# 设备配对

## 场所

创建一个场所对象（Place）（自定义对象），并存入数据库，配对设备需要提供场所属性

Place拥有的个体属性：

String meshKey；Mesh的网络名称

String meshPassword； Mesh的网络密码

String meshLTK；

## 设备配对参数

创建设备配对参数对象（LeAutoPairParameters）（对象来自依赖包）

对象属性可以使用链式风格设置

pairParameters = new LeAutoPairParameters();

pairParameters.setMeshName("ZenggeMesh").

setPassword("ZenggeTechnology").

setNewName(meshPlace.getMeshKey()). 场所的meshKey

setNewPassword(meshPlace.getMeshPassword()). 场所的meshPassword

setLTK(meshPlace.getMeshLTK()); 场所的meshLTK

## 配对任务

创建配对任务对象（AutoPairTask）（对象来自依赖包）

AutoPairTask(Context context, LeAutoPairParameters pairParameters,int startAddress);

Context为当前上下文，pairParameters为设备配对参数

startAddress为设备准备的mesh地址，每配对一个设备后meshAddress都需要加1，传给下一个创建AutoPairTask对象，同一个场所内的设备mesh地址不能相同，可以从1开始累加

配对设备回调接口（OnAutoPairLightRequestListener）（接口来自依赖包）

此接口会回调配对完成的设备信息

调用AutoPairTask对象的setListener（OnAutoPairLightRequestListener listener）

重写方法void onPairedDevice(DeviceInfo deviceInfo)

deviceInfo为设备信息，而暂时只使用设备信息里的macAddress（物理地址，唯一标识）、meshAddress（mesh地址，配对前传入）、ProductUUID（高8位版本号，低8位设备码）

最好自定义一个设备属性对象，把以上信息保存到数据库

分别调用AutoPairTask的startPair（），stopPair（）方法开始配对和停止配对

# 设备连接

## LightService

该类是抽象类，需要创建实体类继承，建立单例模式更合适。

重写onCreate方法new LightAdapter对象赋值到mAdapter，之后调用mAdapter.start（Context context）。

重写onBind方法 new Binder对象赋值到mBinder

该类的重要方法：

自动重连，主动搜索mesh内所有设备

autoConnect（LeAutoConnectParameters）

调用过程：

\* LeAutoConnectParameters params = LeAutoConnectParameters.create();

\* params.setMeshName(mesh.getMeshName());

\* params.setPassword(mesh.getPassword());

\* params.autoEnableNotification(true);

//params.setConnectMac(mac) 可以选择是否调用，也可以传入null不指定搜索目标

\* service.autoConnect(params);

开启自动刷新网络通知

autoRefreshNotify（LeRefreshNotifyParameters）

调用过程：

\* LeRefreshNotifyParameters refreshNotifyParams = LeRefreshNotifyParameters.create();

\* refreshNotifyParams.setRefreshRepeatCount(2);

\* refreshNotifyParams.setRefreshInterval(1000);

\* service.autoRefreshNotify(refreshNotifyParams);

通常autoConnect会和 autoRefresNotify一起使用

空闲模式

IdleMode（boolean disconnect）disconnect代表是否断开当前的连接

向设备发送控制命令

sendCommandNoResponseImmediate（byte opcode，int address，byte[] params）

sendCommandNoResponseDelay(byte opcode, int address , byte[] params, int delay)

opcode 操作码（参考设备协议）

address 设备的地址（0xFFFF时向同一mesh内所有设备发送）

params 参数（参考设备协议）

关闭自动刷新网络通知

disableAutoRefresNotify（）

获取当前工作模式

int getMode（）

MODE\_AUTO\_CONNECT\_MESH = 8 自动搜索模式

MODE\_IDLE = 1 空闲模式

是否已经登录到设备

isLogin（）

连接到指定设备

connect（String mac, int timeOutSeconds）

断开所有连接

disconnect（）

## MeshApplication

该对象需要被启动Application继承，可重写doInit（）方法，并且必须在onCreate（）方法内调用doInit（）和startLightService（LightService）。

设置监听事件：application.addEventListener(String eventType, EventListener<String> listener)

eventType 事件类型 listener 事件监听器

### 事件类型：

MeshEvent.BLE\_ON、MeshEvent.BLE\_OFF 蓝牙的开关

ServiceEvent.SERVICE\_CONNECTED 、ServiceEvent.SERVICE\_DISCONNECTED

服务连接成功和断开

MeshEvent.ERROR 错误

MeshEvent.OFFLINE Mesh连接失败

DeviceEvent.STATUS\_CHANGED

Mesh连接状态改变 监听回调的参数event.getArgs()得到的对象DeviceInfo

DeviceInfo.status为设备状态，分别为：

LightAdapter.STATUS\_LOGIN; //MeshLogin成功

LightAdapter.STATUS\_CONNECTING;//蓝牙连接中

LightAdapter.STATUS\_LOGOUT;

NotificationEvent.ONLINE\_STATUS //在线状态改变

加入监听前需注册解析器application.registerNotificationParser(NotificationParser parser)

NotificationEvent.GET\_GROUP //获取设备分组通知

加入监听前需注册解析器application.registerNotificaitonParser(NotificaitonParser parser)

NotificationParser为抽象类，需被继承重写parse方法，parse方法会传入参数NotificationInfo，notificationInfo内有属性opcode，src（源地址），params，DeviceInfo，具体如何重写，参考demo和协议