16/4/8

Version：draft v0.9

Author：jimmy

Contact me:jimmy.chen@coolkit.cn

COOLKIT SDK REFERENCE

COOLKIT SDK STEP BY STEP:

1:Integration to project

 A jar for android project only is provided，which minSdkVersion is 11, add this jar file to your android studio project or eclipse project.

2:basic use：

2.1 project initiation

2.1.1 apply for APPID and app SECRETKEY
each develop should apply for APPID and SECREAKEY from our developer center, APPID and SECRETLY   are used as an identity of developer, which is unique ,they should be carefully kept secret,

we provide both test host and release host.test host is provided as an develop environment, release host is provided as an release environment,set host to release host when your publishing your app,set host to test host when your are developing.

an test APPID and SECRETKEY are provided, they can be only used in test environment, when your are publishing app,set APPID and SECRETKEY to release ones of yours.

2.1.2 do initiation
we provide
AppHelper to do initiation ,create a singleton an global instance of Apphelper, invoke init method ,see code 1.

2.1.2global account system：
we support global account ,each account belongs to a region, when you get signed in,serve will response an api,at,region and other informations, with these three params, do init host, it will generate urls whit region information,save these to share preference file, so your don't need to login every time to get region information.

see code 1:

2.1.2 important classes：
com.coolkit.protocol.request.ProtocolHandler,
this is a http result abstraction,when your net request ends,sdk will callback this the callback method with a param of class Result,which has a string member mMsg that your can get sever response JSON object ,you should deal with JSON response and refresh ui,see api reference for more information.

CODE 1:
```
public class HkApplication extends Application {

	public AppHelper mHelper;
	public String at;
	public String userName;
	public String apikey;

	@Override
```

```
        public void onCreate() {
                super.onCreate();

                mHelper=new AppHelper(this);

//this is provided for develop version ,remove  setDebug statement and
//set your owner APPID and  SECRETKEY when this is an release version.
mHelper.setDebug(  "54.223.98.144","54.223.98.144");

                String mAppid = "1xMdjbmOBYctEJfye4EjFLR2M6YpYyyJ";
                String mSecret = "hd9yf3DB7Q4UL6gx8iCfUGXwtYoxhCs5";
                mAppHelper.initApp(mAppid,mSecret);



                mHelper.initApp(appid, Secret);
mHelper.setHost(at,apikey,region);//if user has no login,at,apikey,region
//is empty


}
}
```

2.1
account management
account name is found of phone number and country code and plus symbol.for example, country code of china is 86, a user whose phone number 13714758456,when he go register ,a valid account name is “ + 8613714758456” .

2.1.1registration：

please check api reference
example:

```
new UserProtocol(context.app.mHelper).doRegister(mContext, new
ProtocolHandler(mContext, new CallBack() {
                                @Override
                                public void callBack(Result result) {

                        }), phone, pwd, smsCode);
```

smsCode

params phone is account name,pwd is account password, smsCode is a verification code sent to phone .which means your should

request sms verification.see 2.1.3

2.1.2 reset password:
example:

```
final ProtocolHandler handler = new ProtocolHandler(mContext) {



         @Override
         public void callBack(Result result) {


      };
}
     new UserProtocol(context.app.mHelper).doResetPwd(handler,
phone, pwd, smsCode);
```

2.1.3:request a sms verification:

```
SmsProtocal
doSms(ProtocolHandler handler, String phone)

final ProtocolHandler handler = new ProtocolHandler(mContext) {



         @Override
         public void callBack(Result result) {


      };
}
new SmsProtocal(context.app.mHelper).doSms(handler, phone);
```

2.1.4:sign in

```
doLogin(ProtocolHandler handler,
                 String password, String email, String phoneNumber)
```

```
final ProtocolHandler handler = new ProtocolHandler(mContext) {


            @Override
            public void callBack(Result result) {


        };
}
new UserProtocal(context.app.mHelper).doLogin(handler, phone);
```

2.2device management：
device management apis provider searching ,config device,device registration,naming and so on.

2.2.1searching device：
DeviceHelper
getAroudDevice()
make sure device is on config mode, invoke this method will result with an ScanResult list,each ScanResult represent a device。

2.2.2 config device ：
a.connect to device hotspots,when a device in config mode, it is a hotspots at the same time：invoke
connectTOVp(String ssid) to join in device hotspots.

b.listener to phone wifi changes,listener to broadcast ConnectivityManager.CONNECTIVITY_ACTION,check if has connected to device hotspots.

c.get device infos, when device is a hotspots,device is running as an http serve at the same time,request following url to get http response of device infos：

```
new
DeviceProtocol(mContext.app.mHelper).connectToDevice(url,
pHandler);
```

url为：http:// “ + ip + “/device，ip is gateway ip or DHCP serve ip.

d:send device our home wifi infos：
we must send our home wifi ssid and password to let device join to our home wifi .

request to device hotspots with following url：
http://ip/ap，

for example:

```
msg = doPostInfo(final String url, final  String homessid, final  String
homessid_pwd
```

msg will be { “error” :0} when request done successfully.
2.2.3:registration of device:
we must record device of owner when your and registering a device invoke following api to request COOLKIT serve to do this.
  new DeviceProtocol(mContext.app.mHelper).doAddDevice(new Regester(), deviceid, apikey,
                              mPengingName, app.at);

make sure this should be done when you have reconnected to home wifi (did you remenber you just join device hotspots?reconnect to home wifi now.)

2.2.4query current account devices:

see DeviceProtocol
queryUserDevice(Context context,String at, CallBack cal)

2.2.5 delete current account device:
See DeviceProtocol

doDeleteDeviceDetail(String deviceId, ProtocolHandler handler,String at)

2.2.6:rename a device:
See DeviceProtocol
doPostDetail(String sessionId, String deviceId, ProtocolHandler handler, JSONObject json)

3:controll device:

see AppHelper:
postWsRequest(WsRequest wsRequest);
example:
WsRequest res = WsRequest(JSONObject json)
postWsRequest(res);

WsRequest has a JSON object, see following explanation

3.1:listener to device state change ,like device online /offline
broadcast: com.homekit.action.SYSMSG

WEB SOCKET will response a JSON information when device online /offline, the JSON object :

ID	name	type	ALLOW NULL	explanation

1	action	String	N	sysmsg
2	apikey	int	N	user apikey
3	deviceid	String	N	
4	params	PARAMS	N	

PARAMS:

ID	名称	类型	允许NULL	说明
1	online	String	N	true:device online false:device offline

3.2:controll device：
turn on/off device

ID	名称	类型	允许NULL	说明
1	action	String	N	update
2	apikey	int	N	绑定用户的apikey
3	deviceid	String	N	
4	userAgent	String	N	device： sent by device app: sent by app
5	selfApiKey	String	Y	App only：to tell serve if your are owner or shared user
6	sequence	String	Y	App request id ,should be unique
6	params	PARAMS	N	params sent to device ,serve just do transfer, never change this

PARAMS：
params is protocol between device and app, device params are different。for example：
single outlet device ：

```
"params":{
    "switch":"on"
 }
```

multiple outlet device：

```
"params": {
        "switches": [
          {
              "switch": "off",
              "outlet": 0
          },
          {
              "switch": "off",
              "outlet": 1
          },
          {
```

```
                    "switch": "off",
                    "outlet": 2
                },
                {
                    "switch": "off",
                    "outlet": 3
                }
            ]
        }
```

:

send control request important code:

```
        json.put("action", "update");
                                json.put("apikey", mDeviceEntity.mApiKey);
                                json.put("deviceid", mDeviceEntity.mDeviceId);
                                json.put("params", new JSONObject(mParms));
                                json.put("userAgent", "app");
                                json.put("sequence", System.currentTimeMillis() +   “");
json.put("selfApikey", selfkey);
mAppHelper.PosetRequest(new WsRequest(json){
public void callBack(){
//
}}){
}
```