



Unisoc Confidential For hiar

VSIM 功能说明及接口定义介绍

文档版本
发布日期

V1.0
2020-09-25

版权所有 © 紫光展锐（上海）科技有限公司。保留一切权利。

本文件所含数据和信息都属于紫光展锐（上海）科技有限公司（以下简称紫光展锐）所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负任何与本文件相关的直接或间接的、任何伤害或损失。

请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。

Unisoc Confidential For hiar

紫光展锐（上海）科技有限公司



前言

概述

本文档提供紫光展锐平台 VSIM 功能说明及相关接口的定义介绍，更好地帮助开发人员实现 VSIM 卡初始化、注册鉴权，过程中鉴权等功能。

读者对象

本文档主要适用于 VSIM 开发人员，开发人员须具备以下经验和技能：


- 熟悉 VSIM 的基本流程。
- 有一定的 VSIM 开发经历。

缩略语

缩略语	英文全名	中文解释
SIM	Subscriber Identity Module	用户识别模块
VSIM	Virtual Subscriber Identity Module	虚拟用户识别模块
APN	Access Point Name	接入点

符号约定

在本文中可能出现下列标志，它所代表的含义如下。

符号	说明
 说明	用于突出重要/关键信息、补充信息和小窍门等。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害。

变更信息

文档版本	发布日期	修改说明
V1.0	2020-09-25	第一次正式发布。

关键字

VSIM、虚拟卡、云卡、软卡、鉴权。

Unisoc Confidential For hiar

目 录

1 VSIM 功能说明.....	1
2 接口定义.....	3
2.1 libatci 接口定义	3
2.1.1 VSIM_init.....	3
2.1.2 VSIM_exit.....	4
2.1.3 VSIM_set_authid.....	4
2.1.4 VSIM_query_authid	5
2.1.5 VSIM_set_virtual	5
2.1.6 VSIM_query_virtual.....	6
2.1.7 sendATCmd	6
2.1.8 VSIM_get_auth_cause	7
2.1.9 VSIM_set_nv.....	7
2.2 APK 接口定义	8
2.2.1 VSIMInit	8
2.2.2 VSIMSendData.....	9
2.2.3 VSIMExit	9
2.2.4 VSIMSetAuthid	10
2.2.5 VSIMQueryAuthid	10
2.2.6 VSIMSetVirtual	11
2.2.7 VSIMQueryVirtual	11
2.2.8 VSIMGetAuthCause.....	12
2.2.9 setDefaultDataSubId.....	12
2.2.10 setSimPowerStateForSlot	13
2.2.11 attachAPN	13
2.2.12 getSubscriberIdForSlotIdx	14
2.2.13 getSubId	14
2.2.14 getVoiceRegState.....	15
2.2.15 getDataRegState	15
2.2.16 getNetworkOperator	15
2.2.17 getVoiceNetworkType	16
2.2.18 getDataNetworkType.....	16
2.2.19 getSimState.....	17
2.2.20 setDataEnabled	17
2.2.21 setSimNeworkType	18
2.2.22 getSimNeworkType	18
2.2.23 getDefaultDataPhoneId	19
2.2.24 setDefaultDataPhoneId.....	19

2.2.25 restartRadio	19
2.2.26 updatePlmn	20
2.2.27 queryPlmn	21
2.2.28 setImei	21
2.2.29 getImei	22
2.2.30 setPreferredNetworkType	22
2.2.31 powerRadio	23
2.2.32 VSIMSetVirtualWithNV	23
3 注意事项	25
3.1 集成注意事项	25
3.1.1 使用 libatci 集成注意事项	25
3.1.2 使用 APK 集成注意事项	25
3.2 VSIM 场景使用注意事项	25

Unisoc Confidential For hiar

图目录

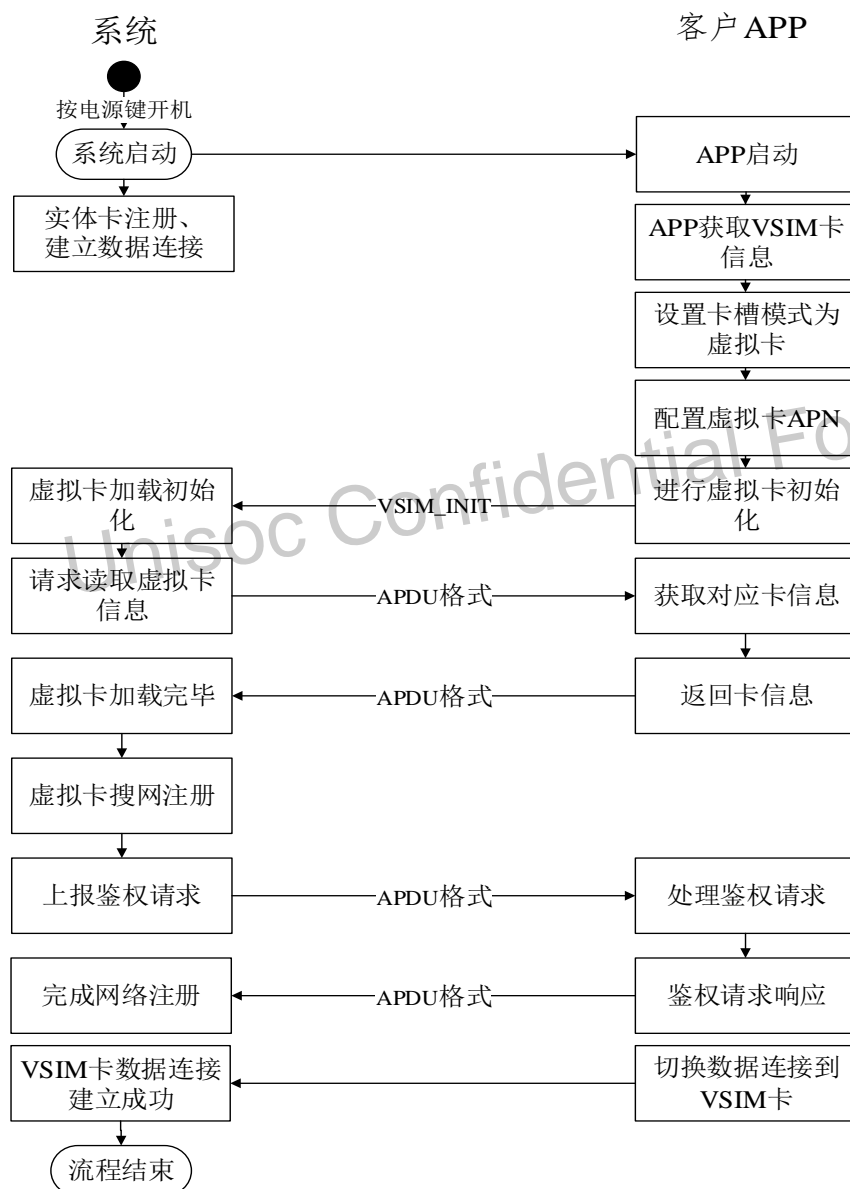
图 1-1 VSIM 注册鉴权流程图	1
--------------------------	---

Unisoc Confidential For hiar

1 VSIM 功能说明

开机 VSIM 注册鉴权流程主要包含开机实体卡注册、实体卡建立数据连接、虚拟卡初始化、虚拟卡处理鉴权请求、虚拟卡完成鉴权、虚拟卡建立数据连接等过程。具体流程如图 1-1。

图1-1 VSIM 注册鉴权流程图



此鉴权流程可以选择 libatci 接口和 APK 接口进行初始化，libatci 接口适用于没有 Android 系统且无法安装 APK 的情况。

使用 libatci 接口初始化

步骤 1 APP 设置卡槽模式

如果需要设置指定卡槽为虚拟卡，请先使用 `VSIM_query_virtual` 查询当前卡槽模式，然后使用 `VSIM_set_virtual` 设置卡槽模式。推荐使用卡 2 作为 VSIM 卡槽，卡 1 作为实体卡槽。

步骤 1 配置 APN 到 Modem

APP 在确定虚拟卡信息后，将当前卡使用的 APN 信息配置到 Modem（需要同步配置到 Android APN 数据库中），Modem 在开协议栈后使用该 APN 做网络注册，能够降低 APN 配置被拒的概率。

配置 APN 需要执行的 AT 命令如下：

```
AT+CGDCONT=1,"IPV4V6","APN","",0,0
AT+CGPCO=0,"user","password",1,authtype
```

其中"IPV4V6"、"APN"、"user"、"password"请根据实际 APN 信息配置。authtype 没有指定时请默认为 3；可以调用接口 7 直接发送 AT，AT 命令中的引号需要保留。

步骤 2 VSIM 初始化

VSIM 卡 APN 配置完成后，使用 `VSIM_init` 接口进行 VSIM 初始化。

步骤 3 鉴权请求响应

APP 在收到鉴权请求后，首先通过 `VSIM_set_authid` 接口设置发送鉴权数据使用的数据连接卡，然后切换数据连接到对应的实体卡上。

---结束

使用 APK 接口初始化

步骤 1 设置卡槽模式

通过接口 `VSIMSetVirtual` 设置卡槽模式。

步骤 2 配置 APN 到 Modem

通过接口 `attachAPN` 配置 APN 到 Modem。此接口只将 APN 发送到 Modem，需要客户 APK 实现将 APN 添加到 APN 数据库中。

步骤 3 VSIM 初始化

VSIM 卡 APN 配置完成后，使用 `VSIMInit` 接口进行 VSIM 初始化。

步骤 4 鉴权请求响应

APP 在收到鉴权请求后，首先通过 `VSIMSetAuthid` 接口设置发送鉴权数据使用的数据连接卡，然后切换数据连接到对应的实体卡上。

---结束

说明

APK 和 libatci 中 VSIM 相关的接口（libatci 和 APK 都定义的接口）不可混用，只能二选一。若选择使用 libatci 接口初始化，APK 中和 VSIM 无关的接口也可以使用。

2 接口定义

2.1 libatci 接口定义

2.1.1 VSIM_init

【函数功能】

对应卡槽的 VSIM 初始化。

【函数原型】

```
int VSIM_init(int phoneId, VSIM_COMMAND pfnCommand, int mode)
```

【参数说明】

参数名称	含义
phoneId	初始化的卡槽 id, $0 \leq \text{phoneId} \leq 1$ 。
pfnCommand	传入调用方处理鉴权请求的函数指针。展锐 VSIM 模块调用该函数将鉴权数据传递给 APP 处理, 需要调用方定义该函数或使用展锐现有通用接口定义, 在 APP 处理完鉴权后将鉴权结果返回给展锐 VSIM 模块。
mode	VSIM 初始化方式。 <ul style="list-style-type: none">0: VSIM 全初始化, 应用在 VSIM 无卡状态下初始化。1: VSIM 卡状态正常情况下, 应用 APP 进程异常退出时, 只初始化 APP 和展锐 VSIM 模块通信 socket, 不重启 VSIM。

【返回值】

- 操作成功: 返回 1。
- 操作失败: 返回-1。

VSIM_COMMAND

【定义】

```
typedef unsigned char u8;  
typedef unsigned short u16;  
typedef int (*VSIM_COMMAND) (u8 slot, u8 *apdu_req, u16 apdu_req_len, u8 *apdu_rsp, u16 apdu_rsp_len);
```

【参数说明】

参数名称	含义
slot	鉴权请求卡槽 id。
apdu_req	鉴权请求数据指针。
apdu_req_len	鉴权请求数据长度。
apdu_rsp	鉴权响应数据指针。
apdu_rsplen	鉴权请求数据指针申请内存长度。

【返回值】

鉴权响应数据实际长度。

2.1.2 VSIM_exit

【函数功能】

关闭对应卡槽的 VSIM。

【函数原型】

```
int VSIM_exit(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

- 操作成功：返回 1。
- 操作失败：返回-1。

2.1.3 VSIM_set_authid

【函数功能】

设置发送 VSIM 鉴权数据使用的数据通道卡。

【函数原型】

```
int VSIM_set_authid(int authid)
```

【参数说明】

参数名称	含义
authid	发送鉴权数据使用数据通道的卡 id, $0 \leq \text{authid} \leq 1$ 。

【返回值】

- 操作成功：返回 0。
- 操作失败：返回-1。

2.1.4 VSIM_query_authid

【函数功能】

查询当前使用哪张卡数据通道发送 VSIM 鉴权数据。

【函数原型】

```
int VSIM_query_authid()
```

【参数说明】

无。

【返回值】

- 操作成功：返回 authid, $0 \leq \text{authid} \leq 1$ 。
- 操作失败：返回-1。

2.1.5 VSIM_set_virtual

【函数功能】

设置对应卡槽的卡模式，默认该模式值会保存 NV，下次开机继续生效。

【函数原型】

```
int VSIM_set_virtual(int phoneId, int mode)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

参数名称	含义
mode	卡的模式，为实体卡还是虚拟卡。 <ul style="list-style-type: none"> 0: 实体卡。 1: 云卡。 2: 软卡。

【返回值】

- 操作成功：返回 0。
- 操作失败：返回-1。

📖 说明

该接口只修改卡槽模式，开关卡操作需要 APP 控制进行。

2.1.6 VSIM_query_virtual

【函数功能】

查询对应卡槽的卡模式。

【函数原型】

```
int VSIM_query_virtual(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

- 操作成功：返回 0, 1, 2。
 - 0: 实体卡
 - 1: 云卡
 - 2: 软卡
- 操作失败：返回-1。

2.1.7 sendATCmd

【函数功能】

发送 AT 命令给 Modem，并接收返回结果，此函数是同步操作。

【函数原型】

```
int sendATCmd(int phoneId, const char *atCmd, char *resp, size_t respLen)
```

【参数说明】

参数名称	含义
phoneId	对应的卡槽 id, $0 \leq \text{phoneId} \leq 1$ 。
atCmd	发送的 AT 命令。
resp	返回结果, 需要调用方分配好内存。
respLen	分配的内存长度。

【返回值】

- 操作成功: 返回 0。
- 操作失败: 返回-1。

2.1.8 VSIM_get_auth_cause

【函数功能】

获取当前卡鉴权类型。

【函数原型】

```
int VSIM_get_auth_cause(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

鉴权类型值。

2.1.9 VSIM_set_nv

【函数功能】

设置对应卡槽的卡模式。该卡槽模式值是否保存 NV 由 writeNv 决定, 写入 NV 后下次开机仍生效, 不写入 NV, 下次开机默认实体卡模式。

【函数原型】

```
int VSIM_set_nv (int phoneId, int mode, int writeNv)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。
mode	卡的模式, 为实体卡还是虚拟卡。 <ul style="list-style-type: none">• 0: 实体卡。• 1: 云卡。• 2: 软卡。
writeNV	<ul style="list-style-type: none">• 0: 不写 NV。• 1: 写 NV。

【返回值】

- 操作成功: 返回 0。
- 操作失败: 返回 -1。

📖 说明

该接口只修改卡槽模式, 开关卡操作需要 APP 控制进行。

2.2 APK 接口定义

2.2.1 VSIMInit

【函数功能】

VSIM 初始化。

【函数原型】

```
int VSIMInit(int phoneId, int mode, IVSIMCallback cb)
```

【参数说明】

参数名称	含义
phoneId	对应的卡槽 id, $0 \leq \text{phoneId} \leq 1$ 。

参数名称	含义
mode	VSIM 初始化方式。 <ul style="list-style-type: none"> 0: VSIM 全初始化，应用 APP 在 VSIM 无卡状态下初始化。 1: VSIM 卡状态正常情况下，应用 APP 进程异常退出时，只初始化 APP 和展锐 VSIM 模块通信 socket，不重启 VSIM。
cb	回调对象，Modem 的读卡、鉴权请求会通过该对象回调给客户 APK。

【返回值】

- 操作成功：返回 1。
- 操作失败：返回-1。

2.2.2 VSIMSendData

【函数功能】

发送 VSIM 读卡、鉴权数据给 Modem。

【函数原型】

```
int VSIMSendData(int phoneId, in byte[] data, int data_len)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。
data	VSIM 读卡、鉴权值。
data_len	数据长度。

【返回值】

- 操作成功：返回 1。
- 操作失败：返回-1。

2.2.3 VSIMExit

【函数功能】

关闭 VSIM。

【函数原型】

```
int VSIMExit(int phoneId)
```


【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

- 操作成功：返回 1。
- 操作失败：返回-1。

2.2.4 VSIMSetAuthid

【函数功能】

设置发送 VSIM 鉴权数据使用的数据通道卡。

【函数原型】

```
int VSIMSetAuthid(int authid)
```

【参数说明】

参数名称	含义
authid	发送鉴权数据使用数据通道的卡 id, $0 \leq \text{authid} \leq 1$ 。

【返回值】

- 操作成功：返回 0。
- 操作失败：返回-1。

2.2.5 VSIMQueryAuthid

【函数功能】

查询当前使用哪张卡数据通道发送 VSIM 鉴权数据。

【函数原型】

```
int VSIMQueryAuthid()
```

【参数说明】

无。

【返回值】

- 操作成功：返回 authid， $0 \leq \text{authid} \leq 1$ 。
- 操作失败：返回-1。

2.2.6 VSIMSetVirtual

【函数功能】

设置对应卡槽的卡模式，默认该模式值会保存 NV，下次开机继续生效。

【函数原型】

```
int VSIMSetVirtual(int phoneId, int mode)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id， $0 \leq \text{phoneId} \leq 1$ 。
mode	卡的模式，为实体卡还是虚拟卡。 <ul style="list-style-type: none"> • 0：实体卡。 • 1：云卡。 • 2：软卡。

【返回值】

- 操作成功：返回 0。
- 操作失败：返回-1。

📖 说明

该接口只修改卡槽模式，开关卡操作需要 APP 控制进行。

2.2.7 VSIMQueryVirtual

【函数功能】

查询对应卡槽的卡模式。

【函数原型】

```
int VSIMQueryVirtual(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id， $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

- 操作成功：返回 0 或 1 或 2。
 - 0：实体卡。
 - 1：云卡。
 - 2：软卡。
- 操作失败：返回-1。

2.2.8 VSIMGetAuthCause

【函数功能】

获取当前卡鉴权类型。

【函数原型】

```
int VSIMGetAuthCause(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

鉴权类型值。

2.2.9 setDefaultDataSubId

【函数功能】

设置数据卡 subId。

【函数原型】

```
void setDefaultDataSubId(int subId)
```

【参数说明】

参数名称	含义
subId	对应的卡槽 subId, 需要将 phoneId 转为 subId。

【返回值】

无。

2.2.10 setSimPowerStateForSlot

【函数功能】

打开或关闭对应卡槽的实体卡，仅限操作实体卡使用。

【函数原型】

```
void setSimPowerStateForSlot(int slotId, boolean state)
```

【参数说明】

参数名称	含义
slotId	对应的卡槽 id。
state	<ul style="list-style-type: none"> • true: 打开实体卡。 • false: 关闭实体卡。

【返回值】

无。

2.2.11 attachAPN

【函数功能】

配置 APN 到 modem。

【函数原型】

```
void attachAPN(int phoneId, String protocol, String apn, String userName, String pwd, int authtype)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。
protocol	当前 VSIM 使用的 APN 信息。
apn	
userName	
pwd	
authtyp	

【返回值】

无。

2.2.12 getSubscriberIdForSlotIdx

【函数功能】

获取对应卡槽 IMSI 信息。

【函数原型】

```
String getSubscriberIdForSlotIdx(int slotIdx)
```

【参数说明】

参数名称	含义
slotIdx	对应的卡槽 id, $0 \leq \text{slotIdx} \leq 1$ 。

【返回值】

IMSI 信息。

2.2.13 getSubId

【函数功能】

根据 phoneId 获取 subId。

【函数原型】

```
int getSubId(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

卡槽对应的当前 subId。

2.2.14 getVoiceRegState

【函数功能】

获取当前卡语音网络注册状态。

【函数原型】

```
int getVoiceRegState(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

语音网络注册状态。对应值见原生接口 API 定义。

2.2.15 getDataRegState

【函数功能】

获取当前卡数据网络注册状态。

【函数原型】

```
int getDataRegState(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

数据网络注册状态。对应值见原生接口 API 定义。

2.2.16 getNetworkOperator

【函数功能】

获取当前卡注册网络运营商 PLMN。

【函数原型】

```
String getNetworkOperator(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

当前卡注册网络运营商 PLMN 。

2.2.17 getVoiceNetworkType

【函数功能】

获取当前卡语音网络注册类型。

【函数原型】

```
int getVoiceNetworkType(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

语音网络注册类型。对应值见原生接口 API 定义。

2.2.18 getDataNetworkType

【函数功能】

获取当前卡数据网络注册类型。

【函数原型】

```
int getDataNetworkType(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

数据网络注册类型。对应值见原生接口 API 定义。

2.2.19 getSimState

【函数功能】

获取当前 sim 卡状态。

【函数原型】

```
int getSimState(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

sim 卡状态。对应值见原生接口 API 定义。

2.2.20 setDataEnabled

【函数功能】

打开或关闭数据连接。

【函数原型】

```
void setDataEnabled(boolean enable)
```

【参数说明】

参数名称	含义
enable	<ul style="list-style-type: none">• true: 打开。• false: 关闭。

【返回值】

无。

2.2.21 setSimNeworkType

【函数功能】

设置卡槽 Modem 驻网能力，在重新开关卡或开关协议栈后生效。

【函数原型】

```
void setSimNeworkType(int phoneId, int type, boolean isPrimary)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id， $0 \leq \text{phoneId} \leq 1$ 。
type	见 Android RILConstants.java 中定义的类型。建议设置如下几种： <ul style="list-style-type: none">• RILConstants.NETWORK_MODE_WCDMA_PREF• RILConstants.NETWORK_MODE_GSM_ONLY• RILConstants.NETWORK_MODE_WCDMA_ONLY• RILConstants.NETWORK_MODE_LTE_GSM_WCDMA• RILConstants.NETWORK_MODE_LTE_ONLY• RILConstants.NETWORK_MODE_LTE_WCDMA
isPrimary	当前卡是否为主卡。

【返回值】

无。

2.2.22 getSimNeworkType

【函数功能】

获取卡槽 modem 驻网能力。

【函数原型】

```
int getSimNeworkType(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id， $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

网络类型。见接口 2.2.21 getSimNeworkType 中 type 说明。

2.2.23 getDefaultDataPhoneId

【函数功能】

获取默认数据卡 phoneId。

【函数原型】

```
int getDefaultDataPhoneId()
```

【参数说明】

无。

【返回值】

phoneId: 对应的卡槽 id, $0 \leq \text{phoneId} \leq 1$ 。

2.2.24 setDefaultDataPhoneId

【函数功能】

设置默认数据卡 phoneId。

【函数原型】

```
void setDefaultDataPhoneId(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

无。

2.2.25 restartRadio

【函数功能】

重启协议栈，配合接口 2.2.21 getSimNeworkType 使用。

【函数原型】

```
void restartRadio()
```

【参数说明】

无。

【返回值】

无。

2.2.26 updatePlmn

【函数功能】

更新 FPLMN、OPLMN 到 Modem。

【函数原型】

```
int updatePlmn(int phoneId, int type, int action, String plmn, int act1, int act2, int act3)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。
type	<ul style="list-style-type: none">0: FPLMN。1: OPEMN。
action	增加或删除 plmn。 <ul style="list-style-type: none">0: 删除。1: 增加。2: 删除所有。
plmn	Sim 卡对应的 plmn 值
注: act1,act2,act3 在操作 OPLMN 时生效, 值是 0 或 1。	

【返回值】

- 操作成功: 返回 1。
- 操作失败: 返回-1。

2.2.27 queryPlmn

【函数功能】

查询 PLMN。

【函数原型】

```
String[] queryPlmn(int phoneId, int type)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。
type	<ul style="list-style-type: none">0: FPLMN。1: OPEMN。

【返回值】

仅支持 OPLMN 查询，返回查询记录。每条记录格式：index1, format, oper1, GSM_AcT1, TD_AcT1, GSM_Compact_AcT1, UTRAN_AcT1。

2.2.28 setImei

【函数功能】

写对应卡槽的 IMEI。

【函数原型】

```
int setImei(int phoneId, String imei)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。
imei	<ul style="list-style-type: none">1: 接受 15 位格式的 IMEI 号。2: imei 传入为 “” 或 null，恢复当前卡槽默认的 IME。

【返回值】

- 操作成功：返回 1。
- 操作失败：返回-1。

2.2.29 getImei

【函数功能】

查询对应卡槽的 IMEI。

【函数原型】

```
String getImei(int phoneId)
```

【参数说明】

参数名称	含义
phoneId	对应的卡槽 id, $0 \leq \text{phoneId} \leq 1$ 。

【返回值】

返回 IMEI。

2.2.30 setPreferredNetworkType

【函数功能】

设置对应卡槽网络模式。

【函数原型】

```
boolean setPreferredNetworkType(int phoneId, int type)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。
type	见接口 2.2.21 getSimNeworkType 中 type 说明。

【返回值】

- 操作成功：返回 true。
- 操作失败：返回 false。

2.2.31 powerRadio

【函数功能】

开、关单卡协议栈。

【函数原型】

```
void powerRadio(int phoneId, boolean on)
```

【参数说明】

参数名称	含义
phoneId	对应的卡槽 Id, $0 \leq \text{phoneId} \leq 1$ 。
on	<ul style="list-style-type: none"> • true: 开协议栈。 • false: 关协议栈。

【返回值】

无。

2.2.32 VSIMSetVirtualWithNV

【函数功能】

设置对应卡槽的卡模式。该卡槽模式值是否保存 NV 由 writeNv 决定，写入 NV 后下次开机还会生效，不写入 NV，下次开机默认实体卡模式。

【函数原型】

```
int VSIMSetVirtualWithNV (int phoneId, int mode, int writeNv)
```

【参数说明】

参数名称	含义
phoneId	对应卡槽的 id, $0 \leq \text{phoneId} \leq 1$ 。
mode	卡的模式，为实体卡还是虚拟卡。 <ul style="list-style-type: none"> • 0: 实体卡。 • 1: 云卡。 • 2: 软卡。
writeNv	<ul style="list-style-type: none"> • 0: 不写 NV。 • 1: 写 NV。

【返回值】

- 操作成功：返回 0。
- 操作失败：返回-1。

📖 说明

该接口只修改卡槽模式，开关卡操作需要 APP 控制进行。

Unisoc Confidential For hiar

3 注意事项

3.1 集成注意事项

3.1.1 使用 libatci 集成注意事项

- 客户 APP 中集成 libatci.so 库文件，在支持 VSIM 的版本上，可以从系统 system/lib 获取该 so 库文件，也可以使用展锐发布的 so 库。建议将 so 库放到 APP 目录下，如果遇到编译失败或库文件加载失败，查看是否有依赖的库需要加载。
- 客户 APP 中集成 libatci 源码，源码路径 vendor/sprd/modules/libatci，在对应 mk 文件中做对应配置：LOCAL_C_INCLUDES += \$(LOCAL_PATH)/../../../../modules/libatci（需要根据源码路径配置）。

3.1.2 使用 APK 集成注意事项

VSIMService.APK 通过 AIDL 方式提供接口，客户 APK 需要 bindservice，对应 service action 为 "com.sprd.VSIMInterface.IVSIMInterface"。

3.2 VSIM 场景使用注意事项

- 卡类型切换（实体卡，云卡，软卡）必须在卡下电的时候切换。
 - 实体卡：实际插入卡槽的 SIM 卡。
 - 云卡：VSIM 卡。
 - 软卡：与云卡作用相同，主要区别在于软卡不需要通过实体卡数据业务和服务器获取数据。

说明

调用 VSIMSetVirtual、VSIMSetVirtualWithNV 接口更改卡槽模式时，需要先调用 VSIMExit、setSimPowerStateForSlot 关闭当前卡槽的卡。

- 不管成功还是失败，虚拟卡的请求必须要有回应。

说明

客户 APK 收到鉴权、读卡、写卡请求后，在进行其他卡操作之前（比如关机），必须回复结果。目前系统只有在关机 VSIMExit 前没有回复请求。为了避免其他场景出问题，请尽量回复请求。

- 点火卡（已注册网络的实体卡）在云卡关闭前不建议关闭。
- 如果点火卡需要做数据业务（云卡鉴权时，发送云卡鉴权数据除外），需要调用 powerRadio 接口关闭云卡协议栈。
- 双卡系统中仅支持一张云卡。
- 连续从实体卡切换为云卡、软卡或从云卡、软卡切换为实体卡时，请在关闭当前卡之后，收到当前卡空缺的状态后再开启新的卡，避免无时间间隔快速切换卡。
- 云卡鉴权时，如果点火卡卡槽没有开启卡，请不要调用 VSIMSetAuthid。

- 调用 `setSimNeworkType` 设置网络模式时，仅支持如下几个模式：
 - `RILConstants.NETWORK_MODE_WCDMA_PREF`
 - `RILConstants.NETWORK_MODE_GSM_ONLY`
 - `RILConstants.NETWORK_MODE_WCDMA_ONLY`
 - `RILConstants.NETWORK_MODE_LTE_GSM_WCDMA`
 - `RILConstants.NETWORK_MODE_LTE_ONLY`
 - `RILConstants.NETWORK_MODE_LTE_WCDMA`
- 当虚拟卡收到鉴权请求时，如果要关闭该虚拟卡，需要先返回鉴权结果（0x9864），然后延时 1s 左右再关闭虚拟卡（接入层网络连接释放，挂起恢复等状态变迁需要 1s 左右的时间）。
- 在云卡鉴权过程中，不要切换网络模式，建议在云卡初始化之前设置好。
- 在云卡鉴权过程中，不要进行协议栈的开关，建议鉴权过程结束后再进行操作。

Unisoc Confidential For hiar