HQconfig

MainActivity的onCreate里面：

initWifi实现了当前wifi的ssid和密码保存，后面配置结束后(中间要切换到hotspot)，可以恢复之前的wifi。

代码追踪：

操作时按下扫描按键，然后就全自动了。

我们根据方法来看：

# 数据下载：

## ParseData：

* 对data处理下，判断是下载配置文件还是Firmware。mDownloadCommond
* checkVersion:检查下HQConfig版本信息，提示用户是否要继续配置。
* startConfig: 版本信息一致，继续往下走

## startConfig：

* saveWifi:把data里的wifi ssid和密码保存下来，why? 了解了，一个是在使用本地导入的时候可以读一次，或者下载config文件过程中，关闭了wifi能够连接回来。
* sendConnect:继续往下走

## sendConnect：

* startScan：搜索可用wifi
* MSG\_CONNECT\_WIFI：发送消息

## MSG\_CONNECT\_WIFI:

判断联网状态，如果联网了，onConnect

如果没联网，connectWifi,然后checkWifiConnect，弯弯绕绕最后还是会调用到onConnect

* connectWifi:联网，连接另一台设备的热点
* onConnect:继续往下走

## onConnect:

根据mDownloadCommond弹不同的tips

## MSG\_CHECK\_NETWORK\_AVAILABLE

这里checkNetworkAvailable，但是无论是false还是true都往下走了。那检查的意义何在呢？

## sendDownload:

直接往下

## initFTPServerSetting：

* initFTPSetting：FTP设置初始化
* download：

## download:

根据mDownloadCommond确定是

* downloadConfigs>>> downLoadFile
* downloadFirmware>>> downLoadFolder

注意，这些download操作都是在HandlerThread这个子线程里面操作的。

## downloadConfigs:

* downLoadFile:核心是ftpClient.retrieveFile
* MSG\_UPDATE\_CONFIG:下载完后,resetWifi，也就是把网关了，继续往下走

## MSG\_UPDATE\_CONFIG:

前面文件下载完了，现在开始解压缩，开始配置。

这里要注意，因为Activity里面的onDestory把

onBarcodeEvent(数据提取)>>>parseData(数据解析)>>> startConfig>>> sendConnect>>>

sendMessage(MSG\_CONNECT\_WIFI)>>>onConnect>>>

sendMessage(MSG\_CHECK\_NETWORK\_AVAILABLE)>>> onNetworkAvailable>>> sendDownload>>> sendMessage(MSG\_DOWNLOAD)>>> initFTPServerSetting>>>download>>>

downloadConfigs>>> downLoadFile>>> MSG\_UPDATE\_CONFIG

# 开始配置

MSG\_UPDATE\_CONFIG>>> updateConfig>>> updateCfg>>> startInstall>>> startSettingsService(这个暂时没啥用)

## updateConfig:

* UnZipFolder:对上面下载过来的configs.zip文件进行解压，直接主线程做的，解压到tmp目录。
* parseSystemConfig :Settings的Global,system,secure.xml配置
* 拷贝文件：

拷贝了HQconfig的sharedPreference.xml(后续要考虑拷贝其它apk的sharedPreference的可行性)

拷贝了apk(这里的apk拷贝应该是可以没有必要的，指定路径就可以了，除非静默安装的过程中客户关机了，没执行完，然后去autoinstall里面手动再check安装一遍。这可能性太小了)

拷贝了xml

/data/user/0/com.honeywell.hqconfig/shared\_prefs

/storage/emulated/0/deployment/autoinstall

/storage/emulated/0/honeywell/ezconfig/generated

* 发广播去配置DeviceConfig.xml

## updateCfg：

* 发广播去配置DataCollectionService.xml，这个看了下主要是扫描设置的

## startInstall：

* 发广播通知autoinstall去静默安装apk

## 优化空间与疑问：

生成Configs.zip，拷贝Configs.zip,解压Configs.zip到tmp目录，然后在tmp目录拷贝到指定目录。

这有必要吗，起码可以直接解压，不用解压后再拷贝啊

结论：拷贝apk的操作可以省略

EzConfig的xml这么多，为什么只配置了DeviceConfig.xml和DataConllectionService.xml?

其它的不用管吗？还是说配置一个文件的时候其他文件也触发到了？

结论：看了下，大部分的配置项在DeviceConfig.xml里面，而扫描设置相关则在DataConllectionService.xml里面。

MainActivity类里面ConfigManager做了些什么？

没搞什么啊，就提供了二维码的string内容

# HQConfigActivity

上面的都是在MainActivity里面，后面看HQConfigActivity

HQConfigActivity大概做了几件事，生成二维码，收集配置文件，配置文件压缩为zip包，创建hotspot以及共享区。

generateBarCode>>>pack>>>register>>>handleStart>>>reopenWifiHot>>>registerPack

## generateBarCode:

这个没什么好说，intent传来的数据生成一个二维码

## pack(false):

sendExplicitBroadcast(xxx.generatexml)：让EzConfig去生成xml文件

com.honeywell.ezprovider.generatexml

## register:

注册hotspot的广播，也就是hotspot被关的时候又给打开

## handlestart：

这是父类的一个方法，开启SshServerService

## reopenWifiHot：

没什么好说的，打开热点

## registerPack：

监听前面EzConfig生成xml文件的广播，知道xml文件生成好后

两个广播：

com.honeywell.ezprovider.generatexmlend

收到这个，去发com.honeywell.ezconfig.intent.action.EXPORT\_XML

以及com.honeywell.tools.deviceconfig.exportxml

这里得去追下，xml文件生成后还要导出？，导出到哪？

com.honeywell.tools.deviceconfig.exportxmlfinished

收到这个后，延迟0.5秒去pack(true),为啥要加延时？

然后走到ConfigManager.pack

Pack里面

**for** (Map.Entry<String, BaseConfig> entry : mConfigs.**entrySet**()) {  
 Message message = Message.**obtain**();  
 message.what = ***MSG\_PACK***;  
 message.obj = entry.**getValue**();  
 message.**getData**().**putString**(Const.***DEPLOYMENT\_CONFIG***, Const.***DEPLOYMENT\_FILE***);  
 handler.**sendMessage**(message);  
}

发了多个MSG\_PACK,至于for循环里面的Map则是在ConfigManager的构造方法调用的时候就加进去了。

ConfigManager的构造方法：

**private ConfigManager**(Context context){  
 mContext = context;  
 **addConfig**(**new WiFiConfig**(context));  
 **addConfig**(**new XMLConfig**(context));  
 **addConfig**(**new SystemConfig**(context));  
 **addConfig**(**new APKConfig**(context));  
 **checkPath**();  
}

checkPath则是去创建了storage/0/emulated/deployment目录和/storage/IPSM/deployment,为什么不把必要文件夹的创建放在MainActivity里呢？回头移一下

## MSG\_PACK:

指定最后的zip包存放位置为storage/0/emulated/deployment

最后会走到WiFiConfig,XMLConfig,SystemConfig,APKConfig的onPack方法。

* APKConfig:提供了

/storage/0/emulated/deployment/autoinstall目录

* SystemConfig:提供了/data/system/users/0/settings\_system&settings\_global&settings\_secure.xml
* XMLConfig:提供了/

/data/user/0/com.honeywell.hqconfig/shared\_prefs

/storage/0/emulated/honeywell/ezconfig/generated

是否可以对这些目录 进行重命名呢？

看到这里也就解释了，为什么要生成configs.zip来进行传输，因为zip包的文件是来自各个目录的，而我们的共享区只有storage/0/emulated/deployment目录，其实还是可以优化的，autoinstall这个目录可以不参与打包的。打包花费的时间太久了。

打包结束后才把创建好的二维码显示出来。

## 优化空间与疑问：

二维码里面包含什么信息

2019-04-26 16:53:05.729 6766-6766/? W/HQCONFIG: generateBarCode mJsonObject={"version":"212.02.00.0021","port":"8888","wifi":{"ssid":"Honeywell\_18102B62BB","passwd":"honeywell"},"ssid":"","password":"","function":1,"ip":"192.168.43.1"}

Server端的共享空间是什么时候创建的？共享空间是什么位置？

Defaults.java的HOME\_DIR

HQConfigActivity的handleStart()>>>ServicesStartStopUtil.startServers>>>

context.startService(createSshServiceIntent(context, prefsBean));

context.startService(createFtpServiceIntent(context, prefsBean));

>>>

SshServerService

FtpServerService

以FtpServerService继续往下，由于FtpServerService继承于AbstractServerService

实际是

AbstractServerService的onCreate创建serviceHandler

>>>

AbstractServerService的onStartCommand中msg.arg1 = MSG\_START;

serviceHandler.sendMessage(msg);

>>>

ServerServiceHandler的handleMessage中走到handleStart(service);

>>>

service.launchServer(shell);也就是FtpServerService的launchServer

>>>

case PLAIN:

return new FsFtpFileSystemView(prefsBean.getStartDir(), user);

为什么是PLAIN 这个case？

其实是LoadPrefsUtil的

**public static** StorageType **storageType**(SharedPreferences prefs) {  
 String storageTypeStr = prefs.**getString**(  
 ***PREF\_KEY\_STORAGE\_TYPE***,  
 StorageType.***PLAIN***.**xmlValue**());  
 **return** StorageType.**byXmlVal**(storageTypeStr);  
}

这个赋值的位置在哪呢？

在PrimitiveFtpdActivity的onCreate中

PrefsBean prefsBean = LoadPrefsUtil.loadPrefs(logger, prefs);

>>>

getStartDir()的值就是Defaults.HOME\_DIR

why?

同上，是LoadPrefsUtil的

**public static** File **startDir**(SharedPreferences prefs) {  
 String prefStr = prefs.**getString**(  
 LoadPrefsUtil.***PREF\_KEY\_START\_DIR***,  
 **null**);  
 **return** prefStr != **null** ? **new File**(prefStr) : Defaults.***HOME\_DIR***;  
}

>>>

ftpServer.start();

这时候Server就创建好了，默认Server目录是HOME\_DIR也就是/storage/0/emulated/deployment

这也解释了FTPUtils中的FTPClient创建后，默认的下载路径就是HOME\_DIR

目前在MainActivity里面，按了扫描，开始配置的时候，back键就失效了，具体看下逻辑。

# 优化空间待办：

1. ConfigManager的checkPath放到MainActivity里面去做，或者MainActivity的创建目录放到ConfigManager里面去做,需要整合一下。而且一旦创建了目录，就要去做一次全盘扫。

新增目录的list可以是：

Autoinstall

Firmware

Apkdata

这样假设有2台机器，打开HQConfig，就会创建上面三个目录，

创建二维码，则会多生成一个configs.zip

里面包含了

* SystemConfig:提供的/data/system/users/0/settings\_system&settings\_global&settings\_secure.xml
* XMLConfig:提供的

/data/user/0/com.honeywell.hqconfig/shared\_prefs

/storage/0/emulated/honeywell/ezconfig/generated

Ok！

2.autoinstall的目录不要打包再传输了，而是直接传输。这样可以节省时间，节约了压缩和解压缩的时间。新增的Apkdata目录既可以打包传，也可以直接传。

就直接传吧。

压缩的原则应该是：

非storage/0/emulated/deployment/的文件都采取压缩的形式。

而storage/0/emulated/deployment/目录下的文件则采用传输的形式。

Ok！

但是这样改的话，涉及到本地导入的逻辑就需要修改，因为autoinstall的目录没了。同步修改下。

Ok！

3. storage/0/emulated/deployment/tmp/autoinstall向 storage/0/emulated/deployment/autoinstall目录的拷贝也要取消.然后发送给autoinstall的目录也确认为storage/0/emulated/deployment/autoinstall，当前是storage/0/emulated/deployment/tmp/autoinstall

Ok！

4.需要确认一个问题，解压缩的过程会进行什么形式的覆盖。

假设解压缩前当前目录有A，压缩文件里面也有A

比如压缩包里的A目录里有a1,a2,a3(new)

当前目录的A目录里面有同名的a1，a2，a3，(old)和a4

解压缩后当前A目录的文件会是什么样的？

讲道理要是a1,a2,a3(new),a4

需要验证下a4还再不再。

Ok，a4还在，这个和我想的是一样的

5.解压缩的tmp目录是否删除？

其实讲道理应该删除掉，什么叫配置同步，那应该所有的文件目录都是一致的。

如果不删除的话，B模仿A配置，配置完后，却又多一个tmp目录。

这不一致。但是tmp目录下的文件又是有用的，而且这些文件配置结束并不会回调。所以很难把握删除的时机，只能大概给个延时。

噢 ，现在看了下，其实是可以删除的，目前

是

* /data/system/users/0/settings\_system&settings\_global&settings\_secure.xml
* /data/user/0/com.honeywell.hqconfig/shared\_prefs
* /storage/0/emulated/honeywell/ezconfig/generated

后两种在tmp目录都拷贝到对应目录去了。

就只有上面的settings\_system&settings\_global&settings\_secure.xml没有拷贝，而是直接生效了，可以加入一个拷贝的操作。

为什么之前没有加入拷贝的操作呢？

是否是考虑到了如果直接拷贝覆盖这三个文件会导致原有的一些数据库文件丢失？

其实是有可能的，假设原来是100条，tmp目录下是50条。如果拷贝的话，就只有50条了，而直接更新的话，就只会重置重复的50条。其实这么做还是比较稳妥的。

不过这三个文件的配置并不是异步，所以最后删除应该是没啥问题的。

Ok！暂时先不删

6.多层目录的拷贝与嵌套

这个要验证下并实现

Ok

7.autoinstall现在触发是用的file.listFiles

这样会忽略掉子目录的迭代，需要修改成allFiles，加入迭代。

Ok！

8.HQConfig的逻辑要修改下，最好先生成configs.zip再去开网，要不然网络的相关修改会同步传输到Configs.zip里面去

Ok

9.5G band下 现在是打开hotspot和连接hotspot都做不到的，所以需要新增逻辑。

Ok

10.文件通过FTP传输如果文件名是中文，会是乱码，需要修改

Ok

11.现在check version只是做了os的版本，而没有HQConfig的versioncode，

比如现在我基于andy的HQConfig1.6版本做的HQConfig1.7差异巨大。

最好加上HQConfig的versioncode校验。