API Document

Version: 1.5.0

Release：2020-05-12

Table of Contents

Revision History

Document Version	Revised By	Date	Description
V1.0.0	HGY	2018-12-21	Create a document
V1.0.1	ZYH	2019-08-15	Layout modification
V1.1.0	HJH	2019-08-26	Modify a document
V1.1.1	HJH	2019-08-27	Modify a document
V1.1.2	HJH	2019-08-28	Modified version number
V1.1.3	HJH	2019-08-30	Adjust the method and class name, and adjust serial port version OTA method
V1.1.4	HJH	2019-09-02	Modify method parameters
V1.1.5	ZYH	2019-09-04	Modify the format of the class mothod
V1.1.6	HJH	2019-09-06	check
V1.1.7	HJH	2019-10-26	（eldbox_api-v2.2.3.aar） Communication protocol update: 1. Received version number changed from Int to String. 2. Determine if there is a bindservice before unbind the Bluetooth service
V1.1.8	HJH	2019-11-25	Communication protocol update： 1. Add fault codes and command switching commands And callback method (data returns byte [], which can be converted into an ASCII string); 2. Increase high-precision odometer (and store it in the box device); 3.Added a method to clear fault codes 4.Add callback methods for onReceiveResponse and onResponseNeedToSend
V1.1.9	HJH	2019-11-26	（eldbox_api-v2.3.0.aar） 1.Update the high precision odometer to long type 2.Added a method to send commands used to query CAN data in command mode; 3.Answer callback adds response type; 4. Remove the int version acquisition

Document Version	Revised By	Date	Description
			method
V1.2.0	HJH	2019-12-05	Major changes 1. Add a new active query fault code command to the V1.2.0 protocol 2. Add a new query work mode command to the V1.2.1 protocol 3. Adjust timestamp and engine hours from int to long; 4. Engine hours adjustment, with type long obtained from the device,should divided by 20(all J1939 and OBD); Other changes 5. Limits each parameter's value range, if the value range is not right,has the corresponding printing; 6 . Modified the HighPrecisionOdemeter to HighPrecisionOdometer 7. Modify the property public of Response.java
V1.2.1	HJH	2019-12-06	（eldbox_api-v2.5.1.aar） Fix engine hours to long type and add comments for API.
V1.3.0	HJH	2019-12-26	eldbox_api-v2.6.3.aar） 1. Added V1.3.0 protocol new command to switch passthrough mode, add SN code read and write 2. Fix crash caused by negative length in serial version 3. Print and skips to the next cycle. when the data from serial port is parsed abnormally. 4. Modify the maximum length of SN code to 32 5. Change type value of WorkModeResponse from 0x6004 to 0x6005
V1.4.0	HJH	2020-01-09	（eldbox_api-v2.7.1.aar） 1. Added the query and set CAN bus protocol, and the response of CAN bus protocol setting, the response of the operating mode

Document Version	Revised By	Date	Description
			setting. 2. Fixed the warning of Serial port parsing library “sending message to a Handler on a dead thread” 3. Add Bluetooth box version.
V1.4.1	HJH	2020-03-13	（eldbox_api-v2.7.2.aar） 1.（Bluetooth version）Refactoring analysis method,optimization processing mechanism,throw out the error frame,can get the number of error frames by Bleservice.getSumErrEnd()，can clear number of error frames by Bleservice.setSumErrEnd().
V1.5.0	HJH	2020-05-11	(eldbox_api-v2.9.1.aar) 1、Add command for query fuel level, and create a new callback function called onReceiveotherparameter used to receive parameter values 2、Add the ADC voltage and GPIO output function added by v1.4.0 and v1.7.2 protocols 3、Add a method to calculate the ADC channel voltage, note that the value of channel 3 needs + 0.5V

1 Configuration Development Environment

1.1 IDE

This sdk supports the use of Android Studio for development, and the rest of the development environment is self-attempted.

1.2 Rely

Use this sdk does not need to rely on other third-party libraries.

1.3 Android Archive(AAR) Import

AAR is a binary archive file unique to the Android domain, and the full name is Android ARchive. It is also based on the ZIP format, encapsulates the dynamic link library, the android resource file, and the android tool class, leaving the interface for direct call by the user application.

Note: eldbox_api.aar is the aar library used for this project, if inconsistent, please change the name to eldbox_api.aar.This article is illustrated only by the Android Studio 3.0 project environment configuration.

①Create a libs folder in the Your_project/app/ directory. If this folder already exists, please skip this step.

② Copy eldbox_api.aar to Your_project/app/libs，Sush as Figure 1:

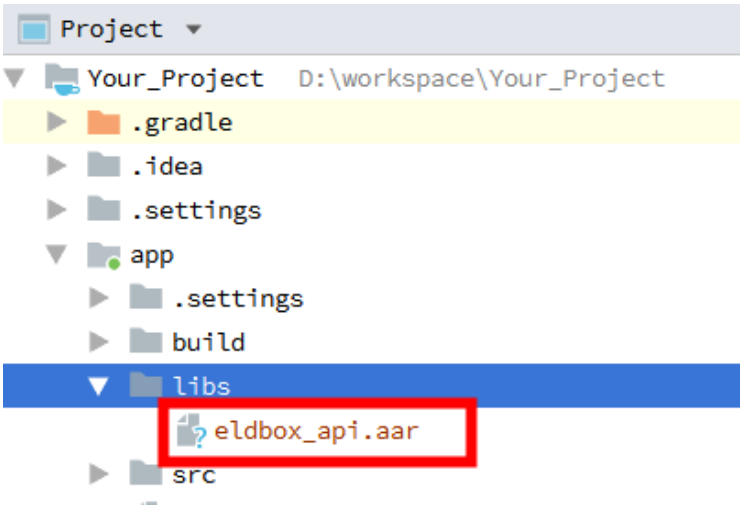

Figure 1

③ In build.gradle (Module:app)

Add the following code to android{} , as Figure 2 shown:

```
repositories {
    flatDir {
        dirs 'libs'
    }
}
```

Add the libraries you need to import in dependencies{}. As Figure 2 shown：

```
implementation(name: "eldbox_api", ext: "aar")
```

```
    }
    repositories {
        flatDir {
            dirs 'libs'
        }
    }
}

dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation(name: "eldbox_api", ext: "aar")
```

Figure 2

④Click Sync Now

1.4 Permissions and services added

1.4.1 ELD BOX-Bluetooth version（for Bluetooth box version）

1.4.1.1 Add permission

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.INTERNET" />
```

To ensure working properly, the above permissions should be declared in Manifest，and apply for it dynamically in the code (Limited to Android 6.0 up) :>

1.4.1.2 Ble Service

A BleService class provided in aar in order to call various APIs. The introduction of how to use this service as follows,

Add in the <application></application> tag of Manifest.xml:

```
<service android:name="com.android.eldbox_api.service.BleService"
android:enabled="true"
android:exported="false" />
```

①Start service

```
startService(new Intent(getApplicationContext(),BleService.class));
```

②Bind service

```
isBleBind = bindService(new Intent(getApplicationContext(),
BleService.class),bleServiceConnection, Context.BIND_AUTO_CREATE);
```

③Complete Service Connection

```
private ServiceConnection bleServiceConnection = new ServiceConnection() {

        @Override

        public void onServiceConnected(ComponentName name, IBinder service)
{

            if (bleService == null) {

                bleService = ((BleService.LocalBinder)
service).getService();

                bleService.startAutoConnect();
            }

        }

        @Override

        public void onServiceDisconnected(ComponentName name) {

            bleService = null;

        }

    };
```

④Unbind service

```
public void unBindAndStopService() {
        bleService.stopAutoConnect();
        if(isBleBind)
            unbindService(bleServiceConnection);
        stopService(new Intent(getApplicationContext(), BleService.class));
    }
```

Ensuring the services are stopped after the application exits is an effective way to reduce errors. Before stopping the service, disconnect it first.

1.4.2 ELD BOX-Serial Port version（for Docking station version）

1.4.2.1 Add permission

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.INTERNET" />
```

1.4.2.2 Serial Handle

```
//--------------Serial ELD-------------------

    public void disConnectSerial() {
        serialAgent.close();
        serialAgent = null;
    }

    public SerialAgent createSerialAgent() {
        serialAgent = new SerialAgent();
        return serialAgent;
    }
```

1.5 Using CallBack to get data

Serial Port Verison（for Docking station version）

```
private void openSerialPort(){
        if (((MApplication) getApplication()).serialAgent == null) {
            ((MApplication)
getApplication()).createSerialAgent().setmEldboxCallBack(mEldboxCallBack);
            try {
                ((MApplication)
getApplication()).serialAgent.openSerialPort().startListening();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
```

Bluetooth version（for Bluetooth box version ）

```
gotoConn.postDelayed(new Runnable() {
            @Override
            public void run() {
                ((MApplication)
getApplication()).getService().setCallBack(mEldboxCallBack);
            }
        }, 1000);
```

The Serial Port version and Bluetooth version share Callback. More details, see Class reference -->EldboxCallBack

```
EldboxCallBack mEldboxCallBack = new EldboxCallBack() {

    /*********    Bluetooth CallBack   *********/
        @Override
        public void onConnected() {

            //todo

        }

        @Override
        public void onDisconnected() {

            //todo

        }

    /*********    ELD Data CallBack   *********/

        @Override
        public void onReceiveVin(long timeStamp, String vin) {
            //todo

        }

        @Override
        public void onReceiveEngineHours(long timeStamp, long time) {
            //todo

        }

        @Override
        public void onReceiveTripDistance(long timeStamp, int distance) {

            //todo

        }

        @Override
        public void onReceiveRpm(long timeStamp, int rpm) {

            //todo

        }

        @Override
        public void onReceiveVss(long timeStamp, int Vss) {

            //todo

        }

        @Override
        public void onReceiveOdometer(long timeStamp, int odometer) {

            //todo

        }

        @Override
        public void onReceiveDTC(long timeStamp, byte[] dtc) {

            //todo

        }
```

```
        @Override
        public void onReceiveHighPrecisionOdometer(long timeStamp, long
highPrecisionOdometer) {

            //todo

        }

        @Override
        public void onReceiveHistoryData(Data data) {

            //todo

        }

        @Override
        public void onReceiveCmdData(byte[] data) {

        }

        @Override
        public void onReceiveVersion(String Version) {

            //todo

        }

        @Override
        public void onReceiveSleepDelay(int sleepDelay) {

            //todo

        }

        @Override
        public void onReceiveResponse(int responseType , byte[] response) {

            //todo

        }

        @Override
        public void onResponseNeedToSend(byte[] response) {

            //todo

        }

        @Override
        public void onReceiveOtherParameter(int paraType, long timeStamp,
long paraValue) {

            //todo

        }

    };
```

Note: We do not recommend a lot of work in the callback. If you need to handle a large number of transactions, you should first consider the operation in a new thread, and we do not recommend that you update ui directly in the callback.

3Rtablet provide custom api services, if the above CallBack does not have the required data, just contact us.

1.6 OTA Upgrade

Objective: To update the device firmware, needs to contact the developer to provide firmware and operation instructions.

1.6.1 Initialize OTA

1）OTA - Bluetooth version（for Bluetooth box version ）

The OTA upgrade function of the device is implemented based on BleService, so please make sure inititialize and bind BleService before using OTA upgrade funtion.

Public methods			
Function Name	Returned Value	Para meter	Functional Description
initialize(BleService var1, File var2, EldboxOTACallBack var3)	boolean		Initialize OTA

If initialization fails, please check you have properly initialized the BleService, EldboxOTACallBack and granted access to the file.

2）OTA - Serial Port Version（for Docking station version）

The device OTA upgrade function is based on the SerialAgent implementation, so please make sure that initialize the SerialAgent before using the OTA upgrade.

Public methods			
Function Name	Returned Value	Para meter	Functional Description
initialize(SerialAgent var1, File var2, EldboxOTACallBack var3)	boolean		Initialise OTA

If initialization fails, please check to initialize SerialAgent、EldboxOTACallBack correctly and grant permission to read files.

1.6.2 Finished EldboxOTACallBack

```
EldboxOTACallBack mEldboxOTACallBack = new EldboxOTACallBack() {

        @Override
        public void onReadyOTA() {
            //todo
        }
        @Override
        public void onOTA(int progress) {
            //todo
        }
        @Override
        public void onOTAFinish() {
            //todo
        }
        @Override
        public void onOTAStop(int error) {
            //todo
        }
    };
```

Note: We do not recommend a lot of work in the callback. If you need to handle a large number of transactions, you should first consider the operation in a new thread, and we do not recommend that you update ui directly in the callback.

1.6.3 Start OTA

We provide a very simple way to upgrade OTA, using the following code in a click event:

Bluetooth Version（for Bluetooth box version ）

```
case R.id.btn_start_ota: {
                    if
(BleOTAHelper.getInstance().initialize(((MApplication)getApplication()).getS
ervice(), otaFile, mEldboxOTACallBack)) {
                        BleOTAHelper.getInstance().startOTA();
                    }
                    break;
                }
```

Serial Port Version（for Docking station version）

```
case R.id.btn_start_ota: {
                    if
(SerialOTAHelper.getInstance().initialize(((MApplication)
getApplication()).serialAgent, otaFile, mEldboxOTACallBack)) {
                        SerialOTAHelper.getInstance().startOTA();
                    }
                    break;
                }
```

Before each OTA upgrade, please make sure that it is initialized, and a successful initialization corresponds to an OTA upgrade.

2 Type Reference

2.1 Service or Agent

2.1.1 BleService（for Bluetooth box version ）

Public methods			
Function Name	Returned Value	Parameter	Functional Description
connectDevice(BluetoothDevice device)	void		Connect the device. If the device is already connected, the previous connection will be disconnected.
disConnect()	void		Disconnect the device.
sendMessage(byte[] data)	void	data：Hex byte array sent	Send data(queries，setting,etc.)
~~**requestDataSync(int address)**~~	~~boolean~~ ~~If there is no connected device, return to false, send successfully return true.~~	~~address：synchronous start address~~	~~Request data synchronization. Abandoned~~
requestDataSync(int begin, int end) throws Exception	boolean If there is no connected device, return to false, send successfully return true.	begin: Begin start time (minutes) end: End time (minutes) Send the time start and end range, in minutes, that needs to be synchronized. The time range does not exceed eight days. For example, you need to synchronize data within 12 hours, begin is 720,end is 0	Request data synchronization Note: Please use the onReceiveResponse callback to get the return value. Return Type is DataSyncResponse(0x0003)
requestVersion()	boolean If there is no connected device, return to false, send successfully return true.		Get the information of the device firmware version. Note：Callback the verion by using onReceiveVersion

Public methods			
requestTimeSync(long time)	boolean If there is no connected device, return to false, send successfully return true.	time:UTC time Range of values: 0 to 4294967295000, unit: millisecond Note: the incoming value will be divided by 1000 and intercepted only for seconds	Initiate time synchronization, it is recommended that the device time be set to NTP network time Note: Use onReceiveResponse callback to get the return values. Return type is TimeSyncRespons
requestDeviceSN()	boolean If there is no connected device, return to false, send successfully return true.		Request device SN code Note: Use onReceiveResponse callback to get the return value, Return type isDeviceSNResponse (0x6008)
requestDTC()	boolean If there is no connected device, return to false, send successfully return true.		Request device failure code Note: It takes a waiting time to get the fault code from the onReceiveDTC callback.
requestClearDTC()	boolean If there is no connected device, return to false, send successfully return true.		Clear the current failure code of the vehicle
requestSleepDelay()	boolean If there is no connected device, return to false, send successfully return true.	After the general vehicle flameout into dormancy, the equipment can not communicate at this time; It takes 6 to 7 seconds for the equipment to work properly by detecting vibration wake-up.	Get the device dormancy time, default 5 minutes; Note : use onReceiveSleepDelay callback to get dormancy time
setSleepDelay(int time)	boolean If there is no connected device, return to false, send successfully return true.	time: Dormancy time Value range: 1~255， unit：minute	Set device sleep delay
~~**setStorageMode(boolea**~~	~~boolean~~	~~True: the deive always~~	~~Set the storage~~

Public methods			
~~n always)~~	~~If there is no connected device, return to false, send successfully return true.~~	~~stores the information.~~ ~~False: the device stores information when disconnected.~~ ~~Always stores by default~~	~~mode~~ ~~Abandon~~
requestWorkMode()	boolean If there is no connected device, return to false, send successfully return true.		Request the current working mode of the device. Note:Use onReceiveResponse callback to get the return value, Return Type is WorkModeResponse(0x6005)
requestBusProtocol()	boolean If there is no connected device, return to false, send successfully return true.		Request for the current BusProtocol, only available for J1939. Note: Use onReceiveResponse callback to get the return value. Return type is BusProtocolResponse (0x6009)
requestFuelLevel()	boolean If there is no connected device, return to false, send successfully return true.		Request fuel level, only available for J1939 and J1587. Note: Use onReceiveOtherParameter callback to get the return value. Return type is DataTypeNotSync.FuelLevel（0xD009）
setWorkMode(WorkMode workMode)	boolean If there is no connected device, return to false, send successfully return true.	workMode:work mode Range of values: CommandEnum.WorkMode.ELD_MODE : ELD Mode CommandEnum.WorkMo	Set (switch) the docking station operating mode; Under the ELD working mode, the docking station automatically

Public methods			
		de.CMD_MODE : CMD Mode `CommandEnum.WorkMode.PTR_MODE : PTR Mode`	acquires and transmits data; The original data is packetized for transmission in command mode; Under the passthrough mode,all raw data will be transferred directly without any processing. Please use the passthrough mode APP。 Note: Use onReceiveResponse callback to get the return value. Return type is WorkModeSetResultResponse(0x0005)
setBusProtocol(BusProtocol busProtocol)	boolean If there is no connected device, return to false, send successfully return true.	busProtocol: the value range: CommandEnum.BusProtocol.J1587_Protocol: use J1587 protocol CommandEnum.BusProtocol.J1939_Protocol : use J1939 protocol.	Set Bus Protocol, only available for J1939. Note: Use onReceiveResponse callback to get the return value. Return type is BusProtocolSetResultResponse(0x0009)
writeDeviceSN(byte[] snString)	boolean If there is no connected device, return to false, send successfully return true.	snString: device SN code Value range: the length of the character does not exceed 32	Write SN to device Note:Use onReceiveResponse callback to get the return value, Return type is DeviceSNSetResultResponse (0x0008)
writeCMDString(byte[] cmdString)	boolean If there is no connected device, return to false, send successfully return true.	cmdString:The command used by cmdString: to query CAN data, sent in binary form "at dp\r" "at rv\r" `Note: Please send`	Send commands in command mode to query CAN data

Public methods			
		"\r" using the hexadecimal of 0x0d	
~~setGpioOutHigh(byte[] gpioString)~~	~~boolean~~ ~~If there is no connected device, return to false, send successfully return true.~~	~~gpioString:~~ ~~* 2 bytes, high byte in the front. bit15 ~bit0 correspond GPIO16~GPIO1~~ ~~* If the bit is "1", the corresponding GPIO port is set to high.~~	~~Set GPIO output high, currently only OBD is available.~~ ~~Note: Use onReceiveResponse callback to get the return value,~~ ~~Return type is Response.SetGpioOutHighSetResultResponse(0XE001)~~
~~setGpioOutLow(byte[] gpioString)~~	~~boolean~~ ~~If there is no connected device, return to false, send successfully return true.~~	~~gpioString:~~ ~~* 2 bytes, high byte in the front. bit15 ~bit0 correspond GPIO16~GPIO1 ;~~ ~~* If the bit is "1", the corresponding GPIO port is set to low.~~	~~Set GPIO output low, currently only OBD is available.~~ ~~Note: Use onReceiveResponse callback to get the return value,~~ ~~Return type is Response.SetGpioOutLowSetResultResponse(0XE002)~~
~~requestGpioVal(byte[] gpioString)~~	~~boolean~~ ~~If there is no connected device, return to false, send successfully return true.~~	~~gpioString:~~ ~~* 2 bytes, high byte in the front. bit15 ~bit0 correspond GPIO16~GPIO1 ;~~ ~~* You can select to query the corresponding GPIO status:~~ ~~If the corresponding bit is "1", query the GPIO value, and 0xFFFF for all queries.~~	~~Get GPIO status, currently only OBD is available.~~ ~~Note: Use onReceiveResponse callback to get the return value,~~ ~~Return type is Response.GetGpioValResponse(0XE003)~~
~~requestAdcVal(int channel)~~	~~boolean~~ ~~If there is no connected device, return to false, send successfully return true.~~	~~channel:~~ ~~Get the ADC value of the corresponding channel~~	~~Obtain the ADC value of the corresponding channel, currently only OBD is available.~~ ~~Note: Use~~

Public methods			
		range in 0~3	onReceiveResponse() callback to get the return value, Return type is Response.GetAdcValResponse(0xE004)
setPwrOut(boolean enable)	boolean If there is no connected device, return to false, send successfully return true.	enable: true: power output enable false:power output disable	External output voltage control, currently only OBD is available. Note: Use onReceiveResponse() callback to get the return value, Return type is Response.SetPwrOutSetResultResponse(0xE005)
computeAdcVal(int data, float vref, float deviation)	float return adc value Voltage calculation formula is Vin=((DATA)*Vref/4096) *106.8/6.8	data：the ADC value of the corresponding channel. Note: Use onReceiveResponse() callback to get the return value, Return type is Response.GetAdcValResponse(0xE004) vref：Reference voltage value deviation：For example, when calculating value for channel 3, after calculating the input voltage, because the circuit has diode voltage drop, it needs to increase about 0.5V to get the correct value	Calculate ADC value, currently only OBD is available
computeAdcVal_for_channel3(int data)	float return adc value Voltage calculation formula is Vin=((DATA)*Vref/	data：the ADC value of the corresponding channel. Note: Use onReceiveResponse()callback to get the return	Calculate ADC value for channel 3, deviation + 0.5V. Currently only OBD is available

Public methods			
	~~4096) *106.8/6.8~~	~~value,~~ ~~Return type is Response.GetAdcValResponse(0xE004)~~	
~~computeAdcVal(int data)~~	~~float~~ ~~return adc value~~ ~~Voltage calculation formula is Vin=((DATA)*Vref/4096) *106.8/6.8~~	~~data：the ADC value of the corresponding channel.~~ ~~Note: Use onReceiveResponse()callback to get the return value,~~ ~~Return type is Response.GetAdcValResponse(0xE004)~~	~~Calculate ADC value, currently only OBD is available~~
getState()	int disconnected：BluetoothProfile.STATE_DISCONNECTED connected：BluetoothProfile.STATE_CONNECTED		Returns Bluetooth connection status
getDevice()	BluetoothDevice Returns the device being connected, and returns null if no device being connected.		Returns the device being connected.

2.1.2 SerialAgent（for Docking station version）

Public methods			
Function Name	Returned Value	Parameter	Function Description
openSerialPort() throws IOException	SerialAgent		Open Serial Port.
close()	void		Close Serial Port.
writeData(byte[] bArr) throws IOException	void	bArr：write data, byte array	Send data(queries,setting,etc.)

Public methods			
~~**requestDataSync(int address) throws IOException**~~	~~boolean~~ ~~If it fails, it returns false, and the success returns true.~~	~~address：synchronous start address~~	~~Request data synchronization. Abandoned~~
requestDataSync(int begin, int end) throws IOException	boolean If it fails, it returns false, and the success returns true.	begin: Begin start time (minutes) end: End time (minutes) Send the time start and end range, in minutes, that needs to be synchronized. The time range does not exceed eight days. For example, you need to synchronize data within 12 hours, begin is 720,end is 0	Request data synchronization Note: Please use the onReceiveResponse callback to get the return value. Return Type is DataSyncResponse(0x0003)
requestVersion() throws IOException	boolean If it fails, it returns false, and the success returns true.		Get device firmware version information Note: Please use onReceiveVersion callback to get the version.
requestTimeSync((long time)) throws IOException	boolean If it fails, it returns false, and the success returns true.	time:UTC time Range of values: 0 to 4294967295000, unit: millisecond Note: the incoming value will be divided by 1000 and intercepted only for seconds	Initiate time synchronization, it is recommended that the device time be set to NTP network time Note: Use onReceiveResponse callback to get the return values. Return type is TimeSyncResponse(0x0001)
requestDeviceSN() throws IOException	boolean If it fails, it returns false, and the success returns true.		Request device SN code Note: Use onReceiveResponse callback to get the return value, Return type isDeviceSNResponse (0x6008)

Public methods			
requestDTC() throws IOException	boolean If it fails, it returns false, and the success returns true.		Request device failure code Note: It takes a waiting time to get the fault code from the onReceiveDTC callback.
requestClearDTC() throws IOException	boolean If it fails, it returns false, and the success returns true.		Clear the current failure code of the vehicle
requestSleepDelay() throws IOException	boolean If it fails, it returns false, and the success returns true.	After the general vehicle flameout into dormancy, the equipment can not communicate at this time; It takes 6 to 7 seconds for the equipment to work properly by detecting vibration wake-up.	Get the device dormancy time, default 5 minutes; Note : use onReceiveSleepDelay callback to get dormancy time
setSleepDelay(int time) throws IOException	boolean If it fails, it returns false, and the success returns true.	time: Dormancy time Value range: 1~255， unit：minute	Set device sleep delay
~~**setStorageMode(boolean always) throws IOException**~~	~~boolean~~ ~~If it fails, it returns false, and the success returns true.~~	~~True: the deive always stores the information.~~ ~~False: the device stores information when disconnected.~~ ~~Always stores by default~~	~~Set the storage mode~~ ~~Abandon~~
requestWorkMode() throws IOException	boolean If it fails, it returns false, and the success returns true.		Request the current working mode of the device. Note:Use onReceiveResponse callback to get the return value. Return Type is WorkModeResponse(0x6005)
requestBusProtocol() throws IOException	boolean If it fails, it returns false, and the success returns		Request for the current BusProtocol, only available for J1939.

Public methods			
	true.		Note:Use onReceiveResponse callback to get the return value. Return type is BusProtocolResponse (0x6009)
requestFuelLevel() throws IOException	boolean If it fails, it returns false, and the success returns true.		Request fuel level, only available for J1939 and J1587. Note: Use onReceiveOtherParameter callback to get the return value. Return type is DataTypeNotSync.FuelLevel（0xD009）
setWorkMode(WorkMode workMode) throws IOException	boolean If it fails, it returns false, and the success returns true.	workMode:work mode Range of values: CommandEnum.WorkMode.ELD_MODE : ELD Mode CommandEnum.WorkMode.CMD_MODE : CMD Mode CommandEnum.WorkMode.PTR_MODE : PTR Mode	Set (switch) the docking station operating mode; Under the ELD working mode, the docking station automatically acquires and transmits data; The original data is packetized for transmission in command mode; Under the passthrough mode,all raw data will be transferred directly without any processing. Please use the passthrough mode APP。 Note:Use onReceiveResponse callback to get the return value. Return type is WorkModeSetResultResponse(0x0005)

Public methods			
setBusProtocol(BusProtocol busProtocol) throws IOException	boolean If it fails, it returns false, and the success returns true.	busProtocol: the value range: CommandEnum.BusProtocol.J1587_Protocol: use J1587 protocol CommandEnum.BusProtocol.J1939_Protocol : use J1939 protocol.	Set Bus Protocol, only available for J1939. Note:Use onReceiveResponse callback to get the return value. Return type is BusProtocolSetResultResponse(0x0009)
writeDeviceSN(byte[] snString) throws IOException	boolean If it fails, it returns false, and the success returns true.	snString: device SN code Value range: the length of the character does not exceed 32	Write SN to device Note:Use onReceiveResponse callback to get the return value, Return type DeviceSNSetResultResponse (0x0008)
writeCMDString(byte[] cmdString) throws IOException	boolean If it fails, it returns false, and the success returns true.	cmdString:The command used by cmdString: to query CAN data, sent in binary form "at dp\r" "at rv\r" `Note: Please send "\r" using the hexadecimal of 0x0d`	Send commands in command mode to query CAN data
setGpioOutHigh(byte[] gpioString) throws IOException	boolean If it fails, it returns false, and the success returns true.	gpioString: * 2 bytes, high byte in the front. bit15 ~bit0 correspond GPIO16~GPIO1 * If the bit is "1", the corresponding GPIO port is set to high.	Set GPIO output high, currently only OBD is available. Note: Use onReceiveResponse callback to get the return value, Return type is Response.SetGpioOutHighSetResultResponse(0XE001)
setGpioOutLow(byte[] gpioString) throws IOException	boolean If it fails, it returns false, and the success returns true.	gpioString: * 2 bytes, high byte in the front. bit15 ~bit0 correspond GPIO16~GPIO1 ;	Set GPIO output low, currently only OBD is available. Note: Use

Public methods			
		* If the bit is "1", the corresponding GPIO port is set to low.	onReceiveResponse callback to get the return value, Return type is Response.SetGpioOutLowSetResultResponse(0XE002)
requestGpioVal(byte[] gpioString) throws IOException	boolean If it fails, it returns false, and the success returns true.	gpioString: * 2 bytes, high byte in the front. bit15 ~bit0 correspond GPIO16~GPIO1 ; * You can select to query the corresponding GPIO status: If the corresponding bit is "1", query the GPIO value, and 0xFFFF for all queries.	Get GPIO status, currently only OBD is available. Note: Use onReceiveResponse callback to get the return value, Return type is Response.GetGpioValResponse(0XE003)
requestAdcVal(int channel) throws IOException	boolean If it fails, it returns false, and the success returns true.	channel: Get the ADC value of the corresponding channel range in 0~3	Obtain the ADC value of the corresponding channel, currently only OBD is available. Note: Use onReceiveResponse() callback to get the return value, Return type is Response.GetAdcValResponse(0xE004)
setPwrOut(boolean enable) throws IOException	boolean If it fails, it returns false, and the success returns true.	enable: true: power output enable false:power output disable	External output voltage control, currently only OBD is available. Note: Use onReceiveResponse() callback to get the return value, Return type is Response.SetPwrOutSetResultResponse(0xE005)
computeAdcVal(int	float	data：the ADC value of	Calculate ADC

Public methods			
data, float vref, float deviation)	return adc value Voltage calculation formula is Vin=((DATA)*Vref/4096) *106.8/6.8	the corresponding channel. Note: Use onReceiveResponse() callback to get the return value, Return type is Response.GetAdcValResponse(0xE004) vref：Reference voltage value deviation：For example, when calculating value for channel 3, after calculating the input voltage, because the circuit has diode voltage drop, it needs to increase about 0.5V to get the correct value	value, currently only OBD is available
computeAdcVal_for_channel3(int data)	float return adc value Voltage calculation formula is Vin=((DATA)*Vref/4096) *106.8/6.8	data：the ADC value of the corresponding channel. Note: Use onReceiveResponse()callback to get the return value, Return type is Response.GetAdcValResponse(0xE004)	Calculate ADC value for channel 3, deviation + 0.5V. Currently only OBD is available
computeAdcVal(int data)	float return adc value Voltage calculation formula is Vin=((DATA)*Vref/4096) *106.8/6.8	data：the ADC value of the corresponding channel. Note: Use onReceiveResponse()callback to get the return value, Return type is Response.GetAdcValResponse(0xE004)	Calculate ADC value, currently only OBD is available

2.2 OTAHelper

2.2.1 BleOTAHelper

Bluetooth version（for Bluetooth box version ）

Public methods			
Function Name	Returned Value	Parameter	Funtional Description
getInstance()	BleOTAHelper		Get BleOTAHelper Object
initialize(BleService,File,EldboxOTACallBack)	boolean If it fails, it returns false, and the success returns true.		Initialize OTAHelper
startOTA()	void		Start OTA upgrade

2.2.2 SerialOTAHelper

Serial Port version（for Docking station version）

Public methods			
Function Name	Returned Value	Parameter	Funtional Description
getInstance()	SerialOTAHelper		Get SerialOTAHelper Object
initialize(SerialAgent serialAgent2, File file2, EldboxOTACallBack eldboxOTACallBack)	boolean If it fails, it returns false, and the success returns true.		Initialize OTAHelper
startOTA()	void		Start OTA Upgrade

2.3 EldboxOTACallBack

Constants	
Variable Name	Functional Description
int SEND_DATA_TO_DEVICE_FAILED	Failed to send data
int REQUEST_DEVICE_OTA_FAILED	Request device OTA failed.
int DEVICE_CHECK_OTA_FILE_FAILED	Device check file failed after OTA file transfer is complete.

Public methods			
Function Name	Returned Value	Parameter	Functional Description
onReadyOTA()	void		Preparing resources for OTA
onOTA(int progress)	void	Progress : range:0-100	OTA upgrade in progress
onOTAFinish ()	void		OTA upgrade finished
onOTAStop(int error)	void		OTA terminates, error is the error code, see constants.

2.4 EldboxCallBack

The callback function, which receives the data from the API library.

Public methods				
Function Name	Return Value	Callback parameter	Span	Functional description
onConnected()	void			Device connection complete
onDisconnected()	void			Device disconnected
onReceiveVin(long timeStamp, String vin)	void	timestamp:timestamp vin: Received engine code		Vin code receive
onReceiveEngineHours(long timeStamp, long time)	void	timestamp:timestamp time: Engine hours received, needs to divide by 20 for two decimal places. See span.	J1939 value range: 0 to 4,211,081,215, unit: 0.05h (data not resolved, upper layer needs to be divided by 20) OBD range: 0 ~ 65,535 * 20, Unit: 0.05s (for compatible processing J1939, data * 20, the upper layer needs to be divided by 20)	Engine hours received

Public methods				
onReceiveTripDistance (long timeStamp, int distance)	void	timeStamp:timeStamp distance:trip distance received in kilometers	J1939 in the range: 0 ~ 526,385,151, unit: km OBD in the range: 0~65,535 (unit: km)	Trip distance received
onReceiveOdometer(long timeStamp, int odometer)	void	timestamp:timestamp odometer:odometer received,in kilometers	J1939 in the range: 0 ~ 526,385,151, unit: km OBD has no odometer parameter	Odometer received Only J1939
onReceiveRpm(long timeStamp, int rpm)	void	timestamp:timestatmp rpm:the rotational speed, unit: r / min	J1939 in the range: 0 ~ 8031, unit: rpm OBD range: 0 to 16,383, unit: rpm	The rotational speed received
onReceiveVss(long timeStamp, int Vss)	void	timestamp:timestamp Vss:speed,unit:km/h	J1939 in the range: 0 ~ 250, unit: km/h OBD range: 0 to 255, unit: km/h	Received the speed
onReceiveDTC(long timeStamp, byte[] dtc);	void	timestamp: timeStamp dtc:vehicle current failure code		Received the current fault code of the vehicle
onReceiveHighPrecisionOdometer(long timeStamp, long highPrecisionOdometer)	void	timestamp:timeStamp highPrecisionOdometer:High precision odometer	J1939 in the range: 0 ~ 21,055,406,000, unit: m OBD does not have this parameter	High precision odometer received,only J1939
onReceiveResponse(int responseType , byte[] response)	void	responseType: Response type VerifyResponse(0x7001) TimeSyncResponse(0x0001) DataSyncResponse(0x0003) SleepDelayResponse(0x0		Answer type and answer data received from the device

Public methods				
		004) WorkModeSetResultResponse(0x0005) DeviceSNSetResultResponse(0x0008) BusProtocolSetResultResponse(0x0009) WorkModeResponse(0x6005) DeviceSNResponse(0x6008) BusProtocolResponse(0x6009) SetGpioOutHighSetResultResponse（0xE001） SetGpioOutLowSetResultResponse（0xE002） GetGpioValResponse（0xE003） GetAdcValResponse（0xE004） SetPwrOutSetResultResponse(0xE005） response：Response data from the device VerifyResponse(0x7001) 0x00: Data reception check failed 0xFF: Data reception check is correct TimeSyncResponse(0x0001) 0x00: failed to set synchronization time 0xFF: The synchronization time is set successfully. DataSyncResponse(0x0003)		

Public methods				
		0x00: Request synchronization failed 0xFF: Request synchronization succeeded 0xAA: Data synchronization ends SleepDelayResponse(0x0004) 0x00: failed to set hibernation time 0xff: Set hibernation time successfully WorkModeSetResultResponse(0x0005) 0x00：Set work mode failed 0xff：Set work mode successful DeviceSNSetResultResponse(0x0008) 0x00: failed to set SN code 0xff: Set SN code successfully BusProtocolSetResultResponse(0x0009) 0x00：Set Bus protocol failed. 0xff：Set Bus protocol successful WorkModeResponse(0x6005) 0x00：ELD Mode 0xFF：CMD Mode 0xDD：PTR Mode DeviceSNResponse(0x6008)		

Public methods				
		n bytes: String, such as: VT7AJN940910001 BusProtocolResponse(0x6009) 0x00：J1587 protocol 0xFF：J1939 protocol SetGpioOutHighSetResultResponse（0xE001） 0x00：Failed to set GPIO output high 0xFF： Set GPIO output high successfully SetGpioOutLowSetResultResponse（0xE002） 0x00：Failed to set GPIO output low 0xFF：Set GPIO output low successfully GetGpioValResponse（0xE003） 4 bytes： ①1-2 bytes are GPIO input status request parameters: bit15 ~bit0 corresponding to gpio16-gpio1; (high byte in the front) If the corresponding bit is "1", query the GPIO value, and 0xFFFF for all queries ②3-4 bytes is GPIO status value: bit15 ~bit0 corresponding to gpio16-gpio1; (high		

Public methods				
		byte in the front) If the corresponding bit is "1", the corresponding GPIO status is high; "0", the corresponding GPIO status is low GetAdcValResponse（0xE004） 3 bytes；(high byte in the front) 1st byte ADC channel value The 2-3 bytes are ADC values: ① Correct: returns the ADC converted value. ② Error: return 0xFFFF Get the ADC value of the corresponding channel The reference voltage is Vref, and the voltage calculation formula is Vin=(（DATA）*Vref/4096) *106.8/6.8 For example, when calculating channel 3, after calculating the input voltage, because the circuit has diode voltage drop, it needs to increase about 0.5V to get the correct value You can use the three functions computeAdcVal() to get the corresponding ADC value. SetPwrOutSetResultResponse(0xE005） 0x00：Enable / close power out output failed		

Public methods				
		0xFF：Enable / close power out output succeeded		
onReceiveOtherParameter(int paraType, long timeStamp, long paraValue)	void	paraType：Parameter type timeStamp：time stamp paraValue：Parameter value paraType： DataTypeNotSync.FuelLevel(0xD009)	paraType 为 DataTypeNotSync.FuelLevel * J1939 in the range: 0 ~ 100, unit: %) * J1587 in the range: 0 ~ 127.5, unit: % ,here use an integer, i.e. 0~127 * OBD does not have this parameter	Other parameters received
onResponseNeedToSend(byte[] response)	void	response：Response data that needs to be sent to the device, complete data to be sent		response：Response data that needs to be sent to the device, complete data to be sent
onReceiveSleepDelay(int sleepDelay)	void	sleepDelay:Time of sleep received, in minutes.		Received device dormancy time
onReceiveVersion(String Version)	void	Version: Received version number		Received device version number

onReceiveHistoryData(Data data)	void	data : eld data
		public class Data {
		public int rpm = 0;
		public int vss = 0;
		public long timeStamp = 0;
		public int tripDistance = 0;
		public int odometer =
				Received device history data,data within 8*24 hours.

Public methods				

		0;//only J1939
		public long enginehours = 0; //Needs to divide by 20 for two decimal places.
		public long highPrecisionOdometer = 0;//only J1939
		}		

onReceiveCmdData(byte[] data)	void	data:Raw CAN data, you need to convert byte [] to an ASCII string		Received CAN raw data (string format) without parsing

Note:The time stamp value of EldboxCallBack callback function always range: 0 to 4294967295, unit: s

Note: Generally speaking, the vehicle information read by the device is sent to the client in real time, but when the device is disconnected or the storage mode is set to always store, a part of the past data is stored in the device. When the client initiates synchronization, the client will receive this history information in onReceiveHistoryData().

3 Others

3.1 Unable to discover Bluetooth device

When Android 6.0 or higher is developed, due to Android restrictions, only apps that have been granted location access can scan nearby devices. So if the device cannot be found, check the permissions.

3.2 Link Supervision timeout

Android default link supervision timeout is 20s, that is, within 20s after Bluetooth disconnects, it will receive the disconnect message. Modify the Bluetooth time, you need to modify the system, please inform us

3.3 Time Stamp

The api that receives the data contains a time stamp, which refers to the (device) time corresponding to this information. If there is no time synchronization, this time will be different from the time of the tablet.The time stamp of the data read after the time synchronization is the same as the tablet data. The data before time synchronization is not the same, so you need to perform time synchronization after connecting the device. Therefore, it is necessary to perform the time synchronization operation after the device is connected.

3.4 Sleep

Equipment sleep is an important measure to reduce energy consumption.When the device goes to sleep, the bluetooth connection will not be disconnected, but the data transmission and reception will stop. Default sleep time is 5 minutes, that is means the device will be enter the sleep state when vehicle rpm is zero within 5 minutes. If necessary, you can modify this sleep time, but not less than 1 minute.

3.5 Odometer

If device can't read the odometer from OBD, it needs the user to input manually, and then accumulate a trip distance.