“HL-HHWUHFController”

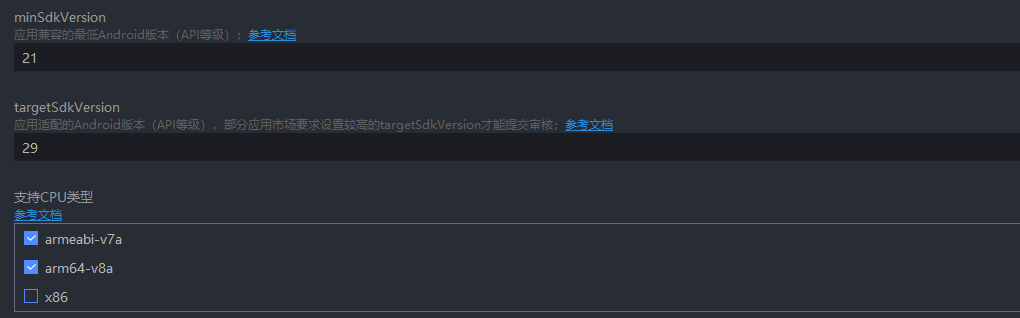
Uni-app插件集成手册

# 引入

HL-HHWUHFController.zip，是一款uni-app原生插件，适用Android平台（只适用特定手持终端），集成该插件后，可使用超高频模组的系列功能。该插件只可在特定手持终端中运行，不适用于模拟器或其它终端设备。

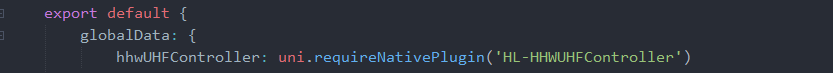
按照uni-app原生插件的集成步骤，将本插件集成到项目，可参见<https://nativesupport.dcloud.net.cn/NativePlugin/use/use>

注意：APP配置中支持的CPU类型勾选arm64-v8a，targetSdkVersino不小于26，如下图



将插件引入放在App.vue中，并将插件对象作为全局变量，然后在使用时，使用该全局变量来调用相关API，如下（详情参照示例工程HHWUHFDemo）：

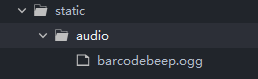
App.vue中引入：



使用：



关于提示音，可更换提示音（格式需为ogg格式），将新提示音文件改为barcodebeep.ogg，放在 static/audio/目录下。

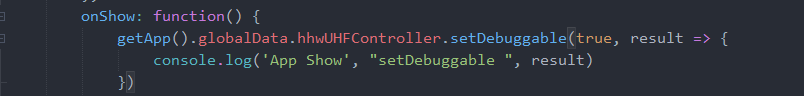


# 接口说明

各接口之间需串行执行，即一个接口返回结果后，再进行另一个接口的调用，以防出现未知错误

文档中所有截图示例，都可以在示例工程中找到

## setDebuggable()

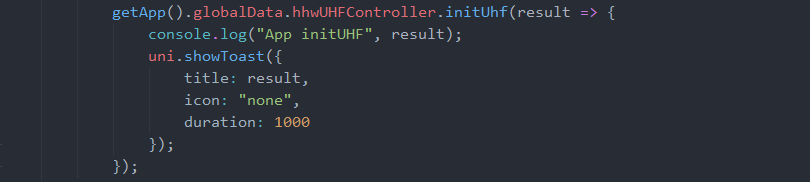


设置插件内部的日志开关

第一个参数为false时，表示关闭java层的日志开关；第一个参数为true时，表示打开java层的日志开关

第二个参数，result，代表了该接口执行的结果，为true时，表示执行成功，为false时，表示执行失败

## initUhf()

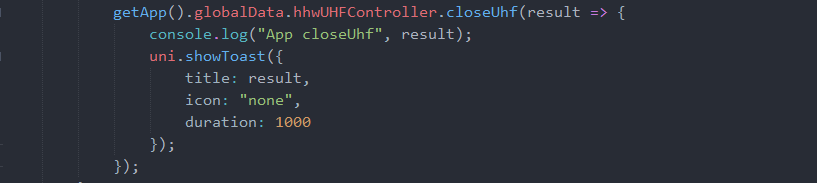


初始化超高频模组

result，为true时，表示初始化成功，为false时，表示初始化失败

注意：该接口为耗时接口，需要大概1600ms左右的时间来完成模组的初始化，从而返回结果。在调用其它接口之前，该接口需要且仅需成功调用一次，多次调用会返回false

## closeUhf()



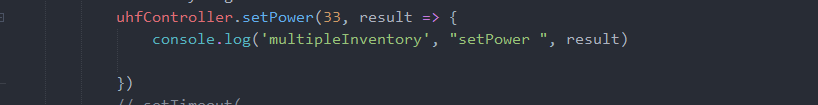
关闭超高频模组

result，为true时，表示关闭模组成功，为false时，表示关闭模组失败。

注意：调用该接口的前提为，已经成功调用[初始化超高频模组](#_初始化超高频模组)接口。在调用该接口后，其它接口都会处于不可使用状态，且模组的功率、工作区域、gen2 session等设置项都会丢失。

因超高频模组在初始化后，会增加手持终端的功耗，所以在不使用超高频模组的时候，务必调用该接口来关闭超高频模组，以降低手持终端的功耗。

## setPower()

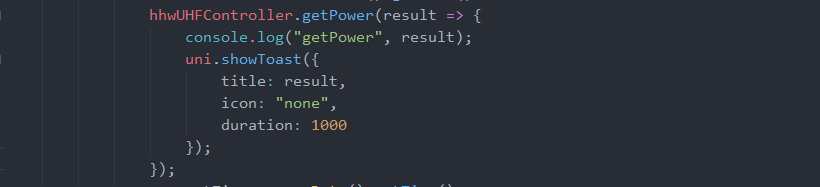


设置模组工作功率

第一个参数为需要设置的模组功率的目标值，范围5-33，默认值为30。模组的工作功率越大，读写范围越大，因超高频受影响的因素繁多，功率和范围并没有固定的对应关系，需根据实际业务需求来定功率值。

第二个参数，result为true时，表示设置成功；result为false时，表示设置失败

## getPower()



获取模组当前的工作功率，result，范围为5-33。

## setGen2session()

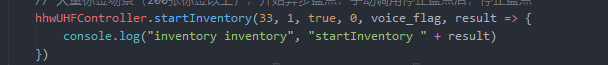


设置异步盘点时的session值

第一个参数为需要设置的目标session值，取值范围：0，1，2，3

第二个参数为设置的结果，true为成功，false为失败

## startInventory()



说明：开始盘点

参数：

1.功率，盘点时采用的功率值，取值范围见[setPower()](#_setPower())

2.session值，取值范围见[setGen2session()](#_setGen2session())

3.异步盘点标志，表示是否开启异步盘点，当标签的数量较大（至少在200张标签）时，可开启异步盘点。true->开启异步盘点，false->不开启异步盘点

4.盘点持续时间，当时间大于0时，会在持续时间后，自动停止盘点；当时间小于等于0时，表示不自动停止盘点，而由[stopInventory()](#_stopInventory())来停止盘点

5.是否播放提示音，true->播放提示音，false->不播放提示音

## stopInventory()

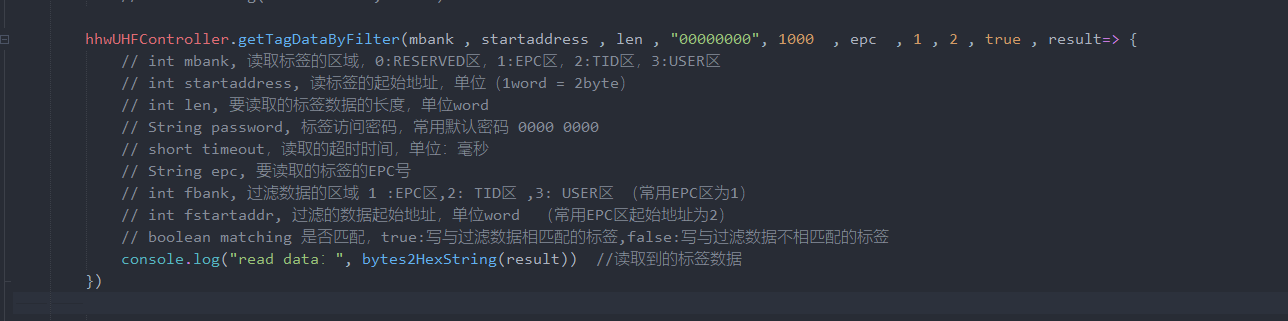


说明：停止盘点，未开启定时盘点时，必须由该方法来停止盘点

参数：

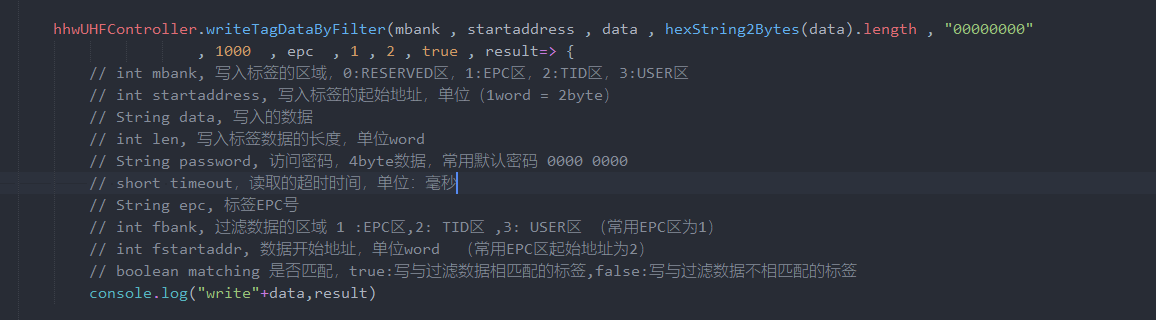
1. 异步盘点标志，该标志必须和调用[startInventory()](#_startInventory())时传入的第一个参数一致

## getTagDataByFilter()



读标签，需要标签epc做过滤条件。（访问密码，延时和过滤条件等默认参数如图所示）

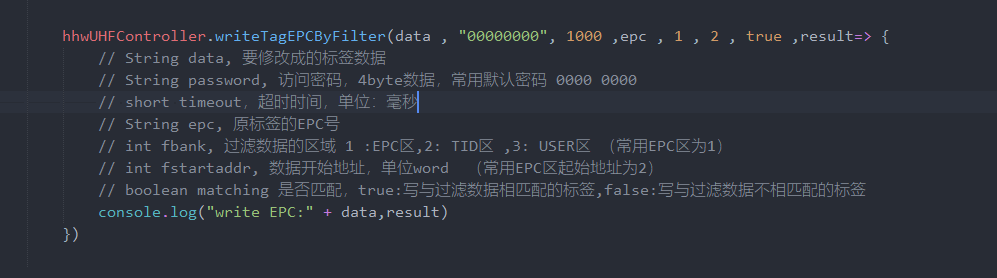
**writeTagDataByFilter()**



写标签，同读标签需要epc做过滤。

result：true 写入成功 ; false 写入失败

**writeTagEPCByFilter（）**

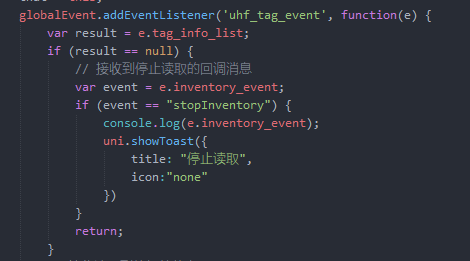


改标签EPC专用，需要先盘存到标签epc后使用。

result：true 写入成功 ; false 写入失败

以上三个接口大部分参数都类似，图上已填的是常用默认值，其他参数可自行根据实际需求传值。

## 监听回调事件



tag\_info\_list，盘点所获取到的标签数据集合，实例说明，详见[附录](#_盘点返回的标签数据数组示例)

inventory\_event，停止盘点时的回调消息，与tag\_info\_list不会同时出现在一个消息事件中，因此可参照示例进行处理

注意：globalEvent也在App.vue中进行引入，然后作为全局变量，其它页面使用时，通过该全局变量来进行事件注册，详情参见示例工程代码

# 代码示例

再次说明，请在上一个接口执行完毕后，再执行下一个接口

示例中的代码都可以在示例工程中找到

## 少量标签盘点

获取少量标签数据（200张以下），适合查找标签对应的库存物品信息等类似场景

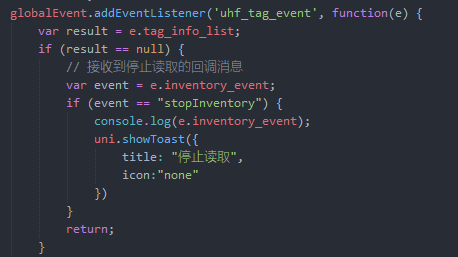
1.获取已初始化的对象，详情参考[initUhf()](#_initUhf())已经示例工程



2.传入功率和session值（session按照图示设置即可，功率可根据实际需求传入），开始同步盘点标签信息 （下图未开启定时盘点）



3.在事件监听中获取标签数据，tag\_info\_list为标签数据的集合，详情见[监听回调事件](#_监听回调事件)



4.如果未开启定时读取，请务必手动停止盘点（注意stopInventory中的参数值）



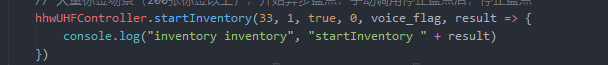
## 大量标签盘点，库存盘点

适合盘库场景，现场的标签比较密集的情况（如服装仓库，服装堆叠存放）的情况

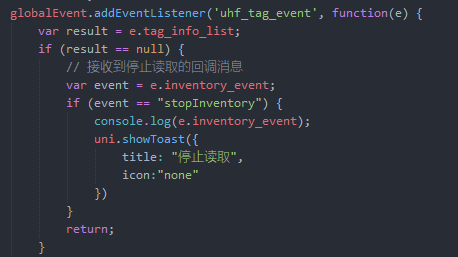
1.获取已初始化的对象，详情参考[initUhf()](#_initUhf())以及示例工程



2.传入功率和session值（session按照图示设置即可，功率可根据实际需求传入），开始同步盘点标签信息 （下图未开启定时盘点）



3.在事件监听中获取标签数据，tag\_info\_list为标签数据的集合，详情见[监听回调事件](#_监听回调事件)

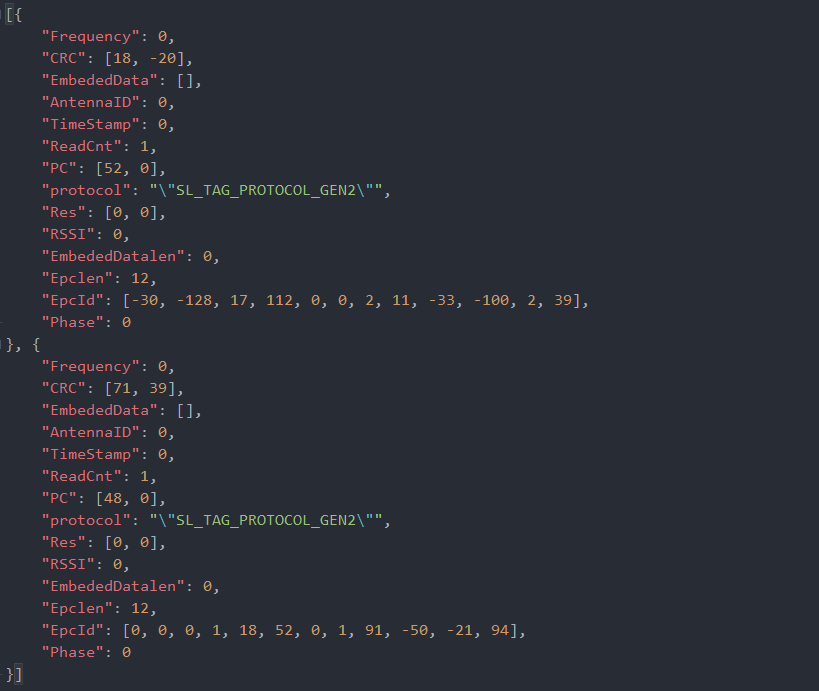


4. 如果未开启定时读取，请务必手动停止盘点（注意stopInventory中的参数值）



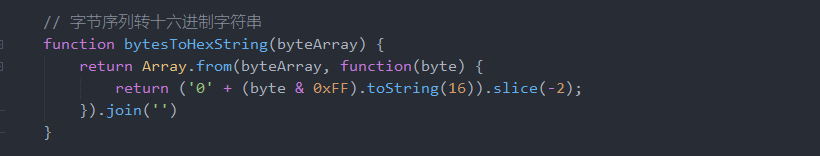
# 附录

## 盘点返回的标签数据数组示例



主要取值：

EpcId，标签的EPC值，需转换为字符串形式，如下为转换成十六进制字符形式：





var epc为标签的EPC的十六进制字符表现形式