**展讯VSIM功能说明及接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Number:** |  | **Document Version:** | V1.1 |
| **Owner:** |  | **Date:** | 2017-3-3 |
| **Document Type:** |  | | |
| **NOTE:** | ALL MATERIALS INCLUDED HEREIN ARE COPYRIGHTED AND CONFIDENTIAL UNLESS OTHERWISE INDICATED. The information is intended only for the person or entity to which it is addressed and may contain confidential and/or privileged material. Any review, retransmission, dissemination, or other use of or taking of any action in reliance upon this information by persons or entities other than the intended recipient is prohibited.  This document is subject to change without notice. Please verify that your company has the most recent specification.  Copyright © 2013 Spreadtrum Communications Inc. | | |



[www.spreadtrum.com](http://www.spreadtrum.com/)

修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **作 者** | **说 明** |
| 1.0.0 | 2017-07-28 | Yinpan.Fu | 接口定义 |
| 2.0.0 | 2018-01-10 | Yinpan.Fu | 增加apk接口定义和使用说明 |
| 2.0.1 | 2018-7-20 | Yinpan.Fu | 增加修改卡槽模式时是否保存NV接口及接口说明 |
| 2.0.2 | 2018-09-24 | Eddie.Wang | 增加鉴权过程中VSIM接口使用注意事项 |
|  |  |  |  |

重要声明

**版权声明**

**版权所有 © 2017, 展讯通信有限公司，保留所有权利。**

**商标声明**

展讯通信有限公司和展讯通信有限公司的产品是展讯通信有限公司专有。在提及其他公司及其产品时将使用各自公司所拥有的商标，这种使用的目的仅限于引用。

**不作保证声明**

展讯通信有限公司不对此文档中的任何内容作任何明示或暗示的陈述或保证，而且不对特定目的的适销性及适用性或者任何间接、特殊或连带的损失承担任何责任。

**保密声明**

本文档（包括任何附件）包含的信息是保密信息。接收人了解其获得的本文档是保密的，除用于规定的目的外不得用于任何目的，也不得将本文档泄露给任何第三方。

**目 录**

修订历史 2

重要声明 3

1. 简介 5

2. VSIM功能说明 6

2.1开机VSIM注册鉴权流程 6

2.1.1 使用libatci中接口初始化流程说明 8

2.1.2 使用apk接口初始化流程说明 8

3. 接口定义 9

3.1 libatci中VSIM接口定义 9

3.2 APK中接口定义 11

4. 注意事项 15

4.1 使用libatci集成注意事项 15

4.2 使用apk集成注意事项 15

4. 3 VSIM场景使用注意事项 16

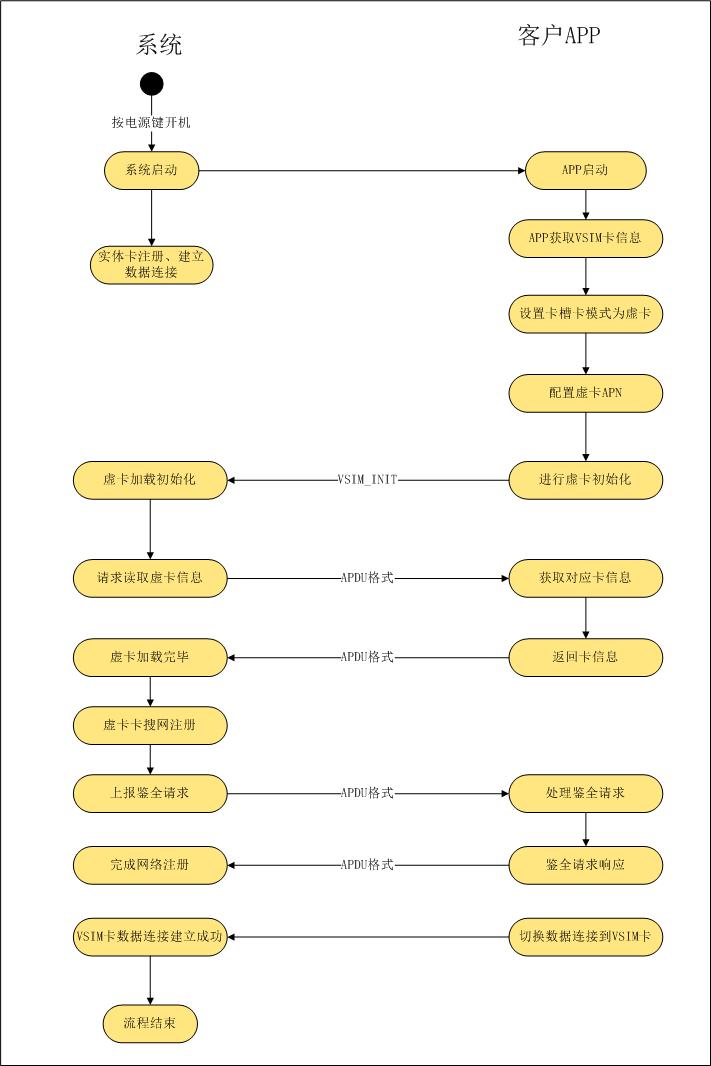
# 简介

本文档提供展讯平台VSIM功能说明、相关接口的定义，给客户在开发VSIM过程中提供指导。目前已经实现VSIM卡初始化、注册鉴权，过程中鉴权等功能。

# VSIM功能说明

## 2.1开机VSIM注册鉴权流程

该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。具体流程如下：



### 2.1.1 使用libatci中接口初始化流程说明

（1）APP设置卡槽模式

如果需要设置为指定卡槽为虚卡，请先使用vsim\_query\_virtual查询当前卡槽模式，然后使用vsim\_set\_virtual设置卡槽卡模式。推荐使用卡2作为VSIM卡槽，卡1作为实体卡槽。

（2）配置attach APN到Modem

APP在确定虚卡信息后，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。目的是降低attch被拒的概率。

AT命令如下：

AT+CGDCONT=1,"IPV4V6","APN","",0,0

AT+CGPCO=0,"user","password",1,authtype

authtype没有指定时请默认为3.​

可以调用接口7直接发送AT。注意AT命令中的引号需要保留。

（3）VSIM初始化

在配置完vsim卡APN后，可以使用vsim\_init接口进行VISM初始化操作。

（4）鉴权请求响应

APP在收到鉴权请求后，需要先调用vsim\_set\_authid设置发送鉴权数据使用的数据连接卡id，然后切换数据连接到对应的卡上。

### 2.1.2 使用apk接口初始化流程说明

（1）设置卡槽模式

通过接口vsimSetVirtual设置卡槽模式，

（2）配置apn到modem

通过接口attachAPN配置apn到modem 。ps：该接口只将APN发送到modem。需要客户apk实现将apn添加到apn数据库中。

（3）VSIM初始化

在配置完vsim卡APN后，可以使用vsimInit接口进行VISM初始化操作。

（4）鉴权请求响应

APP在收到鉴权请求后，需要先调用vsimSetAuthid设置发送鉴权数据使用的数据连接卡id，然后切换数据连接到对应的卡上。

（5）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。Apk中和vsim无关的接口可以用。

# 接口定义

## 3.1 libatci中VSIM接口定义

(1) int vsim\_init(int phoneId, VSIM\_COMMAND pfnCommand, int mode)

函数说明：对应卡槽的Vsim初始化

输入：phoneId为初始化的卡槽id, 0<= phoneId <=1;

pfnCommand: 传入调用方处理鉴权请求的函数指针.展讯VSIM模块调用该函数将鉴权数据传递给APP处理.需要调用方可以定义该函数或使用展讯现有通用接口定义，在APP处理完鉴权后将鉴权结果返回给展讯VSIM模块.

mode: VSIM初始化方式， 0：VSIM全初始化, 应用在VSIM无卡状态下出始化; 1:只初始化APP和展讯VSIM模块通信socket, 不重启VSIM, 应用在APP进程异退出,VSIM卡状态正常情况下.

输出：初始化成功返回 1，初始化失败返回-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：

输入：

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；

输出：返回鉴权响应数据实际长度。

（2）int vsim\_exit(int phoneId)

函数说明：关闭对应卡槽的Vsim

输入：phoneId：对应卡槽的id，0<= phoneId <=1;

输出：成功返回1，失败返回-1；

（3）int vsim\_set\_authid(int phoneId ，nt authid)

函数说明：设置使用哪张卡数据通道发送Vsim鉴权数据

输入：phoneId：当前鉴权的卡id；权数据使用数据通道的卡id，0<= authid <=1

输出：成功返回0， 失败返回-1；

（4）int vsim\_query\_authid(int phoneId)

函数说明：查询当前使用哪张卡数据通道发送Vsim鉴权数据

输入：phoneId：当前鉴权的卡id；

输出：查询成功返回authid，0<= authid <=1，查询失败返回-1；

（5）int vsim\_set\_virtual(int phoneId, int mode)

函数说明：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

输入：phoneId：对应卡槽的id，0<= phoneId <=1;

Mode：卡的模式，为实体卡还是虚拟卡，0：实体卡，1：云卡，2：软卡

说明：该接口只修改卡槽模式，开关卡操作需要APP控制进行。

输出：成功返回0， 失败返回-1；

（6）int vsim\_query\_virtual(int phoneId)

函数说明：查询对应卡槽的卡模式

输入：phoneId：对应卡槽的id，0<= phoneId <=1;

输出：查询成功会0，1，2；0：实体卡，1：云卡， 2：软卡；查询失败返回-1；

（7）int sendATCmd(int phoneId, const char \*atCmd, char \*resp, size\_t respLen)

函数说明：发送AT命令给modem，并接收返回结果，该函数是同步操作。

输入:phoneId： 对应的卡槽id，0<= phoneId <=1; atCmd： 发送的AT命令; resp 返回结果, 需要调用方分配好内存; respLen 内存长度

输出: AT发送成功返回0, 失败返回-1

（8）int vsim\_get\_auth\_cause(int phoneId)

函数说明：获取当前卡鉴权类型

输入:phoneId： 对应的卡槽id，0<= phoneId <=1;

输出：鉴权类型值。

（8）int vsim\_set\_nv (int phoneId, int mode，int writeNv)

函数说明：设置对应卡槽的卡模式, 该卡槽模式值是否包存NV有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。

输入：phoneId：对应卡槽的id，0<= phoneId <=1;

Mode：卡的模式，为实体卡还是虚拟卡，0：实体卡，1：云卡，2：软卡

writeNv:0不写NV，1写NV。

说明：该接口只修改卡槽模式，开关卡操作需要APP控制进行。

输出：成功返回0， 失败返回-1；

## 3.2 APK中接口定义

（1） int vsimInit(int phoneId, int mode, IVSIMCallback cb);

函数说明：VSIM初始化

输入：phoneId： 对应的卡槽id，0<= phoneId <=1;

mode: mode: VSIM初始化方式， 0：VSIM全初始化, 应用在VSIM无卡状态下出始化; 1:只初始化APP和展讯VSIM模块通信socket, 不重启VSIM, 应用在APP进程异退出,VSIM卡状态正常情况下.

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。

输出：初始化成功返回 1，初始化失败返回-1；

（2）int vsimSendData(int phoneId, in byte[] data, int data\_len)

函数说明：发送vsim读卡、鉴权数据给modem

输入:data：VSIM读卡、鉴权值；data\_len：数据长度

输出：成功返回 1，初始化失败返回-1；

（3）int vsimExit(int phoneId)

函数说明：关闭VSIM

输入: phoneId： 对应的卡槽id，0<= phoneId <=1;

输出：成功返回 1，初始化失败返回-1；

（4）int vsimSetAuthid(int phoneId,int authid)

函数说明：设置使用哪张卡数据通道发送Vsim鉴权数据

输入：phoneId: 当前进行鉴权的卡；authid:发送鉴权数据使用数据通道的卡id，0<= authid <=1

输出：成功返回0， 失败返回-1；

（5）int vsimQueryAuthid(int phoneId)

函数说明：查询当前使用哪张卡数据通道发送Vsim鉴权数据

输入：phoneId：当前进行鉴权的卡id

输出：查询成功返回authid，0<= authid <=1，查询失败返回-1；

（6）int vsimSetVirtual(int phoneId, int mode)

函数说明：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

输入：phoneId：对应卡槽的id，0<= phoneId <=1;

Mode：卡的模式，为实体卡还是虚拟卡，0：实体卡，1：云卡， 2：软卡

输出：成功返回0， 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。

（7）int vsimQueryVirtual(int phoneId)

函数说明：查询对应卡槽的卡模式

输入：phoneId：对应卡槽的id，0<= phoneId <=1;

输出：查询成功会0或1，0：实体卡，1：云卡，2：软卡；查询失败返回-1；

（8）int vsimGetAuthCause(int phoneId)

函数说明：获取当前卡鉴权类型

输入:phoneId： 对应的卡槽id，0<= phoneId <=1;

输出：鉴权类型值。

（9）byte[] getAPDUFromRsim(int phoneId, in byte[] apdu\_in)

内部调试使用，客户apk禁用。

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

函数说明：打开或关闭对应卡槽的实体卡， 仅限操作实体卡使用

输入: slotId： 对应的卡槽id，state：true，打开实体卡， false， 关闭实体卡

输出：无

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

函数说明：配置APN到modem

输入: protocol，apn，userName, pwd，authtype 为当前VSIM使用的apn信息

输出：无

（12）String getSubscriberIdForSlotIdx(int soltId)

函数说明：获取对应卡槽IMSI信息

输入: slotId：对应的卡槽id，0<= phoneId <=1

输出：IMSI信息

（13）String getNetworkOperator(int phoneId)

函数说明：获取当前卡注册网络运营商PLMN

输入: phoneId：对应的卡槽id，0<= phoneId <=1

输出：当前卡注册网络运营商PLMN

（14）int getVoiceNetworkType(int phoneId)

函数说明：获取当前卡语音网络注册类型

输入: phoneId：对应的卡槽id，0<= phoneId <=1

输出：语音网络注册类型。对应值见原生接口API定义

（15）int getDataNetworkType(int phoneId)

函数说明：获取当前卡数据网络注册类型

输入: phoneId：对应的卡槽id，0<= phoneId <=1

输出：数据网络注册类型。对应值见原生接口API定义

（16）int getSimState(int phoneId)

函数说明：获取当前sim卡状态

输入: phoneId：对应的卡槽id，0<= phoneId <=1

输出：sim卡状态。对应值见原生接口API定义

（17）void setDataEnabled(boolean enable)

函数说明：打开或关闭数据连接

输入: enable：true，打开；false，关闭

输出：无

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

函数说明：设置卡槽modem驻网能力，在重新开关卡或开关协议栈后生效

输入: type：见Android RILConstants.java中定义的类型。建议设置如下几种：

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:

输出：无

（19）int getSimNeworkType(int phoneId)

函数说明：获取卡槽modem驻网能力

输入: phoneId：对应的卡槽id，0<= phoneId <=1

输出：网络类型。见接口22中type说明

（20）int getDefaultDataPhoneId()

函数说明：获取默认数据卡phoneid

输入: 无

输出：phoneId：对应的卡槽id，0<= phoneId <=1

（21）void setDefaultDataPhoneId(int phoneId)

函数说明：设置默认数据卡phoneid

输入: phoneId：对应的卡槽id，0<= phoneId <=1

输出：无

（22）void restartRadio()

函数说明：重启协议栈， 配合接口22使用

输入: phoneId：对应的卡槽id，0<= phoneId <=1

输出：无

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

函数说明：更新FPLMN、OPLMN到modem。

输入: type：0—FPLMN；1—OPEMN；action，增加或删除plmn，0--删除，1—增加，2—删除所有。Act1,act2,act3在操作OPLMN是生效，值是0或1，

输出：成功返回1，失败返回-1

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

函数说明：查询PLMN

输入: type：0—FPLMN；1—OPEMN

输出：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​

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

函数说明：写对应卡槽的IMEI，

输入: imei：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imei

输出：成功返回1，失败返回-1

（26）String getImei(int phoneId)

函数说明：查询对应卡槽的IMEI，

输入: phoneId：对应的卡槽id，0<= phoneId <=1

输出：返回imei

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

函数说明：设置对应卡槽网络模式

输入: type：见接口22中type说明

输出：成功返回true，失败返回false

（28) void powerRadio(int phoneId, boolean on)​

函数说明：开、关单卡协议栈

输入：phoneId：对应的卡槽id，0<= phoneId <=1，on:开协议栈ture，关协议栈 false。

输出：无

（29）int vsimSetVirtualWithNV (int phoneId, int mode，int writeNv)

函数说明：设置对应卡槽的卡模式, 该卡槽模式值是否包存NV有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。

输入：phoneId：对应卡槽的id，0<= phoneId <=1;

Mode：卡的模式，为实体卡还是虚拟卡，0：实体卡，1：云卡，2：软卡

writeNv:0不写NV，1写NV。

说明：该接口只修改卡槽模式，开关卡操作需要APP控制进行。

输出：成功返回0， 失败返回-1

（30）tring sendATCmd(int phoneId, String cmd)

函数说明：在指定phoneid对应的卡上发送AT

输入：phoneId：对应卡槽的id，0<= phoneId <=1;

cmd：需要发送的AT

输出：AT命令返回结果

# 注意事项

## 使用libatci集成注意事项

1. 客户APP中集成libatci.so库文件，该so库文件可以在支持VSIM的版本上从系统system/lib下获取，也可以使用展讯发布的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。
2. 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）

## 4.2 使用apk集成注意事项

VsimService.pk通过AIDL方式提供接口，客户apk需要bindservice，对应service action为"com.sprd.vsiminterface.IVSIMInterface"。

## 4. 3 VSIM场景使用注意事项

1. 卡类型切换（实体卡，云卡，软卡）必须在卡下电的时候切换：

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。

2. 虚拟卡的请求必须要有回应，不管成功还是失败：

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

3. 点火卡在云卡关闭前不建议关闭；

4. 如果点火卡需要做数据业务，需要关闭云卡协议栈：

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。

5. 双卡系统中仅支持一张云卡

6.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absent状态后再开启新的卡。避免无时间间隔快速更换卡。

7. 云卡鉴权时，如果点火卡卡槽没有开启卡， 请不要调用vsimSetAuthid。

8.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:

9.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）

10.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。

11.云卡鉴权过程中，不要进行协议栈的开关，建议在鉴权过程结束后再操作。