**Рекомендации по подключению библиотеки MariaAPi и настройке приложения**

***Добавьте в манифест*** *вашего приложения секции* **uses-permission** *для определения разрешений.*

*Также обязательным является определения:*

* **intent-filter{.USB\_DEVICE\_ATTACHED},**
* **service{.usbserial.UsbService }**
* **activity{.BT.SelectBluetoothActivity}**
* **meta-data{.USB\_DEVICE\_ATTACHED }{.USB\_DEVICE\_DETACHED}**

*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="…"**>  
  
 <**uses-feature android:name="android.hardware.usb.host"** />  
 <**uses-permission android:name="android.permission.BLUETOOTH"** />  
 <**uses-permission android:name="android.permission.BLUETOOTH\_ADMIN"** />  
 <**uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"** />  
 <**uses-permission android:name="android.permission.READ\_PHONE\_STATE"** />  
 <**uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"** />  
 <**uses-permission android:name="com.android.example.USB\_PERMISSION"**/>  
   
 <**application  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/android\_logo"  
 android:theme="@style/AppTheme"**>  
  
 <**activity android:name=".MainActivity"  
 android:launchMode="singleTask"  
 android:icon="@drawable/android\_logo"**>  
  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
 <**category android:name="android.intent.category.DEFAULT"**/>  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>

<**intent-filter**>  
 <**action android:name="android.bluetooth.device.action.ACL\_DISCONNECTED"** />  
 </**intent-filter**>

<**intent-filter**>  
 <**action android:name="android.hardware.usb.action.USB\_DEVICE\_ATTACHED"** />  
 </**intent-filter**>

<**meta-data  
 android:name="android.hardware.usb.action.USB\_DEVICE\_ATTACHED"  
 android:resource="@xml/device\_filter"** />  
 <**meta-data  
 android:name="android.hardware.usb.action.USB\_DEVICE\_DETACHED"  
 android:resource="@xml/device\_filter"** />  
 </**activity**>

<**service  
 android:name="com.resonance.ekka.mariaapilib.usbserial.UsbService"  
 android:enabled="true"** />

<**activity  
 android:name="com.resonance.ekka.mariaapilib.BT.SelectBluetoothActivity"  
 android:label="@string/title\_activity\_select\_bluetooth"  
 android:noHistory="false"** />  
 </**application**>  
</**manifest**>

**Добавить в файл:** settings.gradle

include **':mariaapilib-1.0'**

**Добавить в файл:** build.gradle

implementation project(**":mariaapilib-1.0"**)

**Главное активити приложения:**

**import** com.resonance.ekka.mariaapilib.MariaAPI;  
**import** com.resonance.ekka.mariaapilib.MariaCommands;  
**import** com.resonance.ekka.mariaapilib.BT.BluetoothWorker;  
**import** com.resonance.ekka.mariaapilib.usbserial.UsbService;

**public class** MainActivity **extends** AppCompatActivity {

**.**

**.**

**.**

Method **mToInvoke**;

Object[] **argsValues**;

@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);

MariaAPI **api**;

**api** = **new** MariaAPI(**this**);

//определим коллбэк для приема сообщений о состоянии канала (USB/Bluetooth) при соединении и разрыве связи   
**api**.onMessageCallBack(**mCallback**);

//определим коллбэк для приема Exceptions при обмене данными с ЭККА

**api**.**MariaCmd**.onErrorCallBack(**mErrCallback**);

//определим коллбэк для приема результата работы функций при обмене данными с ЭККА

**api**.**MariaCmd**.onReadCallBack(**mReadCallback**);

//определим коллбэк снифера данных из канала обмена данными с ЭККА

**api**.**MariaCmd**.onCommandCallBack(**mariaCommandCallback**);

//Следующие функции определяют канал подключения

**api**.SetTypeConnection(MariaAPI.***BLUETOOTH***);

// или

**api**.SetTypeConnection(MariaAPI.***USB***);

}

**private** MariaAPI.MariaAPICallback **mCallback** = **new** MariaAPI.MariaAPICallback()  
{  
 @Override  
 **public void** onReceiveHardwareMessage (Message msg) {  
 **if**(**api**.getTypeConnection() == **api**.***BLUETOOTH***)  
 ParseMessageBT(msg);  
 **else if**(**api**.getTypeConnection() == **api**.***USB***){  
 ParseMessageUSB(msg);  
 }  
 }  
};  
  
**private** MariaCommands.MariaErrorCallback **mErrCallback** = **new** MariaCommands.MariaErrorCallback()  
{  
 @Override  
 **public void** onError (String error) {  
// принимаем ошибку, передаем далее…

EventBus.*getDefault*().post(**new** MessageTransport(error, MessageTransport.RequestTypeUpdate.***SET***, MessageTransport.RequestMessageOptions.***EXCEPTIONS***));

}  
};  
**private** MariaCommands.MariaReadCallback **mReadCallback** = **new** MariaCommands.MariaReadCallback()  
{  
 @Override  
 **public void** onReceiveData (**final** String buf) {

//прием результата работы функции или запроса к ЭККА

runOnUiThread(**new** Runnable() {  
  
 @Override  
 **public void** run() {  
 EventBus.*getDefault*().post(**new** MessageTransport(buf, MessageTransport.RequestTypeUpdate.***APPEND***, MessageTransport.RequestMessageOptions.***RETURN\_VALES***));  
 }  
 });  
 }  
};  
  
**private** MariaCommands.MariaCommandCallback **mariaCommandCallback** = **new** MariaCommands.MariaCommandCallback()  
{  
 @Override  
 **public void** onMariaCommand (**final int** direction,**final** String buf) {  
 runOnUiThread(**new** Runnable() {  
//сниффер канала данных

**if** (direction==1){

//данные от приложения к ЭККА

}else{

//данные от ЭККА к приложению

}

}  
};

@Override  
**protected void** onDestroy() {  
 **super**.onDestroy();  
 **api**.DestructMariaAPI();  
  
}

**private void** ParseMessageBT(Message msg) {  
 **switch** (msg.**what**) {  
 **case** BluetoothWorker.***MSG\_CONNECTED***:  
 Log.*i*(***DBG\_TAG***, **"MSG\_CONNECTED:"** + String.*valueOf*(msg.**obj**));  
 **istConnectEKKA** = **true**;  
 EventBus.*getDefault*().post(**new** MessageTransport(**""**, MessageTransport.RequestTypeUpdate.***SET***, MessageTransport.RequestMessageOptions.***COMMANDS***));  
 EventBus.*getDefault*().post(**new** MessageTransport(**"MSG\_CONNECTED: "** + String.*valueOf*(msg.**obj**)+**" \n"**, MessageTransport.RequestTypeUpdate.***APPEND***, MessageTransport.RequestMessageOptions.***HW***));

**imageViewTypeConnection**.setVisibility(View.***VISIBLE***);  
 **break**;  
 **case** BluetoothWorker.***MSG\_CONNECTING***:  
 Log.*i*(***DBG\_TAG***, **"MSG\_CONNECTING:"** + String.*valueOf*(msg.**obj**));  
 **break**;  
 **case** BluetoothWorker.***MSG\_CONNECTION\_ERROR***:  
  
 showToast(**"Не удается установить подключение с регистратором"**);  
  
 **istConnectEKKA** = **false**;  
  
 **break**;  
 **case** BluetoothWorker.***MSG\_CONNECTION\_FAILED***:  
  
 Log.*e*(***DBG\_TAG***, **"MSG\_CONNECTION\_FAILED "** + String.*valueOf*(msg.**what**));  
 showToast(**"Cвязь с регистратором потеряна"**);  
  
 **istConnectEKKA** = **false**;  
  
 **break**;  
 **case** BluetoothWorker.***MSG\_NOT\_CONNECTED***:  
  
 Log.*i*(***DBG\_TAG***, **"MSG\_NOT\_CONNECTED:"** + String.*valueOf*(msg.**obj**));  
 showToast(**"Нет связи с регистратором"**);  
 **istConnectEKKA** = **false**;  
  
 **break**;  
  
 **default**:  
 Log.*e*(***DBG\_TAG***, **"\*\*Handler default message:"** + String.*valueOf*(msg.**what**));  
 **break**;  
  
 }  
 }  
  
 **private void** ParseMessageUSB(Message msg){  
 **switch** (msg.**what**) {  
 **case** UsbService.***MSG\_USB\_READY***:  
  
 **istConnectEKKA** = **true**;  
 EventBus.*getDefault*().post(**new** MessageTransport(**""**, MessageTransport.RequestTypeUpdate.***SET***, MessageTransport.RequestMessageOptions.***COMMANDS***));  
 EventBus.*getDefault*().post(**new** MessageTransport(**"MSG\_USB\_READY \n"** , MessageTransport.RequestTypeUpdate.***APPEND***, MessageTransport.RequestMessageOptions.***HW***));

**imageViewTypeConnection**.setVisibility(View.***VISIBLE***);  
  
 **break**;  
 **case** UsbService.***MSG\_USB\_DETACHED***:  
 Log.*w*(***DBG\_TAG***, **"MAIN usbHandler MSG\_USB\_DETACHED"**);  
 showToast(**"Устройство USB - отключено"**);  
 **istConnectEKKA** = **false**;  
 **break**;  
 **case** UsbService.***MSG\_USB\_ATTACHED***:  
 Log.*w*(***DBG\_TAG***, **"MAIN usbHandler MSG\_USB\_ATTACHED"**);  
 showToast(**"Устройство USB - подключено"**);  
 **istConnectEKKA** = **false**;  
  
 **break**;  
 **case** UsbService.***MSG\_NO\_USB***:  
 Log.*w*(***DBG\_TAG***, **"MAIN usbHandler MSG\_NO\_USB"**);  
 showToast(**"Устройств USB - НЕ ОБНАРУЖЕНО!"**);  
 **istConnectEKKA** = **false**;  
 **linearLayoutButtons**.setVisibility(View.***GONE***);  
 **linearLayoutParamsResults**.setVisibility(View.***GONE***);  
  
 **break**;  
 **default**:  
 Log.*e*(***DBG\_TAG***, **"\*\*usb Handler default message:"** + String.*valueOf*(msg.**what**));  
 **break**;  
  
 }  
 }

//Рекомендуется делать вызов методов через отдельный поток.

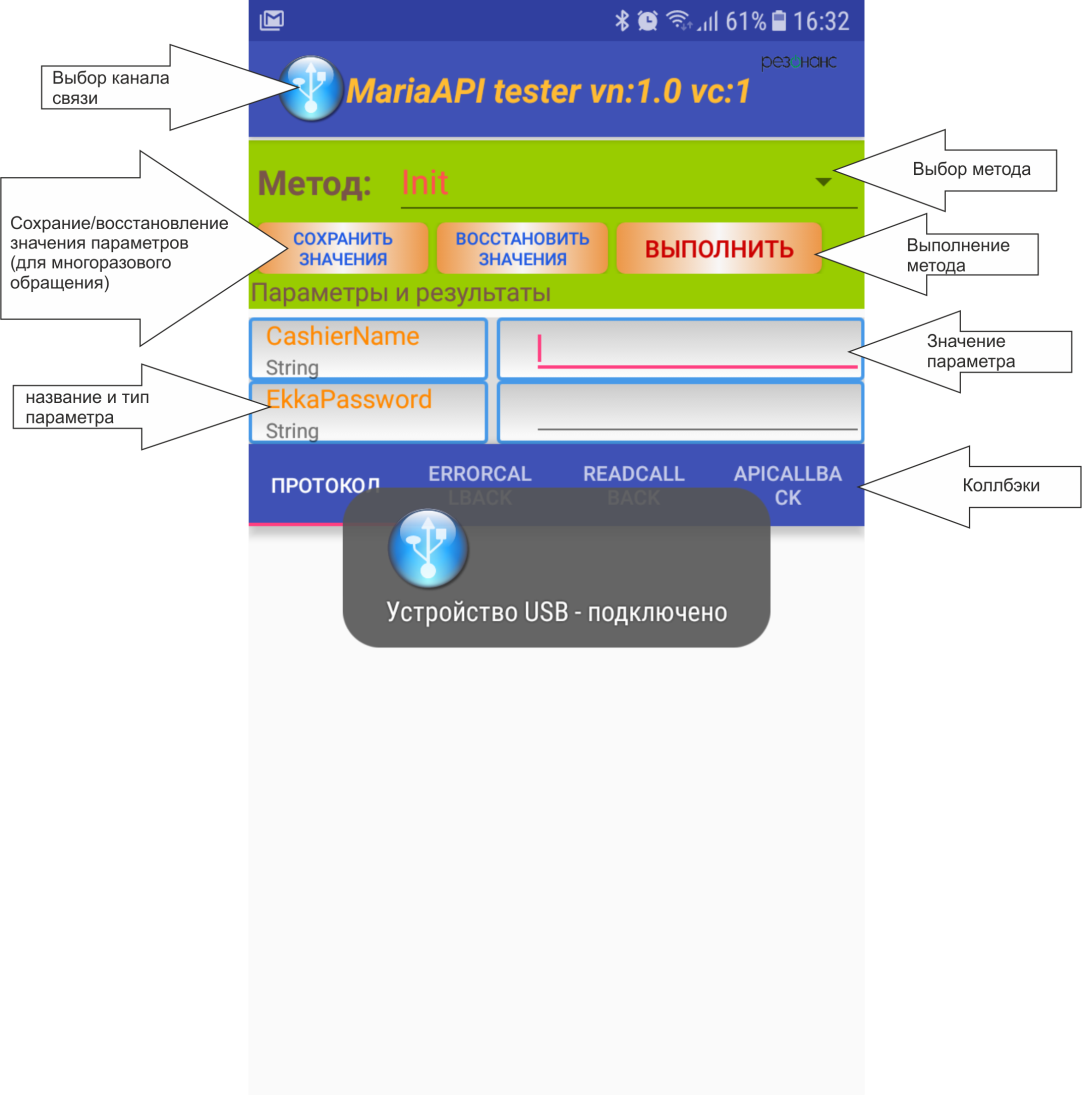
**class** TaskExecutor **extends** AsyncTask<Object, Object, Object> {  
 Object[] **resArgs**;  
.

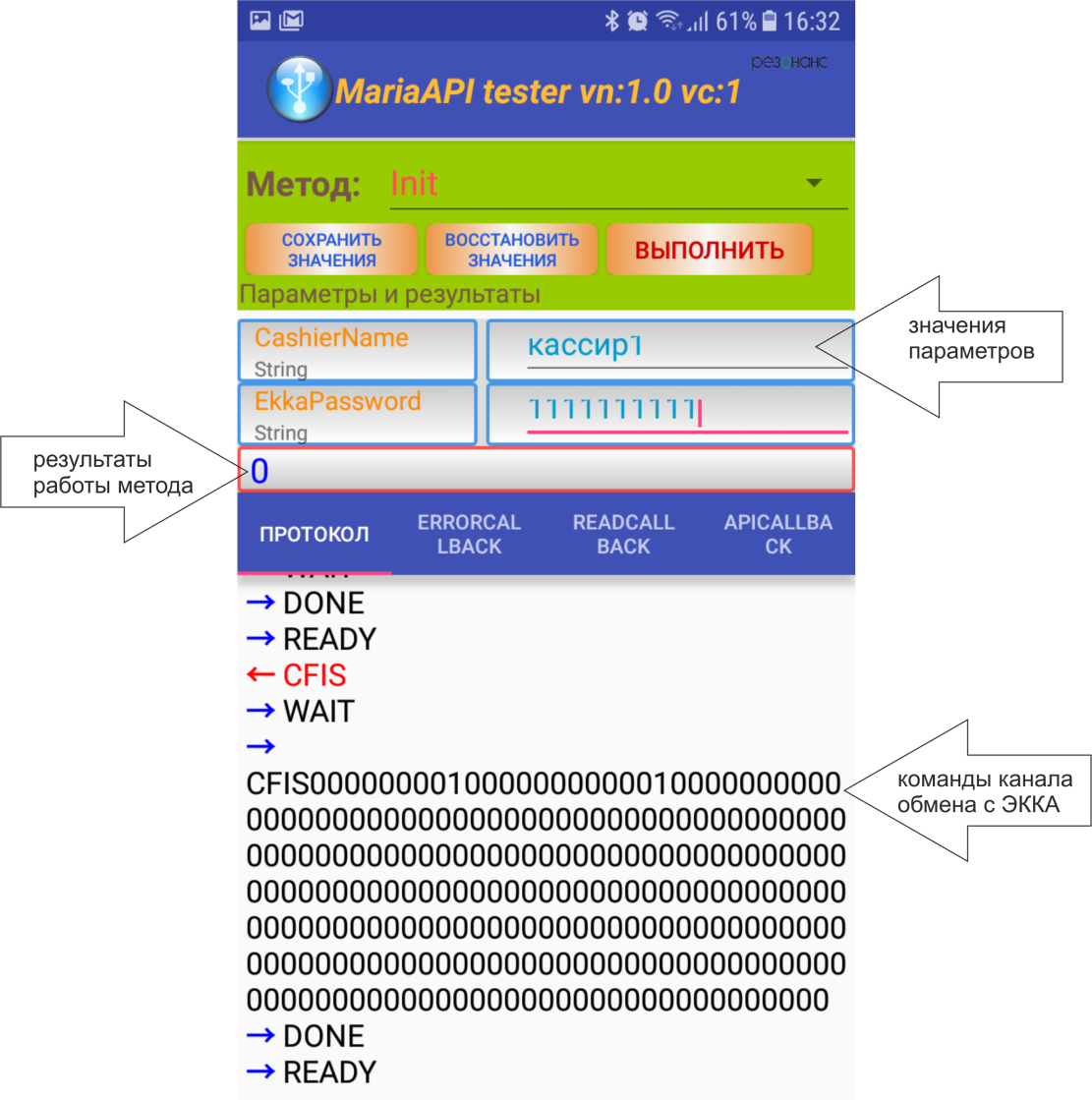
.

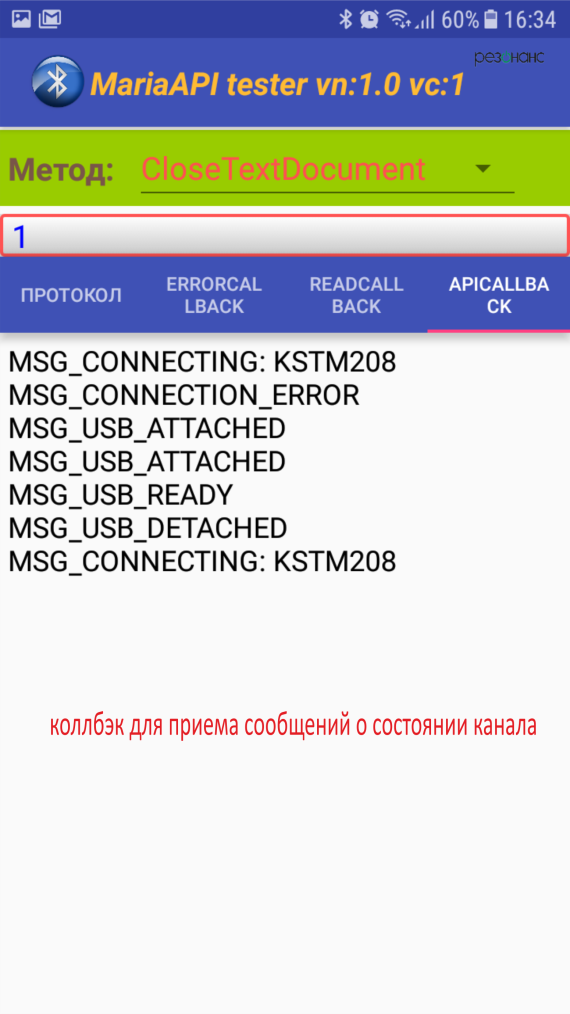
.  
 @Override  
 **protected void** onPreExecute() {  
 **super**.onPreExecute();  
   
 @Override  
 **protected** Object doInBackground(Object... params) {  
  
 **try** {  
  
 ClassInvoke ci = **new** ClassInvoke();  
 ci.setmToInvoke(**mToInvoke**);  
 ci.**resArgs** = **resArgs**.clone();  
 ci.**mc** = **api**.**MariaCmd**;  
  
 **return mToInvoke**.invoke(**api**.**MariaCmd**, **resArgs**);  
  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return null**;  
 }  
  
 @Override  
 **protected void** onPostExecute(Object result) {  
 **super**.onPostExecute(result);  
 **if** (result != **null**) {  
 **if** (result.toString().trim().length()==1) {  
 **if** ((result.toString().contains(**"1"**))||  
 (result.toString().contains(**"2"**))||  
 (result.toString().contains(**"3"**))||  
 (result.toString().contains(**"4"**)))  
 showToast(**"Обратите внимание на результат выполнения операций!"**);  
 }  
  
 }  
 }  
}

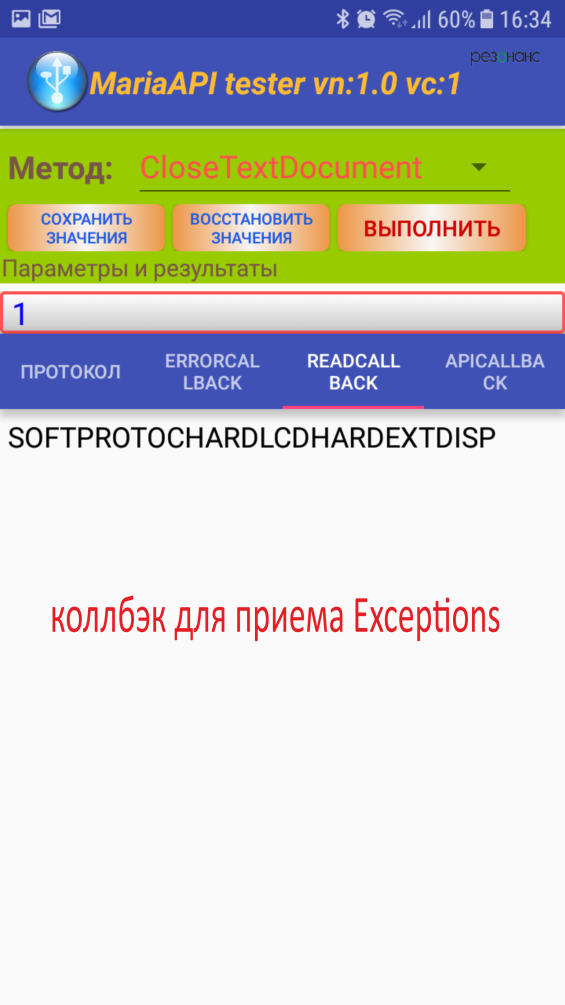
***Интерфейс MariaAPI Tester***

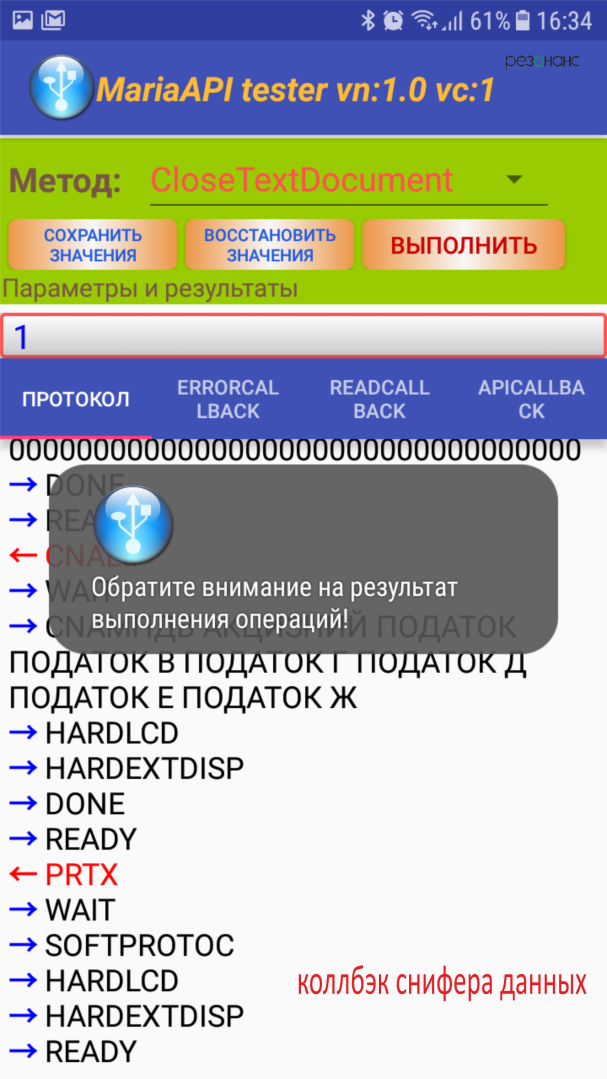
Описание элементов окна

**

******

******

******

******

Обновление: mariaapilib-1.1

Добавлено событие при отключении Bluetooth устройства: ***MSG\_CONNECTION\_FAILED (***onReceiveHardwareMessage***)***

Добавьте в манифест:

<**intent-filter**>  
 <**action android:name="android.bluetooth.device.action.ACL\_DISCONNECTED"** />  
 </**intent-filter**>

Обновление: mariaapilib-1.2

Корректировки в интерфейсе выбора Блютуз – устройства. Добавлена языковая поддержка: EN/UA/RU

Обновление: mariaapilib-1.3

Установка наименования налога

Программирование cхемы налогообложения

Обновление: mariaapilib-1.5.1

Запрос количества пакетов данных, ожидающих отправку на сервер эквайера.

Генерация события ***MSG\_CONNECTION\_REJECT*** при отказе подключения к ЭККР

Обновление системных функций



Рекомендуется выполнять обработку событий(коллбэков) с вынесением в отдельный поток

**private** MariaAPI.MariaAPICallback **mCallback** = **new** MariaAPI.MariaAPICallback()  
{  
 @Override  
 **public void** onReceiveHardwareMessage (**final** Message msg) {  
  
 runOnUiThread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 ParseMessage(msg);  
 }  
 });

};

В случае, если был отказ от установления связи – возобновить процедуру подключения возможно нажатием (дважны с периодичностью) на пиктограмму выбора канала связи

**

В кастомном окне выбора устройства подключения, при отказе от подключения, через MariaAPI.MariaAPICallback посылается сообщение BluetoothWorker.MSG\_CONNECTION\_REJECT. Далее попытки установить соединение прекращаются.

Возобновить процедуру подключения, необходимо вызовом api.SetTypeConnection(api.BLUETOOTH);