**Q/HR**

海 尔 集 团 企 业 标 准

Q/HR XXXX XXX—2018

SmartHomeSDK使用文档

2018 - XX – XX 发布 2018 - XX - XX 实施

前 言

该文档适用于需要使用SmartHomeSDK的开发者，文档中详细讲解了SmartHomeSDK的初始化、账号的登录、MDT的获取、设备列表，判断设备是否在线，获取设备属性以及控制设备，并有相关的示例代码。

SmartHomeSDK的设计模式（MVVM）是基于LiveData和ViewModel做的数据绑定和业务处理，使用者只需要关注View层即可。

LiveData 是一个可以感知 Activity 、Fragment生命周期的数据容器。当 LiveData 所持有的数据改变时，它会通知相应的界面代码进行更新。

ViewModel 将视图的数据和逻辑从具有生命周期特性的实体（如 Activity 和 Fragment）中剥离开来。直到关联的 Activity 或 Fragment 完全销毁时，ViewModel 才会随之消失。

本文档由海尔智海尔优家智能科技（北京）有限公司负责起草

文档修订记录

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **版本编 号** | **变化**  **状态** | **变更内容和范围** | **变更日期** | **变更人** | **批准日期** | **批准人** |
| 1.1.1 | A | 添加使用说明 | 2018.9.5 | 章冰 |  |  |

\*变化状态：A——增加，M——修改，D——删除，N——正式发布

SmartHomeSDK使用文档

1 前提

使用该SDK前需先在设备上安装GE登录APP。

2 阅读对象

适用于使用SmartHomeSDK开发的相关开发人员和UI设计人员。

3 API调用

3.1 SDK初始化

|  |  |
| --- | --- |
| 接口 | SmartHomeSdk.getInstance().init(Context context, Boolean isShowLog, Boolean isProduction) |
| 功能 | 初始化SmartHomeSDK，需在Application类的onCreate方法中调用 |
| 备注 |  |

3.1.1 参数说明

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| context | Context | 上下文 |
| isShowLog | Boolean | 是否显示log日志（开发建议显示）true：显示；false不显示 |
| isProduction | Boolean | 测试环境还是生产环境，true：生产环境；false测试环境（注意：环境和EG登录APP环境保持一致，开发中建议使用测试环境） |

3.1.2 示例代码

public class SmartHomeApplication extends Application {  
 @Override  
 public void onCreate() {  
 super.onCreate();  
 SmartHomeSdk.*getInstance*().init(this, true, false);  
 }  
}

3.2 登录GE账号获取MDT

在应用的启动页面中登录GE账号，登录成功后会把MDT保存在本地SharedPreferences，下次进入应用直接从SharedPreferences拿到MDT，不再登录，getMDT(final Activity activity, final LoginCallBack mloginCallBack)。

3.2.1 参数说明

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| activity | Activity | 当前的activity |
| loginCallBack | LoginCallBack | 账号登录的回调，  onAddAccountFailed() 登录失败  onAddAccountSucceed（）登录成功  onUserCancelLogin（）用户取消登录  onUserSkipLogin（）用户跳过登录  onLoginAppNotInstall（）GE账号APP未安装 |

3.2.2 示例代码

/\*\*  
 \* 此方法在Activity初始时调用。

\*/

private void initMdt() {

SmartHomeSdk.getInstance().getMDT(this, new LoginCallBack() {

@Override

public void onAddAccountFailed(String errMsg, boolean isUserManuallyCancel) {

//todo 登录失败后的操作

finish();

}

@Override

public void onAddAccountSucceed(String mdt) {

deviceListViewModel.refresh();

}

@Override

public void onUserCancelLogin() {

finish();

}

@Override

public void onUserSkipLogin() {

finish();

}

@Override

public void onLoginAppNotInstall() {

finish();

}

});

}

3.3 获取设备列表（DeviceListViewModel）

在onChanged（）回调方法中进行UI刷新。如果数据为null，则请求失败，不为null，则请求成功。

如果想再次刷新数据，可以调用refresh()，上述3.2.2中的示例代码在获取MDT成功后调用refresh()进行了数据的刷新。

3.3.1 参数说明

无

3.3.2 示例代码

/\*\*  
 \* 此方法在Activity初始时调用。

\*/

private void observeDeviceListViewModel() {

deviceListViewModel = ViewModelProviders.of(this).get(DeviceListViewModel.class);

deviceListViewModel.getObservable().observe(this, new Observer<DeviceListBean>() {

@Override

public void onChanged(@Nullable DeviceListBean deviceListBean) {

if (deviceListBean != null && !deviceListBean.items.isEmpty()) {

//TODO 刷新UI列表

}

}

});

}

3.4 判断设备是否在线（JudgeOnlineViewModel）

在onChanged（）回调方法中进行UI刷新。如果数据为null，则请求失败，不为null，则请求成功。

如果想再次刷新数据，可以调用refresh(String applianceId)

3.4.1 参数说明

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| applianceId | String | 设备id |

3.4.2 示例代码

/\*\*  
 \* 此方法在Activity初始时调用。

\*/

private void observeJudgeOnlineViewModel() {

judgeOnlineViewModel = ViewModelProviders.of(this).get(JudgeOnlineViewModel.class);

judgeOnlineViewModel.getObservable().observe(this, new Observer<DeviceListEntryBean>() {

@Override

public void onChanged(@Nullable DeviceListEntryBean applianceListEntryBean) {

if (applianceListEntryBean != null && !TextUtils.isEmpty(applianceListEntryBean.online)) {

if (applianceListEntryBean.online.equals("ONLINE")) {

//TODO 设备在线

} else {

//TODO 设备离线

}

}

}

});

}

3.5 获取设备属性（DeviceAttributesViewModel）

在onChanged（）回调方法中进行UI刷新。如果数据为null，则请求失败，不为null，则请求成功。

如果想再次刷新数据，可以调用refresh()。

3.5.1 参数说明

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| applianceId | String | 设备id |

3.5.2 示例代码

/\*\*  
 \* 此方法在Activity初始时调用。

\*/

public void observeDeviceAttributes() {

DeviceAttributesViewModel.Factory factory = new DeviceAttributesViewModel.Factory(getApplication(), deviceInfo.applianceId);

deviceAttributesViewModel = ViewModelProviders.of(this, factory).get(DeviceAttributesViewModel.class);

deviceAttributesViewModel.getObservable().observe(this, new Observer<DeviceAttributesBean>() {

@Override

public void onChanged(@Nullable DeviceAttributesBean deviceAttributesBean) {

if (deviceAttributesBean != null) {

//TODO刷新UI

}

}

});

}

3.6 控制设备（DeviceControlViewModel）

在onChanged（）回调方法中进行UI刷新。如果数据为null，则请求失败，不为null，则请求成功。

如果想再次刷新数据，可以调用refresh(String command, List<CommandBodyBean.DeviceProperty> properties)。

3.6.1 参数说明

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| applianceId | String | 设备id |
| userId | String | 用户id |
| command | String | 指令 |
| properties | List<CommandBodyBean.DeviceProperty> | 修改的配置 |

3.6.2 示例代码

/\*\*  
 \* 此方法在Activity初始时调用。

\*/

public void observeDeviceControl() {

DeviceControlViewModel.Factory factory = new DeviceControlViewModel.Factory(getApplication(), deviceInfo.applianceId, deviceInfo.userId);

deviceControlViewModel = ViewModelProviders.of(this, factory).get(DeviceControlViewModel.class);

deviceControlViewModel.getObservable().observe(this, new Observer<ControlResultBean>() {

@Override

public void onChanged(@Nullable ControlResultBean controlResultBean) {

if (controlResultBean != null) {

//TODO 请求成功

} else {

//TODO 请求失败

}

}

});

}