OneNet 接入文档

OneNet 信息入口:
https://github.com/cm-heclouds SDK
http://open.iot.10086.cn/doc 开发文档
http://open.iot.10086.cn/product 开发者中心

一、接入 OneNet SDK 方法

1.从 GitHub 上 OneNet_SDK 链接上获取 Android_SDK，导入 SDK (附带 sample 教程)，中间需要在 SDK 上添加 httpclient-4.2.5.jar，httpcore-4.2.4.jar 后编译。

2.在 OneNetDemo 项目中导出 onenetapi-library，并进行编译。

二、获取 OneNet MasterKey

由于 OneNet SDK 没有提供注册登录的 API，所以使用者必须先获取 MasterKey 才能进行获取设备列表。具体步骤见《OneNet 开发文档》->《接入帮助 》->注册/登录->新增产品->新增设备

接入流程总览：

获取 MasterKey(即 APIKey)，如图，拷贝应用在应用程序中。

或者可以在 Application 上修改默认 MasterKey:

```
public static String sApiKey = "BCl6eQcvLr6Jx82hoCyUORSuBZY="; // 此处填写账户的 MasterKey
```

三、使用的 API 接口解析

1.获取设备列表

oneNetApi.getDevices(myApplication.sApiKey, null, null, null, null, null, null, new ResponseListener() ;

解析结果：

```
    }
03-10 10:09:59.487 17453-17453/com.aithinker.tamas.onenetdemo I/MainActivity:
[getDevices] getError {"errno":0,
"data":{"per_page":30,"devices":[
{"private":true,"protocol":"HTTP","create_time":"2017-02-27 10:14:55","online":false,"location":{"lon":0,"lat":0},
"auth_info":"A12517","id":"4723940","title":"安信可","desc":"安信可之家","tags":[]},
{"private":true,"protocol":"HTTP","create_time":"2017-01-04 11:50:32","online":false,"location":{"lon":0,"lat":0},
"auth_info":"AI10001","id":"4600434","title":"AISmart","desc":"安信可科技，智能控制家电设备。","tags":[]}],
"total_count":4,"page":1},"error":"succ"}
```

用 Gson 解析方法获取以下数据

```
@SerializedName("private")
private boolean isPrivate;
private String protocol;
private String route_to;
private String create_time;
private boolean online;
private String id;
private String title;
private String desc;
```

2.设备列表控制

主要是添加设备，删除设备，编辑设备

a. 添加设备

当 getError() =0 表示添加成功。

```
//自动添加设备
oneNetApi.addDevice(myApplication.sApiKey, "OneNet_demo", "OneNetApi_Demo", false, "http://www.baidu.com", null, new ResponseListener() {

    @Override
    public void onResponse(OneNetResponse response) {
        Log.i(TAG, "[addDevice]" + response.getRawResponse());
        if (response.getErrno() == 0) {
            // 请求成功
            getDevices();
        }
    }

    @Override
    public void onError(OneNetError error) {

    }
});
```

b.删除设备

当 getError() =0 表示删除成功。

```
//删除一个设备
oneNetApi.deleteDevice(myApplication.sApiKey, device.getId(), new ResponseListener() {
    @Override
    public void onResponse(OneNetResponse response) {
        if (response.getErrno() == 0) {
            lvDevice.hiddenRight(lvDevice.getmCurrentItemView());
            getDevices();
        }
    }

    @Override
    public void onError(OneNetError error) {

    }
});
```

c.编辑设备信息

当 getError() =0 表示编辑成功。

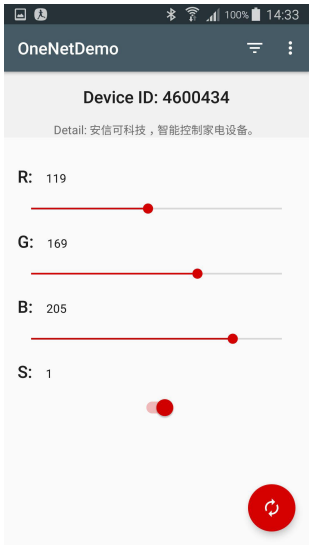
```
//编辑设备信息
oneNetApi.editDevice(myApplication.sApiKey, device.getId(), name, desc, isPrivate, new ResponseListener() {
    @Override
    public void onResponse(OneNetResponse response) {
        if (response.getErrno() == 0) {
            lvDevice.hiddenRight(lvDevice.getmCurrentItemView());
            getDevices();
        }
    }

    @Override
    public void onError(OneNetError error) {

    }
});
```

3.设备数据流控制

每次新建设备后，并进行设备通信的过程中，发送相应的数据流到服务器，服务器会相应新建数据流。例如：新建一个新的设备点，发送”age”的数据流，假使服务器没有”age”则会主动新建设备流。当然也可以自己新建自己想要的数据流，通过数据流来跟服务器通信。这里我们为做球泡灯的透传通信，将数据流固定为datasStreamIds={”r”,”g”,”b”,”s”}

```
    private final String key_r = "r";
    private final String key_g = "g";
    private final String key_b = "b";
    private final String key_s = "s";
    private final String[] datasStreamsId = {key_r, key_g, key_b, key_s};//数据流
```

设备数据流控制界面：获取数据流，接受数据，发送数据



a. 获取数据流列表

```
oneNetApi.getDatastreams(myApplication.sApiKey, deviceId, datasStreamsId, new ResponseListener() {
    @Override
    public void onResponse(OneNetResponse response) {
        swipeRefresh.setRefreshing(false);
        Log.i(TAG, "[getDatastreams]" + response.getData());
        if (response.getErrno() == 0) {
            Gson gson = new Gson();
            Stream stream = gson.fromJson(response.getRawResponse(), Stream.class);
            streamList = stream.getData();
            mHandler.sendEmptyMessage(1001);
        }
    }

    @Override
    public void onError(OneNetError error) {
        swipeRefresh.setRefreshing(false);
    }
});
```

解析结果

```
{"create_time":"2017-03-07 10:27:51","update_at":"2017-03-10 14:33:35","id":"s",
"uuid":"f78dc8e6-49d2-456a-9d14-834cf99d79b0","current_value":1},
{"create_time":"2017-03-07 10:27:49","update_at":"2017-03-07 10:27:49","id":"b",
"uuid":"e9e11ea1-baf7-4a7a-8b65-caa519a4b806","current_value":205},
{"create_time":"2017-03-07 10:27:46","update_at":"2017-03-07 10:27:45","id":"g",
"uuid":"98f3ef49-2e6e-449c-810b-20dbf47492eb","current_value":169},
{"create_time":"2017-03-07 10:27:44","update_at":"2017-03-07 10:27:43","id":"r",
"uuid":"c1c8971f-0367-4ddd-a9d0-539166d8a648","current_value":119},
```

用 Gson 解析方法获取以下数据

```
public static class Bean{
    private String at;
    private String value;
```

b. 获取单个数据流

获取的单个数据流的数据：

[getDatastream]{"create_time":"2017-03-07 10:27:44","update_at":"2017-03-10 15:23:31","id":"r","current_value":64}

```
oneNetApi.getDatastream(myApplication.sApiKey, deviceId, stream, new ResponseListener() {
    @Override
    public void onResponse(OneNetResponse response) {
        Log.i(TAG, "[getDatastream]" + response.getData());
        if (response.getErrno() == 0) {
            Gson gson = new Gson();
            Stream.streamBean itemBean = gson.fromJson(response.getData(), Stream.streamBean.class);
            updateUI(itemBean);
        }
    }

    @Override
    public void onError(OneNetError error) {

    }
});
```

c. 发送数据

发送数据分为两步进行：(1)发送数据到服务器，(2)EDP 发送数据到设备。

(1) 发送数据到服务器，如：{"value",100}

当 getError() =0 表示添加成功。

```
oneNetApi.addDatapoint(myApplication.sApiKey, deviceId, streamId, datapoints, new ResponseListener() {
    @Override
    public void onResponse(OneNetResponse response) {
        Log.i(TAG, "[addDatapoint]" + response.getRawResponse());
        if (response.getErrno() == 0) {
            mHandler.sendEmptyMessageDelayed(1002, 1000);
        }
    }

    @Override
    public void onError(OneNetError error) {

    }
});
```

2)EDP 发送数据到设备

当 getError() =0 表示添加成功。

```
oneNetApi.sendToEdp(myApplication.sApiKey, deviceId, value, new ResponseListener() {
    @Override
    public void onResponse(OneNetResponse response) {
        Log.i(TAG, "[sendToEdp]" + response.getRawResponse());
    }

    @Override
    public void onError(OneNetError error) {

    }
});
```

备注：

1.OneNet 暂时还没提供让手机直接控制设备，只能分两步来完成，发送消息到服务器，服务器不会转发到相应的设备，则手机 APP 必须再通过 EDP 发送到设备。

2.设备发送数据到服务器，服务器没有自动反馈到手机 APP 上，则无法监听到设备的变化，只能通过轮询的方式每隔 1000ms 向服务器获取数据流的信息。

3.应用嵌入:手机通过 webview 嵌入应用模块直接控制模块，暂时无法完成。

附录：

《JavaForOneNet 接口文档.doc》