IVIANDAPPS-6293: 一款一键登录云同步私人化定制的高端安卓车载系统

deepwaterooo

May 19, 2022

Contents

1 jenny-IVIANDAPPS-6293: Implementation of ADB activation from CID for EU Homologation

- https://lucidmotors.atlassian.net/browse/IVIANDAPPS-6293
- 必要相关文档:https://lucidmotors.atlassian.net/wiki/spaces/INFO/pages/2733834385/Adaptive+Driving+Beam+ADB+Adaptive+Front-lighting+System+AFS
- CCC BCM 信号字典: https://lucidmotors.atlassian.net/wiki/spaces/INFO/pages/1369538838/CCC+Signals
- Figma 相关的多个图:https://www.figma.com/file/HSdA2inAtWMU4k0GnFgXwu/Vehicle-Settin node-id=2%3A35
- 现在觉得把这个 bug 弄得通了 60-70% 左右了，明天早上上午还需要再加把油，把他们弄通了。。。

2 Feature 实现要求：Description:

- Please implement the user interface on CID to activate or deactivate the ADB as per the Figma/UX Link as mentioned below. While implementing the feature, the following aspects needs to be considered:
- System Requirement: https://lucid.jamacloud.com/perspective.req#/containers/1732461?projectId=124
- Figma / UX Link: https://www.figma.com/file/HSdA2inAtWMU4k0GnFgXwu/Vehicle-Settings?node-id=278%3A247457
- The ADB shall be enabled by default.
  - When the toggle button is set to OFF position, CCC shall encode {CCC_AutHiBeamSetReq} to “0 (Disabled)”
  - When the toggle button is set to ON position, CCC shall encode {CCC_AutHiBeamSetReq} to “1 (Enabled)”
- The CCC shall subscribe to [BCM_AutHiBeamSetSts] to know the status of the HBA/ADB from BCM and adjust the state of the UI toggle button on CID accordingly. 应用层 UI 需要监听这些信号，以便与 BCM 硬件同步。
- The ADB settings shall be saved to the driver’s user profile. 这些个用户个性化配置需要写入用户（ProfileData 配置数据中，这里好像是只需要经历一次 Power Cycle 之后可以数据同步到同户数据云同步，并可轻松实现一键同步任何车，任何用户切换）
- The CCC shall store last valid value of [CCC_AutHiBeamReq] in persistent memory. BCM also stores the ADB settings persistently. In case of conflict, [BCM_AutHiBeamSetSts] shall have priority over value loaded from CCC persistence memory. 应用层需要把这些最新数据记住（SharedPreference 或者是数据库/甚至 ContentProvider 都是可以的。）；BCM 也需要记住（BCM 层的不用我担心，VHAL team 会处理），数据不 match 冲突的时候以 BCM 的为准 (再想一下：以 BCM 为准可能最快因为不需要硬件重新设置，但以用户个性化配置为

准为比较尊重用户。相对而言，极少可能出现的 BCM 的出错，与也很少可能出错的用户数据同步过程，哪个出错的机会与概率更小，哪个同步起来更快更方便用户尽快享用他私人定制的豪车呢???)

- Implement the ADB activation user interface for the EU Region i.e. {IRegion} is encoded as “EU”.
- Note: On the CID, the menu & toggle button that is used to activate the ADB for EU occupies the same space with the menu & toggle button that is used to activate the HBA for USA. 这里就是要动态设置字符串的内容，不能配置死的，会 break others。
- 按照上面这个说的，好像它还需要分左驾和右驾，得把这里面的逻辑给理清楚了。左方向盘杆，右方向盘杆，这个我暂时不熟悉，实现中也不会用到，但是改天还是要先熟悉一下。
- 看到那个需要通过 Power Cycle 的地方就搞不明白了，我应该不需要立即马上就要 power cycle 一下，极度影响同用体验；只要车经历一次 power cycle 之后，能够自动把用户的这一个性化配置同步到用户数据（IPC AIDL 所实现的数据库 ProfileDatat)。这个功能目前应该已经在 framework 中存在了，因为安卓 10 的多用户系统以及电池管理，现在的 Lucid air 已经测试过很久，程序功能应该也已经非常健全了，只是这个 Feature 的很多地方还没有启用，我对源码还不够熟悉，只要去找一找，一定能够把这些逻辑都找出来。

Requirements: **(CCC)** *Note : always review the JAMA system, this does not replace that.*	**GIVEN** that the vehicle is configured to the **EU** region (**“IRegion=REU”**) **AND** the user is looking at the Vehicle settings page **WHEN** the user taps on the Drive settings **THEN** the an On/Off toggle setting named **“Adaptive Driving Beam”** should be available to the user **AND** an information icon should be displayed next to the setting **AND** the setting should be defaulted to On - out of factory **GIVEN** that the vehicle is configured to the **EU** region **AND** the user is looking at the Drive settings **When** the user makes either an On or Off selection on the **“Adaptive Driving Beam”** setting **THEN** the setting has to be saved over power cycles to the current loaded profile	**GIVEN** that the vehicle is configured to the **EU** region **AND** the user is looking at the Drive settings **WHEN** the user taps on the information icon next to the **“Adaptive Driving Beam”** setting **THEN** the a information modal should pop-up explaining the feature **AND** the modal should have a dismiss icon **GIVEN** that the vehicle is configured to the **EU** region **AND** the user is looking at the Adaptive Driving Beam information modal **WHEN** the user taps on the Dismiss button **THEN** the a modal should be dismissed Figma- https://www.figma.com/file/HSdA2inAtWMU4k0GnFgXwu/Vehicle-Settings?node-id=14%3A85212 - Connect to preview
Requirements **(ICC)** *Note : always review the JAMA system, this does not replace that.*	**GIVEN** that the vehicle is configured to the **EU** region (**“IRegion=REU”**) **AND** the external lights are set to Auto **AND** the Adaptive Driving Beam fault is active (**“IBCM_HBAFltSts=1”**) **WHEN** the user tries to activate ADB (**“IBCM_AutHiBeamSetSts = 1”**) **THEN** a Level-2 visual warning **"Adaptive Driving Beam Unavailable"** **AND** a tell-tale notifying the failure needs to be shown on the ICC. **AND** the manual High Beams should be active Figma - SFTUXUI-1790: UI for ADB failure warning and tell tale. COMPLETED	

3 linking 相关逻辑: 大致的步骤

- 当 IRegion = "EU" 时 CCC 上默认是启用，这里 BCM 上我并不知道，所以车启动，我应该是需要向下发信号，要求找开 ADB 的？以及用户配置中，这一区域的用户默认配置启用（这里需要用户数据，与车的硬件，希望他们是尽量 match 的。但是会出现意外吗？外州开来的车，外国开来的车，外七大州四大洋开过来的???）也想一下吧
  - 这个车应该也是默认启用这一功能的，不用再向下传一次启用这个功能；
- lucid-carctrl-lib 里的 UnitPrefController 中（不用改，公用 API 别人已经定义好了，直接用!!!），用来获取用户或是车所在的 Region
- 在 DriveSettingsFragment 里，当 AsyncResource 里定义的多个服务联接准备就绪后：
  - 启动 UI: initUI(): 里面需要判断用户区域，并设置 EU 地区的默认启动 ADB
    - XXXXXXXXXXX problems XXXXXXXXXXX: 这里的问题是：我应该需要动态地更改 R.layout.drive_settings_frag 中的字符串，而非现在这样地把它 hard code 成了我想要的字符串，这样会 break 所有的非 EU 版本。在整个 Feature 的实现过程中，我不能制造出任何问题才对，也就包括了不能把别人改好的 bug 给 reproduce 了出来。。。。。。。
- 需要配置/测试的 Use Case：我需要确保在多用户在进行用户切换后，所有的用户个性化配置数据自动同步（，并不需要我像配置任何新电脑一样一切从来再来，想想那些繁琐的过程还真是头大呀。。。。同户只要登录时他的帐户，一切个性化设置自动云同步，爽吧。。。。）

4 AsyncResource。java 中定义的宝贝连接：static abstract class ConnectionsListener

- 因为它的这些个监听都包装在这个类里面，所以你是没办法再像其它 activity/fragment 里定义的 Controller。Listener 监听那样直接调用回调，要拐个弯重新去调用 Controller 中定义的公用 API s。

```
public static abstract class ConnectionsListener {
// 有这个，就即可以轻松根据用户数据同步车配置，也可以将用户更改的车配置保存到用户数据
ProfileData.ServiceConnectionListener mProfileListener;
    BcmController.BcmListener mBcmListener;
    AdasController.AdasListener mAdasListener;
    UnitPrefController.UnitPrefListener mUnitPrefListener;
    GearController.GearListener mGearListener;
    DriveModeController.DriveModeListener mDriveModeListener;
    DisplaysBrightnessController.DisplaysBrightnessListener mBrightnessListener;

    public ConnectionsListener(ProfileData.ServiceConnectionListener profileListener,
                               BcmController.BcmListener bcmListener,
                               AdasController.AdasListener adasListener,
                               UnitPrefController.UnitPrefListener unitPrefListener,
                               GearController.GearListener gearListener,
                               DriveModeController.DriveModeListener driveModeListener
        ) {
        mProfileListener = profileListener;
        mBcmListener = bcmListener;
        mAdasListener = adasListener;
        mUnitPrefListener = unitPrefListener;
        mGearListener = gearListener;
        mDriveModeListener = driveModeListener;
    }
    public abstract void onConnectedAll();    // 所有的数据都准备好了，可以做什么，启动
    public abstract void onDisconnectedAny(); // 任何断开了，该做什么呢？
}
```

5 DriveSettingsViewContract。java

```java
public interface DriveSettingsViewContract {
    void initUi();
    // void updateStoppingMode(boolean isOn);
    // void updateRegenBrakingLevel(int level);
    void updateHighBeamSensitivity(boolean isOn);
    void updateHighBeamPower(boolean isOn);
    // void initGear();
    // void updateGearStatus(boolean isInPark);
    // void initDriveMode();
    // void updateDriveModeStatus(DriveMode driveMode);
    void updateLscStatus(LSCStatus lscStatus); // 不知道这个 Lsc 是什么意思
    // void updateCreepStatus(boolean isOn);
}
```

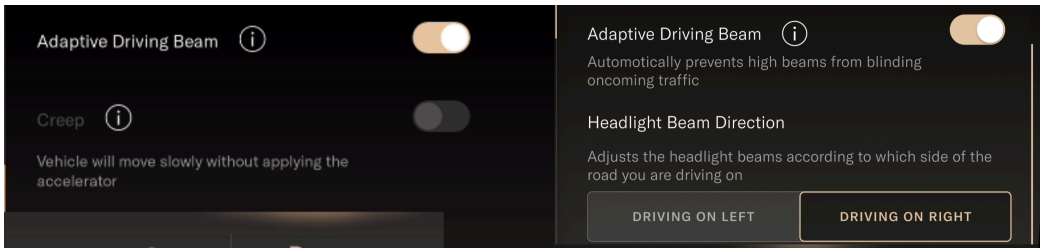
6 DriveSettingsFragment 的 layout 配置：R.layout.drive_settings_ BUG todo: 动态改变字符串

- 这里字符串的改变需要在代码中动态地来完成，要不然会 break 其它配置的系统。

```xml
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

  <!-- < DRIVING SETTINGS -->
  <com.lucid.resource.layout.BackButton
      android:id="@+id/layout_drive_settings_back_button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginLeft="37px"
      app:title="@string/option_drive_item_2" />

  <!-- Regenerative Braking -->
  <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      style="@style/text_title_06.default"
      android:text="@string/drive_settings_breaking"
      android:layout_marginTop="64px"
      android:layout_marginLeft="48px" />
  <com.lucid.resource.switches.Large2Toggle
      android:id="@+id/breaking_2_toggle"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      app:textLeft="@string/drive_settings_breaking_option_1"
      app:textRight="@string/drive_settings_breaking_option_2"
      android:layout_marginTop="24px"
      android:layout_marginLeft="48px" />
  <!-- Stop Mode: 只有到止为止两个 大 的 Toggle -->
  <TextView
      android:id="@+id/tv_stop_title"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      style="@style/text_title_06.default"
      android:text="@string/drive_settings_stop"
      android:layout_marginTop="48px"
      android:layout_marginLeft="48px" />
  <com.lucid.resource.switches.Large2Toggle
      android:id="@+id/stop_mode_2_toggle"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      app:textLeft="@string/drive_settings_stop_mode_1"
      app:textRight="@string/drive_settings_stop_mode_2"
      android:layout_marginTop="24px"
      android:layout_marginLeft="48px" />


<!-- Headlight Beam Direction: 感觉这里好像不对?????????? -->
```

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="40px">
  <View
      android:id="@+id/hba_hide"
      android:layout_width="315px"
      android:layout_height="80px"
      android:layout_marginLeft="48px" />
  <!--  它说是要显示在这个以前显示 High Beam Assist 的同一个地方:  app:featureName = "High Beam Assist" -->
  <!-- 那么, 不是把这个文本换一下 UI 的部分就算基本完成了吗？剩下的就是信号传递回调连接 -->
  <com.lucid.resource.layout.FeatureToggleList
      android:id="@+id/high_beam_list"
      android:layout_height="wrap_content"
      android:layout_width="wrap_content"
      android:layout_marginLeft="48px"
      <!-- app:featureName="@string/drive_settings_high_beam_assist" -->
      app:featureName="@string/drive_settings_adaptive_driving_beam"
      app:showInfoIcon="true" />
      <!-- string.xml 中加个字符串: 注意确保只加在 EU 版本中 -->
      <string name="drive_settings_adaptive_driving_beam">Adaptive Driving Beam</string>
    </RelativeLayout>
    <!-- 这部分我暂时把它们放在一起, 只是提醒不要制造附加问题 -->
    <!-- app:featureName = "Reduced High Beam Sensitivity" 问题来了: 这里是否与上面相关, 有没有必要的逻辑是需要跟这部分也联接好的
    <com.lucid.resource.layout.FeatureToggleList
        android:id="@+id/high_beam_sensitivity_list"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_marginTop="40px"
        android:layout_marginLeft="48px"
        app:featureName="@string/drive_settings_high_beam_assist_sensitivity"
        android:visibility="gone" />
        <!-- 这里说它暂时不显示, 查一下什么地方可以从代码里控制它们显示与否 -->


    <!-- Creep -->
<com.lucid.resource.layout.FeatureToggleList
    android:id="@+id/creep_list"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_marginTop="40px"
    android:layout_marginLeft="48px"
    app:showInfoIcon="true"
    app:description="@string/drive_settings_creep_info"
    app:featureName="@string/drive_settings_creep" />

<!-- app:featureName = "Low-Traction Start" -->
<com.lucid.resource.layout.FeatureToggleList
    android:id="@+id/wheelspin_list"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_marginTop="636px"
    android:layout_marginLeft="48px"
    app:showDisabled="true"
    android:visibility="gone"
    app:description="@string/drive_settings_wheelspin_info"
    app:featureName="@string/drive_settings_wheelspin" />
<!-- _info: Makes starting on loose or slippery surfaces easier -->

    <!-- <!-\- app:featureName = "Reduced High Beam Sensitivity" 原本的位置 : 问题来了: 这里是否与上面相关, 有没有必要的逻辑是需
    <!-- <com.lucid.resource.layout.FeatureToggleList -->
    <!--     android:id="@+id/high_beam_sensitivity_list" -->
    <!--     android:layout_height="wrap_content" -->
    <!--     android:layout_width="wrap_content" -->
    <!--     android:layout_marginTop="40px" -->
    <!--     android:layout_marginLeft="48px" -->
    <!--     app:featureName="@string/drive_settings_high_beam_assist_sensitivity" -->
    <!--     android:visibility="gone" /> -->

    <!-- Lucid Stability Control: 大的, 分三截, 左中右 -->
    <TextView
        android:id="@+id/tv_lucid_stability_control"
        android:layout_width="wrap_content"
        android:layout_height="80px"
        android:gravity="center"
```

```
        style="@style/text_title_06.default"
        android:layout_marginTop="40px"
        android:text="@string/drive_settings_stability_control"
        android:layout_marginLeft="48px"/>
    <com.lucid.resource.switches.Large3Toggle
        android:id="@+id/lsc_stability_mode_toggle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="48px"
        android:layout_marginTop="8px"
        app:textLeft="@string/drive_settings_stability_control_option_3"
        app:textMiddle="@string/drive_settings_stability_control_option_1"
        app:textRight="@string/drive_settings_stability_control_option_2" />
    <!-- Recommended for most drivinbg conditions -->
    <TextView
        android:id="@+id/tv_lsc_msg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignStart="@id/lsc_stability_mode_toggle"
        android:layout_alignEnd="@id/lsc_stability_mode_toggle"
        android:layout_marginTop="8px"
        android:layout_marginLeft="48px"
        android:text="@string/drive_setting_stability_msg"
        android:textAppearance="@style/text_title_07.dim" />
</LinearLayout>
```



- 这时注意代码的逻辑里应试会全权处理左右驾的问题，我不需要考虑。但是可以把这部分的代码也熟悉一下

7 AppFeature：后面 UnitPrefController 的问题是，如果同一应用 CarSetting 配置自不同地方，我可能会 break 掉某些东西

- UnitPreController 的确有 IRegion 的监听，但是我也同样需要遵循 LucidCarSetting 现有的配置逻辑，来自 AppFeature 的
- 要在现有逻辑的基础上去分割出 EU 版本，同时不 break 其它任何版本
- 我需要把 AppFeature 的逻辑与来自硬件的 UnitPrefController 统一起来，确保不会出问题。
- 明天早上上午的时候就把代码再好好熟悉一下，尤其是 IPC AIDL 配置同步用户数据的部分，以及 AppFeature Settings.Global Content Provider 的部分。把这些有些没看懂的地方都弄清楚了，早一点儿心理比较踏实。下周的测试过程还需要配置 configure 一个 bench 用来专测这个 EU 版本的 Configuration，可能还会有很多问题
- 还想要再早一天出发早点儿去表哥家看表哥。。。。。

7.1 Feature：bug todo

- 当我在配置定制这么一个 Feature 的时候，我————是这个需要把这些个配置写入 Lucid air 安卓系统的系统层面的任何所需要的配置文件的，以及系统层面 ContentProvider 的人!!!我不能以自己看那些层面的代码看得有点儿不太懂为由只求看懂就可以了，我不能/也不应该

指望有其它任何的高层或是 group lead 已经早就帮助我们配置了这个 Feature，它应该是需要我在配置这样的一个 Feature 的时候需要自己去考虑/去一一配置实现的!!! 我居然到今天下午当有一个 teammate 帮助我提示这一点儿之后，才一再去想想到我还有这些/这么多个一不小心就会 break 掉非 EU 版本的问题。也要快点儿写呀，我还想早点儿写完早点儿放假早点儿回家看表哥呢!!!

```
public enum DynamicFeature {
    AUTO_HIGH_BEAMS(ENABLE_AUTO_HIGH_BEAMS, Feature.DISABLED),
    TRAFFIC_DRIVE_OFF_ALERT(ENABLE_TRAFFIC_DRIVE_OFF_ALERT, Feature.ENABLED),
    CRUISING_SPEED_UPDATE_ALERT(ENABLE_CRUISING_SPEED_UPDATE_ALERT, Feature.ENABLED),
    COMFORT_BRAKING(ENABLE_PARK_COMFORT_BRAKE, Feature.DISABLED);

    public String featureName;
    public int defaultValue;
    DynamicFeature(String featureName, int defaultValue) {
        this.featureName = featureName;
        this.defaultValue = defaultValue;
    }
}
public class Feature {
    public static final boolean DEVELOPER_BUILD = "userdebug".equals(Build.TYPE)
        || "eng".equals(Build.TYPE);
    public static final boolean RELEASE_BUILD = ! DEVELOPER_BUILD;

    public static final int ENABLED = 1;
    public static final int DISABLED = 0;

    public static final boolean isEnabled(ContentResolver cr, String feature_name, int default_value) {
        boolean enabled = (Settings.Global.getInt(cr, feature_name, default_value) == ENABLED);
        Log.i(TAG, feature_name + " default: " + default_value + " enabled: " + enabled);
        return enabled;
    }
}
```

7.2 AppFeature

```
public class AppFeature extends Feature {
    public static final boolean BUILD_CELLULAR = "userdebug".equals(Build.TYPE) || "eng".equals(Build.TYPE);
    public static final String ENABLE_CELLUAR = "ENABLE_CELLULAR";

    public static final String ENABLE_AUTO_HIGH_BEAMS = "ENABLE_AUTO_HIGH_BEAMS"; // <<<<<<<<<<<<<==============

    public static final String ENABLE_ALL_LANGUAGES = "ENABLE_ALL_LANGUAGES";
    public static final String ENABLE_AUTO_EMERGENCY_BREAKING = "ENABLE_AUTO_EMERGENCY_BREAKING";
    public static final String ENABLE_CROSS_TRAFFIC_PROTECTION = "ENABLE_CROSS_TRAFFIC_PROTECTION";
    public static final String ENABLE_PARK_COMFORT_BRAKE = "ENABLE_PARK_COMFORT_BRAKE";
    public static final String ENABLE_TRAFFIC_DRIVE_OFF_ALERT = "ENABLE_TRAFFIC_DRIVE_OFF_ALERT";
    public static final String ENABLE_CRUISING_SPEED_UPDATE_ALERT = "ENABLE_CRUISING_SPEED_UPDATE_ALERT";
    public static final String ENABLE_LANE_DEPARTURE_PROTECTION = "ENABLE_LANE_DEPARTURE_PROTECTION";
    public static final String ENABLE_SPEED_LIMIT_WARNING = "ENABLE_SPEED_LIMIT_WARNING";

    private ContentResolver mContentResolver;
    public AppFeature(Context context) {
        mContentResolver = context.getContentResolver();
    }
    public boolean isFeatureEnabled(DynamicFeature feature) {
        if (feature == null)
            return true;
        return isEnabled(mContentResolver, feature.featureName, feature.defaultValue);
    }
    public boolean isFeatureDisabled(DynamicFeature feature) {
        return !isFeatureEnabled(feature);
    }
}
```

8 DriveSettingsFragment.java 中的主要逻辑：那么与会与 UI 层的 View Contract 相关？这里的源码还没有整理/没划重点

```
public class DriveSettingsFragment extends RightPanelFrag
//  Fragment 实现了接口，那么 fragment instance 就可以视作监听了 (Fragment 也就需要实现这个接口类中所定义的所有接口)
implements DriveSettingsViewContract, // 实现它的所有接口，就可以当回调来配置 UI
    LscWarningModal.LscModalCallbacks {
    private final static String TAG = DriveSettingsFragment.class.getName();

    private AppFeature mAppFeature; // 这里何止是相关，是一不小心就会出错。。。。。
    private final AsyncResources asyncResources;

    private HbaSensitivityModal hbaSensitivityModal; // 大概是一个确认打开高光灯之类的会话框
    private boolean mIsHighBeamAssistSensitivityOn = false;
    private boolean mIsHighBeamEnabled = false; // local variable

    private final Handler handler = new Handler();
    private static final long HBA_HIDDEN_VIEW_HOLD_TIME = 30000; // 30 seconds
    private final Runnable mLongPressed = () -> showHighBeamSensitivity(true);

    public DriveSettingsFragment(AsyncResources asyncResources) {
        this.asyncResources = asyncResources;
    }
    @Override public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        mAppFeature = new AppFeature(this.getContext()); // 需要知道这个 AppFeature 会涉及到一些区域相关的定制吗????????
// supposedly:  整个 lucid air 的安卓系统，不管来自于哪里的配置，硬件软件，应该都是各个方向互相支持的
// 我应该需要去找到配置 LucidCarSetting 应用的 ContentProvider 的源头，一定弄确定，可是我现在还没有弄明白更高层面，他们是怎么从 Co
// 我应该也可以把所有的源头都查找一遍，以确保他们互相支持，并找出可能有冲突的地方，加以解决
        mIsHighBeamEnabled = mAppFeature.isFeatureEnabled(DynamicFeature.AUTO_HIGH_BEAMS); // 远光灯自适应 启用了吗？
    }
    private void initViews(View view) { // 初始化
        highBeamList = view.findViewById(R.id.high_beam_list);
        highBeamSensitivityList = view.findViewById(R.id.high_beam_sensitivity_list);
        View hbaHiddenView = view.findViewById(R.id.hba_hide); // 标在上面的隐藏着 ?????
        // IVIANDAPPS-947: 这是别人之前改过的 bug 的标记，我不能把别人改好的 bug 再 reproduce 出来。。。。。
        // if (mIsHighBeamEnabled) { // 这里我打算把它改掉
// 设置 UI event 的监听：向下传递，要求 BCM 同步
            highBeamList.setCheckedChangedCallback(((buttonView, isChecked) -> highBeamIsToggled(isChecked)));
            highBeamList.setInfoIconCallback( // 小圆圈叹号 的 UI Event 监听：应该是再打开一个确认什么之类的小会话框
                () -> new HbaInfoModal().show(getFragmentManager(), "HbaInfoModal"));
            highBeamSensitivityList.setCheckedChangedCallback( // 这个视图的长按：会显示某个什么东西，只在按的过程中显示
                ((buttonView, isChecked) -> highBeamSensitivityIsToggled(isChecked)));
            hbaHiddenView.setOnTouchListener(new View.OnTouchListener() {
                    @Override
                        public boolean onTouch(View v, MotionEvent event) {
                        switch (event.getAction()) {
                        case MotionEvent.ACTION_DOWN:
                            handler.postDelayed(mLongPressed, HBA_HIDDEN_VIEW_HOLD_TIME);
                            break;
                        case MotionEvent.ACTION_UP:
                            handler.removeCallbacks(mLongPressed);
                            break;
                        }
                        return true;
                    }
                });
        // } else { // 这里我打算把它改掉
        //     highBeamList.setVisibility(View.GONE);
        //     highBeamSensitivityList.setVisibility(View.GONE);
        //     hbaHiddenView.setVisibility(View.GONE);
        // }
    }
    private final Runnable mLongPressed = () -> showHighBeamSensitivity(true);
    private void showHighBeamSensitivity(boolean comesFromLongTouch) {
        if (mIsHighBeamEnabled) {
            if (allowToShowHbaSensitivity(comesFromLongTouch)) {
                highBeamSensitivityList.setVisibility(View.VISIBLE);
                boolean isOn = SystemPrefs.getHbaSensitivity(context) || mIsHighBeamAssistSensitivityOn;
                Log.d(TAG, "updateHighBeamSensitivityToggle() is called with status isOn=" + isOn);
                highBeamSensitivityList.setChecked(isOn);
            } else {
```

```
                highBeamSensitivityList.setVisibility(View.GONE);
            }
        }
    }
    private boolean allowToShowHbaSensitivity(boolean comesFromLongTouch) {
        if (mIsHighBeamEnabled) {
            boolean isInPark = asyncResources.isGearInPark();
            boolean isGuestProfile = ProfileAccountType.isGuestProfile(context);
            // if it comes from long touch, show regardless of ON status IF gear is park, 这里的逻辑有点儿说不清，暂时不管
            // otherwise, only show if this is ON regardless of gear
            boolean allowBasedOnGear = (comesFromLongTouch && isInPark) || SystemPrefs.getHbaSensitivity(context); // 你需要
            boolean allowToShowHbaSen = highBeamList.isChecked() && !isGuestProfile && allowBasedOnGear;
            Log.d(TAG, "allow to show hba=" + allowToShowHbaSen);
            return allowToShowHbaSen;
        } else return false;
    }
// 下面的两个方法来自于： prefs。SystemPref：是把这些配置写入了 LucidCarSetting 应用的 SharedPreference
// high beam assist sensitivity true means this is ON, false means this is off
public static boolean getHbaSensitivity(Context context) {
    SharedPreferences sharedPreferences = context.getSharedPreferences(HomeActivity.PREF_SETTINGS_APP, Context.MODE_PRIVATE
    return sharedPreferences.getBoolean(PREF_HBA_SENSITIVITY, false);
}
// high beam assist sensitivity true means set this to ON, false means set this to off
public static void setHbaSensitivity(Context context, boolean isOn) {
    Log.d(TAG, "setting HBA Sensitivity isOn=" + isOn);
    SharedPreferences sharedPreferences = context.getSharedPreferences(HomeActivity.PREF_SETTINGS_APP, Context.MODE_PRIVATE
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putBoolean(PREF_HBA_SENSITIVITY, isOn);
    editor.apply();
}
    private void highBeamIsToggled(boolean isChecked) { // CCC ==>>>> BCM
        if (mIsHighBeamEnabled) { // 如果 远光灯自适应 是启用 的前提下
            saveHighBeamPrefs(isChecked); // 这里保存会不会太早？想一下，如果 BCM 同步失败，应该也会再写一次用户数据配置
            Log.d(TAG, "updateHighBeamAssistance with status of on=" + isChecked); // 这里的 log 看得不清不楚。。。。。
// 向下传，要求 BCM 同步配置硬件 on/off，调用控制器公共方法并启用计时器
            asyncResources.getBcmController().enableHighBeamAssistance(isChecked);
            showHighBeamSensitivity(false); // ????
        }
    }
    private void highBeamSensitivityIsToggled(boolean isChecked) {
        if (mIsHighBeamEnabled) { // 如果远光灯是打开的
            if (isChecked) {
                hbaSensitivityModal = new HbaSensitivityModal(asyncResources);
                hbaSensitivityModal.show(getFragmentManager(), "HbaSensitivityModal"); // 显示这个窗口
            } else {
                SystemPrefs.setHbaSensitivity(context, false);
//  上面：写入用户数据：不是数据库不是 ProfileData，是 SystemPref，但是 Power Cycle 后应该需要写入 ProfileData
                // 向下：要求硬件配置同步
                asyncResources.getBcmController().updateHighBeamSensitivity(BcmController.HIGH_BEAM_SENSITIVITY_NORMAL);
            }
        }
    }
    private void saveHighBeamPrefs(boolean isHighBeamAssistOn) { // BCM ==>>>> CCC ==>>>> ProfileData 保存用户的系统个性化配置
        Log.d(TAG, "PREF - saving to profile for " + ProfileKey.SETTINGS_HIGH_BEAM_ASSIST_PREF + "=" + isHighBeamAssistOn);
// 这里应该涉及 IPC AIDL，好像是，明天早上把这部分的源码看懂!!!
        asyncResources.getProfileData().persistBoolean(ProfileKey.SETTINGS_HIGH_BEAM_ASSIST_PREF, isHighBeamAssistOn);
    }
    @Override public void onResume() {
        super.onResume();
        //this most likely be ready by the time user gets to this screen since asyncResources
        // is initialized when this app is first opened. If race condition happens, may want to show
        // a spinner or something so UI doesn't look clunky. If this screen is opened before
        // asyncResources is ready, initUi() gets called via the AdasViewContract.
        if (asyncResources.isReadyAll()) // 它说，等用户个性化配置的所有数据都 Ready 了，那么就启动 UI 吧
            initUi();
    }
    @Override public void initUi() {
        Log.d(TAG, "initUi was called");
        updateRegenBrakingLevel(asyncResources.getBcmController().getRegenBrakingLevel());
        // boolean isOnePedalDrivingOn = asyncResources.getBcmController().isOnePedalDrivingOn();
        // if (isOnePedalDrivingOn) {
        //     stop2Toggle.selectLeft();
        // } else {
        //     stop2Toggle.selectRight();
        // }
```

```
        if (mIsHighBeamEnabled) { // 它很特殊，它感光，它是一个 Runable，大概不能简单地只用一个监听回调；但它也是一个最普通的公用
            updateHighBeamToggle(asyncResources.getBcmController().isHighBeamPowerOn());
            // high beam sensitivity is updated within updateHighBeamToggle()
        }
        // updateCreepToggle(asyncResources.getBcmController().getCreepModeStatus() != 0);
        // initGear(); // 这两个不知道是在干什么，暂时不管
        // initDriveMode();
    }
    // @Override public void initGear() {
    //     updateGearStatus(asyncResources.isGearInPark());
    // }
    @Override public void updateHighBeamSensitivity(boolean isOn) {
        if (mIsHighBeamEnabled) {
            mIsHighBeamAssistSensitivityOn = isOn;
            highBeamSensitivityList.setChecked(isOn);
            if (!isOn)
                showHighBeamSensitivity(false);
        }
    }
    @Override
        public void updateHighBeamPower(boolean isOn) {
        if (mIsHighBeamEnabled)
            updateHighBeamToggle(isOn);
    }

    public void updateHighBeamToggle(boolean isOn) {
        if (mIsHighBeamEnabled) {
            Log.d(TAG, "updateHighBeamToggle() is called with status isOn=" + isOn);
            highBeamList.setChecked(isOn);
            showHighBeamSensitivity(false);
        }
    }
    private void showHighBeamSensitivity(boolean comesFromLongTouch) {
        if (mIsHighBeamEnabled) {
            if (allowToShowHbaSensitivity(comesFromLongTouch)) {
                highBeamSensitivityList.setVisibility(View.VISIBLE);
                boolean isOn = SystemPrefs.getHbaSensitivity(context) || mIsHighBeamAssistSensitivityOn;
                Log.d(TAG, "updateHighBeamSensitivityToggle() is called with status isOn=" + isOn);
                highBeamSensitivityList.setChecked(isOn);
            } else  highBeamSensitivityList.setVisibility(View.GONE);
        }
    }
    private boolean allowToShowHbaSensitivity(boolean comesFromLongTouch) {
        if (mIsHighBeamEnabled) {
            boolean isInPark = asyncResources.isGearInPark();
            boolean isGuestProfile = ProfileAccountType.isGuestProfile(context);
            //if it comes from long touch, show regardless of ON status IF gear is park
            //otherwise, only show if this is ON regardless of gear
            boolean allowBasedOnGear = (comesFromLongTouch && isInPark) || SystemPrefs.getHbaSensitivity(context);
            boolean allowToShowHbaSen = highBeamList.isChecked() && !isGuestProfile && allowBasedOnGear;
            Log.d(TAG, "allow to show hba=" + allowToShowHbaSen);
            return allowToShowHbaSen;
        } else  return false;
    }
    @Override public void disable() {
        asyncResources.getBcmController().setLSCMode(LSCRequest.OFF.getValue());
        Log.d(TAG, "Disable event triggered");
    }
    @Override public void onDestroy() {
        super.onDestroy();
        handler.removeCallbacks(mLongPressed);
    }
    @Override public void onStart() {
        super.onStart();
        Activity activity = getActivity();
        if (activity instanceof HomeActivity)
            ((HomeActivity) activity).setDriveSettingsVc(this); // 设置 view 层的监听回调
    }
    @Override public void onStop() {
        super.onStop();
        Activity activity = getActivity();
        if (activity instanceof HomeActivity)
            ((HomeActivity) activity).clearDriveSettingVc(); // clean up
    }
}
```

9 UnitPrefContoller.java 相关的 LucidCarSettings 中的控制器

- 只是看了一下，摘了一点儿，未必重要，有不懂的，改天再去把它们看懂

```
// profileAndAccess.LoginService.java
AsyncResources.ConnectionsListener mConnectionsListener =
    new AsyncResources.ConnectionsListener(
        mProfileListener,
        mBcmControllerListener,
        mAdasControllerListener,
        mUnitPrefListener, //
        null,
        null) {
    @Override
    public void onConnectedAll() {
        Log.d(TAG, "onConnectedAll()");
        // Upon connecting, get user preference
        ProfileData profileData = AsyncResources.getInstance(getBaseContext()).getProfileData();
        initUnitPrefs(profileData); // 这里可能没看懂
        initBcmPrefs(profileData);
        initAdasPrefs(profileData);
        initAudioPrefs( profileData );
    }

    @Override
    public void onDisconnectedAny() {
        // Do nothing.
    }
};
// 在 CarSetting 中，把车的起始化配置，同步写入到了这些个用户相关的 Pref 用户数据库中，也 ?????
// 从车的配置，向用户数据中去写 ?????????。。。。。。。。。。
// 可是车配置中的初始化，来自于哪里呢？改天再来看这些吧
// CarSettings.IS_BATTERY_PERCENTAGE_UNIT_DEFAULT 等是来自 lucid-settings-lib static 定义
private void initUnitPrefs(ProfileData profileData) {
    updatePref(profileData, CarSettings.IS_BATTERY_PERCENTAGE_UNIT, CarSettings.IS_BATTERY_PERCENTAGE_UNIT_DEFAULT);
    updatePref(profileData, CarSettings.IS_PRESSURE_PSI_UNIT, CarSettings.IS_PRESSURE_PSI_UNIT_DEFAULT);
    updatePref(profileData, CarSettings.IS_TIME_24_HR, CarSettings.IS_TIME_24_HR_DEFAULT);
    updatePref(profileData, CarSettings.IS_TEMP_F_UNIT, CarSettings.IS_TEMP_F_UNIT_DEFAULT);
    notifyIccForUnitPrefs(profileData);
}
private void updatePref(ProfileData profileData, String key, boolean defaultVal) {
    boolean val = profileData.retrieveBoolean(key, defaultVal);
    Settings.Global.putString(getApplicationContext().getContentResolver(), key, String.valueOf(val));
}
// AsyncResource UnitPrefController wrapper
private final UnitPrefController.UnitPrefListener mUnitPrefControllerListener =
    new UnitPrefController.UnitPrefListener() {
    @Override
    public void onVhalConnectedW() {
        super.onVhalConnectedW();
        synchronized (mConnectionsListeners) {
            for (ConnectionsListener listener : mConnectionsListeners) {
                if (listener.mUnitPrefListener != null)
                    listener.mUnitPrefListener.onVhalConnectedW();
            }
        }
        mUnitPrefControllerReady = true;
        checkConnectedAll();
    }
    @Override
    public void onVhalDisconnectedW() {
        super.onVhalDisconnectedW();
        mUnitPrefControllerReady = false;
        checkDisconnectedAny();
        synchronized (mConnectionsListeners) {
            for (ConnectionsListener listener : mConnectionsListeners) {
                if (listener.mUnitPrefListener != null)
                    listener.mUnitPrefListener.onVhalDisconnectedW();
            }
        }
    }
};
```

10 先去把信号找到，确定好是能够传上来的！不要等到最后一分钟: 应该是可以传得上来的

- 它的意思是说，作个替换，把 US 版本的 High Beam Assist 换成 EU 版本的 Adaptive Driving Beam(ADB)，不该是换汤不换药吗?
- 信号也是用现有的，但是可能细节的要求：左驾右驾，ICC 与 CID 的要求全都需要满足

```
ICCC_AutHiBeamSetReq
IBCM_AutHiBeamSetSts
IBCM_AutHiBeamSts
```

  - 需要搞清楚这三个信号分别是什么意思，怎么起作用的，见下面的截图：

HeadLight		BCM_to_HMI/IBCM_HiBeamSts BCM_to_HMI/IBCM_MainBeamSts	Off:0, On:1
Auto high beam setting (HBA high beam assist)	ICCC_AutHiBeamReq S2R-16: CCC_to_BCM_1/ICCC_AutHiBeamSetReq	IBCM_AutHiBeamSts S2R-16: IBCM_AutHiBeamSetSts	request: Idle:0, disable:1, enable:2, reserved:3 status: Disabled:0, Enabled:1
High Beam Assist sensitivity setting (HBA)		IHCU_L_HLASnvty	Unknown:0, Normal:1, US:2, Invalid:3

11 首先找到 LucidCarSettings app 中的最相关的界面 driving/DriveSettingsFragment。java

```java
package com.lucid.car.carsettings.driving;

public class DrivingListFragment extends RightPanelFrag implements View.OnClickListener {
    @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        View view = inflater.inflate(R.layout.drive_list_frag, container, false);
        initViews(view);
        return view;
    }
    private void initViews(View view) {
        // view.findViewById(R.id.list_item_1).setOnClickListener(this);
        // view.findViewById(R.id.list_item_2).setOnClickListener(this);
        view.findViewById(R.id.list_item_3).setOnClickListener(this);
    }
    @Override
        public void onClick(View view) {
        int id = view.getId();
        // if (id == R.id.list_item_1) {
        //     swapRightPanel(TripInfoFragment.newInstance(), Utils.SettingsType.VEHICLE);
        // }else if(id == R.id.list_item_2){
        //     swapRightPanel(TirePressureFragment.newInstance(), Utils.SettingsType.VEHICLE);
        // } else
        if (id == R.id.list_item_3) { //  注意这里需要获取资源的地方，Context 相关的
            swapRightPanel(new DriveSettingsFragment(getAsyncResources()), Utils.SettingsType.VEHICLE);
        }
    }
}
```

- 注意这里 getAsyncResources() 是调用的 RightPanelFrag 的方法

```
protected AsyncResources getAsyncResources() {
    Activity activity = getActivity();
    if (activity instanceof HomeActivity)
        return ((HomeActivity) activity).getAsyncResources(); // 从 activity 再调用 getAsyncResources()
    return null;
}
```

12 BcmController.java 具体三个信号的接收与监听

- 在 LucidCarSettings 应用中的定位是在 AsyncResources。java 的 mBcmListener 中

```
public class BcmController {
    private static final int VCU_SIGNAL_TIMEOUT = 1 * 1000; // this timeout will be used for REGEN, STOPMODE, and CREEP MOD

    public static class BcmListener {
        public void onVhalConnectedW() { }
        public void onVhalDisconnectedW() { }
        /**
         * Callback for High Beam Power
         * @param isOn = true means ON, false means OFF
         */
        public void onUpdateHighBeamPowerW(boolean isOn) {  }
        /**
         * Callback for High Beam Sensitivity Level
         * @param val = HIGH_BEAM_SENSITIVITY_NORMAL or HIGH_BEAM_SENSITIVITY_US
         */
        public void onUpdateHighBeamSensitivityW(int val) {  }
    }
    /**
     * High Beam Request
     * @param enable :0: Idle - 0x00
     *               (1-0): Disable - 0x01
     *               (2-0): Enable - 0x02
     *               (3-0): Reserved
     */
    public void enableHighBeamAssistance(boolean enable) { //  这个公共方法，是在 AsyncResource.java 中的 mBcmListener 中调用
        // 当你要取消定时器的时候，只需要调用 handler.removeCallbacks(runnable) 就可以了
        mSignalBroadcastHandler.removeCallbacks(mHighBeamRunnable);
        int val = enable ? 2 : 1;
        mSignalBroadcastHandler.sendMessage(mSignalBroadcastHandler
                                            .obtainMessage(MSG_BROADCAST_SIGNAL, ICCC_AUT_HI_BEAM_SET_REQ, val));
// 调用 handler.post(runnable); 就能启动定时器，750 ms 内完成不了就取消操作，维持原状
        mSignalBroadcastHandler.postDelayed(mHighBeamRunnable, 750);
// 延迟 750 ms 执行：以便 BCM 运行不力的时候，取消息上次操作，转为 Idle，即上次 request 作废，继续维持上次 request 之前的状态
    }
    final Runnable mHighBeamRunnable = new Runnable() {
// 启用计时器：当 BCM 反应不过来，或是出错了，不能在规定的最长时间内完成我们想要的设置
// 我们有充足的理由取消操作，向用户反馈设置失败，更新同步 UI 为硬件状态值
        @Override public void run() { // 它没事的时候，就是常年地闲逛吗？发一条消息 idle，对应 BCM 的 matching；然后再向用户反馈
            // set to idle val
            mSignalBroadcastHandler.sendMessage(
                mSignalBroadcastHandler.obtainMessage(MSG_BROADCAST_SIGNAL, ICCC_AUT_HI_BEAM_SET_REQ, 0)); // 0 0 0 0 0 idl
            // update UI for current status signal
            if (mBcmListener != null) {
                synchronized (mPropertiesLock) {
                    mBcmListener.onUpdateHighBeamPowerW(isHighBeamPowerOn);
                }
            }
        }
    };
    /**
     * High Beam Sensitivity Request
     * @param sensitivity
     * Unknown:0, Normal:1, US:2, Invalid:3
     */
    public static int HIGH_BEAM_SENSITIVITY_NORMAL = 1;
    public static int HIGH_BEAM_SENSITIVITY_US = 2;
// 这个方法定义出来，谁在用呢？公共方法，在应用层调用
    public void updateHighBeamSensitivity(int sensitivity) {
        if (sensitivity != HIGH_BEAM_SENSITIVITY_NORMAL && sensitivity != HIGH_BEAM_SENSITIVITY_US) {
```

```
        Log.e(TAG, "DEVELOPER ERROR - INVALID REQUEST INT SENT FOR HIGH BEAM SENSITIVITY");
        return;
    }
    mSignalBroadcastHandler.removeCallbacks(mHighBeamSensitivityRunnable);
    mSignalBroadcastHandler.sendMessage(mSignalBroadcastHandler
                                        .obtainMessage(MSG_BROADCAST_SIGNAL, HIGH_BEAM_ASSIST_SENSITIVITY_SEL, sensitiv
    mSignalBroadcastHandler.postDelayed(mHighBeamSensitivityRunnable, 750);
}
final Runnable mHighBeamSensitivityRunnable = new Runnable() {
    @Override public void run() {
        // update UI for current status signal
        if (mBcmListener != null) {
            synchronized (mPropertiesLock) {
                mBcmListener.onUpdateHighBeamSensitivityW(highBeamSensitivitySts);
            }
        }
    }
};

//================================================================
// Init / Cleanup
//================================================================
private Context mContext;
private BcmListener mBcmListener;
public BcmController() {
    // Signal broadcasting worker thread.
    mHandlerThread = new HandlerThread( TAG );
    mHandlerThread.start();
    mSignalBroadcastHandler = new SignalBroadcastHandler( mHandlerThread.getLooper() );
}
public void init(Context context, BcmListener bcmListener) {
    mContext = context;
    mBcmListener = bcmListener;
    mCarApiClient = Car.createCar( mContext, mCarConnectionCallback );
    mCarApiClient.connect();
}
public void cleanup() {
    if ( mCarApiClient != null ) {
        mCarApiClient.disconnect();
    }
    mBcmListener = null;
    mContext = null;
}

//================================================================
// Connection to VHal
//================================================================
private Car mCarApiClient;
private CarPropertyManager mPropertyManager;
private final ServiceConnection mCarConnectionCallback = new ServiceConnection() {
    @SuppressLint("StaticFieldLeak")
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        if ( mBcmListener != null ) {
            if (IS_DEBUG_BUILD) Log.d( TAG, "mBcmListener.onVhalConnectedW()" );
            mBcmListener.onVhalConnectedW();
        }
        try {
            mPropertyManager = (CarPropertyManager) mCarApiClient
                .getCarManager( android.car.Car.PROPERTY_SERVICE );
            for ( int propId : S_NOTIFY_PROP_IDS )
                mPropertyManager.registerCallback(mPropertiesListener, propId, 100 );
            // Fetch initial values.
            new AsyncTask<Void, Void, Void>() {
                @Override
                    protected Void doInBackground(Void... nulls) {
                    fetchProperties();
                    return null;
                }
                @Override
                    protected void onPostExecute(Void aVoid) {
                    // Notify
                    if ( mBcmListener != null ) {
                        if (IS_DEBUG_BUILD) Log.d( TAG, "mBcmListener.onPropertiesReadyW()" );
                        mBcmListener.onPropertiesReadyW();
```

```
                    }
                }
            }.execute();
        } catch ( CarNotConnectedException e ) {
            Log.e(TAG, "onConnected() : Car not connected.");
        } catch (IllegalStateException e) {
            Log.e(TAG, "Illegal State Exception... "+ e.getMessage());
        }
    }
    @Override
    public void onServiceDisconnected(ComponentName name) {
        for( int propId : S_NOTIFY_PROP_IDS) {
            mPropertyManager.unregisterCallback(mPropertiesListener, propId );
        }
        if ( mBcmListener != null ) {
            if (IS_DEBUG_BUILD) Log.d( TAG, "mBcmListener.onVhalDisconnectedW()" );
            mBcmListener.onVhalDisconnectedW();
        }
    }
};
private void fetchProperties() { // 初始化监听的两个属性本地变量值
    Log.d( TAG, "fetchProperties() : BEGIN" );
    int value;
    float fValue;
    synchronized (mPropertiesLock) {
        try {
            value = mPropertyManager.getIntProperty(IBCM_AUT_HI_BEAM_SET_STS, 0);
            isHighBeamPowerOn = value == 1;
            value = mPropertyManager.getIntProperty(HCU_HLAS_SENSITIVITY_STS, 0);
            highBeamSensitivitySts = value;
        } catch ( android.car.CarNotConnectedException e ) {
            Log.e(TAG, "onConnected() : Car not connected.");
        }
    }
    Log.d( TAG, "fetchProperties() : END" );
}

//================================================================
// Incoming BCM Signals Handling
//================================================================
private CarPropertyManager.CarPropertyEventCallback mPropertiesListener =
    new CarPropertyManager.CarPropertyEventCallback() {
    @Override
    public void onErrorEvent(int i, int i1) {
        Log.e( TAG, "onErrorEvent() : i=" + i + ", i1=" + i1 );
    }
    @Override
    public void onChangeEvent(CarPropertyValue carPropertyValue) { // 更新监听的两个属性本地变量的值
        if (IS_DEBUG_BUILD) Log.d(TAG, " onChangeEvent " + carPropertyValue.getPropertyId()
                                + " (0x" + Integer.toHexString(carPropertyValue.getPropertyId()) + ")");
        switch ( carPropertyValue.getPropertyId() ) {
        case IBCM_AUT_HI_BEAM_SET_STS: // BCM_to_HMI/IBCM_HiBeamSts: on/off
            handleHighBeamPowerW((Integer) carPropertyValue.getValue());
            break;
        case HCU_HLAS_SENSITIVITY_STS:
            handleHighBeamSensitivityPowerW((Integer) carPropertyValue.getValue());
            break;
        }
    }
};
// boolean parseCarPropertyValue(CarPropertyValue carPropertyValue) {
//     return (Integer) carPropertyValue.getValue() == 1 ? true : false;
// }
void handleHighBeamPowerW(Integer value) { // 只有有效值的时候，才会收到回调，idle 与 matching 不会收到回调
    if (value != 0 && value != 1) {
        Log.e(TAG, "VHAL ERROR - INVALID VAL IS SENT FOR HIGH BEAM POWER");
        return;
    }
    synchronized (mPropertiesLock) {
        isHighBeamPowerOn = value == 1;
    }
    if (mBcmListener != null) { // 不管是 ON，还是 OFF，都会收到回调
        boolean isOn = value == 1;
        mBcmListener.onUpdateHighBeamPowerW(isOn);
    }
```

```
    }
    void handleHighBeamSensitivityPowerW(Integer value) {
        synchronized (mPropertiesLock) {
            highBeamSensitivitySts = value;
        }
        if (mBcmListener != null)
            mBcmListener.onUpdateHighBeamSensitivityW(value);
    }

    //================================================================
    // Car Property IDs
    //================================================================
// 好像是可以找到要求的三个中的两个信号
    private static final int ICCC_AUT_HI_BEAM_SET_REQ = 0x2140114A; // high beam request signal - ARXML ICCC_AutHiBeamSetRe
    private static final int IBCM_AUT_HI_BEAM_SET_STS = 0x2140114B; // high beam status signal  - ARMXL IBCM_AutHiBeamSetSt
    private static final int HIGH_BEAM_ASSIST_SENSITIVITY_SEL = 0x21401149; // high beam sensitivity request signal - ARXML

    //STATUS Signal to ICC
    private static final int TRIP_INF_DISP_SETTING_REQ = 0x2140170E;
    private static final int IVCU_LSC_MODE_STAT = 0x21401610; //FULL = 0, SEMI = 1, OFF = 2
    private static final int ICCC_LSC_MODE_REQ = 0x21401611; //Idle:0, Mode0:1, Mode1:2, Mode2:3.
    public static final int ICCC_CRP_MOD_REQ = 0x21401620;
    public static final int IVCU_CRP_MOD_ST = 0x21401621;
    private final static int[] S_NOTIFY_PROP_IDS = new int [] {
        IBCM_AUT_HI_BEAM_SET_STS,
    };

    private CidState mCidState;
    private int mGloveBoxStatus;
    private int mSteerColTele = 0;
    private int mSteerColTilt = 0;
    private float mTargetSOCStatus;
    private float realTimePowerInfoStatus;
    private int onePedalDrivingStatus; // 它依耐一只脚吗????????
    // private int regenBreakingLevel = REGEN_BRAKING_LVL_MED;
    private boolean isHighBeamPowerOn; // 打开远光灯了吗？初始化的时候 fetch 初始值; 然后收到回调的时候变更
    private int highBeamSensitivitySts;
    private int lscModeStatus;
    private int screenCleaningMode;
    public static final int HIGH_BEAM_SENS_STS_ON = 2;
    // private static boolean isRccdLockOn;
    private final Object mPropertiesLock = new Object();

    //================================================================
    // Handling Broadcasting of Signals (from CCC): 这个部分比较熟悉，可以暂时不用看了
    //================================================================
    private static final int MSG_BROADCAST_SIGNAL = 100;
    private HandlerThread mHandlerThread;
    private SignalBroadcastHandler mSignalBroadcastHandler;
    class SignalBroadcastHandler extends Handler {
        SignalBroadcastHandler(Looper looper) {
            super(looper);
        }
        @Override
            public void handleMessage(Message msg) {
            if ( msg.what == MSG_BROADCAST_SIGNAL ) {
                broadcastCccSignalW( msg.arg1, msg.arg2 );
            }
        }
        void broadcastCccSignalW(int propId, int value) {
            try {
                mPropertyManager.setIntProperty( propId, 0, value );
            } catch ( android.car.CarNotConnectedException e ) {
                Log.e( TAG, "broadcastCccSignalW() : VHAL not connected." );
            }
        }
    }
}
```

13 AsyncResources。java 的 mBcmListener

14 IMarket IRegion: lucid-carctrl-lib UnitPrefController.java

- 上次改 UnitsFragment 的内存泄露里为了 enable UI 我试图找过这两个，但是当时没能找到；今天早上再去找，就一下子就找出来了。
- 所有重要的相关逻辑都在这里了。如果需要添加监听，应该还是不是在这里，还需要再去找，这里只是硬件软件配制初始化的源头??? 感觉不一定吧，我还能再找出更多的第三第四个源头吗？再多找一找

```java
public class UnitPrefController {
    public static class UnitPrefListener {
        public void onVhalConnectedW() { }
        public void onVhalDisconnectedW() {}
        // public void onDistanceUnitReady(boolean isDistanceInMiles) {} // 我不需要这个回调，定义自己的 IMarket IRegion 回调

        // hack: temp hack IRegion in LucidCarSettings app even in US to be harded coded to be
        // mRegion.value = "EU", so that I can test locally in Pixel3 and bench in US
        // They could help build/configure the bench to simulate EU too, as team suggested
        public Region onRegionReady() { }
    }
    //==================================================================
    // Init / Cleanup
    //==================================================================
    private Context mContext;
    private UnitPrefController.UnitPrefListener mUnitPrefListener;
    private final static float SIGNAL_REFRESH_RATE = 0;
    private static final int VEHICLE_GLOBAL_AREA = 0;

    private Market mMarket = Market.UNKNOWN;
    private Region mRegion = Region.UNKNOWN;
    private boolean mIsDebuggable = false;
    private final static String MARKET_CANADA = "CAN";
    private final static String MARKET_SAUDI_ARABIA = "KSA";
    private final static String MARKET_CHINA = "CHI";
    private final static String MARKET_US = "USA";
    private final static String MARKET_UK = "GBR";
    private final static String REGION_NORTH_AMERICA = "RNA";
    private final static String REGION_EUROPE = "REU"; // <<<<<<<<<<<<<<<<<<<<<<<<<<==========================
    private final static String REGION_ASIA = "RAP";
    private final static String REGION_MIDDLE_EAST = "RME";
    public static final String IS_DISTANCE_IMPERIAL_MI_UNIT = "IS_DISTANCE_IMPERIAL_MI_UNIT";

    public UnitPrefController() {
        // Signal broadcasting worker thread.
        mHandlerThread = new HandlerThread(TAG);
        mHandlerThread.start();
        mSignalBroadcastHandler = new UnitPrefController.SignalBroadcastHandler(mHandlerThread.getLooper());
    }
    public void init(Context context, UnitPrefController.UnitPrefListener unitPrefListener) { // 这个方法总是会被调用，因为要
        mContext = context;
        TAG = mContext.getPackageName() + " | " + TAG;
        mUnitPrefListener = unitPrefListener; //
        mCarApiClient = Car.createCar(mContext, mCarConnectionCallback); // 准备一个车
        mCarApiClient.connect(); // 连上这个车
        mIsDebuggable = isDebuggable(); //
    }
    public void cleanup() {
        if (mCarApiClient != null)
            mCarApiClient.disconnect(); // 断开与车的连接：因为是为了在一个更高的层面上 clean up，免得 Application 不能回收，浪费
        mUnitPrefListener = null;
        mContext = null;
    }
    //==================================================================
    // Connection to VHal
    //==================================================================
    private Car mCarApiClient;
    private CarPropertyManager mPropertyManager;
    private final ServiceConnection mCarConnectionCallback = new ServiceConnection() {
        @Override public void onServiceConnected(ComponentName name, IBinder service) {
            try {
                Log.d(TAG, "onConnected..");
                mPropertyManager = (CarPropertyManager) mCarApiClient
```

```
            .getCarManager(android.car.Car.PROPERTY_SERVICE);
            loadDefaultValues(); // 这里应试是车的初始化的起始配置: 这就涉及到了车/用户所在的版块分区语言与车的硬件 FW 配置等
        } catch (android.car.CarNotConnectedException e) {
            Log.e(TAG, "onConnected() : Car not connected.");
        }
        if (mUnitPrefListener != null) {
            Log.d(TAG, "mUnitPrefListener.onVhalConnectedW()");
            mUnitPrefListener.onVhalConnectedW();
        }
    }
    @Override public void onServiceDisconnected(ComponentName name) {
        if (mUnitPrefListener != null) {
            Log.d(TAG, "mUnitPrefListener.onVhalDisconnectedW()");
            mUnitPrefListener.onVhalDisconnectedW();
        }
    }
};
private CarPropertyManager.CarPropertyEventCallback mPropertiesListener =
    new CarPropertyManager.CarPropertyEventCallback() {
    @Override public void onErrorEvent(int i, int i1) {
        Log.e(TAG, "onErrorEvent() : i=" + i + ", i1=" + i1);
    }
    @Override public void onChangeEvent(CarPropertyValue carPropertyValue) {
        int propId = carPropertyValue.getPropertyId();
        Log.d(TAG, "onChangeEvent...." + propId);
        switch (propId) {
        case BCM_TO_CCC_IMARKET:
        case BCM_TO_CCC_IREGION:
            String value = parseStringSafely(carPropertyValue);
            if (BCM_TO_CCC_IMARKET == propId) {
                log("Received Market changed, value:" + value);
                if (!TextUtils.isEmpty(value)) {
                    Market tempMarket = Market.valueOfMarket(value);
                    Log.d(TAG, " tempMarket: " + tempMarket + " mMarket: " + mMarket);
                    if (tempMarket != mMarket)
                        mMarket = tempMarket;
                }
            } else { // BCM_TO_CCC_IREGION == propId
                log("Received Region changed, value:" + value);
                if (!TextUtils.isEmpty(value)) {
                    Region tempRegion = Region.valueOfRegion(value);
                    Log.d(TAG, " tempRegion: " + tempRegion + " mRegion: " + mRegion);
                    if (tempRegion != mRegion)
                        mRegion = tempRegion;
                }
            }
            Log.d(TAG, " mUnitPrefListener: " + mUnitPrefListener + " mMarket: " + mMarket + " mRegion: " + mRegion);
            if (mUnitPrefListener != null && (mMarket != Market.UNKNOWN || mRegion != Region.UNKNOWN))
                mUnitPrefListener.onDistanceUnitReady(isDistanceUnitInMiles());
            break;
        }
    }
};
private void loadDefaultValues() throws CarNotConnectedException { //
    CarPropertyValue<String> marketValue = mPropertyManager.getProperty(BCM_TO_CCC_IMARKET, VEHICLE_GLOBAL_AREA); // IM
    String strMarket = parseStringSafely(marketValue);
    log("Get Market, value:" + strMarket);
    if (!TextUtils.isEmpty(strMarket))
        mMarket = Market.valueOfMarket(strMarket);
    CarPropertyValue<String> regionValue = mPropertyManager.getProperty(BCM_TO_CCC_IREGION, VEHICLE_GLOBAL_AREA); // IR
    String strRegion = parseStringSafely(regionValue);
    log("Get Region, value:" + strRegion);
    if (!TextUtils.isEmpty(strRegion))
        mRegion = Region.valueOfRegion(strRegion);
    if (mUnitPrefListener != null && (mMarket != Market.UNKNOWN || mRegion != Region.UNKNOWN))
        mUnitPrefListener.onDistanceUnitReady(isDistanceUnitInMiles());
    mPropertyManager.registerCallback(mPropertiesListener, BCM_TO_CCC_IMARKET, SIGNAL_REFRESH_RATE); // 注册回调
    mPropertyManager.registerCallback(mPropertiesListener, BCM_TO_CCC_IREGION, SIGNAL_REFRESH_RATE);
}

//==============================================================
// Handling Broadcasting of Signals (from CCC)
//==============================================================
private static final int MSG_BROADCAST_SIGNAL = 100;
private HandlerThread mHandlerThread;
```

```
private UnitPrefController.SignalBroadcastHandler mSignalBroadcastHandler;
class SignalBroadcastHandler extends Handler {
    SignalBroadcastHandler(Looper looper) {
        super(looper);
    }
    @Override
        public void handleMessage(Message msg) {
        if (msg.what == MSG_BROADCAST_SIGNAL)
            broadcastCccSignalW(msg.arg1, msg.arg2);
    }
    void broadcastCccSignalW(int propId, int value) {
        try {
            mPropertyManager.setIntProperty(propId, 0, value);
        } catch (android.car.CarNotConnectedException e) {
            Log.e(TAG, "broadcastCccSignalW() : VHAL not connected.");
        }
    }
}

//=============================================================
// Request signals
//=============================================================
public boolean isUKMarket() {
    return mMarket == Market.UNITED_KINGDOM;
}
public boolean hasValidMarket() {
    return mMarket != Market.UNKNOWN;
}
public boolean hasValidRegion() {
    return mRegion != Region.UNKNOWN;
}
public Market getDefaultMarket() {
    return mMarket;
}
public Region getDefaultRegion() { // 硬件配置的区域
    return mRegion;
}
private String parseStringSafely(CarPropertyValue<String> carPropertyValue) {
    String value = "";
    if (carPropertyValue == null)
        return value;
    return carPropertyValue.getValue();
}
private void log(String msg) {
    Log.d(TAG, msg);
}
private boolean isDebuggable() {
    boolean debuggable = false;
    PackageManager pm = mContext.getPackageManager();
    try {
        ApplicationInfo appInfo = pm.getApplicationInfo(mContext.getPackageName(), 0);
        debuggable = (0 != (appInfo.flags & ApplicationInfo.FLAG_DEBUGGABLE));
    } catch (Exception e) {
    }
    return debuggable;
}
private static final int BCM_TO_CCC_IMARKET = 0x2110116B;
private static final int BCM_TO_CCC_IREGION = 0x2110116C;
public boolean isEuRegion() { // 这个公共接口可以一用
    return mRegion == Region.EU;
}
// PM hasn't defined all the default values for all the countries. we will use this for now. may change it in the futur
public enum Market {
    UNKNOWN("", DistanceUnit.MILES, TimeFormat.TWELVE_HOUR, TemperatureFormat.FAHRENHEIT, TirePressureFormat.PSI),
    CANADA(MARKET_CANADA, DistanceUnit.KILOMETERS, TimeFormat.TWENTY_FOUR_HOUR, TemperatureFormat.CELSIUS, TirePressure
    SAUDI_ARABIA(MARKET_SAUDI_ARABIA, DistanceUnit.KILOMETERS, TimeFormat.TWELVE_HOUR, TemperatureFormat.FAHRENHEIT, Ti
    CHINA(MARKET_CHINA, DistanceUnit.KILOMETERS, TimeFormat.TWELVE_HOUR, TemperatureFormat.CELSIUS, TirePressureFormat.
    UNITED_STATES(MARKET_US, DistanceUnit.MILES, TimeFormat.TWELVE_HOUR, TemperatureFormat.FAHRENHEIT, TirePressureForm
    UNITED_KINGDOM(MARKET_UK, DistanceUnit.MILES, TimeFormat.TWELVE_HOUR, TemperatureFormat.FAHRENHEIT, TirePressureFor
    public String value;
    public DistanceUnit unit;
    public TemperatureFormat temperatureFormat;
    public TimeFormat timeFormat;
    public TirePressureFormat tirePressureFormat;
    Market(String val, DistanceUnit unit, TimeFormat timeFormat, TemperatureFormat temperatureFormat, TirePressureForma
```

```
            this.value = val;
            this.unit = unit;
            this.temperatureFormat = temperatureFormat;
            this.timeFormat = timeFormat;
            this.tirePressureFormat = tirePressureFormat;
        }
        public static Market valueOfMarket(String value) {
            for (Market market : Market.values()) {
                if (market.value.equals(value))
                    return market;
            }
            return Market.UNKNOWN;
        }
    }
    public enum Region {
        UNKNOWN("", DistanceUnit.MILES),
        ASIA(REGION_ASIA, DistanceUnit.KILOMETERS),
        EU(REGION_EUROPE, DistanceUnit.KILOMETERS),
        MIDDLE_EAST(REGION_MIDDLE_EAST, DistanceUnit.KILOMETERS),
        NORTH_AMERICA(REGION_NORTH_AMERICA, DistanceUnit.MILES);
        public String value;
        public DistanceUnit unit;
        Region(String val, DistanceUnit unit) {
            this.value = val;
            this.unit = unit;
        }
        public static Region valueOfRegion(String value) {
            for (Region region : Region.values()) {
                if (region.value.equals(value))
                    return region;
            }
            return Region.UNKNOWN;
        }
    }
}
```

15 DriveModeController.java

- 这个应该还是不相关的，中是走过路过不想错过，看了看，捡了这点儿记在这里

```
public class DriveModeController {
    private static final int SIGNAL_TIMEOUT_MILLISEC = 3000;

    public static class DriveModeListener {
        public void onVhalConnectedW() { }
        public void onVhalDisconnectedW() { }
        public void onPropertiesReadyW() { }

        public void onUpdateDriveModeW(Integer state) { }
        public void onUpdateVehicleSpeedW(Float speed) { }
    }
    //================================================================
    // Connection to VHal
    //================================================================

    private Car mCarApiClient;
    private CarPropertyManager mPropertyManager;
    private ValuesDumper mDumper;

    private final ServiceConnection mCarConnectionCallback = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            if (mDriveModeListener != null) {
                Log.d(TAG, "mDriveModeListener.onVhalConnectedW()");
                mDriveModeListener.onVhalConnectedW();
            }
            try {
                mPropertyManager = (CarPropertyManager) mCarApiClient
                    .getCarManager(android.car.Car.PROPERTY_SERVICE);
                for (int propId : S_NOTIFY_PROP_IDS)
                    mPropertyManager.registerCallback(mPropertiesListener, propId, 100);
                // Fetch initial values. 它只监听了两个信号，并且都不相关
```

```
                int driveMode = mPropertyManager.getIntProperty( CCC_DRIVE_MODE_STATUS, 0 );
                float speed = mPropertyManager.getFloatProperty( PERF_VEHICLE_SPEED, 0 );
                synchronized ( mPropertiesLock ) {
                    mDriveMode = driveMode;
                    mLastSpeed = speed;
                }

            } catch (android.car.CarNotConnectedException e) {
                Log.e(TAG, "onConnected() : Car not connected.");
            }

            mPropertiesReady = true;

            // Notify
            if (mDriveModeListener != null) {
                Log.d(TAG, "mDriveModeListener.onPropertiesReadyW()");
                mDriveModeListener.onPropertiesReadyW();
            }
            if( IS_DEBUG_BUILD ) {
                mDumper = new ValuesDumper(TAG, S_NOTIFY_PROP_IDS, mSignalBroadcastHandler, mPropertyManager);
            }
        }
    }}
```

16 LucidCarSettings 应用中 HomeActivity 中与 Controller 控制器相关/UI 回调相关/bug 相关的最重要逻辑

- 这里，我应该是需要往更上层地去找，是谁在设置这些个 View 层面的监听，比如: HomeActivity 是在什么时候什么情况下设置了某个它的 UI Contract 给某个特定的 Fragment? 这里理解起来有点儿绕，但是就是那么个意思

```
public class HomeActivity extends AppCompatActivity {
    private static final String TAG = HomeActivity.class.getName();
    // private static final String OWNER_MANUAL_URL = "OWNER_MANUAL_URL";

    // 在 DriveSettingsFragment.java 中设置这个，在 onStart() 方法中设置:
    // ((HomeActivity) activity).setDriveSettingsVc(this);
    public void setDriveSettingsVc(DriveSettingsFragment frag) {
        driveSettingVc = frag;
    }
    public void clearDriveSettingVc() {
        driveSettingVc = null;
    }

    // private ProfileData mProfileData;
    // private AsyncResources asyncResources;
    private Handler uiHandler = new Handler(Looper.getMainLooper()); // handler 机制崩出来了，如何动作的呢？
    // 它有一堆 UI 相关的 Contract，同时它也有一堆的 Model，它们是怎么联系起来的呢？
    // 定义些 Fragment 中的监听回调，为的就是根据传递上来的信号，来改变 Fragment UI 显示
    private AdasViewContract adasVc;
    private DriveSettingsViewContract driveSettingVc;
    private AccessibilityViewContract accessibilityVc;
    private TripInfoViewContract tripInfoVc;
    private DisplaysViewContract langVc;

    // private HomeViewModel mHomeViewModel;

    //================================================================
    // Controller Listeners + Async Resources connection listener
    //================================================================
    private final ProfileData.ServiceConnectionListener profileListener =
        new ProfileData.ServiceConnectionListener() {
// 它定义了一个接口在这里，可是它并没有在回调中去处理什么逻辑，那么定义在这里是为什么呢？方便别人来连它 ??????
        @Override public void onConnected() {  }
        @Override public void onDisconnected() {  }
    };
    AsyncResources.ConnectionsListener connectionsListener =
        new AsyncResources.ConnectionsListener(
            profileListener,
            bcmListener,
```

```
            adasListener,
            null, // 这一个是什么呢？
            gearListener,
            driveModeListener
            ) {
        @Override public void onConnectedAll() {
            Log.d(TAG, "onConnectedAll");
            adasListener.onPropertiesReadyW();
            bcmListener.onPropertiesReadyW();
            gearListener.onPropertiesReadyW();
            driveModeListener.onPropertiesReadyW();
        }
        @Override public void onDisconnectedAny() {}
    };
// 留这些 controller 在这里，就是给你留几个葫芦，你可以照着画个瓢
    private final DriveModeController.DriveModeListener driveModeListener
        = new DriveModeController.DriveModeListener() {
        @Override
        public void onPropertiesReadyW() {
            if (uiHandler != null) {
                uiHandler.post(() -> {
                        // this handles the race condition if the connection has not been made
                        // yet before the user navigates to the DreamDrive tab. In this case, the UI will flash.
                        // May want to consider using a spinner IF this is happens too frequently.
                        if (driveSettingVc != null)
                            driveSettingVc.initUi();
                    });
            }
        }
        @Override public void onUpdateDriveModeW(Integer state) {
            if (uiHandler != null) {
                uiHandler.post(() -> {
                        if (driveSettingVc != null)
                            driveSettingVc.updateDriveModeStatus(asyncResources.mapToDriveMode(state));
                    });
            }
        }
    };
    private final GearController.GearListener gearListener = new GearController.GearListener() {
        public void onPropertiesReadyW() { }
        public void onUpdateGearStateChangedW(Integer state) {}
    };
    private final BcmController.BcmListener bcmListener = new BcmController.BcmListener() {
        public void onPropertiesReadyW() {}
        public void onUpdateOnePedalDrivingW(boolean isOn) {}
        public void onUpdateRegenBrakingLevelW(int value) {}
        @Override // 这个回调是作什么用的呢？
        public void onUpdateHighBeamPowerW(boolean isOn) {}
        public void onUpdateHighBeamSensitivityW(int val) {}
        public void onCreepModeStatusChangeW(int value) {}
        public void onUpdateRccdLockPowerW(boolean isOn) {}
        public void onUpdateLSCModeStatusW(int status) {}
        public void onUpdateTripADistanceW(float value) {}
        public void onUpdateTripAEnergyW(float value) {}
        public void onUpdateTripAEfficiencyW(float value) {}
        public void onUpdateTripBDistanceW(float value) {}
        public void onUpdateTripBEnergyW(float value) {}
        public void onUpdateTripBEfficiencyW(float value) {}
        public void onUpdateLastChargeDistanceW(float value) {}
        public void onUpdateLastChargeEnergyW(float value) {}
        public void onUpdateLastChargeEfficiencyW(float value) {}
    };
    private final AdasController.AdasListener adasListener = new AdasController.AdasListener() {
        public void onPropertiesReadyW() {}
        public void onUpdateHighwayAssistW(boolean isHwaOn) {}
        public void onUpdateLdpPowerW(boolean isOn) {}
        public void onUpdateLdpLevelW(int level) {}
        public void onUpdateCollisionProtectionPowerW(boolean isOn) {}
        public void onUpdateCollisionProtectionLevelW(int level) {}
        public void onUpdateCrossTrafficProtectionW(boolean isOn) {}
        public void onUpdateDriveOffAlertPowerW(boolean isOn) {}
        public void onUpdateDrowsyDriverStatusW(boolean isOn) {}
        public void onHWAStatusChanged(boolean isActive) {}
        public void onUpdateRadarStatusW(AdasController.Radar radar, Integer value){}
    };
```

```
}
```

17 LucidCarSettings 应用中 HomeActivity 中重要逻辑再稍微熟悉一下

```
public class HomeActivity extends AppCompatActivity {
    private static final String TAG = HomeActivity.class.getName();

    private AsyncResources asyncResources;
    public AsyncResources getAsyncResources() {
        return asyncResources;
    }

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

        asyncResources = AsyncResources.getInstance(this.getApplicationContext()); // 全局唯一 Singleton 吗，还是静态的呢？
        asyncResources.registerConnection(connectionsListener); // 注册监听者

        mProfileData = new ProfileData(getApplicationContext()); // IPC AIDL 的服务联接交互
        mProfileData.bindService(new ProfileData.ServiceConnectionListener() {
                @Override public void onConnected() {
                    mIsBindServiceCalled = true;
                    UserProfileData.setProfileDataInstance(mProfileData);
                }
                @Override public void onDisconnected() {
                    UserProfileData.setProfileDataInstance(null);
                }
            });
        boolean showProfileTab = getIntent().getBooleanExtra("SHOW_PROFILE_TAB", false);
        //boolean shouldShowDialog = getIntent().getBooleanExtra("SHOW_DIALOG", false);
        // 用户帐户相关的部分
        IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction(Utils.ACTION_OPEN_PROFILE_SETTINGS);
        intentFilter.addAction(Utils.ACTION_UPDATE_LUCID_ID_CONNECTION_STATUS);
        registerReceiver(mProfileSettingsUIReceiver, intentFilter);
        IntentFilter filter = new IntentFilter();
        filter.addAction(ACTION_CREATE_NEW_USER_PROFILE);
        filter.addAction(Utils.ACTION_DISPLAY_HOME_ACTIVITY_VIEW);
        getApplicationContext().registerReceiver(mProfileReceiver, filter);

        setHomeActivityView(showProfileTab); // 默认显示用户帐户

        registerMediaApplicationBroadcast();
        registerDeepLinkBroadcast();
        setUpViewModel();
        setUpObserver(); // 这里可能也需要去看一下

        // // This is required so that whenever the car settings are opened,
        // // application can essentially "invoke" the bluetooth service to initialize.
        // BluetoothUtils.getLocalBtManager(getApplicationContext());//暂时不管
        // AtmosReceiver.connectReceiver(getApplicationContext()); // 不知道这个是在干什么
    }
    // @Override public void onPause() {
    //     super.onPause();
    // }

    @Override
        protected void onStart() {
        // we need to initialize controllers before calling super.onStart()
        // as some of the fragments may be dependent on those
        Context context = getApplicationContext(); // 获得应用层面的 Context
        if (context == null) {
            Log.e(TAG, "ERROR - CONTEXT IS NULL, SO CONTROLLERS WILL CRASH");
            return;
        }
        initControllers(context); //
        // context.registerReceiver(mThemeBroadcastReceiver, CarTheme.getThemeIntentFilter());//我觉得目前这个不相关
        super.onStart();
    }
```

```
@Override
    protected void onStop() {
    super.onStop();
    // getApplicationContext().unregisterReceiver(mThemeBroadcastReceiver);//这个东西目前对我不重要
    cleanUpControllers(); //
}
@Override
    protected void onDestroy() { // "HomeActivity onDestroy called");
    getViewModelStore().clear(); //  ??????? 这是在干什么呢？
    if (mIsBindServiceCalled) {
        mIsBindServiceCalled = false;//如果建立了远程服务联接，现在断开
        mProfileData.unbindService();//断开联接
    }
    // getApplicationContext().unregisterReceiver(mProfileReceiver); // 前面不是注册了这个监听吗？！
    // unregisterReceiver(mProfileSettingsUIReceiver); // 它也只干两件事：负责显示用户帐户相关 Fragment，以及
    // unregisterMediaApplicationBroadcast();
    // unregisterDeepLinkBroadcast(); // 它只干一件事
    asyncResources.unregisterConnections(connectionsListener);
    AtmosReceiver.disconnectReceiver(getApplicationContext());
    super.onDestroy();
}
public void initControllers(Context context) { // 三个 !!!
    Log.d(TAG, "init controllers");
    bcmPresenter = new BcmPresenter(context);
    gearPresenter = new GearPresenter(context);
    alertPresenter = new AlertPresenter(context);
}
private void cleanUpControllers() {
    Log.d("TAG", "cleanUpControllers");
    if (bcmPresenter != null) {
        bcmPresenter.cleanup();
        bcmPresenter = null;
    }
    if (gearPresenter != null) {
        gearPresenter.cleanup();
        gearPresenter = null;
    }
    if (alertPresenter != null) {
        alertPresenter.cleanup();
        alertPresenter = null;
    }
}
// 它只干一件事：LucidSystemUI 应用中：IcrSettingsWindow 中几个 Tabs 中有每个 Tab 中那个键点击后，会广播要求 CID LucidCarSet
private void registerDeepLinkBroadcast(){
    IntentFilter filter = new IntentFilter();
    filter.addAction(CarSettingsConstants.ACTION_DEEP_LINK_TRIGGERED);
    registerReceiver(mDeepLinkReceiver, filter);
}
private void unregisterDeepLinkBroadcast(){
    unregisterReceiver(mDeepLinkReceiver);
}
private BroadcastReceiver mDeepLinkReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) { // 它只干一件事：要打开哪个就打开哪个
        String action = intent.getAction();
        if(CarSettingsConstants.ACTION_DEEP_LINK_TRIGGERED.equals(action)){
            Log.d(TAG, "Deep Link Triggered");
            loadStartScreenFromDeepLink();
        }
    }
};
public static boolean getPrimaryLucidIdStatus(Context context) { // 来自 Utils.java: IPC 进程间通信，用户帐户连起来了吗？
    String isLucidIdConnected = Settings.Global.getString(context.getContentResolver(),
                                                          IS_LUCID_ID_CONNECTED);
    Log.d(TAG, "getPrimaryLucidIDStatus() " + isLucidIdConnected + " boolean value: " + Boolean.parseBoolean(isLucidIdC
    return Boolean.parseBoolean(isLucidIdConnected);
}
private void refreshUserProfileLucidIdStatus(){
    UserInfo userInfo = mCarUserManagerHelper.getCurrentForegroundUserInfo(); // 前台用户
    Bitmap bitmap = mCarUserManagerHelper.getUserIcon(userInfo); // 用户圆形头像
    if (bitmap != null) {
        Bitmap circleBitmap = Utils.getCircularBitmap(bitmap);
        mProfilePicture.setImageDrawable(new BitmapDrawable(getResources(), circleBitmap));
    }
    mProfileName.setText(userInfo.name); // 用户名
```

```
    if (ProfileAccountType.ProfileType.PRIMARY_USER == Utils.getProfileType()) { // PRIMARY DRIVER
        mOwnerText.setVisibility(View.VISIBLE);
        mProfilePicture.setEnabled(true);
        mLucidIdConnectionStatus = Utils.getPrimaryLucidIdStatus(getApplicationContext()); // 定义在上面
        if (mLucidIdConnectionStatus) // 自动导出 PRIMARY DRIVE 的邮箱
            mProfileEmailId.setTitle(getApplicationContext().getString(R.string.alexa_sign_out_button).toUpperCase());
        else // 邮箱缺省为: "CONNECT YOUR LUCID ID"
            mProfileEmailId.setTitle(getString(R.string.connect_lucid_id_text));
    }
    // 不知道这些监听回调现在在干什么鬼事, 现在不用管它们
    mProfileEmailId.setOnClickListener(new View.OnClickListener() {
            @Override
                public void onClick(View v) {
                if (!mLucidIdConnectionStatus){
                    mLucidIdConnectionModal = new LucidIdConnectionModal(getApplicationContext(), mMessenger);
                    if (getSupportFragmentManager()!=null)
                        mLucidIdConnectionModal.launch(getSupportFragmentManager());
                }else{
                    UserInfo userInfo = mCarUserManagerHelper.getCurrentForegroundUserInfo();
                    Bitmap bitmap = mCarUserManagerHelper.getUserIcon(userInfo);
                    LucidIdSignOutRequestModal lucidIdSignOutRequestModal = new LucidIdSignOutRequestModal(getApplicati
                    if(getSupportFragmentManager()!=null)
                        lucidIdSignOutRequestModal.show(getSupportFragmentManager(),"LucidIdSignOutRequestModal");
                }
            }
        });
    mProfilePicture.setOnClickListener(new View.OnClickListener() {
            @Override
                public void onClick(View v) {
                if (ProfileAccountType.ProfileType.PRIMARY_USER == Utils.getProfileType()){
                    if(!mLucidIdConnectionStatus){
                        EditProfileNameAvatarModal editProfileNameAvatarModal = new EditProfileNameAvatarModal(getAppli
                        if(getSupportFragmentManager()!=null)
                            editProfileNameAvatarModal.show(getSupportFragmentManager(),"EditProfileNameAvatarModal");
                    }else{
                        UserInfo userInfo = mCarUserManagerHelper.getCurrentForegroundUserInfo();
                        Bitmap bitmap = mCarUserManagerHelper.getUserIcon(userInfo);
                        MobileAppModal mobileAppModal = new MobileAppModal(getApplicationContext(), userInfo.name, bitm
                        if (getSupportFragmentManager() != null)
                            mobileAppModal.show(getSupportFragmentManager(), "MobileAppModal");
                    }
                }else if (ProfileAccountType.ProfileType.SECONDARY_USER == Utils.getProfileType()){
                    EditProfileNameAvatarModal editProfileNameAvatarModal = new EditProfileNameAvatarModal(getApplicati
                    if(getSupportFragmentManager()!=null)
                        editProfileNameAvatarModal.show(getSupportFragmentManager(),"EditProfileNameAvatarModal");
                }
            }
        });
}
BroadcastReceiver mProfileSettingsUIReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        Log.d(TAG,"onReceive() with action: "+ intent.getAction());
        if(Utils.ACTION_OPEN_PROFILE_SETTINGS.equals(intent.getAction())) { // 如果收到要求显示用户帐户设置, 那若隐藏当前界
            mLayoutApplication.setVisibility(View.INVISIBLE);
            mCustomViewApplication.setVisibility(View.INVISIBLE);
            mCustomView.setVisibility(View.VISIBLE);
            if (currentlySelected != null && currentlySelected != accessProfileListItem) {
                currentlySelected.unhighlight();
                currentlySelected = accessProfileListItem;
                accessProfileListItem.highlight();
            }
            FragmentManager fragmentManager = getSupportFragmentManager(); // 应该是到用户帐户界面相关的什么东西
            Fragment profileFragment = fragmentManager.findFragmentByTag("AccessAndProfileFragment");
            if (fragmentManager.getFragments().size() == 0) {
                Log.d(TAG, "No fragments found");
                FragmentTransaction ft = fragmentManager.beginTransaction();
                ft.add(R.id.home_activity_secondary_view, new AccessAndProfileFragment(), "AccessAndProfileFragment");
                ft.commitNow();
            } else if (profileFragment == null || !profileFragment.isVisible()) {
                Log.d(TAG, "There is no profile UI visible to user");
                FragmentTransaction ft = fragmentManager.beginTransaction();
                ft.replace(R.id.home_activity_secondary_view, new AccessAndProfileFragment(), "AccessAndProfileFragment
                ft.commitNow();
            }
```

```
            mProfileRecentlyOpened = true;
            new Handler().postDelayed(() -> mProfileRecentlyOpened = false, 1000);
        } else if (Utils.ACTION_UPDATE_LUCID_ID_CONNECTION_STATUS.equals(intent.getAction())){
            Log.d(TAG,"Broadcast received ACTION_UPDATE_LUCID_ID_CONNECTION_STATUS");
            refreshUserProfileLucidIdStatus();
            getApplicationContext().sendBroadcast(new Intent(Utils.ACTION_UPDATE_RESET_PIN_ACCESS)); // 为什么这里崩出来
        }
    }
};
BroadcastReceiver mProfileReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) { // 它只做两件事
        Log.d(TAG, "onReceive() with action: " + intent.getAction());
        if (ACTION_CREATE_NEW_USER_PROFILE.equals(intent.getAction())) {
            int imageId = intent.getIntExtra(EXTRA_AVATAR_INFO, 0);
            int[] images = Utils.getImages();
            String profileName = intent.getStringExtra(EXTRA_USER_PROFILE_NAME);
            mProfilePicture.setImageResource(images[imageId]);
            mProfileName.setText(profileName);
        } else if(Utils.ACTION_DISPLAY_HOME_ACTIVITY_VIEW.equals(intent.getAction())){
            setHomeActivityView(false); // 大致是一大堆跟显示这个活动以及用户帐户相关的东西
        }
    }
};
```