knox.knoxsdk.SampleAdminReceiver"

android:description="@string/enterprise_device_admin_description"
            android:label="@string/enterprise_device_admin"
            android:permission="android.permission.BIND_DEVICE_ADMIN" >
            <meta-data
                android:name="android.app.device_admin"
                android:resource="@xml/device_admin_receiver" />
            <intent-filter>
                <action
android:name="android.app.action.DEVICE_ADMIN_ENABLED" />
            </intent-filter>
        </receiver>
        <!-- Activate License TODO Provide receiver for Knox license
activation results -->
        <receiver
android:name="com.samsung.knox.knoxsdk.SampleLicenseReceiver" >
            <intent-filter>
                <action
```

```
android:name="com.samsung.android.knox.intent.action.KNOX_LICENSE_STATUS
" />
            </intent-filter>
        </receiver>

    </application>

</manifest>
```