

Rockchip RKDevInfoWriteTool使用指南

文件标识：RK-YH-YF-275

发布版本：V1.0.6

日期：2020-08-21

文件密级：☐绝密 ☐秘密 ☐内部资料 ☒公开

免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自所有者所有。

版权所有 © 2020 瑞芯微电子股份有限公司

超越合理使用范畴，非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址：福建省福州市铜盘路软件园A区18号

网址：www.rock-chips.com

客户服务电话：+86-4007-700-590

客户服务传真：+86-591-83951833

客户服务邮箱：fae@rock-chips.com

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.4	lsh	2020-06-08	

版本号	作者	修改日期	修改说明
V1.0.5	lsh	2020-08-21	
V1.0.6	lsh	2021-04-23	
V1.2.0	lsh	2024-01-22	增加写Attestation Key、Widevine Key、Hardware identifiers的功能 获取RKP的功能

目录

Rockchip RKDevInfoWriteTool使用指南

- 一、简介
- 二、工具配置
- 三、SN/MAC/IMEI... 设置
 - 2.1 写号配置
 - 通用设置
 - 写入项配置
 - 手动模式
 - 自增模式
 - 文件模式
 - 2.2 注意事项
 - 2.2.1 SN配置
 - 2.2.2 多MAC配置
- 四、安全Kyes
 - Hardware identifiers
 - widevine keybox
 - attestation key
 - RKP
 - OEM HUK
 - TA
- 五、HDCPKey
 - 通用配置
- 六、插件
- 七、二次开发需求
- 八、问题解析
 - 1 提示 获取xxx失败
 - 2 提示写入失败
 - 3 MAC写入后不能生效问题
 - 4 rtl wifi模组mac地址不一致问题
 - 5 RK预分配的ID

一、简介

RKDevInfoWriteTool用于向VendorStorage（勾选兼容模式的话，写在IDB分区）分区写入用户定义数据，如机器的SN、Wi-Fi、IMEI、用户自定义等数据，这些数据在设备恢复出厂设置后，不会丢失。

两种设备模式：

- 1、MASKROM模式：短接FLASH CLK引脚，这个时候，SOC无法从可用的FLASH加载固件，会切换到USB下载状态（MASKROM模式）。这个时候，是需要用户选择正确的MiniLoaderAll.bin代码，烧写进去，MiniLoaderAll.bin初始化USB之后，才能写号。
- 2、LOADER模式：按住“音量+” 开机或者通过adb reboot loader命令，uboot在检测相应条件之后，切换到Loader模式。在该模式下面写号，设备必须是已经有烧写过固件的（至少MiniLoaderAll.bin、parameter、uboot.img），目前写号工具大部分的操作都是在LOADER模式下操作。

二、工具配置

工具功能分为三大块：SN/MAC/IMEI...，安全Keys，HDCPKey，对应的配置可以通过设置菜单进行单独设置



主界面



1 单次读写：勾选“单次读写”，每次接入设备之后，需要再次按“读取”或者“写入”才能读号或者写号；

2 连续读写：不勾选“单次读写”，接入新设备自动触发读写操作。写号时候，在手动输入模式下面，全部项都输完之后，敲回车键，也会触发写入操作，**这个操作，方便用户使用扫描枪写号**，注意要设置扫描枪末尾自动回车功能；

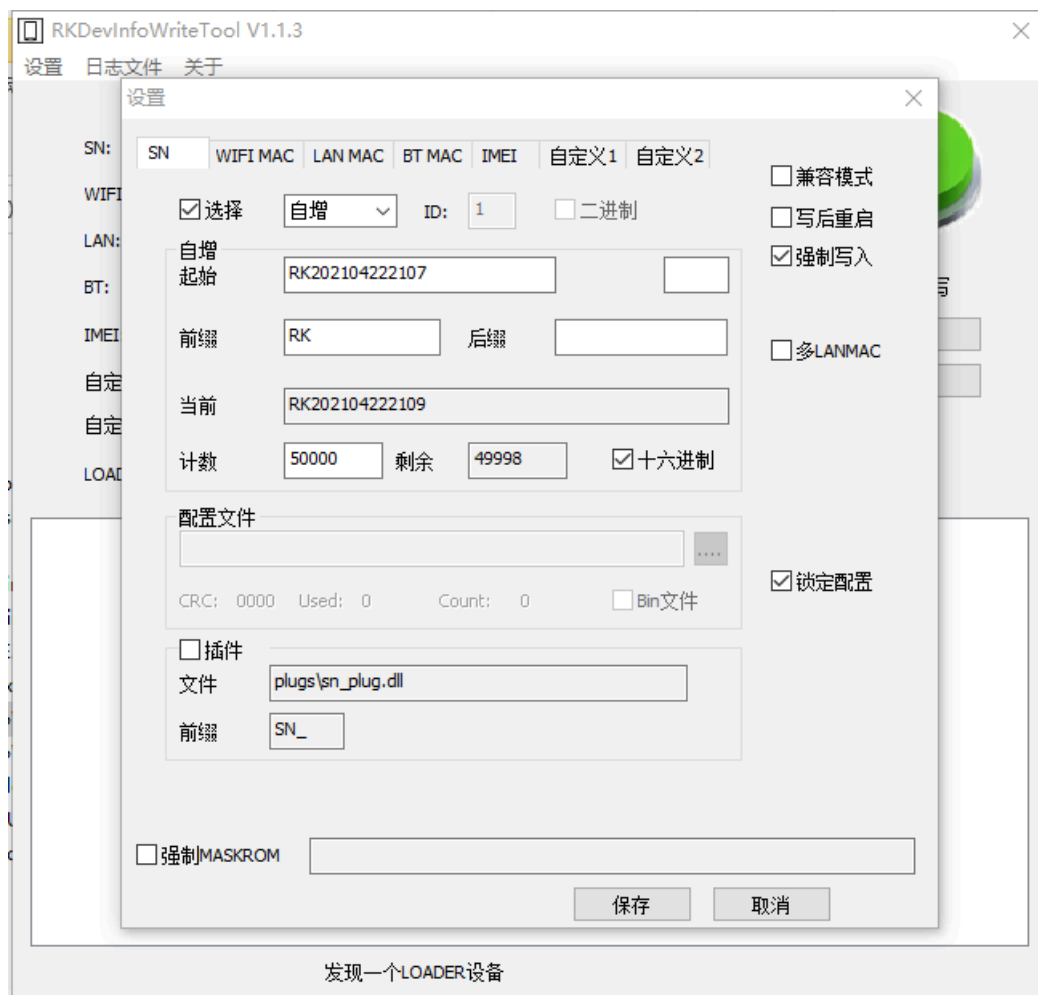
3 LOADER：选择MiniLoader文件，设备在“MASKROM模式”的话，必须选择正确的MiniLoader文件。

三、SN/MAC/IMEI... 设置

这部分主要是设置SN MAC IMEI 以及一些自定义项

2.1 写号配置

单击主界面菜单栏“设置”进入写号工具配置界面：



通用设置

- 1、兼容模式：选择“兼容模式”下面写号，在这种模式下面，可以选择“强制MASKROM”强制机器到maskrom下面写号，或者不选择“强制MASKROM”，那么机器需要先切换到loader模式写号（适用于在空片情况下使用）；
- 2、写后重启：写完后重启机器；
- 3、强制写入：不勾选的话，写入前会检查有没有写过号，如果写过了，会跳过；如果勾选，不管设备有没有写过号，直接写入；

写入项配置

下面两项仅仅针对自定义可配置：

- 1、ID：SN、Wi-Fi、LAN MAC、BT MAC等数据项存储VendorStorage分区，每个数据项通过一个唯一的ID 进行检索。只有自定义项支持自定义ID，且不可与前面定义的重复；
- 2、二进制：选定输入的项是否是二进制格式。如“11223344”，如果是二进制格式，则实际写入到设备的是“0x11 0x22 0x33 0x44”四个字节；没有勾选二进制格式的话，则按文本方式写入，实际写入的是“11223344”字符串；

对于写入项的获取，可以在设置里面，通过下拉框选择三种模式：“手动”、“自增”、“文件”

手动模式

直接在主界面从键盘或者扫描枪输入要写入的数据项

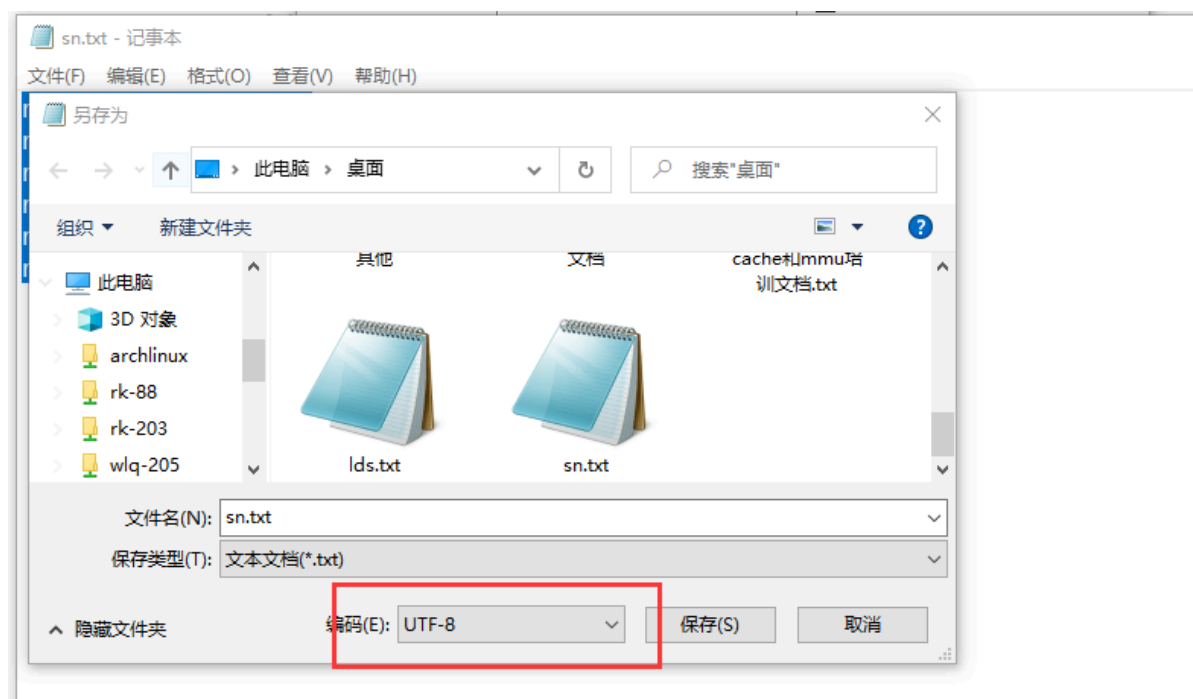
自增模式

在这个模式下，用户可以设置一个包含数字的字符串作为起始字符串，如“rockchip201810150029sn”，每次成功写完一个号之后，软件都会自动给数字部分自增1（+1）。默认是按十进制自增的，如果勾选“十六进制”，那么是按十六进制自增。“前缀”，“后缀”是起始字符串的前面部分，和后面部分，用户可以通过设置“前缀”，“后缀”，让这部分内容不会自增。

文件模式

在这个模式下，写入项可以通过一个文本获取。

文本格式：如果没有勾选“bin文件”选项的话，那么文件内容必须是文本文件，软件每次从文件里面获取一行，然后写到设备。注意文本必须以UTF-8编码，比如在Windows下面的记事本里面，必须按下图选择格式保存：



SN号文件文本实例：

```
RK202104222107
RK202104222108
RK202104222109
RK202104222110
RK202104222111
RK202104222112
```

MAC号文件文本实例(1个mac地址的情况)：

```
0a0b11223344
0a0b11223345
0a0b11223346
0a0b11223347
0a0b11223348
0a0b11223349
0a0b11223350
```

BIN文件：勾选“Bin文件”选项的话，那么软件每次写入会把整个文件内容写到设备里面。

2.2 注意事项

2.2.1 SN配置

SN默认都是要求符合GMS的规范，**要求长度不大于14且字母开头的数字字母组合**，如果没有按要求写入，重启设备之后，看到的SN不是自己写入的SN，而是系统随机生成的。对于这个问题的解决办法

1. 重新写入符合GMS的规范的SN
2. 对于不需要使用GMS服务的场景，可以从SDK里面去掉SN校验的功能：

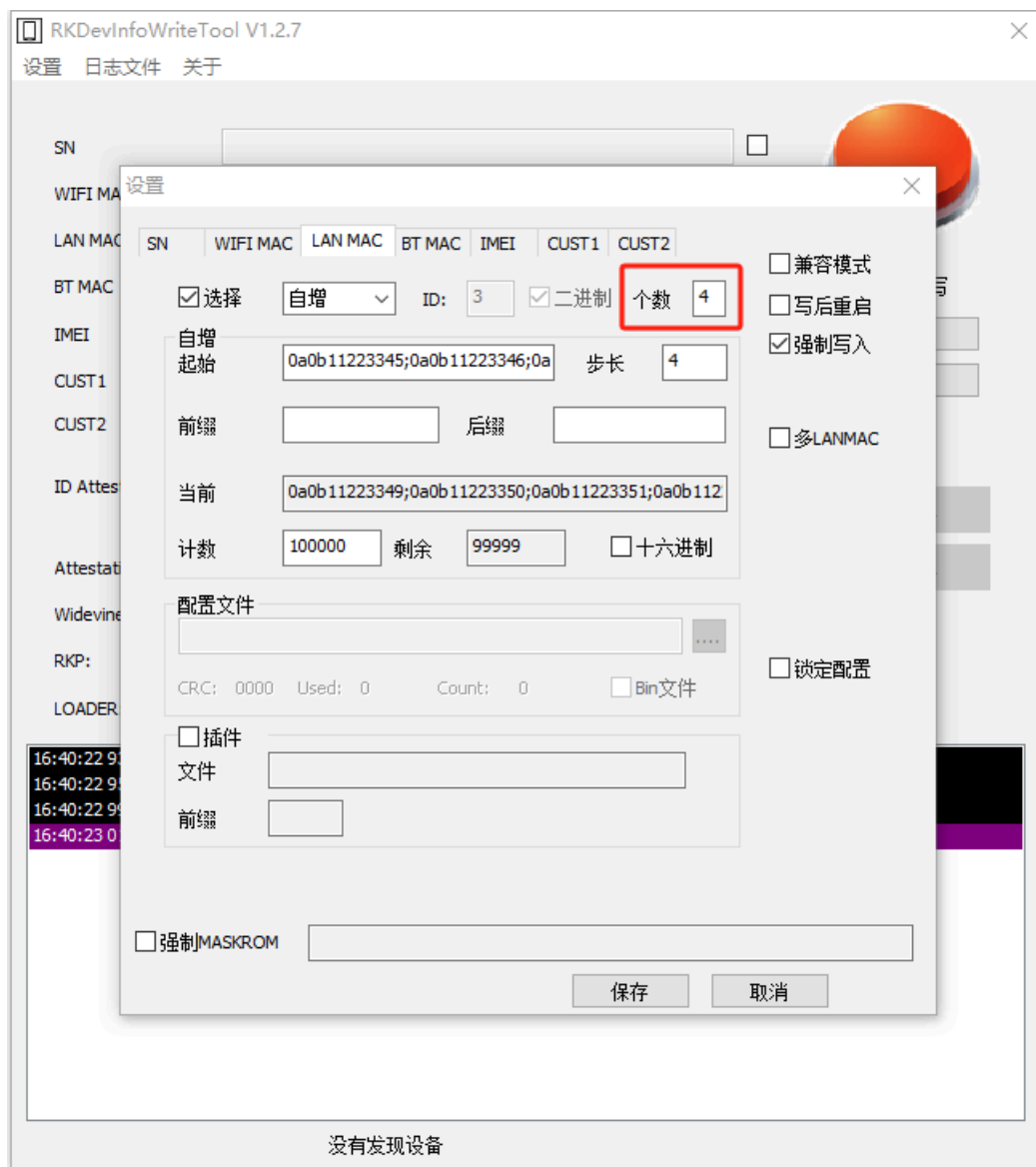
```
lsh@rk-intel-1:~/rk-sdk/android10/hardware/rockchip/drmsservice$ git diff
diff --git a/Android.mk b/Android.mk
index 2606cae..6f18054 100644
--- a/Android.mk
+++ b/Android.mk
@@ -22,7 +22,7 @@ LOCAL_CPPFLAGS := \

# API 28 -> Android 9.0
ifeq (1,$(strip $(shell expr $(PLATFORM_SDK_VERSION) \>= 29)))
-LOCAL_CFLAGS += -DENABLE_CMDLINE_VERIFY
+#LOCAL_CFLAGS += -DENABLE_CMDLINE_VERIFY
endif

include $(BUILD_EXECUTABLE)
```

2.2.2 多MAC配置

有些设备有多个MAC的需求，因此工具对于MAC地址的写入，会有一个 **个数** 的选项，用来选择需要写入MAC的数量：

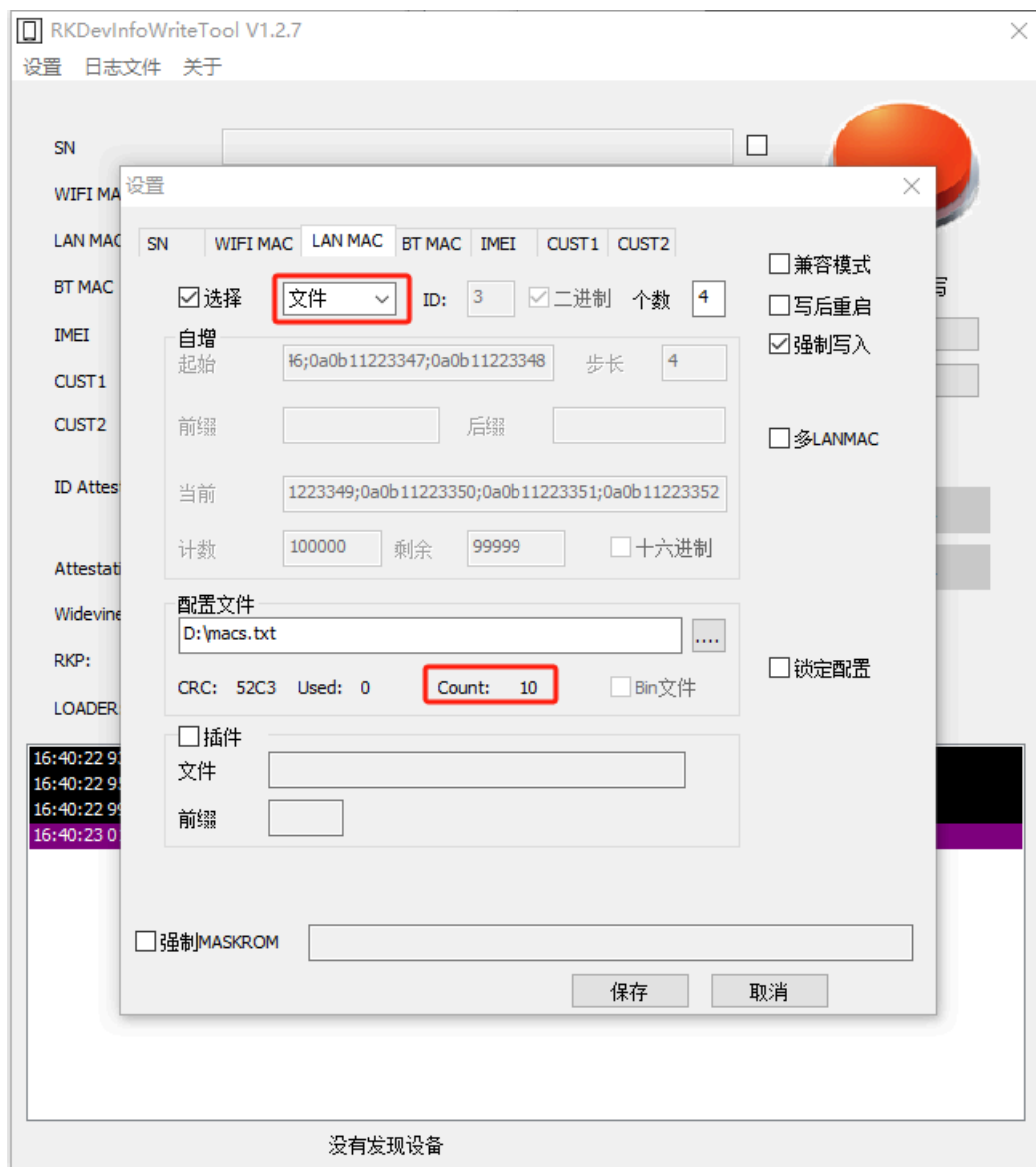


如上图，设置写入4个MAC地址，每个MAC地址都是以“;”号间隔，写完之后，每个mac地址自增4(通过步长设定)

```
0a0b11223345;0a0b11223346;0a0b11223347;0a0b11223348
```

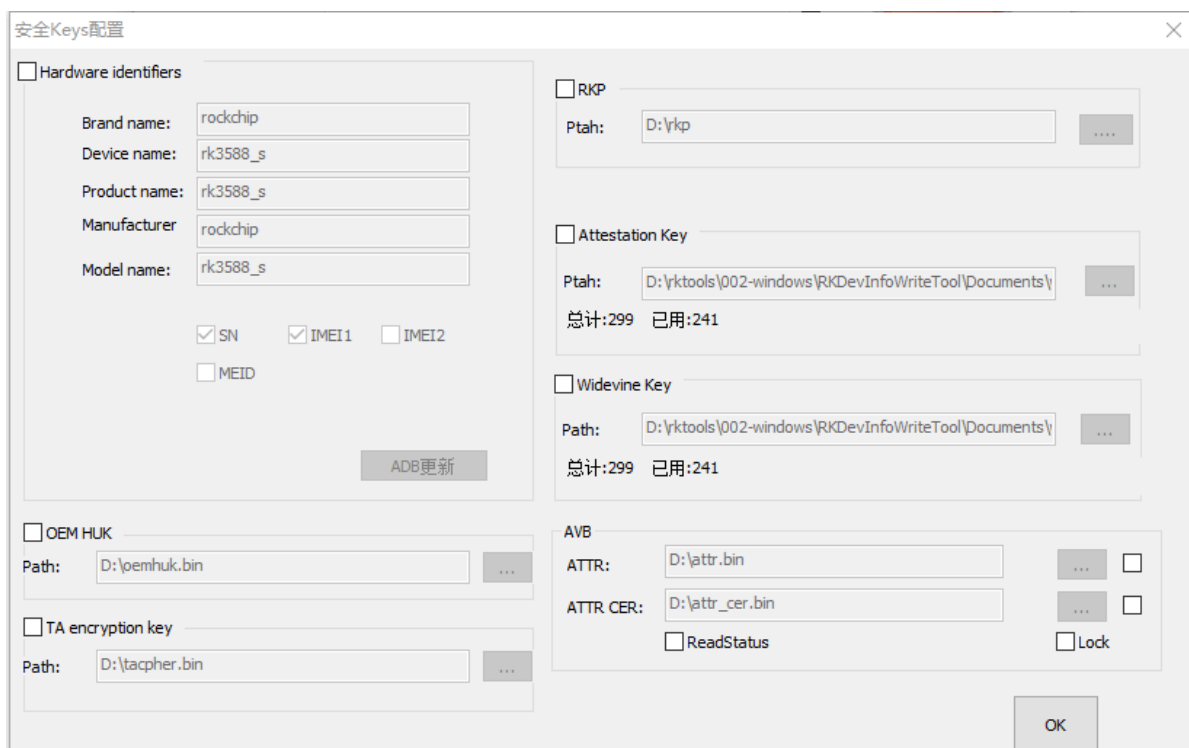
也可以通过文本方式添加，如下可以写10台机器的mac地址文本（10行）

```
$ cat macs.txt
0a0b11223349;0a0b11223350;0a0b11223351;0a0b11223352
0a0b11223359;0a0b11223360;0a0b11223361;0a0b11223362
0a0b11223369;0a0b11223370;0a0b11223371;0a0b11223372
0a0b11223379;0a0b11223380;0a0b11223381;0a0b11223382
0a0b11223389;0a0b11223390;0a0b11223391;0a0b11223392
0a0b11223399;0a0b11223300;0a0b11223301;0a0b11223302
0a0b11223309;0a0b11223310;0a0b11223311;0a0b11223312
0a0b11223319;0a0b11223320;0a0b11223321;0a0b11223322
0a0b11223329;0a0b11223330;0a0b11223331;0a0b11223332
0a0b11223339;0a0b11223340;0a0b11223341;0a0b11223342
```

四、安全Kyes

这里主要是写入Device Identifiers、widevine keybox、attestation key、RKP



Hardware identifiers

Hardware identifiers一般是在SN、IMEI（可选）一起写到设备

ID 认证支持以下硬件标识符：

1. 品牌名称，由 Android 中的 `Build.BRAND` 返回
2. 设备名称，由 Android 中的 `Build.DEVICE` 返回
3. 产品名称，由 Android 中的 `Build.PRODUCT` 返回
4. 制造商名称，由 Android 中的 `Build.MANUFACTURER` 返回
5. 型号名称，由 Android 中的 `Build.MODEL` 返回
6. 序列号
7. 所有无线装置的 IMEI
8. 所有无线装置的 MEID
9. 型号名称，由 Android 中的 `Build.MODEL` 返回

这些都是需要从Android系统里面获取，为了避免误操作，建议先让一台机器，进入Android系统，然后连接电脑，单击按键“**ADB更新**”从设备获取

对于需要写 序列号、IMEI、MEID等ID的设备，需要勾选对应的选择框

widevine keybox

...：加载文件

attestation key

...：加载文件

RKP

...：选择保存路径即可

OEM HUK

...：加载文件

TA

...：加载文件

五、HDCPKey

HDCP设置

通用配置

☐ 写后重启 ☒ 强制写入 ☒ 单次写入 ☐ 锁定配置 ☐ 支持ADB设备

☐ 兼容模式1 ☐ 兼容模式2 ☐ CRC校验 ☐ RK3588

☐ Loader: ...

☒ hdcp 1.4 Hdmirx

HDCP 1.4 HDMIRX: ... ☒ AES 16/50

☒ hdcp 1.4 Hdmits

hdcp 1.4 HdmitsX: ... ☒ AES 16/50

☐ hdcp 2x hdmirx

hdcp 2x hdmirx: ... ☒ AES 0/50

☐ hdcp 1.4 dp

hdcp 1.4 dp: ... ☒ AES 0/50

☐ hdcp 2.X wfd

hdcp 2.X wfd: ... ☒ AES 0/50

保存

通用配置

- 写后重启：烧写完之后，是否重启设备；
- 强制写入：写入之前，判断设备是否已经又写过，如果有写，则跳过。目前hdcp因为取消了读取的功能，因此，这个是必须要勾选的；
- 单次写入：勾选的话，每次写号都要手动按主界面的 **写入** 按键才开始烧写，不勾选的话，每次插入设备，会自动烧写；
- 锁定配置：主界面不能使能烧写项
- 支持ADB设备：是否支持在adb模式下烧写，注意这个功能实际上是写号之前，先调用了adb reboot loader，让设备从adb切换到loader模式
- 兼容模式1 兼容模式2：兼容早期平台，如果烧写时候失败，可以尝试勾选这两个其中一个尝试
- RK3588：因为控制器不一样，key的加密方式不一样，因此这里需要区分是否RK3588
- AES：对于RK3588的HDCP key烧写，用于使能在传输中使用AES加密功能

六、插件

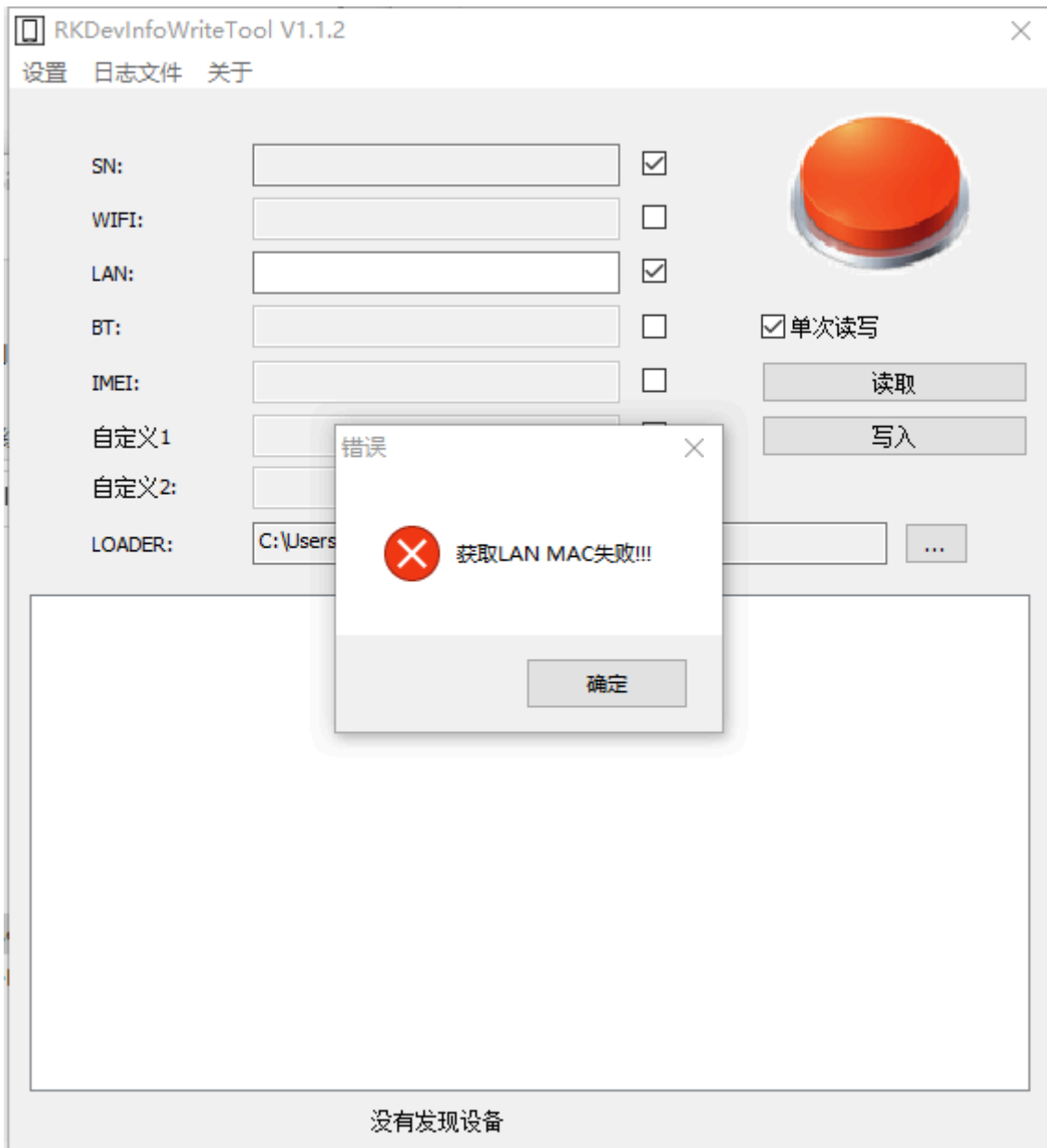
该项用于写入项的一些预检查，如果带GMS包固件，对SN有14位长度限制，必须以字母开头等。可以通过勾选插件来确保。修改该项需要重启软件。

七、二次开发需求

如果以上配置不能满足需求，可以申请源码进行二次开发。申请方式，写邮件给业务，抄送lsh@rock-chips.com。

八、问题解析

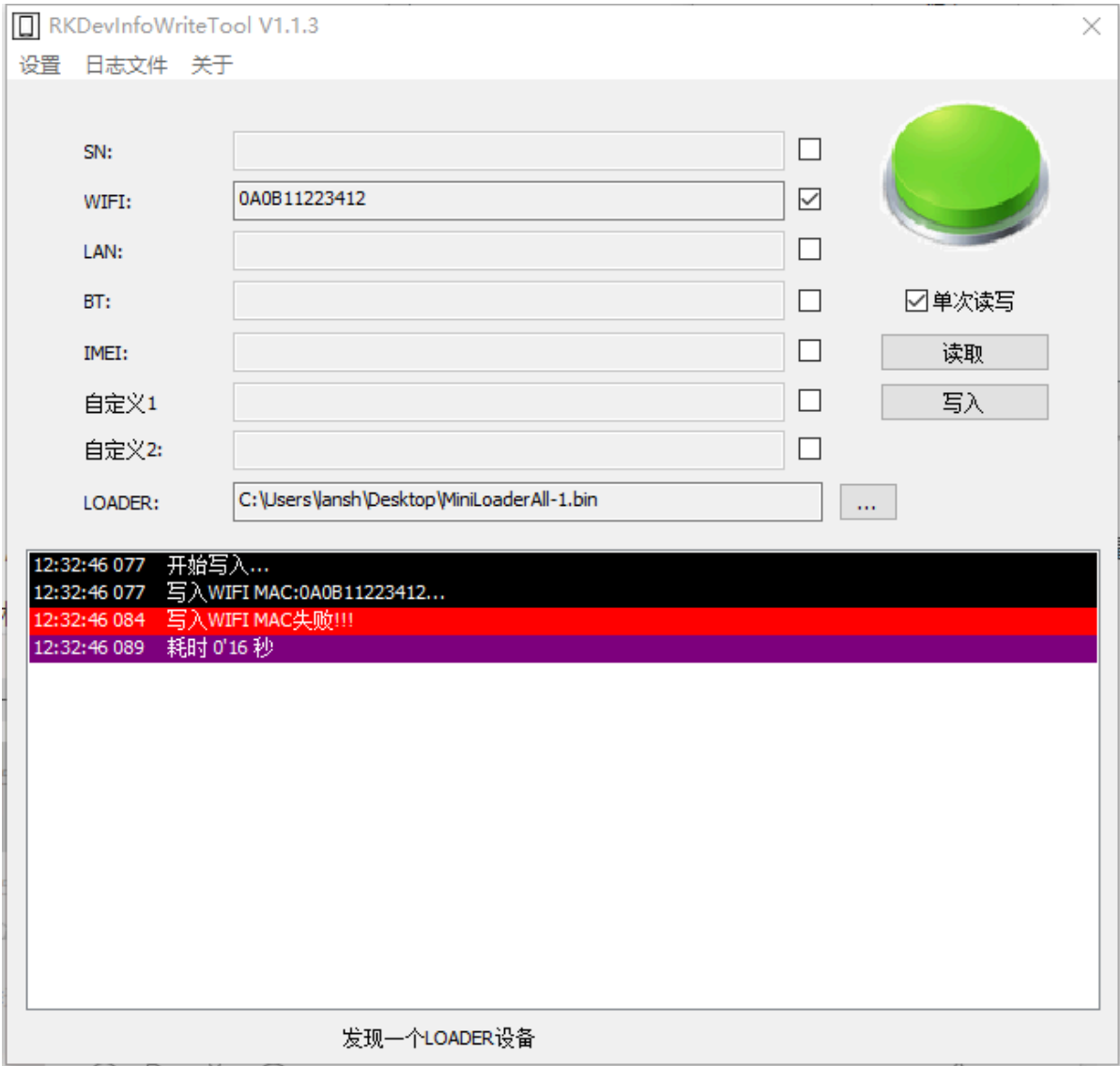
1 提示 获取XXX失败



这个问题，是根据用户配置，获取写入数据时候获取数据失败，不是在写号的阶段失败的。检查下配置就好了。如

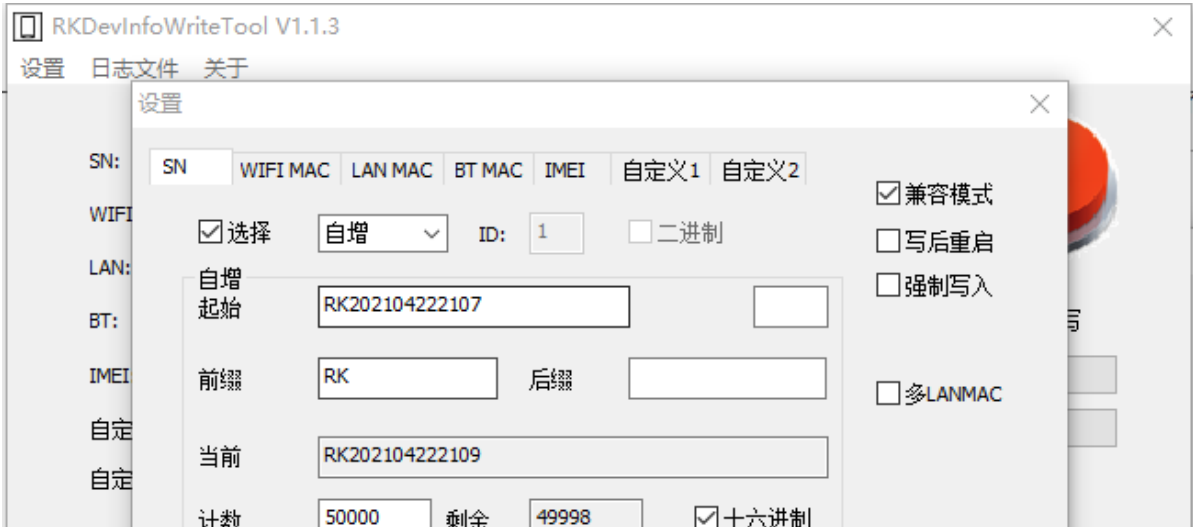
- 手动 模式，检查下是不是输入框没有输入
- 自增 模式，检查是不是配号用光了，或者起始字段没写
- 文件 模式，检查文件是不是被删了，或者是不是配号用光了

2 提示写入失败



因为UBOOT版本升级的改动，写号的命令有做过升级，一般默认工具会自动选择对应命令，如果写号一直提示失败的问题，有可能是选择失败，这个时候可以通过一下方法尝试：

1. 确认下是不是勾选了 **兼容模式**，如果有勾选，去掉试试。这个模式在Android7.1之后，基本不用了。



2. 修改配置文件，强制使用某个命令。配置文件是安装目录下面config.ini，在修改前，记得要先关闭软件：

```
[System]
ForceApi=0
ForceApiType=0
```

如上，默认 ForceApi=0，改为1，ForceApiType 的值可以是 0 - 2

ForceApiType 跟uboot对应一般如下：

- 0- uboot分支为next-dev
- 1 - uboot分支为rkdevelop
- 2 - 适用于Android5.1之前版本，写在IDB中

3 MAC写入后不能生效问题

工具往vendorstorage写入有效的MAC地址后，会由UBOOT读取，然后通过dtb传递给内核。如果写完设备的MAC地址显示不对，可以先在rockchip_set_ethaddr函数里面，把MAC地址打印出来，先做第一步判断，对应uboot代码

```
/* ./arch/arm/mach-rockchip/board.c */
static int rockchip_set_ethaddr(void)
/* ./common/fdt_support.c */
void fdt_fixup_ethernet(void *fdt)
```

4 rtl wifi模组mac地址不一致问题

改模组会强制清零MAC地址第一字节的bit[1]，对应提交如下：

```
lsh@rk-intel-1:~/rk-sdk/android12-rkdevelop/kernel-4.19$ git show
f37e3b555cf3e497d52bcc2f26596e0b57636539
commit f37e3b555cf3e497d52bcc2f26596e0b57636539
Author: hwg <hwg@rock-chips.com>
Date: Mon Apr 20 16:09:08 2015 +0800

    wifi: support random mac address and save to file

diff --git a/net/rfkill/rfkill-wlan.c b/net/rfkill/rfkill-wlan.c
index ce9e3cf7f0ef9..d96d702d9e97e 100755
--- a/net/rfkill/rfkill-wlan.c
+++ b/net/rfkill/rfkill-wlan.c
....
int rockchip_wifi_mac_addr(unsigned char *buf)
{
    char mac_buf[20] = {0};
@@ -572,11 +610,12 @@ int rockchip_wifi_mac_addr(unsigned char *buf)

wifi_custom_mac_addr[2],wifi_custom_mac_addr[3],wifi_custom_mac_addr[4],wifi_cust
om_mac_addr[5]);
    LOG("falsh wifi_custom_mac_addr=[%s]\n", mac_buf);

-    if (is_valid_ether_addr(wifi_custom_mac_addr)) {
-        if (2 == (wifi_custom_mac_addr[0] & 0x0F)) {
-            LOG("This mac address come into conflict with the address of direct,
ignored...\n");
```

```

-         return -1;
-     }
+ #ifdef ENABLE_WIFI_RAND_MAC
+     rockchip_wifi_rand_mac_addr(buf);
+ #endif
+
+     if (is_valid_ether_addr(wifi_custom_mac_addr) &&
!strcmp(wifi_chip_type_string, "rtl", 3)) {
+         wifi_custom_mac_addr[0] &= ~0x2; // for p2p
    } else {
        LOG("This mac address is not valid, ignored...\n");
        return -1;
    }

```

5 RK预分配的ID

如果用户需要写入自定义项的话，需要避免使用这些预分配好的ID

```

#define VENDOR_SN_ID 1 /* serialno */
#define VENDOR_WIFI_MAC_ID 2 /* wifi mac */
#define VENDOR_LAN_MAC_ID 3 /* lan mac */
#define VENDOR_BLUETOOTH_ID 4 /* bluetooth mac */
#define VENDOR_HDCP14_HDMI_ID 5 /* HDCP 1.4 HDMI */
#define VENDOR_HDCP14_DP_ID 6 /* HDCP 1.4 DP */
#define VENDOR_HDCP2X_WFD_ID 7 /* HDCP 2.x WFD */
#define VENDOR_WIDEVINE_ID 8
#define VENDOR_PLAYREADY_CERT_ID 9
#define VENDOR_ATTENTION_KEY 10
#define VENDOR_PLAYREADY_ROOT_SL3000_ID 11
#define VENDOR_PLAYREADY_ROOT_SL2000_ID 12
#define VENDOR_HDCP14_HDMIRX_ID 13 /* HDCP 1.4 HDMIRX */
#define VENDOR_IMEI_ID 15 /* imei */
#define VENDOR_CUST1_ID 16
#define VENDOR_CUST2_ID 17
#define VENDOR_TACIPHER 120

```