Q/CUP

中 国 银 联 股 份 有 限 公 司 企 业 标 准

Q/CUP 009.7—2015

中国银联银联卡受理终端应用规范
第 7 部分 智能销售点终端应用规范

Application Specification for Terminal Accepting CUP Cards

Part 7 Application Specification for Smart Point of Sale Terminal

2015-12-8 发布 2015-12-9 实施

中国银联股份有限公司 发布

中国银联股份有限公司（以下简称“中国银联”）对该规范文档保留全部知识产权权利，包括但不限于版权、专利、商标、商业秘密等。任何人对该规范文档的任何使用都要受限于在中国银联成员机构服务平台（http://member.unionpay.com/）与中国银联签署的协议之规定。中国银联不对该规范文档的错误或疏漏以及由此导致的任何损失负任何责任。中国银联针对该规范文档放弃所有明示或暗示的保证,包括但不限于不侵犯第三方知识产权。

未经中国银联书面同意，您不得将该规范文档用于与中国银联合作事项之外的用途和目的。未经中国银联书面同意，不得下载、转发、公开或以其它任何形式向第三方提供该规范文档。如果您通过非法渠道获得该规范文档，请立即删除，并通过合法渠道向中国银联申请。

中国银联对该规范文档或与其相关的文档是否涉及第三方的知识产权（如加密算法可能在某些国家受专利保护）不做任何声明和担保，中国银联对于该规范文档的使用是否侵犯第三方权利不承担任何责任，包括但不限于对该规范文档的部分或全部使用。

目　　次

前　言

《银联卡受理终端应用规范》（Q/CUP 009-2014）对受理银联卡（包括磁条卡和IC卡）终端应用软件作具体规定，分为7个部分：

——第1部分：销售点（POS）终端应用规范

——第2部分：ATM终端应用规范

——第3部分：预留

——第4部分：密钥分发专用POS终端规范

——第5部分：电话支付终端应用规范

——第6部分：脚本POS技术规范

——第7部分：智能销售点终端应用规范

本部分为《银联卡受理终端应用规范》的第7部分。

本部分主要依据《中国集成电路（IC）卡规范》（JR/T 0025-2010），《银联卡受理终端安全规范 第一部分 销售点（POS）终端安全规范》（Q/CUP 007.1-2014）、《银联卡受理终端安全规范 第八部分 智能销售点终端安全规范》（QCUP 007.8-2014）和《银联卡受理终端应用规范 第一部分 销售点（POS）终端应用规范》（Q/CUP 009.1-2014）。

本标准由中国银联提出。

本标准由中国银联技术管理部组织制定和修订。

本标准的主要起草单位：中国银联技术管理部、电子支付研究院、上海慧银信息科技有限公司。

本标准的主要起草人：何朔、刘国宝、程志强、张少飞、金海青

中国银联银联卡受理终端应用规范
第 7 部分 智能销售点终端应用规范

1 范围

本标准对智能销售点终端的系统功能、银行卡支付服务层、应用开发要求及应用管理等做具体规范。

本标准所指智能销售点终端的特性须符合以下几方面要求：

-- 智能销售点终端之上运行的是智能操作系统，该系统须支持多应用的运行、下载、升级等。

-- 运行在智能销售点终端之上的应用须可控，可管理。

本标准适用于商户收单使用的支付终端，不适用于持卡人本人使用的支付终端。

2 适用对象

本规范适用对象包括终端厂商、第三方应用开发商、机构。

-- 终端厂商应遵照规范中第6章终端智能操作系统的要求，定制终端智能操作系统。

-- 第三方应用开发商应遵循规范中的第8章应用开发要求开发第三方应用；第三方应用开发商可参考规范中第6章终端智能操作系统中的标准外设驱动接口、外设访问权限要求开发具备标准外设访问能力的应用；第三方应用开发商可参考规范中第7章银行卡支付服务层开发具备银行卡支付能力的应用。

-- 机构须遵照规范中第7章银行卡支付服务层，为智能销售点终端提供银行卡支付服务层。机构须遵照规范中第9章应用管理要求，为智能销售点终端提供银行卡支付客户端、银行卡支付平台、多应用管理客户端、多应用管理平台。

银联按照规范的要求，开发了银行卡支付服务层（参见附件B.1 银联支付服务层）、银行卡支付客户端（参见附件B.2 银联支付客户端）、银行卡支付平台（参见附件B.3 银联支付平台）、多应用管理客户端（参见附件B.4 银联应用商店客户端）、多应用管理平台（参见附件B.5 银联应用商店平台），可供机构直接使用。

3 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

《银联卡受理终端安全规范第八部分智能销售点终端安全规范》（QCUP 007.8-2014）

4 术语和定义

4.1 银行卡

商业银行等金融机构及邮政储汇机构向社会发行的，具有消费信用、转账结算、存取现金等全部或部分功能的信用支付工具。

4.2 磁条卡

物理特性符合GB/T 14916标准，磁条记录符合GB/T 15120、GB/T 15694-1、ISO 7812-2、GB/T17552标准的银行卡片。

4.3 集成电路（IC）卡

内部封装一个或多个集成电路用于执行处理和存储功能的卡片。

4.4 终端智能操作系统

管理和控制终端硬件与软件资源的计算机程序，是直接运行在“裸机”上的最基本的系统软件，能支持多应用的管理、运行。

4.5 机构

智能销售点终端生态圈中的重要参与方之一，是终端的拥有者，如收单机构、第三方专业化服务机构等，负责终端状态的管理及终端应用的管理。

4.6 终端厂商

智能销售点终端生态圈中的重要参与方之一，是智能销售点终端的提供商，负责开发智能销售点终端硬件及终端智能操作系统。

4.7 第三方应用开发商

智能销售点终端生态圈中的重要参与方之一， 负责为机构、商户提供软件解决方案，并负责将应用上传至多应用管理平台。

4.8 多应用管理平台

多应用管理平台包括三方面的功能：为第三方应用开发商提供应用上传、应用统计等功能；为多应用管理平台运营商提供应用生命周期的管理功能；为多应用管理客户端提供应用浏览、下载功能。

4.9 多应用管理平台运营商

多应用管理平台的提供商，负责多应用管理平台的研发、运维等。

4.10 多应用管理客户端

多应用管理客户端由多应用管理平台运营商开发、维护，并由终端厂商在终端出厂时预置于终端智能操作系统中。多应用管理客户端通过安全的方式接入多应用管理平台，至少提供应用下载、安装、升级、应用卸载功能。

4.11 终端管理员

负责终端运行参数的配置、终端应用的下载安装、升级、卸载等。

4.12 进程管理工具

智能销售点终端应用运行状态的查看及管理工具。

5 终端及系统架构

智能销售点终端及系统由五部分组成：终端硬件平台、终端智能操作系统、银联支付服务层、应用层、后台。终端及系统架构如图1所示：

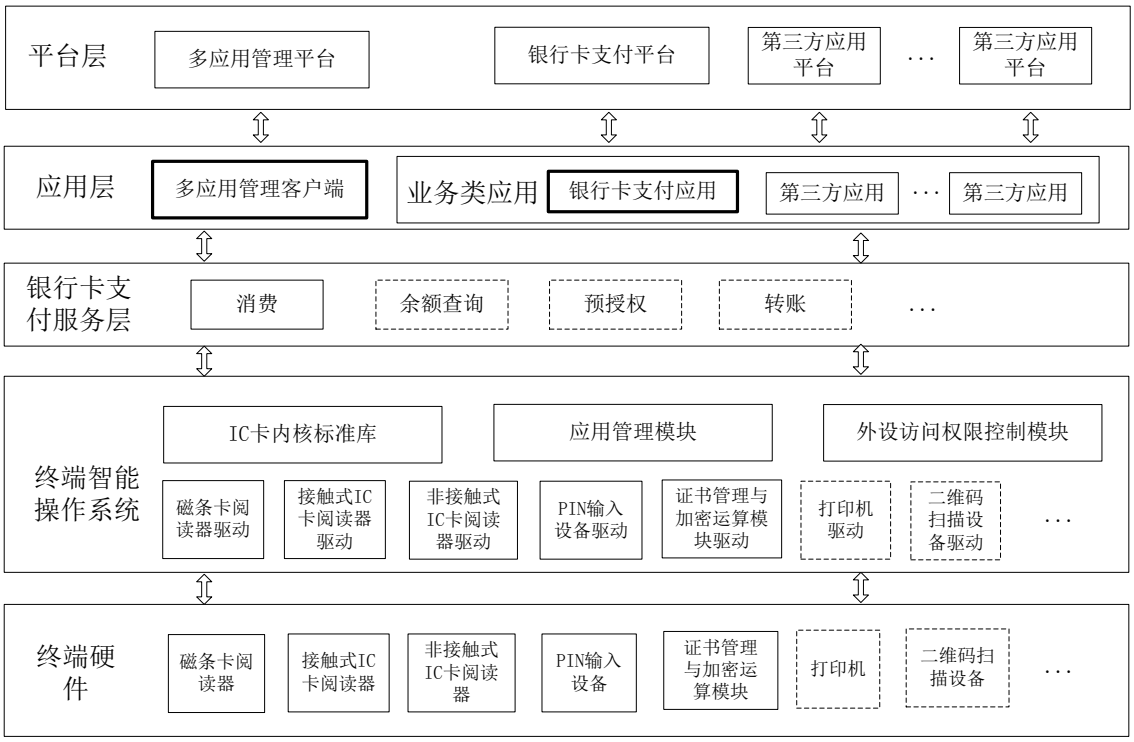


图1　智能销售点终端及系统架构图

-- 终端硬件，至少包括磁条卡阅读器、接触式IC卡阅读器、非接触式IC卡阅读器、PIN输入设备、证书管理与加密运算模块等外设，可选打印机、二维码扫描设备等外设。终端硬件由终端厂商按照银联标准开发。硬件开发的要求，参见《银联卡受理终端安全规范第八部分智能销售点终端安全规范》（QCUP 007.8-2013）；

-- 终端智能操作系统，由终端厂商根据本规范的要求进行定制，定制的内容至少包括磁条卡阅读器驱动、接触式IC卡阅读器驱动、非接触式IC卡阅读器驱动、PIN输入设备驱动、安全存储及运算设备驱动、IC卡内核标准库、应用管理模块、外设访问权限控制模块，可选打印机驱动、二维码扫描设备驱动；

-- 银行卡支付服务层，该层将银行卡消费等交易进行封装，为第三方应用提供标准化支付调用接口。该层至少须提供银行卡消费服务，可选提供银行卡余额查询、预授权、转账等服务。

-- 应用层，运行于智能销售点终端系统之上，应用分为两类：多应用管理客户端、业务类应用。多应用管理客户端由机构开发、维护。从支付属性出发，业务类应用可分为两类：银行卡支付应用、第三方应用。 银行卡支付应用由机构开发、维护，提供线下收单功能及界面操作。第三方应用，由第三方应用开发商根据市场需求或机构委托开发。

-- 平台层，包括多应用管理平台、银行卡支付平台、第三方应用平台。第三方应用平台为第三方应用提供内容、数据的访问接口，应由第三方应用提供商开发、运维。第三方应用与第三方应用平台之间宜采用安全的数据传输方式，以避免数据被篡改、被窃取。

6　终端智能操作系统

6.1　系统选型

规范不对系统进行特别限制。本规范所涉及的终端智能操作系统，除具备运行多应用、系统扩展性、安全性等通用能力外，还须具备足够的开放性以满足本章节所涵盖的系统定制要求。

6.2　屏幕分辨率要求

智能销售点终端屏幕分辨率须采用800*480、800*600、1024*768、1280*720、1920*1080、1280*800中的一种。

6.3 系统参数

终端智能操作系统参数应至少包括互联网通信参数、时间和日期等，可选参数包括地理位置服务设置、存储查看设置。终端智能操作系统参数须具备参数安全访问控制，安全级别至少不低于口令的安全级别。

6.4 终端自检

终端智能操作系统应提供终端设备模块检测功能，其中应包括磁条卡阅读器、非接触式IC卡阅读器、打印机、PIN输入设备等模块的状态检测：

表1 设备检测表

设备模块	状态
磁条卡阅读器	正常、故障
接触式 IC 卡阅读器	正常、故障
非接触式 IC 卡阅读器	正常、故障
PIN 输入设备（内置或外置，二选一）	正常、故障
证书管理与加密运算模块	正常、故障
打印机	存在、不存在、正常、故障、缺纸
二维码扫描设备	存在、不存在、正常、故障

6.5 系统模块管理

为了确保终端布放时系统的安全性，本规范要求终端智能操作系统包含两部分：系统加载模块、系统映象。终端智能操作系统模块管理如图2所示：

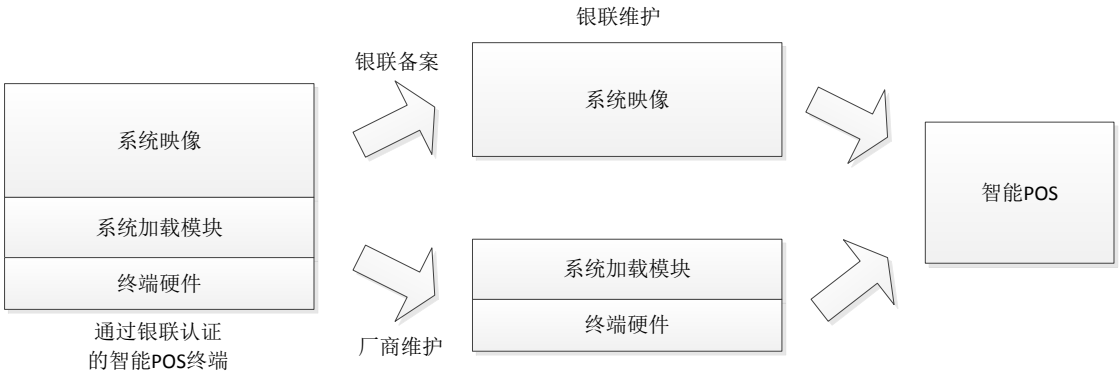


图2 终端智能操作系统模块管理图

其中，系统映象须在银联报备并由银联维护，系统加载模块由终端厂商维护。在智能销售点终端布放时，机构从银联和终端厂商分别获取系统映象及系统加载模块。

6.6 外设驱动标准 C 接口

智能销售点终端操作系统须为上层服务及应用提供统一的外设驱动标准接口。

统一的外设驱动标准接口包括磁条卡阅读器驱动接口、接触式IC卡阅读器驱动接口、非接触式阅读器驱动接口、PIN输入设备驱动接口以及打印机驱动接口。以下是外设驱动标准接口功能的最小集合要求：

6.6.1 磁条卡阅读器

6.6.1.1 打开磁条卡阅读器

功能描述	定义打开的磁条卡阅读设备功能，存在设备打开成功，打

	开异常两种情况，需要提供返回码，错误定义。 没有调用打开设备的接口时，调用本设备的其他接口均会返回错误。 磁条卡阅读器是独占设备，一旦某个应用打开后，本设备不能再被其他应用打开。 尝试打开一个已经被打开的独占设备会返回错误。
输入	无
输出	处理结果返回码

6.6.1.2 关闭磁条卡阅读器

功能描述	定义关闭一个磁条卡阅读器设备功能，存在成功关闭设备和关闭异常情况，需要提供返回码，错误定义。 磁条卡阅读器是独占设备，因此对于处于打开状态的设备，只有打开该设备的应用调用本接口关闭本设备的情况下，其他应用才能再次打开本设备。 如果已经打开本设备的应用进程消失，则自动关闭对设备的占用。
输入	无
输出	处理结果返回码

6.6.1.3 刷卡回调函数登记

功能描述	记回调函数，回调函数登记好之后，如果用户刷卡，回调函数会被调用，用来通知用户来取数据。
输入	用户信息
输出	处理结果返回码，处理后数据

6.6.1.4 撤销刷卡回调函数登记

功能描述	撤销登记的函数，撤销之后，刷卡后上层程序不会收到通知。
输入	无
输出	处理结果返回码

6.6.1.5 查询磁道解码错误

功能描述	查询对应磁道在解码过程中发生的错误。
输入	磁道索引
输出	处理结果返回码

6.6.1.6 查询磁道数据长度

功能描述	查询对应磁道数据的长度
输入	磁道索引
输出	处理结果和返回长度

6.6.1.7 获取磁道数据

功能描述	定义获取对应磁道数据功能，具备第 1,2，3 磁道读取能力

输入	磁道索引，磁道数据长度
输出	处理结果和返回内容

定义获取对应磁道数据功能，具备第1,2，3磁道读取能力。

6.6.2 接触式 IC 卡阅读器

6.6.2.1 初始化接触式 IC 卡阅读器

功能描述	定义初始化接触式读卡设备
输入	无
输出	处理结果

6.6.2.2 回收被接触式 IC 卡阅读器占用的资源

功能描述	回收被接触式读卡设备占用的资源
输入	无
输出	处理结果

6.6.2.3 查询可用卡槽（slot）数量

功能描述	查询可用卡槽（slot）数量，终端至少需要具备一个 IC 卡槽
输入	无
输出	处理结果，可用卡槽数

6.6.2.4 打开指定卡槽（slot）

功能描述	定义打开指定卡槽（slot）设备功能 没有调用打开卡槽接口时，调用本卡槽相关的其他操作接口均会返回错误。 IC 卡阅读器的每个卡槽（slot）均被视作一个独立的独占设备，一旦某个应用打开该卡槽后，该卡槽不能再被其他应用打开。 尝试打开一个已经被打开的独占设备会返回错误。
输入	卡槽索引
输出	处理结果 和 已打开卡槽结果

6.6.2.5 关闭已打开的指定卡槽（slot）

功能描述	定义关闭指定卡槽（slot）设备功能 IC 卡阅读器每个卡槽是一个独占设备，因此对于处于打开状态的卡槽，只有打开该卡槽的应用调用本接口关闭本卡槽的情况下，其他应用才能再次打开本卡槽。 如果已经打开本卡槽的应用进程消失，则自动关闭对卡槽的占用。
输入	卡槽句柄
输出	处理结果

6.6.2.6 查询指定卡槽中卡（和该句柄相关联的卡槽）是否存在

功能描述	查询指定卡槽中卡（和该句柄相关联的卡槽）是否存在
输入	卡槽句柄

输出	处理结果

6.6.2.7 给指定卡槽中的卡上电

功能描述	定义给指定卡槽中的卡上电功能，需要传入供电电压能参数
输入	卡槽句柄
输出	处理结果，卡片返回 ATR 信息

6.6.2.8 给指定卡槽（和该句柄相关联的卡槽）中的卡下电

功能描述	给指定卡槽中的卡下电，需要传入卡槽句柄
输入	卡槽句柄
输出	处理结果

6.6.2.9 设置相关信息

功能描述	设置相关信息，具体包括但不仅限于等待时间延长、校验计算方式、协议类型、供电电压、传输字节的方式等
输入	卡槽句柄，设置的信息
输出	处理结果

6.6.2.10 卡片信息交互

功能描述	实现与卡片进行交互功能，主要是实现 APDU 指令操作
输入	卡槽句柄，APDU 信息
输出	处理结果

6.6.3 非接触式 IC 卡阅读器

6.6.3.1 创建或者打开一个已经存在的非接触读卡设备

功能描述	定义创建或者打开一个已经存在的非接触读卡设备。 没有调用打开设备接口时，调用本设备相关的其他接口均会返回错误。 非接触式 IC 卡阅读器是独占设备，一旦某个应用打开本设备后，本设备不能再被其他应用打开。 尝试打开一个已经被打开的独占设备会返回错误。
输入	无
输出	处理结果，非接触读卡器信息，设备句柄

6.6.3.2 关闭非接触读卡设备

功能描述	定义关闭非接触读卡设备。 非接触式 IC 卡阅读器是一个独占设备，因此对于处于打开状态的设备，只有打开设备的应用调用本接口关闭设备的情况下，其他应用才能再次打开本设备。 如果已经打开本设备的应用进程消失，则自动关闭对本设备的占用。
输入	设备句柄
输出	处理结果

6.6.3.3 检卡

功能描述	搜索非接触 IC 卡

输入	读卡器设备句柄
输出	处理结果，IC 卡信息

6.6.3.4 停止检卡

功能描述	停止搜索非接触 IC 卡
输入	读卡器设备句柄
输出	处理结果

6.6.3.5 连接非接触式 IC 卡

功能描述	连接搜索非接触 IC 卡
输入	读卡器设备句柄
输出	处理结果，卡片信息

6.6.3.6 解除连接非接触 IC 卡

功能描述	解除连接搜索非接触 IC 卡
输入	读卡器设备句柄
输出	处理结果

6.6.3.7 卡片交互

功能描述	实现卡片交互
输入	读卡器设备句柄，APDU 信息
输出	处理结果，APDU 处理返回结果

6.6.3.8 发送非接触 IC 卡控制命令

功能描述	实现发送非接触 IC 卡控制命令
输入	读卡器设备句柄，控制命令信息
输出	处理结果，卡处理返回结果

定义发送非接触IC卡控制命令。

6.6.4 PIN 输入设备

6.6.4.1 打开 PIN 输入设备

功能描述	定义打开存在的 PIN 输入设备，包括内置 PIN 输入设备和外置 PIN 输入设备，需要传入采用的 PIN 输入设备类型。 没有调用打开设备接口时，调用本设备相关的其他接口均会返回错误。 PIN 输入设备是独占设备，一旦某个应用打开本设备后，本设备不能再被其他应用打开。 尝试打开一个已经被打开的独占设备会返回错误。
输入	无
输出	处理结果

6.6.4.2 关闭 PIN 输入设备

功能描述	定义关闭已打开的 PIN 输入设备，包括内置 PIN 输入设备和外置 PIN 输入设备，需要传入采用的 PIN 输入设备类型。 PIN 输入设备是一个独占设备，因此对于处于打开状态的设

	备，只有打开设备的应用调用本接口关闭设备的情况下，其他应用才能再次打开本设备。 如果已经打开本设备的应用进程消失，则自动关闭对本设备的占用。
输入	无
输出	处理结果

6.6.4.3 显示数据

功能描述	定义写入显示数据同时显示在 PIN 输入设备 LCD 上，无 LCD 或内置 PIN 输入设备的需要提供回调显示到终端屏幕
输入	显示内容，声音提示设置
输出	处理结果

6.6.4.4 选择一个 PIN 输入设备主密钥和用户密钥

功能描述	定义加密数据前先选择主密钥（TMK）和用户密钥（PIK、MAK、TDK），选择时需要指定主密钥类型、主密钥 ID、用户密钥 ID、算法
输入	密钥类型，主密钥 ID，用户密钥 ID，算法选择
输出	处理结果

——主密钥类型（0：dukpt，1：Tdukpt，2：master key，3：public key，4：fix key）；
——主密钥ID［0x00, … ,0x09］，当nKeyType为master key时使用；
——用户密钥ID，［0x00,0x01］，当nKeyType为master key时使用；
——算法选择（3DES，DES）

6.6.4.5 加密数据

功能描述	定义使用用户密钥加密数据，使用前要先选择一个 PIN 输入设备主密钥和用户密钥
输入	需加密字串
输出	处理结果，已加密字串

6.6.4.6 下载主密钥

功能描述	定义终端主密钥（masterKey）注入功能，正常情况下需要指定旧主密钥索引、旧密钥值、旧密钥长度、新密钥、新密钥长度。
输入	主密钥 ID，PIN 输入设备用户 ID，待更新的主密钥
输出	处理结果

6.6.4.7 更新用户密钥

功能描述	定义更新用户密钥功能，用户密钥数据是经主密钥加密过的密文，因此注入时需要指定所使用的主密钥索引。
输入	主密钥 ID，PIN 输入设备用户 ID，待更新的工作密钥
输出	处理结果

6.6.4.8 计算 pin block

功能描述	定义使用用户密钥提供计算 Pin Block 功能。

输入	银联卡卡号，PIN Block 缓冲区，超时和提示音设置
输出	处理结果

定义使用用户密钥提供计算Pin Block功能。

——针对没有LCD显示屏PIN输入设备，输密需要回调显示到终端主屏幕上，回调显示密码输入*号个数。

6.6.4.9 Mac 计算

功能描述	定义使用用户密钥提供计算 Mac 功能,以对报文进行完整性校验。
输入	Mac 计算数据，Mac 计算方式
输出	处理结果

常见算法：

——0 ： X9.19 算法 ,后补 80 ；

——1 ： 银联 ECB 算法；

——2 ： X9.19 算法（不足后补 0x00），移动支付项目使用。

——3 ： 中总行扩展算法。

——4 ： X9.19 算法 ,后补 00 ；

——5 ： 异或后 3DES 结果；

6.6.4.10 设置 Pin 长度

功能描述	定义设置 Pin 的最大限制长度，包括最小长度，最大长度。
输入	长度信息
输出	处理结果

6.6.5 打印机

6.6.5.1 打开打印设备

功能描述	定义打开打印设备。 没有调用打开设备接口时，调用本设备相关的其他接口均会返回错误。 打印设备是独占设备，一旦某个应用打开本设备后，本设备不能再被其他应用打开。 尝试打开一个已经被打开的独占设备会返回错误。
输入	无
输出	处理结果

6.6.5.2 关闭打印设备

功能描述	定义关闭打印设备。 打印设备是一个独占设备，因此对于处于打开状态的设备，只有打开设备的应用调用本接口关闭设备的情况下，其他应用才能再次打开本设备。 如果已经打开本设备的应用进程消失，则自动关闭对本设备的占用。
输入	无

输出	处理结果

6.6.5.3 打印文本数据

功能描述	定义从当前点连续打印，打印的数据存放在缓冲区中
输入	打印文本
输出	处理结果

6.6.5.4 打印位图

功能描述	定义从当前点连续打印，打印的数据存放在缓冲区中
输入	打印位图缓存
输出	处理结果

6.6.6 证书管理与加密运算模块

6.6.6.1 打开硬件证书管理与加密运算模块设备

功能描述	打开硬件证书关于和加密运算模块设备。 没有调用打开设备接口时，调用本设备相关的其他接口均会返回错误。 安全模块设备是一种特殊的独占设备，一旦某个具有读写权限的应用打开本设备后，本设备不能再被其他具有读写权限的应用打开，但是本设备可以被多个具有只读权限的应用同时打开。 尝试打开一个已经被打开的独占设备会返回错误。
输入	无
输出	处理结果

6.6.6.2 关闭硬件证书管理与加密运算模块设备

功能描述	关闭硬件证书关于和加密运算模块设备。 安全模块设备是一中特殊的独占设备，因此对于处于已经被具有读写权限的应用打开的设备，只有打开设备的应用调用本接口关闭设备的情况下，其他具有读写权限的应用才能再次打开本设备。 如果已经打开本设备的具有读写权限的应用进程消失，则自动关闭对本设备的占用。
输入	无
输出	处理结果

6.6.6.3 保存证书

功能描述	保存安全证书至硬件证书管理与加密运算模块设备
输入	证书属性，证书数据，证书格式
输出	处理结果

6.6.6.4 删除指定证书

功能描述	删除模块设备中某个证书

输入	证书编号等信息，证书 PIN 码
输出	处理结果

6.6.6.5 删除所有证书

功能描述	删除模块设备中所有证书
输入	无
输出	处理结果

6.6.6.6 读取证书

功能描述	从硬件证书管理与加密运算模块内读取证书
输入	证书索引号
输出	处理结果 ， 证书信息数据

6.7 IC 卡内核 C 接口

6.7.1 PBOC tag 存取功能

6.7.1.1 检查 PBOC Tag

功能描述	检查指定 PBOC Tag 的值是否存在
输入	Tag 编号
输出	处理结果

6.7.1.2 读取 PBOC Tag 值

功能描述	在 PBOC 应用被选取后，读取指定 Tag 的值
输入	Tag 编号
输出	处理结果 ， Tag 对应的值

6.7.1.3 设置 PBOC Tag 值

功能描述	在 PBOC 应用被选取后，设置指定 Tag 的值
输入	Tag 编号
输出	处理结果

6.7.2 PBOC 交易处理功能

6.7.2.1 PBOC 应用参数初始化

功能描述	PBOC 应用参数初始化
输入	PBOC 参数
输出	处理结果

6.7.2.2 读取应用 Kernel 版本

功能描述	提供读取当前应用核心版本
输入	无
输出	处理结果 ，返回的核心版本号

6.7.2.3 设置交易参数

功能描述	在 PBOC 应用被选取后，设置当前交易参数
输入	交易参数，包括交易金额，交易类型等
输出	处理结果

6.7.2.4 设置终端商户参数

功能描述	在 PBOC 应用被选取后，配置当前终端参数
输入	终端参数，包括终端商户名，商户类型等
输出	处理结果

6.7.2.5 设置黑名单卡片

功能描述	在终端中清除或设置黑名单卡片
输入	黑名单数据
输出	处理结果

6.7.2.6 设置终端证书

功能描述	在终端中清除或设置证书
输入	证书数据
输出	处理结果

6.7.2.7 查询卡片交易记录

功能描述	读取 IC 卡中交易记录
输入	无
输出	处理结果 ，卡片交易记录

6.7.2.8 设置持卡人验证方式

功能描述	设置持卡人身份的验证方式
输入	验证方式，包括：onlinepin，bypasspin 等
输出	处理结果

6.7.2.9 设置联机交易结果

功能描述	设置联机交易结果
输入	终端与后台联机交易得到的返回数据

输出	处理结果

6.8 外设驱动标准 Java 接口

6.8.1 card

6.8.1.1 ATR 类

功能描述	本类用于 IC 卡的 ATR 对象，通过 ATR 可以识别不同的卡类别等。
输入	无
输出	处理结果返回码

6.8.1.1.1 ATR 方法

功能描述	取得所有 ATR 数据
输入	1. atr 2. histBytes
输出	ATR 数据 buffer

6.8.1.1.2 getBytes 方法

功能描述	取得所有 ATR 数据
输入	无
输出	ATR 数据 buffer

6.8.1.1.3 getHistoricalBytes 方法

功能描述	取得包含发卡人、卡类型、操作系统等信息的 TLV 格式的数据,如果通讯协议不支持则返回 null。 Historical Bytes 是用于描述卡片的物理特性或额外信息比如厂商或芯片类型或卡片序列号。 ISO/IEC 7816-4 定义了 Historical Bytes 的结构和含义，他是一个可变长度的 15 个字节长度以内的 byte 数组。 EMV 中虽然规定了 Historical Bytes 的结构，但是并不是所有卡片都会返回该数据。 另外目前 android 的原声 NFC 接口中是含有该接口.
输入	无
输出	数据 buffer

6.8.1.2 card 接口

功能描述	Card 接口提供了一些卡片服务的通用的定义。 通过这个接口，可以实现卡片和读卡设备之间的连接或断开操作，并且获得卡片的 ATR，ID 和状态等基本信息。
输入	无
输出	

6.8.1.2.1 getID 方法

功能描述	返回卡片 ID
输入	无
输出	卡片 ID 的字节流

6.8.1.2.2 getProtocol 方法

功能描述	返回表达当前卡槽和卡片通讯协议的常量。 本方法是个同步方法。当卡槽中有卡片存在，并且成功 open，并得到了卡对象，可以被随时读取。
输入	无
输出	1. 卡片通讯协议的常量： * <dl> * <dd>{@link #PROTOCOL_RFCARD_TYPE_A} <dd>{@link #PROTOCOL_RFCARD_TYPE_B} <dd> * {@link #PROTOCOL_T_0} <dd>{@link #PROTOCOL_T_1} <dd>{@link #PROTOCOL_UNKNOWN} * </dl>

6.8.1.2.3 getCardStatus 方法

功能描述	返回卡操作结果对象的状态。
输入	无
输出	结果

6.8.1.3 CPUCard 接口

功能描述	CPU 卡的接口，通过 APDU 命令和智能卡读卡器之间传递信息。 CPU 智能卡，是 IC 卡的一种，内有微处理器 CPU、存储单元（包括随机存储器 RAM、程序存储器 ROM（FLASH）、用户数据存储器 EEPROM）以及芯片操作系统 COS，通常 CPU 卡内含有随机数发生器，硬件 DES，3DES 加密算法。 CPU 卡既有接触式也有非接触式的，接触式 CPU 卡主要采用两种通讯协议：T=0 和 T=1 通讯协议。T=0 是异步半双工字符传输协议，T=1 是异步半双工块传输协议。 通过接触式设备 SmartCardReaderDevice 对象或非接触式设备 RFCardReaderDevice 对象都有可能得到 CPU 卡对象。
输入	无
输出	无

6.8.1.3.1 transmit 方法

功能描述	在 CPU 卡和 CPU 卡读卡器之间传递信息。 APDU--ApplicationProtocolDataUnit--应用协议数据单元，是智能卡与智能卡读卡器之间传送的信息单元. 如果在发送命令前，需要选卡、唤醒等操作，由底层自动实现。

	如果与读卡器之间断开连接，不能执行 APDU 命令。
输入	1. apdu APDU 数据流
输出	返回的 APDU 数据

6.8.1.3.2 connect 方法

功能描述	连接卡片，并返回卡片的 ATR 信息
输入	无
输出	卡片的 ATR 信息

6.8.1.3.3 disconnect 方法

功能描述	断开卡片
输入	无
输出	无

6.8.1.4 MemoryCard 接口

功能描述	MemoryCard接口提供了 Memory 卡服务的通用定义。 这里没有具体的接口定义，仅用于分类。
输入	无
输出	

6.8.1.5 MifareCard 接口

功能描述	非接触式 Mifare 卡的接口定义。符合 MIFAREI 国际标准，容量为 8K 位.是 memory 卡的一种。
输入	无
输出	无

6.8.1.5.1 verifyKeyA 方法

功能描述	验证给定分区的 Key A。 无论验证失败还是成功，前面卡片被打开的分区均会被关闭。
输入	1. sectorIndex Mifare 卡的分区号 2. key 6 个字节的 key
输出	{@code true} 成功，{@code false} 失败。

6.8.1.5.2 verifyKeyB 方法

功能描述	验证给定分区的 Key B。 无论验证失败还是成功，前面卡片被打开的分区均会被关闭。
输入	1. sectorIndex Mifare 卡的分区号 2. key 6 个字节的 key
输出	{@code true} 成功，{@code false} 失败。

6.8.1.5.3 readBlock 方法

功能描述	读取分区中某个 block 的数据。
输入	1. sectorIndex 分区 ID 2. blockOfSector block 索引号
输出	the data buffer block 的数据

6.8.1.5.4 writeBlock 方法

功能描述	写某个分区的某个 block。
输入	1. sectorIndex 分区号 2. blockOfSector block 索引号 3. buffer 数据流
输出	无

6.8.1.5.5 writeValue 方法

功能描述	往电子钱包里面写入钱包数据和用户数据。
输入	1. sectorIndex 分区号 2. blockOfSector block 索引号 3. value 钱包数据及用户数据
输出	无

6.8.1.5.6 readValue 方法

功能描述	从电子钱包中读取钱包数据和用户数据。
输入	1. sectorIndex 分区号 2. blockOfSector block 索引号
输出	3. 钱包数据及用户数据

6.8.1.5.7 increaseValue 方法

功能描述	设置电子钱包的数据值，做加法。
输入	1. sectorIndex 分区号 2. blockOfSector block 索引号 3. value 钱包数据，加值
输出	{@code true} 成功，{@code false} 失败.

6.8.1.5.8 decreaseValue 方法

功能描述	设置电子钱包的数据值，做减法
输入	1. sectorIndex 分区号 2. blockOfSector block 索引号 3. value 钱包数据，减去值
输出	{@code true} 成功，{@code false} 失败.

6.8.1.6 MifareUltralightCard 接口

功能描述	Mifare UltraLight 又称为 MF0，遵守 ISO14443A 协议，存储容量只有 512bit(Mifare S50 有 8192bit)。 Mifare UltraLight 的卡序列号有 7 个字节，没有密码，不需要验证，有 16 个 BLOCK，且每个 BLOCK 只有 4 个字节，没有电子钱包功能. 适合一次性、不需要回收的低成本的电子票证、景区门票等场合的解决方案。属于 memory card 的一种.
输入	无
输出	无

6.8.1.6.1 read 方法

功能描述	读出 4 个 blocks (16 字节). Mifare 轻型协议每次总是读 4 个 blocks，以减少需要读取整个标签命令的数目.如果读到最后一个可读的 blocks，标记将返回已被包装的前面的 blocks，Mifare 轻型协议标记包含可读 blocks 从 0x00 到 0x0F.所以读到 blocks0x0E 将返回到 blocks0x0E, 0x0F, 0x00, 0x01. Mifare 轻型协议 C 标记包含可读 blocks 从 0x00 到 0x2B.0x2A 将返回到 blocks0x2A, 0x2B, 0x00, 0x01.
输入	1. blockIndex 4 个 blocks 中第一个的索引, 从 0 开始.
输出	4 个 blocksbuffer (16 字节)

6.8.1.6.2 write 方法

功能描述	写一个 blocks (4 bytes). Mifare 轻型协议每次总是写 1 个 blocks, 减少 EEPROM 写周期。
输入	1. blockIndex 写入 blocks 的索引,从 0 开始 2. data 4 字节数据
输出	无

6.8.1.6.3 MoneyValue 接口

功能描述	电子钱包数据结构
输入	无
输出	无

6.8.1.6.4 MoneyValue 方法

功能描述	电子钱包构造函数
输入	userData money
输出	无

6.8.1.6.5 getUserData 方法

功能描述	返回 userdata
输入	无
输出	userdata

6.8.1.6.6 getMoney 方法

功能描述	返回 money
输入	无
输出	money

6.8.1.6.7 SLE4442Card 接口

功能描述	接触式 IC 卡的一种。有三项安全机制用户密码，唯一代码，固化写入。密码若未核对正确，则无法写入数据。写入的数据一经写保护则无法再更改，采用唯一代码作为系统所使用 IC 卡的标识，可避免相同型号的假冒卡闯入系统。 属于 memory card 的一种.
输入	无
输出	无

6.8.1.6.8 connect 方法

功能描述	连接卡片，并返回卡片的 ATR 信息。
输入	无
输出	卡片的 ATR 信息

6.8.1.6.9 disconnect 方法

功能描述	断开卡片。
输入	无
输出	无

6.8.1.6.10 verify 方法

功能描述	使用给定的密钥，验证 memory 卡。
输入	key
输出	{@code true} 成功,{@code false} 失败

6.8.1.6.11 read 方法

功能描述	读取内存卡的数据。
输入	1. area 读取的区域标志，包括以下常量： * <dl> * <dd>{@link #MEMORY_CARD_AREA_MAIN} <dd>{@link #MEMORY_CARD_AREA_PROTECTED} <dd> * {@link #MEMORY_CARD_AREA_SECURITY} * </dl>

	2. address 地址，由 0 开始 3. length 读取数据的长度
输出	读取数据的内容，长度为 length 指定长度。

6.8.1.6.12 write 方法

功能描述	向内存卡写入数据。
输入	1. area 目标的区域标志，包括以下常量： * <dl> * <dd>{@link #MEMORY_CARD_AREA_MAIN} <dd>{@link #MEMORY_CARD_AREA_PROTECTED} <dd> * {@link #MEMORY_CARD_AREA_SECURITY} * </dl> 2. address 地址，由 0 开始 3. data 写入的数据
输出	无

6.8.2 cashdrawer

6.8.2.1 CashDrawerDevice 接口

功能描述	定义了钱箱设备的接口
	设备对象通过<code>POSTerminal</code>的对应方法获得，如下所示：
	<pre>CashDrawerDevice cashDrawerDevice =
	(CashDrawerDevice)
	POSTerminal.getInstance().getDevice("cloudpos.device.cashdrawer");
	</pre>
	其中，"cloudpos.device.cashdrawer"是标识钱箱设备的字符串，由具体的实现定义。
	使用钱箱设备对象控制钱箱的弹开操作。
	为了正常访问钱箱设备，请在 Android Manifest 文件中设置钱箱访问权限，具体如下所示：<pre>
	<uses-permission
	android:name="android.permission.CLOUDPOS_MONEYBOX"
	/> </pre>

输入	无
输出	无

6.8.2.1.1 open 方法

功能描述	打开某个逻辑 ID 的钱箱设备
输入	logicalID 钱箱设备逻辑 ID
输出	无

6.8.2.1.2 kickOut 方法

功能描述	弹出钱箱

输入	无
输出	无

6.8.2.1.3 queryStatus 方法

功能描述	返回钱箱状态
输入	无
输出	{@code 0} 钱箱开启， {@code 1} 钱箱关闭， {@code -1} 钱箱状态未知

6.8.2.2 CashDrawerDeviceSpec 接口

功能描述	定义了对钱箱设备的详细描述。此设备暂无详细描述
输入	无
输出	无

6.8.3 hsm

6.8.3.1 HSMDevice 接口

功能描述	安全模块存放终端所需要的各种证书，主要有四类： 终端所有者证书（Terminal Owner Root Cert）：用于表明表示该终端属于哪个所有者，它是终端中其他根证书的签发人；在导入其他根证书时可以用它来进行验证，包括更新终端所有者证书自己。 应用根证书（App Root Cert）：由终端所有者证书签发；用于验证终端所有应用的根签名。向终端安装的 apk 都必须由该证书签发，或者该证书的子证书签发（子证书需要包含在 apk 中）。 通讯根证书（SSL Comm Root Cert）：由终端所有者证书签发；用于在 SSL 连接时验证通讯服务器. 终端公钥证书（Terminal Pub Key Cert）：终端内部私钥对应的公钥证书；由其他第三方 CA 颁发，用来提供给第三方服务端来确认终端身份。不属于 truststore 的范围，仅仅存放在安全模块中。 安全模块中证书使用别名进行区分。主要提供以下功能： 生成多组 RSA 公私钥对，至少 2048 位；对外部提交的数据进行私钥运算，并返回结果；产生真随机数；证书管理； 加密解密。 硬件安全模块设备对象用来连接并使用硬件安全模块设备的功能。 设备对象通过<code>POSTerminal</code>的对应方法获得，如下所示：HSMDevice hsmDevice =(HSMDevice) POSTerminal.getInstance().getDevice("cloudpos.device.hsm"); 其中，"cloudpos.device.hsm"是标识安全模块设备的字符串，由具体的实现定义。 硬件安全模块设备操作分为只读权限和读写权限。在只读权

	限下，它是非独占设备，可以多个设备对象同时使用；但是，在读写权限下同时只能有一个设备对象打开设备。 为了正常访问硬件安全模块设备，请在 Android Manifest 文件中设置硬件安全模块访问权限，具体如下所示： <uses-permission android:name="android.permission.CLOUDPOS_SAFEMODUAL" />
输入	无
输出	无

6.8.3.1.1 open 方法

功能描述	打开某个逻辑 ID 的安全模块设备。 安全模块设备是一种特殊的独占设备，一旦某个具有读写权限的应用打开本设备后，本设备不能再被其他具有读写权限的应用打开，但是本设备可以被多个具有只读权限的应用同时打开。
输入	logicalID 安全模块逻辑 ID
输出	无

6.8.3.1.2 isTampered 方法

功能描述	检查安全模块是否已经触发。 硬件安全模块有自动保护机制，如果发生对安全模块的攻击，会自动触发自毁并删除所有敏感信息。 该操作需要只读权限。
输入	无
输出	{@code true}触发，{@code false} 未触发.

6.8.3.1.3 generateRandom 方法

功能描述	从安全模块返回真随机数。 该操作只需要只读权限。
输入	length 需要返回随机数的长度 < 64
输出	包含随机数的数据流

6.8.3.1.4 generateKeyPair 方法

功能描述	让安全模块生成密钥对。 该操作需要读写权限。
输入	1. aliasPrivateKey 需要生成的私钥（密钥对）的别名 2. algorithm 密钥对支持的算法 3. keySize 密钥长度，目前只支持 2048 位
输出	无

6.8.3.1.5 injectPublicKeyCertificate 方法

功能描述	注入终端公钥证书。 终端公钥证书一般有终端产生 CSR 后，向 CA 提出证书申请，CA 为终端签发的表明终端身份的证书。 该操作只需要读写权限。
输入	1. alias 证书的别名。 2. aliasPrivateKey 证书对应的私钥别名 3. bufCert 证书数据流。 4. dataFormat 证书数据格式，目前只支持{@link #FORMAT_PEM}
输出	{@code true} 成功。{@code false} 失败

6.8.3.1.6 injectRootCertificate 方法

功能描述	注入根证书接口。终端所有者证书，应用根证书和通讯根证书都是通过本方法注入。 所有根证书必需由终端所有者根证书签发。而且证书的 keyUsage 必须符合下面规则： 1）{@link #CERT_TYPE_TERMINAL_OWNER} 终端所有者根证书的 keyUsage 标志：critical、KeyEncipherment、CertificateSign 和 CRLSign 标识位必须被设置，其他标志不能设置。 2）{@link #CERT_TYPE_APP_ROOT} 终端应用根证书的 keyUsage 标志：critical、DigitalSignature、CertificateSign 标识位必须被设置，其他标识位不能设置 3）{@link #CERT_TYPE_COMM_ROOT} 终端通讯根证书的 keyUsage 标志：DigitalSignature、KeyEncipherment 和 DataEncipherment 标识位必须被设置，其他标识位不能设置。 该操作需要读写权限
输入	1.certType 证书类型：{@link #CERT_TYPE_TERMINAL_OWNER}，{@link #CERT_TYPE_APP_ROOT}或者{@link #CERT_TYPE_COMM_ROOT}. 2. alias 证书别名 3. bufCert 证书数据流 . dataFormat 数据流格式，目前只支持{@link #CERT_FORMAT_PEM}
输出	{@code true} 成功。{@code false} 失败

6.8.3.1.7 getCertificate 方法

功能描述	返回证书数据。通过证书类型及证书别名，可返回找到匹配的唯一一张证书。 该操作需要只读权限

输入	1. certType 证书类型 2. alias 证书别名。 3. dataFormat 数据流格式，目前只支持{@link #CERT_FORMAT_PEM}。
输出	证书数据流

6.8.3.1.8 deleteCertificate 方法

功能描述	删除证书。通过证书类型及证书别名，可找到匹配的唯一一张证书删除。 终端所有者证书和银联的应用根证书不能被删除。 该操作只需要读写权限。
输入	1. certType 证书类型 2. alias 证书别名
输出	{@code true} 成功。{@code false} 失败

6.8.3.1.9 queryCertificates 方法

功能描述	从硬件证书管理与加密运算模块内查询证书。 该操作只需要读权限。
输入	certType 证书类型
输出	证书别名数组

6.8.3.1.10 deleteKeyPair 方法

功能描述	删除终端私钥（密钥对）。 该操作只需要读写权限。
输入	aliasPrivateKey 私钥（密钥对）别名
输出	{@code true} 成功。{@code false} 失败

6.8.3.1.11 generateCSR 方法

功能描述	生成终端公钥的证书签名请求 CSR。 该操作只需要读写权限。
输入	1. aliasPrivateKey 私钥别名 2. subject CSR 中的主体名称等
输出	PEM 格式的 CSR 数据流

6.8.3.1.12 encrypt 方法

功能描述	使用终端私钥加密数据。加密结果默认使用 PKCS1Padding。 该操作只需要只读权限。
输入	1. aigorithm 加密算法 2. aliasPrivateKey 私钥别名 3. bufPlain 加密数据明文
输出	加密结果

6.8.3.1.13 decrypt 方法

功能描述	使用终端私钥解密数据。解密结果默认使用 PKCS1Padding。 该操作只需要只读权限。
输入	1. aigorithm 解密算法 2. aliasPrivateKey 私钥别名 3. bufCipher 解密数据密文
输出	解密结果

6.8.3.1.14 getFreeSpace 方法

功能描述	返回硬件加密接口的剩余或可用空闲空间。 该操作只需要只读权限。
输入	无
输出	空间大小单位为 byte

6.8.3.2 HSMDeviceSpec 接口

功能描述	定义了对安全模块的详细描述。 通过调用{@link #getAlgorithms()}返回安全模块的算法
输入	无
输出	无

6.8.3.2.1 getAlgorithms 方法

功能描述	返回安全模块中的支持的所有加密算法。
输入	无
输出	返回支持的所有加密算法，不支持返回 null。

6.8.4 idcardreader

6.8.4.1 IDCard 类

功能描述	本接口主要用于表示身份证
输入	无
输出	无

6.8.4.1.1 getName 方法

功能描述	获取姓名
输入	无
输出	string 类型

6.8.4.1.2 getSex 方法

功能描述	获取性别

输入	无
输出	string 类型

6.8.4.1.3 getNation 方法

功能描述	获取国籍
输入	无
输出	string 类型

6.8.4.1.4 getBorn 方法

功能描述	获取出生日期
输入	无
输出	string 类型

6.8.4.1.5 getAddress 方法

功能描述	获取地址
输入	无
输出	string 类型

6.8.4.1.6 getIDCardNo 方法

功能描述	获取身份证号
输入	无
输出	string 类型

6.8.4.1.7 getGrantDept 方法

功能描述	获取颁发部门
输入	无
输出	string 类型

6.8.4.1.8 getValidFromDate 方法

功能描述	获取有效期开始日期
输入	无
输出	string 类型

6.8.4.1.9 getValidToDate 方法

功能描述	获取有效期结束日期
输入	无
输出	string 类型

6.8.4.1.10 getReserved 方法

功能描述	获取其他保留信息
输入	无
输出	string 类型

6.8.4.1.11 getPicture 方法

功能描述	获取对加密数据解密后的身份证相片
输入	无
输出	string 类型

6.8.4.2 IDCardReaderDevice 类

功能描述	定义了所有身份证读取设备的接口. 设备对象通过<code>POSTerminal</code>的对应方法获得，如下所示： IDCardReaderDevice idCardReaderDevice =(IDCardReaderDevice) POSTerminal.getInstance().getDevice("cloudpos.device.idcard"); 其中，"cloudpos.device.idcard"是标识非接读卡器的字符串，由具体的实现定义。 身份读取设备对象主要进行身份证卡读取操作。其中等卡及移卡都包括同步和异步两种方式。同步方式会将主线程锁定，直到有结果返回，超时或者被取消。异步方式不会锁定主线程，当有结果时，会回调监听者{@link OperationListener#handleResult(OperationResult) handleResult()}方法。 为了正常访问身份证读取设备，请在 Android Manifest 文件中设置身份证读取设备访问权限，具体如下所示： uses-permission android:name="android.permission.CLOUDPOS_IDCard"
输入	无
输出	无

6.8.4.2.1 open 方法

功能描述	打开某个逻辑 ID 的身份证读取设备。
输入	logicalID 模块的逻辑 ID
输出	无

6.8.4.2.2 listenForCardPresent 方法

功能描述	启动一次扫描身份证的过程。 一次只能查找一张身份证。

	本方法会正确响应{@link #cancelRequest()}方法来取消操作。本操作是个异步调用。当找到身份证后，结果通过操作监听者{@link OperationListener#handleResult(OperationResult) handleResult()}方法返回，通常程序必须定义自己的OperationListener，在回调函数 handleResult()中对返回结果进行处理。如下所示：<pre> OperationListener operationListener = new OperationListener(){@Override public void handleResult(OperationResult result) { // handleResult } }); </pre> 方法通过设置 timeout 来决定最多等待多长时间。请求开始执行的时候 timeout 开始计时。如果 timeout 时间到了，但还没有扫描到卡，也会回调函数 handleResult()。此时 OperationResult 会返回错误：{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何卡片返回，如果 timeout 是{@link TimeConstants#FOREVER FOREVER}，方法会一直等待卡，直到扫描到卡或取消。如果 timeout 是{@link TimeConstants#IMMEDIATE IMMEDIATE}，方法会马上返回。
输入	listener 操作监听者 timeout 最大扫描时间，通过时间常量设定{@link TimeConstants#SECOND SECOND},{@link TimeConstants#MilliSECOND MilliSECOND},{@link TimeConstants#FOREVER FOREVER},{@link TimeConstants#IMMEDIATE IMMEDIATE}。
输出	无

6.8.4.2.3 waitForCardPresent 方法

功能描述	本方法是上述对应的（listenForCardPresent 方法）。 {@link #listenForCardPresent(OperationListener, int)}方法的同步版本。 只有当超时发生或者操作正常完成，本次调用才会返回。 由于带有超时，本方法会响应{@link #cancelRequest()}方法。 如果超时发生，会返回这个操作结果：{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何卡片返回。
输入	timeout 最大扫描时间，通过时间常量设定。
输出	操作结果

6.8.4.3 IDCardReaderDeviceSpec 接口

功能描述	定义了对身份证读取设备的详细描述。 此设备暂无详细描述。
输入	无
输出	无

6.8.4.4 IDCardReaderOperationResult 接口

功能描述	IDCardReaderOperationResult 是被设备产生，用来返回身份证读取设备的操作结果。 {@link OperationResult#getResultCode() getResultCode()} 方法继承至{@link OperationResult OperationResult}的对应方法。 通过{@link OperationResult#getResultCode() getResultCode()}返回成功或失败，返回值参见<code>OperationResult</code>中定义。 通过{@link #getIDCard()}返回身份证上的个人信息。
输入	无
输出	无

6.8.4.4.1 getIDCard 方法

功能描述	返回身份证对象。
输入	无
输出	IDCard

6.8.5 led

6.8.5.1 LEDDevice 接口

功能描述	定义了 LED 设备的接口。 设备对象通过<code>POSTerminal</code>的对应方法获得，如下所示： <pre> LEDDevice lEDDevice = (LEDDevice) POSTerminal.getInstance().getDevice("cloudpos.device.led"); </pre> 其中，"cloudpos.device.led"是标识 LED 设备的字符串，由具体的实现定义。 可以通过以下步骤对 LED 设备对象进行操作: 通过前面的介绍，得到 LED 设备对象。 调用{@link #open(int)},打开成功后，和对应的 LED 设备建立了连接。 调用{@link #turnOn()},可以点亮该 LED 设备，直到程序调用了{@link #turnOff()},灭掉该 LED 设备，或者调用{@link #close()},关闭某个 LED 设备 。 有两种方法可以让某个 LED 灯闪烁，一为同步方法：<code>blink</code>，一为异步方法<code>startBlink</code>,程序可以自行选择闪烁方式。值

	得注意的是，如果选择异步方法，可以通过调用{@link #cancelBlink()}来取消闪烁。同步方法不能取消，会按照设定的开关时间及频次闪烁下去。
	调用{@link #close()},关闭某个 LED 设备。
	父类接口{@link #cancelRequest()},在这里是没有效果的。由于闪烁的异步方法不带有超时概念，和其他设备的异步监听响应机制不同。这里用{@link #cancelBlink()}来取消异步闪烁。
	为了正常访问 LED 设备，请在 Android Manifest 文件中设置 LED 访问权限，具体如下所示：
	<pre><uses-permission
	android:name="android.permission.CLOUDPOS_LED"
	/></pre>

输入	无
输出	无

6.8.5.1.1 open 方法

功能描述	打开 LED 设备,和对应的 LED 设备建立连接。
输入	logicalID 设备逻辑 ID
输出	无

6.8.5.1.2 getLogicalID 方法

功能描述	返回正在使用的 logicalID
输入	无
输出	logicalID 设备逻辑 ID -1 未打开任何 led 灯

6.8.5.1.3 startBlink 方法

功能描述	使某个 led 灯进行闪烁，异步方法，可以被取消。
输入	1. delayTurnOn-指定 Led 灯所需要打开的时间。单位 ms 2. delayTurnOff-指定 Led 灯所需要关闭的时间。单位 ms 3. counts-周期，持续次数。
输出	无

6.8.5.1.4 startBlink 方法

功能描述	使某个 led 灯进行闪烁，异步方法，可以被取消。Led 灯的颜色参见{@link LEDDeviceSpec LEDDeviceSpec}颜色定义。
输入	1. color 指定 Led 灯的颜色。 2. delayTurnOn-指定 Led 灯所需要打开的时间。单位 ms 3. delayTurnOff-指定 Led 灯所需要关闭的时间。单位 ms 4. counts-周期，持续次数。

输出	无

6.8.5.1.5 startBlink 方法

功能描述	使某个 led 灯进行闪烁，异步方法，可以被取消。
输入	1. delayTurnOn-指定 Led 灯所需要打开的时间。单位 ms 2. delayTurnOff-指定 Led 灯所需要关闭的时间。单位 ms
输出	无

6.8.5.1.6 cancelBlink 方法

功能描述	取消 LED 设备闪烁，调用该方法，当前闪烁的 LED 设备会停止闪烁。
输入	无
输出	无

6.8.5.1.7 blink 方法

功能描述	使某个 led 灯进行闪烁，同步方法，不能被取消。
输入	1. delayTurnOn-指定 Led 灯所需要打开的时间。单位 ms 2. delayTurnOff-指定 Led 灯所需要关闭的时间。单位 ms 3. counts-周期，持续次数。
输出	无

6.8.5.1.8 blink 方法

功能描述	使某个 led 灯进行闪烁，同步方法，不能被取消。 Led 灯的颜色参见{@link LEDDeviceSpec LEDDeviceSpec}颜色定义。
输入	1. color -指定 Led 灯的颜色。 2. delayTurnOn-指定 Led 灯所需要打开的时间。单位 ms 3. delayTurnOff-指定 Led 灯所需要关闭的时间。单位 ms 4. counts-周期，持续次数。
输出	无

6.8.5.1.9 getStatus 方法

功能描述	返回 LED 的当前状态。
输入	无
输出	状态常量：{@link #STATUS_OFF}或者{@link #STATUS_ON}。

6.8.5.1.10 turnOn 方法

功能描述	打开 led 灯。 LED 灯会一直亮下去,直到调用{@link #turnOff()}或{@link #close()}

输入	无
输出	无

6.8.5.1.11 turnOff 方法

功能描述	关闭 led 灯. Led 灯会灭掉。
输入	无
输出	无

6.8.5.2 LEDDeviceSpec 接口

功能描述	定义了对 LED 的详细描述。 可以得到终端中关于 LED 定义的信息。
输入	无
输出	无

6.8.5.2.1 getCounts 方法

功能描述	返回可以操作的 LED 灯数量
输入	无
输出	返回 LED 灯的数量。不支持返回 0.

6.8.5.2.2 getColors 方法

功能描述	返回 LED 设备的颜色
输入	logicalID 设备逻辑 ID
输出	int 颜色，参数错误及不支持返回 null。

6.8.5.2.3 canQuickBlink 方法

功能描述	是否支持快速闪烁，500ms 闪烁一次
输入	logicalID 设备逻辑 ID
输出	{@code true} 支持， {@code false} 不支持，参数错误也返回 false.

6.8.5.2.4 canSlowBlink 方法

功能描述	是否支持慢速闪烁，1s 闪烁一次
输入	logicalID 设备逻辑 ID
输出	{@code true} 支持， {@code false} 不支持，参数错误也返回 false.

6.8.6 msr

6.8.6.1 MSRDevice 接口

功能描述	定义了对磁条卡阅读器的操作方法。任何具体的实现都必须实现这个接口。设备对象通过<code>POSTerminal</code>的对应方法获得，如下所示：<pre> MSRDevice msrDevice = (MSRDevice) POSTerminal.getInstance().getDevice("cloudpos.device.msr"); </pre> 其中，"cloudpos.device.msr"是标识磁条卡阅读器的字符串，由具体的实现定义。 磁条卡阅读器设备对象主要进行刷卡操作。其中等待刷卡包括同步和异步两种方式。同步方式会将主线程锁定，直到有结果返回，超时或者被取消。异步方式不会锁定主线程，当有结果时，会回调监听者{@link OperationListener#handleResult(OperationResult) handleResult()}方法。 为了正常访问磁条卡阅读器，请在 Android Manifest 文件中设置磁条卡阅读器访问权限，具体如下所示：<uses-permission android:name="android.permission.CLOUDPOS_MSR"/>
输入	无
输出	无

6.8.6.1.1 open 方法

功能描述	打开磁条卡阅读器的指定卡槽 。 打开成功，设备对象就和相应的磁条卡阅读器的卡槽建立了连接。此后可以进行后面的各项操作。 设备对象去打开某个已经打开（被当前设备对象或其他设备对象）的磁条卡阅读器的卡槽会抛出异常{@link DeviceException#BAD_CONTROL_MODE BAD_CONTROL_MODE}。 设备对象打开磁条卡阅读器的某个卡槽，再打开该磁条卡阅读器的另外一个卡槽，会抛出异常{@link DeviceException#BAD_CONTROL_MODE BAD_CONTROL_MODE}。
输入	logicalID 读卡器逻辑 ID（卡槽 ID）。
输出	无

6.8.6.1.2 listenForSwipe 方法

功能描述	等待用户刷卡。 本操作是个异步调用，调用后立即返回。当用户刷卡发生后，结果通过操作监听者{@link OperationListener#handleResult(OperationResult) handleResult()}方法返回。 本方法会正确响应{@link #cancelRequest()}方法来取消操作。 通常程序必须定义自己的 OperationListener，在回调函数 handleResult()中对返回结果进行处理。如下所示：<pre> OperationListener operationListener = new OperationListener(){@Override public void

	handleResult(OperationResult result) { // handleResult } }); </pre>
	方法通过设置 timeout 来决定最多等待多长时间。请求开始执行的时候 timeout 开始计时。如果 timeout 时间到了，但用户还没有刷卡，也会回调函数 handleResult()。此时 OperationResult 会返回错误：{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何卡片返回，如果 timeout 是{@link TimeConstants#FOREVER FOREVER}，方法会一直等待刷卡，直到刷卡或取消。如果 timeout 是{@link TimeConstants#IMMEDIATE IMMEDIATE}，方法会马上返回。

输入	1. listener 操作监听者。 2. timeout 最大等待时间，通过时间常量设定{@link TimeConstants#SECOND SECOND},{@link TimeConstants#MilliSECOND MilliSECOND},{@link TimeConstants#FOREVER FOREVER},{@link TimeConstants#IMMEDIATE IMMEDIATE}。
输出	无

6.8.6.1.3 waitForSwipe 方法

功能描述	本方法是上述对应的（listenForSwipe）。 {@link #listenForSwipe(OperationListener,int)}方法的同步版本。 只有当超时发生或者操作正常完成，本次调用才会返回。 由于带有超时，本方法会响应{@link #cancelRequest()}方法。 如果超时发生，会返回这个操作结果：信息为{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何卡片返回。
输入	timeout 最大等待时间，通过时间常量设定{@link TimeConstants#SECOND SECOND},{@link TimeConstants#MilliSECOND MilliSECOND},{@link TimeConstants#FOREVER FOREVER},{@link TimeConstants#IMMEDIATE IMMEDIATE}。
输出	操作结果

6.8.6.2 MSRDeviceSpec 接口

功能描述	定义了对磁条卡阅读器的详细描述。 设备暂时没有详细的描述。
输入	无

输出	无

6.8.6.3 MSROperationResult 接口

功能描述	是被磁条卡阅读器设备产生，用来返回磁条卡的操作结果。 {@link OperationResult#getResultCode() getResultCode()}方法继承至{@link OperationResult}的对应方法。 这里通过"ERR_"设置了本设备相关的自定义错误，可以在{@link OperationResult#getResultCode() getResultCode()}返回，通过{@link MSROperationResult#getMSRTrackData() getMSTrackData()}返回磁条卡数据对象。
输入	无
输出	无

6.8.6.3.1 getMSRTrackData 方法

功能描述	返回磁条卡数据。
输入	无
输出	磁条卡数据对象

6.8.6.3.2 MSRTrackData 接口

功能描述	磁道数据对象
输入	无
输出	无

6.8.6.3.3 getTrackData 方法

功能描述	返回磁道信息。 磁道信息参考：ISO 7811-2 and JIS X 6302.
输入	无
输出	磁道信息数据

6.8.6.3.4 getTrackError 方法

功能描述	返回本磁道的错误标识
输入	无
输出	定义的磁道错误常量 #NON_SPECIFIC_ERROR #TRACK_NOT_SUPPORTED #READ_ERROR #PARITY_ERROR #LRC_ERROR

	#NO_DATA #NO_STRIPE

6.8.6.4 MSRUtils 类

功能描述	集成了对磁条卡阅读器设备的使用，用户在不关心接口含义的情况下，可以调用 waitForSwipe 方法，然后继续刷卡操作就可以了
输入	无
输出	无

6.8.6.4.1 waitForSwipe 方法

功能描述	在方法中先调用{@link MSRDevice#open() open()}，然后调用{@link MSRDevice#waitForSwipe() waitForSwipe()}等待用户刷卡。 只有当超时发生或者操作正常完成，本次调用才会返回。 由于带有超时，本方法会响应{@link #cancelRequest()}方法。 如果超时发生，会返回 null，同时关闭该设备。如果返回 MSROperationResult，开发者后续自行关闭该设备。
输入	timeout 最大等待时间，通过时间常量设定{@link TimeConstants#SECOND SECOND},{@link TimeConstants#MilliSECOND MilliSECOND},{@link TimeConstants#FOREVER FOREVER},{@link TimeConstants#IMMEDIATE IMMEDIATE}。
输出	操作结果

6.8.7 pinpad

6.8.7.1 KeyInfo 类

功能描述	为后续的加密操作选择密钥和算法
输入	无
输出	无

6.8.7.2 PINPadDevice 接口

功能描述	定义了 PIN 输入设备的接口。 PIN 输入设备是智能销售点终端的核心组件之一，用于加密持卡人的 PIN、计算交易报文 MAC，加解密数据等，可做为共享资源供终端上所有应用使用。为了唯一标识 PIN 输入设备中的终端主密钥，PIN 输入设备为每一套终端主密钥分配了一个密钥索引号。PIN 输入设备的详细描述参考《银联卡受理终端安全规范第八部分智能销售点终端安全规范》（QCUP 007.8-2013）。

	设备对象通过<code>POSTerminal</code>的对应方法获得，如下所示：<pre> PINPadDevice pinPadDevice = (PINPadDevice) POSTerminal.getInstance().getDevice("cloudpos.device.pinpad"); </pre> 其中，"cloudpos.device.pinpad"是标识 PIN 输入设备的字符串，由具体的实现定义。 使用 PIN 输入设备对象，可以在 PIN 输入设备上显示文本，修改用户密钥，对数据加密，计算 MAC，返回随机数。PIN 输入设备输入 PIN 的操作有两种方式，一种是同步，一种是异步。同步方式会将主线程锁定，直到有结果返回，超时或者被取消。异步方式不会锁定主线程，当有结果时，会回调监听者{@link OperationListener#handleResult(OperationResult) handleResult()}方法。 为了正常访问 PIN 输入设备，请在 Android Manifest 文件中设置 PIN 输入设备访问权限，具体如下所示： <uses-permission android:name="android.permission.CLOUDPOS_PIN_GET_PIN_BLOCK"/>应用程序声明这个权限，可以调用{@link #listenForPinBlock(KeyInfo, String, boolean, OperationListener, int)}及{@link #waitForPinBlock(KeyInfo, String, boolean, int)}方法。<uses-permission android:name="android.permission.CLOUDPOS_PIN_MAC"/>应用程序声明这个权限，可以调用{@link #calculateMac(KeyInfo, int, byte[])}方法。<uses-permission android:name="android.permission.CLOUDPOS_PIN_ENCRYPT_DATA"/>，应用程序声明这个权限，可以调用{@link #encryptData(KeyInfo, byte[])}方法。<uses-permission android:name="android.permission.CLOUDPOS_PIN_UPDATA_USER_KEY"/>，应用程序声明这个权限，可以调用{@link #updateUserKey(int, int, byte[],int, byte[])}及{@link #updateUserKey(int, int, byte[])}方法。 PIN 输入设备可以是外接的，也可以是内置的。对于内置的情况，终端系统提供密码输入界面。
输入	无
输出	无

6.8.7.2.1 open 方法

功能描述	打开某个逻辑 ID 的 PIN 输入设备。
输入	logicalID 设备逻辑 ID
输出	无

6.8.7.2.2 showText 方法

功能描述	在 PIN 输入设备上显示文本
输入	lineIndex 行号 message 本行显示的信息
输出	无

6.8.7.2.3 showText 方法

功能描述	在 PIN 输入设备上显示文本
输入	1. lineIndex 行号 2. message 本行显示的信息 3. voicePrompt 是否提示声音
输出	无

6.8.7.2.4 clearText 方法

功能描述	清空文本显示。
输入	无
输出	无

6.8.7.2.5 updateUserKey 方法

功能描述	修改用户密钥，一般用于主密钥/会话密钥算法。 用来更新的用户密钥数据是密文，是由给定的主密钥加密，因此更新时需要指定所使用的主密钥索引。 用户密钥约定使用的是 3des 算法
输入	1. masterKeyID 用来解密的主密钥索引。 2. userKeyType 需要更新的用户密钥类型。 3. cipherNewUserKey 被主密钥加过密的用户密钥数据。 4. checkType 校验位校验类型：{@link #CHECK_TYPE_NONE}, {@link #CHECK_TYPE_CUP }。 5. checkValue 校验 pinpad。
输出	无

6.8.7.2.6 updateUserKey 方法

功能描述	修改用户密钥，一般用于主密钥/会话密钥算法。 用来更新的用户密钥数据是密文，是由给定的主密钥加密，因此更新时需要指定所使用的主密钥索引。 用户密钥约定使用的是 3des 算法
输入	1. masterKeyID 用来解密的主密钥索引。 2. userKeyType 需要更新的用户密钥类型。 3. cipherNewUserKey 被主密钥加过密的用户密钥数据。 4. cipherNewUserKey 被主密钥加过密的用户密钥数据

输出	无

6.8.7.2.7 encryptData 方法

功能描述	按照 KeyInfo 中选定的密钥进行数据加密，通常用于计算用户数据等操作。
输入	1. keyInfo ： 设置加密配置信息。 2. plain ： 需要加密的明文
输出	加完密的密文

6.8.7.2.8 listenForPinBlock 方法

功能描述	让用户输入 PIN，并且按照 KeyInfo 中选定的密钥进行 PIN block 的加密。最后返回加密结果。 本方法是一个异步方法，应用程序等待 PIN 输入设备输入命令后，终端通过操作监听者{@link OperationListener#handleResult(OperationResult) handleResult()}方法返回结果 每个应用程序必须定义自己的 OperationListener，在回调函数 handleResult()中对返回结果进行处理。如下所示： 〈pre〉 OperationListener operationListener = new OperationListener(){@Override public void handleResult(OperationResult result) { // handleResult } }); 方法通过设置 timeout 来决定最多等待多长时间。请求开始执行的时候 timeout 开始计时。如果 timeout 时间到了，但还没有任何输入，也会回调函数 handleResult()。此时 OperationResult 会返回错误：{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何数据返回。如果 timeout 是{@link TimeConstants#FOREVER FOREVER}，方法会一直等待，直到有数据返回或取消。如果 timeout 是{@link TimeConstants#IMMEDIATE IMMEDIATE}，方法会马上返回。本方法会正确响应{@link #cancelRequest()}方法来取消操作。
输入	1. keyInfo 密钥和算法配置信息。 2. pan 用户卡号的 ASCII 字符串。 3. voicePrompt 是否语音提示用户。 4. listener 本操作的动作监听者。 5. timeout 等待用户输入的时间。
输出	无

6.8.7.2.9 waitForPinBlock 方法

功能描述	本方法是上述对应的（listenForPinBlock 方法）。{@link #listenForPinBlock(KeyInfo, String, boolean, OperationListener, int)}方法的同步版本。只有当超时发生或者操作正常完成，本次调用才会返回。由于带有超时，

	本方法会响应{@link #cancelRequest()}方法。如果超时发生，会返回这个操作结果：{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何数据返回。
输入	1. keyInfo 密钥和算法配置信息。 2. pan 用户卡号的 ASCII 字符串。 3. voicePrompt 是否语音提示用户。 4. timeout 等待用户输入 PIN 的时间。
输出	PIN block 的密文。

6.8.7.2.10 calculateMac 方法

功能描述	按照 KeyInfo 中指定的密钥进行计算 MAC。 这里 KeyInfo 中，algorithm 一般不需要指定。具体的 MAC 算法由 macFlag 参数指定。
输入	1. keyInfo 指定计算 MAC 的密钥。 2. macFlag 计算 MAC 的算法 * * X9.19 算法，后补 80：{@link AlgorithmConstants#ALG_MAC_METHOD_X919_80}。 * 银联 ECB 算法：{@link AlgorithmConstants#ALG_MAC_METHOD_ECB}。 * X9.19 算法（不足后补 0x00）：{@link AlgorithmConstants#ALG_MAC_METHOD_X919_X00}移动支付项目使用。 * 中总行扩展算法：{@link AlgorithmConstants#ALG_MAC_METHOD_BOCE}。 * X9.19 算法，后补 00：{@link AlgorithmConstants#ALG_MAC_METHOD_X919_00}。 * 异或后 3DES：{@link AlgorithmConstants#ALG_MAC_METHOD_XOR_3DES}。 * X9.9：{@link AlgorithmConstants#ALG_MAC_METHOD_X99}。 * 3. plain 数据明文
输出	MAC

6.8.7.2.11 setPINLength 方法

功能描述	设置可输入 PIN 的长度限制。
输入	1. minLen 最短有效 PIN 长度。 2. maxLen 最长有效 PIN 长度。
输出	无

6.8.7.2.12 getSN 方法

功能描述	返回 PINPad 序列号
输入	无
输出	PINPad 序列号

6.8.7.2.13 getRandom 方法

功能描述	返回随机数
输入	随机数的长度
输出	随机数 buffer 流

6.8.7.3 PINPadDeviceSpec 接口

功能描述	定义了对 PIN 输入设备的详细描述。 可以得到终端中关于 PINPad 的定义的信息。
输入	无
输出	无

6.8.7.3.1 getCounts 方法

功能描述	可支持的最大 PIN 输入设备数量
输入	无
输出	返回支持的数量，不支持返回 0

6.8.7.3.2 isInternal 方法

功能描述	是否是内置 PIN 输入设备，如果是内置，那么系统提供密码输入界面。
输入	logicalID 设备逻辑 ID，默认 1
输出	{@code true} 内置 PIN 输入设备，{@code false} 外置 PIN 输入设备，参数错误及不支持也返回 false.

6.8.7.3.3 canGetRandom 方法

功能描述	是否支持获取随机数
输入	logicalID 设备逻辑 ID，默认 1
输出	{@code true} 支持取随机数， {@code false} 不支持取随机数，参数错误也返回 false.

6.8.7.3.4 canShowText 方法

功能描述	是否支持显示文本
输入	logicalID 设备逻辑 ID，默认 1
输出	{@code true} 支持， {@code false} 不支持，参数错误也返回 false.

6.8.7.4 PINPadOperationResult 接口

功能描述	PIN 输入设备的操作结果对象。 {@link OperationResult#getResultCode() getResultCode()}方法继承至{@link OperationResult}的对应方法。 通过{@link OperationResult#getResultCode() getResultCode()}返回 PIN 输入设备的结果状态，通过{@link PINPadOperationResult#getEncryptedPINBlock() getEncryptedPINBlock()}返回加过密的 PIN Block。
输入	无
输出	无

6.8.7.4.1 getEncryptedPINBlock 方法

功能描述	返回加过密的 PIN Block
输入	无
输出	结果 buffer 流

6.8.8 printer

6.8.8.1 Format 类

功能描述	打印格式根接口，程序定义的打印格式必须实现该接口
输入	无
输出	无

6.8.8.1.1 setParameter 方法

功能描述	以键-值对的方式设置打印格式. 目前定义的键-值如下所示： 打印浓度（density）：对应的值为：<code>light</code>(淡)，<code>medium</code>(中)，<code>dark</code>(深)。对字符，图形及条码打印都有效。 加粗（bold）：对应的值为：<code>true</code>(是)，<code>false</code>(否)。其中，<code>true</code>，<code>false</code>，不区分大小写。对字符打印有效。 反白（reverse）：对应的值为：<code>true</code>(是)，<code>false</code>(否)。其中，<code>true</code>，<code>false</code>，不区分大小写。对字符打印有效。 上下倒置（inversion）：对应的值为：<code>true</code>(是)，<code>false</code>(否)。其中，<code>true</code>，<code>false</code>，不区分大小写。对字符打印有效。 删除线（line-through）：对应的值为：<code>1</code>(连续的删除线)，<code>2</code>(断开的删除线)。对字符打印有效。

	大小（size）：对应的值为：<code>extra-small</code>(特小)，<code>small</code>(小)，<code>medium</code>(中)，<code>large</code>(大)，<code>extra-large</code>(特大)。对字符打印有效。 对齐方式(align)：对应的值为：<code>left</code>(靠左)，<code>right</code>(靠右)，<code>center</code>(居中)。对字符及图形打印有效。 斜体（italic）：对应的值为：<code>true</code>(是)，<code>false</code>(否)。其中，<code>true</code>，<code>false</code>，不区分大小写。对字符打印有效。 HRI 字符的打印位置（HRI-location）：对应的值为：<code>none</code>(不打印)，<code>up</code>(条码上方)，<code>down</code>(条码下方)，<code>up-down</code>(条码上下方)。对条码打印有效。
输入	1. key　　打印格式主键 2. value　打印格式值
输出	无

6.8.8.1.2　getParameter 方法

功能描述	返回打印格式.
输入	key 传入主键
输出	打印格式

6.8.8.1.3　remove 方法

功能描述	移除某一个参数
输入	key 传入主键
输出	无

6.8.8.1.4　clear 方法

功能描述	清除所有打印格式
输入	无
输出	无

6.8.8.2　PrinterDevice 类

功能描述	定义了对打印机的操作方法。任何具体的实现都必须实现这个接口。设备对象通过<code>POSTerminal</code>的对应方法获得，如下所示：<pre> PrinterDevice printerDevice = (PrinterDevice) POSTerminal.getInstance().getDevice("cloudpos.device.printer"); </pre> 其中，"cloudpos.device.printer"是标识磁条卡读卡器的字符串，由具体的实现定义。使用打印机设备对象可以打印文本，图片，条码，二维码，并且可以发送 ESC 指令。为了

	正常访问打印机设备，请在 Android Manifest 文件中设置打印机访问权限，具体如下所示：〈pre〉<uses-permission android:name="android.permission.CLOUDPOS_PRINTER"/〉〈/pre〉
输入	无
输出	无

6.8.8.2.1 open 方法

功能描述	打开某个逻辑 ID 的打印机。 打开成功，设备对象就和相应的逻辑 ID 的打印机建立了连接。此后可以进行后面的各项操作。设备对象去打开某个已经打开（被当前设备对象或其他设备对象）的逻辑 ID 的打印机会抛出异常{@link DeviceException#BAD_CONTROL_MODE BAD_CONTROL_MODE}。设备对象打开某个逻辑 ID 的打印机，再打开另外一个逻辑 ID 的打印机，会抛出异常{@link DeviceException#BAD_CONTROL_MODE BAD_CONTROL_MODE}。
输入	logicalID 打印机逻辑 ID
输出	无

6.8.8.2.2 printText 方法

功能描述	打印字符串。
输入	message 打印的字符串数据。
输出	无

6.8.8.2.3 printText 方法

功能描述	打印字符串。 通过 format 对象来控制打印字符串的格式。如果通过这个接口传入了 format 对象，那么打印机后续的打印也按照这个 format 对象所包含的格式来执行。 如果有新的带有 format 对象的接口被调用，那么会合并新的格式。打印机后续的打印也按照新的合并后的格式执行。 合并的原则是：新的 format 对象中存在旧的 format 对象中不存在的格式，那么该格式会包含进去；新旧 format 中都存在的，用新的格式替换旧的。
输入	format 用于控制字符串格式。 message 打印的字符串数据。
输出	无

6.8.8.2.4 printBitmap 方法

功能描述	打印图片。

输入	bitmap。
输出	无

6.8.8.2.5 printBitmap 方法

功能描述	打印图片。 通过 format 对象来控制打印图片的格式。如果通过这个接口传入了 format 对象，那么打印机后续的打印也按照这个 format 对象所包含的格式来执行。 如果有新的带有 format 对象的接口被调用，那么会合并新的格式。打印机后续的打印也按照新的合并后的格式执行。 合并的原则是：新的 format 对象中存在旧的 format 对象中不存在的格式，那么该格式会包含进去；新旧 format 中都存在的，用新的格式替换旧的。
输入	1. bitmap。 2. format 参考{@link Format Format}中的定义。
输出	无

6.8.8.2.6 printBarcode 方法

功能描述	打印条码或者二维码。 通过 format 对象来控制打印条码的格式。如果通过这个接口传入了 format 对象，那么打印机后续的打印也按照这个 format 对象所包含的格式来执行。 如果有新的带有 format 对象的接口被调用，那么会合并新的格式。打印机后续的打印也按照新的合并后的格式执行。 合并的原则是：新的 format 对象中存在旧的 format 对象中不存在的格式，那么该格式会包含进去；新旧 format 中都存在的，用新的格式替换旧的。
输入	1. format 条码格式，参考{@link Format Format}中的定义。 2. barcodeType，见本接口定义的常量。 3. barcode 条码内容。
输出	无

6.8.8.2.7 sendESCCommand 方法

功能描述	通用发送 ESC 指令，并获得可能的返回结果。 打印指令的详细定义由厂商给出。
输入	esc ESC 指令数据
输出	发送结果返回值

6.8.8.2.8 cutPaper 方法

功能描述	切纸，但仅仅是支持的打印机才会有此功能。
输入	无

输出	无

6.8.8.2.9 queryStatus 方法

功能描述	查询打印机纸张状态
输入	无
输出	{@code -101} 缺纸 ,{@code 1}打印机状态正常 ,{@code -102} 打印机异常 .

6.8.8.2.10 getDefaultParameters 方法

功能描述	返回默认的打印格式。
输入	无
输出	打印格式

6.8.8.3 PrinterDeviceSpec 接口

功能描述	定义了对打印机的详细描述。 可以得到终端中关于打印机定义的信息。
输入	无
输出	无

6.8.8.3.1 getCounts 方法

功能描述	返回当前终端有几个打印机。
输入	无
输出	终端中打印机个数，不支持返回 0

6.8.8.3.2 canSetDensity 方法

功能描述	是否支持打印浓度的设定
输入	logicalID 打印机逻辑 ID，默认为 0
输出	{@code true} 支持，{@code false}不支持，参数错误也返回 false

6.8.8.3.3 getBaseFontHeight 方法

功能描述	返回标准字体的点数宽度
输入	logicalID 打印机逻辑 ID，默认为 0
输出	打印机的标准字体的高度，参数错误及不支持返回 0

6.8.8.3.4 getWidth 方法

功能描述	返回可打印的最大宽度
输入	logicalID 打印机逻辑 ID，默认为 0

输出	打印机的最大显示宽度，参数错误及不支持返回 0

6.8.8.3.5 getBarCodeFormat 方法

功能描述	返回可打印 barcode 的格式。
输入	logicalID 打印机逻辑 ID，默认为 0
输出	可打印条码的格式，参数错误及不支持返回 null

6.8.8.3.6 canCutPaper 方法

功能描述	打印机是否能切纸
输入	logicalID 打印机逻辑 ID，默认为 0
输出	{@code true} 支持切纸功能，{@code false} 不支持，参数错误也返回 false

6.8.8.4 PrinterOperationResult 接口

功能描述	打印机的操作结果对象，暂时未使用。
输入	无
输出	无

6.8.9 rfcardreader

6.8.9.1 RFCardReaderDevice 接口

功能描述	定义了所有非接触式 IC 卡阅读器的接口. 设备对象通过<code>POSTerminal</code>的对应方法获得，如下所示：<pre> RFCardReaderDevice rFCardReaderDevice = (RFCardReaderDevice) POSTerminal.getInstance().getDevice("cloudpos.device.contactlesscard"); </pre> 其中，"cloudpos.device.contactlesscard"是标识非接读卡器的字符串，由具体的实现定义。非接触式 IC 卡阅读器对象主要进行非接卡读卡操作。其中等卡及移卡都包括同步和异步两种方式。同步方式会将主线程锁定，直到有结果返回，超时或者被取消。异步方式不会锁定主线程，当有结果时，会回调监听者 {@link OperationListener#handleResult(OperationResult) handleResult()}方法。为了正常访问非接触式 IC 卡阅读器，请在 Android Manifest 文件中设置非接触式 IC 卡阅读器访问权限，具体如下所示： <uses-permission android:name="android.permission.CLOUDPOS_CONTACTLESS_CARD"/>
输入	无
输出	无

6.8.9.1.1 open 方法

功能描述	打开某个逻辑 ID 的非接触式 IC 卡阅读器，并指定模式。只有符合模式的卡会被发现。
输入	1. logicalID 读卡器逻辑 ID 2. mode 通讯模式
输出	空

6.8.9.1.2 getMode 方法

功能描述	返回当前通讯模式。
输入	无
输出	以下常量： {@link #MODE_AUTO MODE_AUTO} {@link #MODE_NFC_PASSIVE MODE_NFC_PASSIVE} {@link #MODE_NFC_ACTIVE MODE_NFC_ACTIVE} {@link #MODE_ISO14443_TYPE_A MODE_ISO14443_TYPE_A} {@link #MODE_ISO14443_TYPE_B MODE_ISO14443_TYPE_B} {@link #MODE_ISO15693 MODE_ISO15693} {@link #MODE_MIFARE MODE_MIFARE} {@link #MODE_FELICA MODE_FELICA}

6.8.9.1.3 setSpeed 方法

功能描述	设置通讯速率参数：
输入	无
输出	无

6.8.9.1.4 getsetSpeed 方法

功能描述	返回通讯速率参数：
输入	无
输出	无

6.8.9.1.5 listenForCardPresent 方法

功能描述	按照非接卡的通讯参数设定和模式设定，启动一次扫描非接卡的过程。本操作是个异步调用。当找到非接卡后，结果通过操作监听者 {@link OperationListener#handleResult(OperationResult) handleResult()}方法返回。根据底层定义只能返回一张卡片。结果可以通过 {@link RFCardReaderOperationResult#getCard()}获得。如果读卡设备上放入多张卡片，可能会有以下两种实现方式：返回错误： {@link RFCardReaderOperationResult#ERR_MULTI_CARD

	ERR_MULTI_CARD}，不返回任何卡片。 返回某一卡片。 本方法会按照{@link #setSpeed(int) setSpeed(int)}定义的参数扫描卡片。如果没有定义，本方法会按照读卡器默认参数扫描。本方法会正确响应{@link #cancelRequest()}方法来取消操作。通常程序必须定义自己的OperationListener，在回调函数 handleResult()中对返回结果进行处理。如下所示：<pre> OperationListener operationListener = new OperationListener(){@Override public void handleResult(OperationResult result) { // handleResult } }); </pre>方法通过设置 timeout 来决定最多等待多长时间。请求开始执行的时候 timeout 开始计时。如果 timeout 时间到了，但还没有扫描到卡，也会回调函数 handleResult()。此时 OperationResult 会返回错误:{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何卡片返回如果 timeout 是{@link TimeConstants#FOREVER FOREVER}，方法会一直等待，直到扫描到卡或取消。如果 timeout 是{@link TimeConstants#IMMEDIATE IMMEDIATE}，方法会马上返回。
输入	1. listener 操作监听者。 2. timeout 最大等待时间，通过时间常量设定{@link TimeConstants#SECOND SECOND},{@link TimeConstants#MilliSECOND MilliSECOND}, {@link TimeConstants#FOREVER FOREVER},{@link TimeConstants#IMMEDIATE IMMEDIATE}。
输出	无

6.8.9.1.6 waitForCardPresent 方法

功能描述	本方法是上述对应的（listenForCardPresent）{@link #listenForCardPresent(OperationListener, int)}方法的同步版本。 只有当超时发生或者操作正常完成，本次调用才会返回。 由于带有超时，本方法会响应{@link #cancelRequest()}方法 如果超时发生，会返回这个操作结果：状态为{@link OperationResult#EVT_ERROR EVT_ERROR}，信息为{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何卡片返回。
输入	timeout 最大扫描时间，通过时间常量设定。
输出	操作结果

6.8.9.1.7 listenForCardAbsent 方法

功能描述	监听卡片移除动作。 本操作是个异步调用。当用户移除卡发生后，结果通过操作监听者 {@link OperationListener#handleResult(OperationResult) handleResult()}方法返回。 本方法会正确响应{@link #cancelRequest()}方法来取消操作。通常程序必须定义自己的 OperationListener，在回调函数 handleResult()中对返回结果进行处理。如下所示：<pre> OperationListener operationListener = new OperationListener(){@Override public void handleResult(OperationResult result) { // handleResult } }); </pre>方法通过设置 timeout 来决定最多等待多长时间。请求开始执行的时候 timeout 开始计时。如果 timeout 时间到了，但用户还没有移除卡，也会回调函数 handleResult()。此时 OperationResult 会返回错误：{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何卡片返回如果 timeout 是{@link TimeConstants#FOREVER FOREVER}，方法会一直等待移除卡，直到移除卡或取消。如果 timeout 是{@link TimeConstants#IMMEDIATE IMMEDIATE}，方法会马上返回。
输入	1. listener 操作监听者。 2. timeout 最大等待时间，通过时间常量设定{@link TimeConstants#SECOND SECOND},{@link TimeConstants#MilliSECOND MilliSECOND},{@link TimeConstants#FOREVER FOREVER},{@link TimeConstants#IMMEDIATE IMMEDIATE}。
输出	无

6.8.9.1.8 waitForCardAbsent 方法

功能描述	本方法是上述对应的(listenForCardAbsent). {@link #listenForCardAbsent(OperationListener, int)}方法的同步版本。 只有当超时发生或者操作正常完成，本次调用才会返回。 由于带有超时，本方法会响应{@link #cancelRequest()}方法。 如果超时发生，会返回这个操作结果：{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何卡片返回。
输入	timeout 最大等待时间，通过时间常量设定。

输出	操作结果

6.8.9.2 RFCardReaderDeviceSpec 接口

功能描述	定义了对非接卡读卡器的详细描述。 可以得到终端中关于非接触式 IC 卡阅读器定义的信息。
输入	无
输出	无

6.8.9.2.1 isRemovable 方法

功能描述	是否支持等待卡片移除.
输入	无
输出	{@code true} 支持，{@code false} 不支持。

6.8.9.2.2 getSupportedModes 方法

功能描述	返回非接卡模式，模式定义见{@link RFCardReaderDevice RFCardReaderDevice}常量部分定义。
输入	无
输出	返回模式，不支持返回 null

6.8.9.3 RFCardReaderOperationResult 接口

功能描述	是被非接卡读卡设备产生，用来返回非接卡的操作结果。{@link OperationResult#getErrorCode() getStatus()}方法继承至{@link OperationResult OperationResult}的对应方法。 这里通过"ERR_"设置了本设备相关的自定义错误，可以在{@link OperationResult#getResultCode() getResultCode()}返回。通过{@link RFCardReaderOperationResult#getCard() getCard()}返回非接卡数据对象。得到卡对象后，应用程序可以自行区分不同类别的卡，进行卡的后续操作。一般返回的卡类型为{@link CPUCard CPUCard},{@link MifareCard MifareCard},{@link MifareUltralightCard MifareUltralightCard}, 其中后两种属于<code>MemoryCard</code>
输入	无
输出	无

6.8.9.3.1 getCard 方法

功能描述	返回扫描到的卡片。
输入	无
输出	扫描到的非接卡片

6.8.10 seconddarydisplay

6.8.10.1 SecondaryDisplayDevice 接口

功能描述	定义了客显设备的接口。
	设备对象通过<code>POSTerminal</code>的对应方法获得，如下所示：<pre>
	SecondaryDisplayDevice secondaryDisplayDevice = (SecondaryDisplayDevice) POSTerminal.getInstance().getDevice("cloudpos.device.secondaryDisplay"); </pre>
	其中，"cloudpos.device.secondaryDisplay"是标识客显设备的字符串，由具体的实现定义。使用客显设备对象，可以将内容以图片的形式显示在客显屏幕上。也可以让客显屏发出声音。为了正常访问客显设备，请在 Android Manifest 文件中设置客显设备访问权限，具体如下所示：
	<uses-permission android:name="android.permission.CLOUDPOS_CUSTOMER_DISPLAY"/>

输入	无
输出	无

6.8.10.1.1 open 方法

功能描述	打开某个逻辑 ID 的客显服务。
输入	logicalID 客显模块逻辑 ID
输出	无

6.8.10.1.2 setBackgroundColor 方法

功能描述	设置客显设备背景颜色。
输入	color 颜色值
输出	无

6.8.10.1.3 display 方法

功能描述	设置客显设备背景图片。
输入	1. bitmap 位图 2. offsetX 图片显示位置：距离左上角的 x 方向位置 3. offsetY 图片显示位置：距离左上角 y 方向位置
输出	无

6.8.10.1.4 display 方法

功能描述	设置客显设备背景图片。
输入	bitmap 位图

输出	无

6.8.10.1.5 beep 方法

功能描述	让客显设备发声音。
输入	无
输出	无

6.8.10.1.6 resetDisplay 方法

功能描述	恢复客显设备默认设置。
输入	无
输出	无

6.8.10.2 SecondaryDisplayDeviceSpec 接口

功能描述	定义了对客显设备的详细描述。可以得到终端中关于客显定义的信息。
输入	无
输出	无

6.8.10.2.1 getWidth 方法

功能描述	返回客显屏幕的宽度。
输入	无
输出	宽度，不支持返回 0

6.8.10.2.2 getHeight 方法

功能描述	返回客显屏幕的高度。
输入	无
输出	高度，不支持返回 0

6.8.11 serialport

6.8.11.1 SerialPortDevice 接口

功能描述	定义了串口设备的接口。
	设备对象通过<code>POSTerminal</code>的对应方法获得，如下所示：<pre>
	SerialPortDevice serialPortDevice =
	(SerialPortDevice)
	POSTerminal.getInstance().getDevice("cloudpos.device.serialport"); </pre>
	其中，"cloudpos.device.serialport"是标识串口设备的字符串，由具体的实现定义。可以通过以下步骤对串口设备对

	象进行操作: 通过前面的介绍，得到串口设备对象。调用{@link #open(int)},打开成功后，和对应的串口设备建立了连接。调用{@link #write(byte[], int, int)()},可以将数据 buffer 流通过该打开的串口设备写到连接到该串口的某个设备中 。有两种方法可以读取串口中传过来数据，一为同步方法：{@link #waitForRead(int, int)},一为异步方法{@link #listenForRead(int, OperationListener, int)},程序可以根据自身的业务逻辑自己选择哪种方式。 值得注意的是，如果选择异步方法，可以通过调用{@link #cancelRequest()}来取消。同步方法不能取消。这两种方法都定义了超时，因此即使不取消，在超时到来时，不管有没有数据过来，都会返回结果。调用{@link #close()},关闭某个串口设备 。为了正常访问串口设备，请在 Android Manifest 文件中设置串口访问权限，具体如下所示： <uses-permission android:name="android.permission.CLOUDPOS_SERIAL"/>
输入	无
输出	无

6.8.11.1.1 open 方法

功能描述	打开某个逻辑 ID 的串口设备
输入	logicID 串口设备逻辑 ID
输出	无

6.8.11.1.2 getBaudRate 方法

功能描述	返回串口波特率
输入	无
输出	integer 波特率

6.8.11.1.3 getDataBits 方法

功能描述	返回当前数据位。
输入	无
输出	SerialPortDevice中定义的如下常量：DATABITS_5, DATABITS_6, DATABITS_7, or DATABITS_8

6.8.11.1.4 getStopBits 方法

功能描述	返回停止位。
输入	无
输出	停止位是一下常量： STOPBITS_1, STOPBITS_2, or STOPBITS_1_5

6.8.11.1.5 getParity 方法

功能描述	返回奇偶校验位
输入	无
输出	如下常量： PARITY_NONE, PARITY_ODD, or PARITY_EVEN

6.8.11.1.6 changeRTS 方法

功能描述	设置 RTS
输入	<code>true</code>设置 RTS, <code>false</code>清楚 RTS
输出	无

6.8.11.1.7 retrieveRTS 方法

功能描述	返回当前 RTS 状态
输入	无
输出	true, RTS 已经被设置 false, RTS 已被清除

6.8.11.1.8 retrieveCTS 方法

功能描述	返回 CTS 状态
输入	无
输出	true, CTS 已经被设置，false, CTS 已经被清除

6.8.11.1.9 changeFlowControlMode 方法

功能描述	设置流控
输入	flowControl 在SerialPortDevice中定义的常量： FLOWCONTROL_NONE - 没有流控 FLOWCONTROL_RTSCTS_IN - RTS/CTS 输入硬件流控 FLOWCONTROL_RTSCTS_OUT - RTS/CTS 输出 硬件流控 FLOWCONTROL_RTSCTS_IN_OUT - RTS/CTS 双向硬件流控 FLOWCONTROL_XONXOFF_IN - XON/XOFF 输入软流控 FLOWCONTROL_XONXOFF_OUT - XON/XOFF 输出软流控 FLOWCONTROL_XONXOFF_IN_OUT - XON/XOFF 输入/输出 软流控
输出	无

6.8.11.1.10 getFlowControlMode 方法

功能描述	返回当前的流控设置

输入	无
输出	返回SerialPortDevice中定义的常量： FLOWCONTROL_NONE - 没有流控 FLOWCONTROL_RTSCTS_IN - RTS/CTS 输入硬件流控 FLOWCONTROL_RTSCTS_OUT - RTS/CTS 输出 硬件流控 FLOWCONTROL_RTSCTS_IN_OUT - RTS/CTS 双向硬件流控 FLOWCONTROL_XONXOFF_IN - XON/XOFF 输入软流控 FLOWCONTROL_XONXOFF_OUT - XON/XOFF 输出软流控 FLOWCONTROL_XONXOFF_IN_OUT - XON/XOFF 输入/输出 软流控

6.8.11.1.11 changeSerialPortParams 方法

功能描述	修改串口参数。
输入	baudrate 波特率 dataBits 数据位 DATABITS_5 - 5 bits DATABITS_6 - 6 bits DATABITS_7 - 7 bits DATABITS_8 - 8 bits stopBits 停止位 STOPBITS_1 - 1 stop bit STOPBITS_2 - 2 stop bits STOPBITS_1_5 - 1.5 stop bits parity 奇偶校验位 PARITY_NONE - no parity PARITY_ODD - odd parity PARITY_EVEN - even parity
输出	无

6.8.11.1.12 write 方法

功能描述	将指定<code>data</code>数组中从偏移量<code>offset</code>开始的<code>len</code>个字节写入串口。
输入	1. data 数据 2. offset 数据中的起始偏移量。 3. len 要写入的字节数。
输出	无

6.8.11.1.13 listenForRead 方法

功能描述	从包含的输入流中将最多<code>len</code>个字节读入一个<code>data</code>数组中。尽量读取<code>len</code>个字节，但读取的字节数可能少于<code>len</code>个，也可能为零。以整数形式返回实际读取的字节数。如果

	<code>len</code>为零，则不读取任何字节并返回 0；否则，尝试读取至少一个字节。如果因为流位于文件未尾而没有字节可用，则返回值-1；否则，至少读取一个字节并将其存储到<code>data</code>中。
	将读取的第一个字节存储到元素<code>data[offset]</code>中，将下一个字节存储到<code>data[offset+1]</code>中，依此类推。读取的字节数至多等于<code>len</code>。设 k 为实际读取的字节数；这些字节将存储在<code>data[offset]</code>到<code>data[offset+k-1]</code>的元素中，<code>data[offset+k]</code>到<code>data[offset+len-1]</code>的元素不受影响。
	在所有情况下，<code>data[0]</code>到<code>data[offset]</code>的元素和<code>data[offset+len]</code>到<code>data[data.length-1]</code>的元素都不受影响。本方法会正确响应{@link #cancelRequest()}方法来取消操作。本方法是一个异步方法，应用程序发出读取命令后，终端通过操作监听者{@link OperationListener#handleResult(OperationResult) handleResult()}方法返回结果。每个应用程序必须定义自己的 OperationListener，在回调函数 handleResult()中对返回结果进行处理。如下所示：<pre>
	OperationListener operationListener = new OperationListener(){
	@Override
	public void handleResult(OperationResult result) {
	// handleResult } });
	方法通过设置 timeout 来决定最多等待多长时间。请求开始执行的时候 timeout 开始计时。
	如果 timeout 时间到了，但还没有数据读到，也会回调函数 handleResult()。此时 OperationResult 会返回错误:{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何数据返回，如果 timeout 是{@link TimeConstants#FOREVER FOREVER}，方法会一直等待，直到有数据返回或取消。如果 timeout 是{@link TimeConstants#IMMEDIATE IMMEDIATE}，方法会马上返回。

输入	1. len 需要读取的最大长度 2. listener 操作监听者。 3. timeout 最大等待时间，通过时间常量设定{@link TimeConstants#SECOND SECOND},{@link TimeConstants#MilliSECOND MilliSECOND},{@link TimeConstants#FOREVER FOREVER},{@link TimeConstants#IMMEDIATE IMMEDIATE}。

输出	无

6.8.11.1.14 waitForRead 方法

功能描述	本方法是上述对应的（listenForRead）{@link #listenForRead(int, OperationListener, int)}方法的同步版本。 只有当超时发生或者操作正常完成，本次调用才会返回。 由于带有超时，本方法会响应{@link #cancelRequest()}方法。 如果超时发生，会返回这个操作结果：{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}。
输入	len 需要读取的最大长度 timeout 超时
输出	操作结果<code>SerialPortOperationResult</code>

6.8.11.2 SerialPortDeviceSpec 接口

功能描述	定义了对串口设备的详细描述。 可以得到终端中关于串口设备定义的信息。
输入	无
输出	无

6.8.11.2.1 getCounts 方法

功能描述	返回串口数量
输入	无
输出	终端中串口数量，不支持返回 0

6.8.11.3 SerialPortOperationResult 接口

功能描述	是被串口设备产生，用来返回串口的操作结果。 {@link OperationResult#getResultCode() getResultCode()}方法继承至{@link OperationResult OperationResult}的对应方法。 可以在{@link OperationResult#getResultCode() getResultCode()}返回操作结果是成功还是失败，通过{@link #getData()}返回通过串口读到的数据，通过{@link #getDataLength()}返回通过串口读到的数据长度。
输入	无
输出	无

6.8.11.3.1 getData 方法

功能描述	返回读到的数据

输入	无
输出	数据 buffer

6.8.11.3.2 getDataLength 方法

功能描述	返回数据长度
输入	无
输出	长度

6.8.12 smartcardreader

6.8.12.1 SmartCardReaderDevice 接口

功能描述	定义了接触式 IC 卡阅读器的使用接口。 设备对象通过<code>POSTerminal</code>的对应方法获得，如下所示：<pre> SmartCardReaderDevice martCardReaderDevice = (SmartCardReaderDevice) POSTerminal.getInstance().getDevice("cloudpos.device.smartcard"); </pre> 其中，"cloudpos.device.smartcard"是标识接触式 IC 卡阅读器的字符串，由具体的实现定义。接触式 IC 卡阅读器设备对象主要进行 SmartCard 读卡操作。其中等卡及移卡都包括同步和异步两种方式。同步方式会将主线程锁定，直到有结果返回，超时或者被取消。异步方式不会锁定主线程，当有结果时，会回调监听者{@link OperationListener#handleResult(OperationResult) handleResult()}方法。 为了正常访问接触式 IC 卡阅读器设备，请在 Android Manifest 文件中设置接触式 IC 卡阅读器设备的访问权限，具体如下所示：<uses-permission android:name="android.permission.CLOUDPOS_SMARTCARD"/>
输入	无
输出	无

6.8.12.1.1 open 方法

功能描述	打开某个逻辑 ID 的 IC 卡读卡槽。每个终端可能有多个 IC 卡卡槽，在打开时，要指定 logicalID. logicalID 的范围从 0 到设备支持的最大卡槽数-1
输入	logicalID 打开 IC 卡读卡器的设备编号，0 对应底部的插槽，1、2、3 对应终端背面的 PSAM 卡槽
输出	无

6.8.12.1.2 listenForCardPresent 方法

功能描述	监听插卡动作。本操作是个异步调用。当用户插卡发生后，结果通过操作监听者{@link OperationListener#handleResult(OperationResult) handleResult()}方法返回。本方法会正确响应{@link #cancelRequest()}方法来取消操作。通常程序必须定义自己的 OperationListener，在回调函数 handleResult()中对返回结果进行处理。如下所示：<pre>
	OperationListener operationListener = new OperationListener(){@Override public void handleResult(OperationResult result) { // handleResult } }); </pre>
	方法通过设置 timeout 来决定最多等待多长时间。请求开始执行的时候 timeout 开始计时。如果 timeout 时间到了，但用户还没有插卡，也会回调函数 handleResult()。此时 OperationResult 会返回错误：{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何卡片返回如果 timeout 是{@link TimeConstants#FOREVER FOREVER}，方法会一直等待插卡，直到插卡或取消。如果 timeout 是{@link TimeConstants#IMMEDIATE IMMEDIATE}，方法会马上返回。

输入	1. listener 操作监听者。 2. timeout 最大等待时间，通过时间常量设定{@link TimeConstants#SECOND SECOND},{@link TimeConstants#MilliSECOND MilliSECOND},{@link TimeConstants#FOREVER FOREVER},{@link TimeConstants#IMMEDIATE IMMEDIATE}。
输出	无

6.8.12.1.3 waitForCardPresent 方法

功能描述	本方法是上述对应的（listenForCardPresent）。{@link #listenForCardPresent(OperationListener, int)}方法的同步版本。 只有当超时发生或者操作正常完成，本次调用才会返回。 由于带有超时，本方法会响应{@link #cancelRequest()}方法。 如果超时发生，会返回这个操作结果：信息为{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何卡片返回。
输入	timeout 最大等待时间，通过时间常量设定。
输出	操作结果

6.8.12.1.4 listenForCardAbsent 方法

功能描述	监听卡片移除动作。
	本操作是个异步调用。当用户移除卡发生后，结果通过操作监听者
	{@link OperationListener#handleResult(OperationResult) handleResult()}方法返回
	本方法会正确响应{@link #cancelRequest()}方法来取消操作。通常程序必须定义自己的 OperationListener，在回调函数 handleResult()中对返回结果进行处理。如下所示：
	<pre>
	OperationListener operationListener = new OperationListener(){@Override public void handleResult(OperationResult result) { // handleResult } }); </pre>
	方法通过设置 timeout 来决定最多等待多长时间。请求开始执行的时候 timeout 开始计时。
	如果 timeout 时间到了，但用户还没有移除卡，也会回调函数 handleResult()。此时 OperationResult 会返回错误：{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何卡片返回。
	如果 timeout 是{@link TimeConstants#FOREVER FOREVER}，方法会一直等待,直到移除卡或取消。如果 timeout 是{@link TimeConstants#IMMEDIATE IMMEDIATE}，方法会马上返回。

输入	1. listener 操作监听者。 2. timeout 最大等待时间，通过时间常量设定{@link TimeConstants#SECOND SECOND},{@link TimeConstants#MilliSECOND MilliSECOND}, {@link TimeConstants#FOREVER FOREVER},{@link TimeConstants#IMMEDIATE IMMEDIATE}。
输出	无

6.8.12.1.5 waitForCardAbsent 方法

功能描述	本方法是上述对应的（listenForCardAbsent），{@link #listenForCardAbsent(OperationListener, int)}方法的同步版本。 只有当超时发生或者操作正常完成，本次调用才会返回。由于带有超时，本方法会响应{@link #cancelRequest()}方法。如果超时发生，会返回这个操作结果：状态为{@link OperationResult#EVT_ERROR EVT_ERROR}，信息为{@link OperationResult#ERR_TIMEOUT ERR_TIMEOUT}，同时没有任何卡片返回。

输入	timeout 最大等待时间，通过时间常量设定。
输出	操作结果

6.8.12.2 SmartCardReaderDeviceSpec 接口

功能描述	定义了对 SmartCard 读卡器的详细描述。 可以得到终端中关于接触式 IC 卡阅读器设备定义的信息。
输入	无
输出	无

6.8.12.2.1 getCounts 方法

功能描述	返回有多少个卡槽
输入	无
输出	卡槽数量,不支持返回 0

6.8.12.2.2 canRemovable 方法

功能描述	是否支持中断插入和移除卡片操作
输入	logicalID 打开 IC 卡读卡器的设备编号，0 对应底部的插槽，1、2、3 对应终端背面的 PSAM 卡槽
输出	{@code true} 支持,{@code false} 不支持，参数错误也返回 false

6.8.12.3 SmartCardReaderOperationResult 接口

功能描述	是被 SmartCard 读卡设备产生，用来返回得到的卡对象。 {@link OperationResult#getResultCode() getResultCode()}方法继承至{@link OperationResult}的对应方法。 这里通过"ERR_"设置了本设备相关的自定义错误，可以在{@link OperationResult#getResultCode() getResultCode()}返回错误或正确的操作结果。这里通过"ERR_"设置了本设备相关的自定义错误，可以在{@link OperationResult#getResultCode() getResultCode()}返回错误或正确的操作结果。得到卡对象后，应用程序可以自行区分不同类别的卡，进行卡的后续操作。一般返回的卡类型为{@link CPUCard CPUCard}或者{@link SLE4442Card SLE4442Card}两种。
输入	无
输出	无

6.8.12.3.1 getCard 方法

功能描述	返回卡。

输入	无
输出	卡对象。

6.9 外设访问权限控制模块

系统应实现可通过应用声明的配置文件来控制终端应用访问外设权限,即倘若应用未申明对某外设的访问权限，系统应限制该应用访问该外设。系统需对外设定义的权限控制功能至少包括：

序号	权限
1	访问证书管理与加密运算模块权限
2	访问磁条卡读卡器设备权限
3	访问接触式 IC 卡阅读器权限
4	访问非接触 IC 卡读卡设备权限
5	访问打印机设备权限
6	访问 PIN 输入设备权限

7 银行卡支付服务层

7.1 支付安全通道

详细参考《银联卡受理终端安全规范第八部分智能销售点终端安全规范》（QCUP 007.8-2013）。

7.2 银行卡支付调用接口

智能销售点终端须提供统一的支付接口供第三方应用完成支付调用。支付接口的种类、形式、内容可由支付接口提供商自行定制。支付接口应防止敏感信息的泄露，其中，敏感信息至少包括卡号、PIN、CVN2号等。

7.3 PIN 输入设备密钥索引管理

PIN输入设备是智能销售点终端的核心组件之一，用于加密持卡人的PIN、计算交易报文MAC，加解密数据等，可做为共享资源供终端上所有应用使用。为了唯一标识PIN输入设备中的终端主密钥，PIN输入设备为每一套终端主密钥分配了一个密钥索引号。PIN输入设备的详细描述参考《银联卡受理终端安全规范第八部分智能销售点终端安全规范》（QCUP 007.8-2013）；

PIN输入设备应能被运行于智能POS终端上的授权应用访问，为了避免PIN输入设备密钥索引使用冲突问题，银联建议预留0x00~0x09等10个索引号，机构可申请剩余的密钥索引号，并向银联进行报备。

8 应用开发要求

应用开发要求对第三方开发者在应用开发过程中所涉及的各方面和环节进行规范和约束,确保开发的应用在功能、安全、友好性上满足用户的需求。

8.1 通用要求

——第三方应用开发者应严格遵循智能销售点终端应用开发要求开发各类应用。

——第三方应用开发者开发的应用不能损害机构、商户的利益，包括但不限于泄漏个人隐私、破坏用户数据等。

8.2 应用信息

应用信息至少须包括：应用唯一标识、应用图标、应用名称。

——应用唯一标识须采用字符类型，最大长度应不超过128个字符；

——应用名称不能包括非法字符（\ / ： * ？“ 〈 〉 |），最大长度应不超过128个字符；

——应用图标的长宽须采用48p*48px、72*72px、 128*128px中的一种。

——应用图标格式须采用png、jpg、jpeg、ico图片格式中的一种。

——应用图标的大小应不超过30KB；

8.3 应用进程

——应用的进程不允许隐藏，需能在进程管理工具中查找到并接受管理。

——如果该应用是开机自启动应用，则终端智能操作系统须允许终端管理员通过界面对其进行配置。进程管理工具须能查找开机自启动应用并能中止开机自启动应用的运行。

8.4 用户体验相关建议

此章节只作建议，不作强行规定。

8.4.1 用户界面设计原则

智能销售点终端应用界面的设计应该是基于对应终端智能操作系统的整体风格。界面中的图标、按钮等须表现出一定的个性，来体现它们各自的功能，从而降低用户的认知难度。

8.4.2 用户交互设计原则

用户交互遵循的原则包括但不限于：

——提供快速反馈，对用户的操作要尽快反馈信息。

——要有清楚、易懂的错误提示。

——提供导航功能，用户能方便进行功能切换。

8.4.3 应用界面文字格式建议

——对于在智能销售点终端中显示的应用程序名称，可以用中文表示的应用名，须用中文表示；

——对于应用向用户呈现的所有显示界面，所有可以用中文表示的内容，须用中文表示。

8.5 其他

第三方应用开发者在开发应用中涉及的其他要求如下：

——开发的应用中不能含有恶意代码（没作用却会带来危险的代码），导致智能销售点终端服务不可用。

——开发的应用中不能含有违背国家规定的黄色非法信息。

——开发的应用所用的密码不可以明码显示。

——开发的应用各个业务流程需能执行完成，各种异常情况下不得导致终端不可使用。

——开发的应用中不能插入与本应用或智能销售点终端无关的广告信息。

8.6 应用管理要求

8.7 应用生命周期管理

业务类应用的整个生命周期如图3所示：

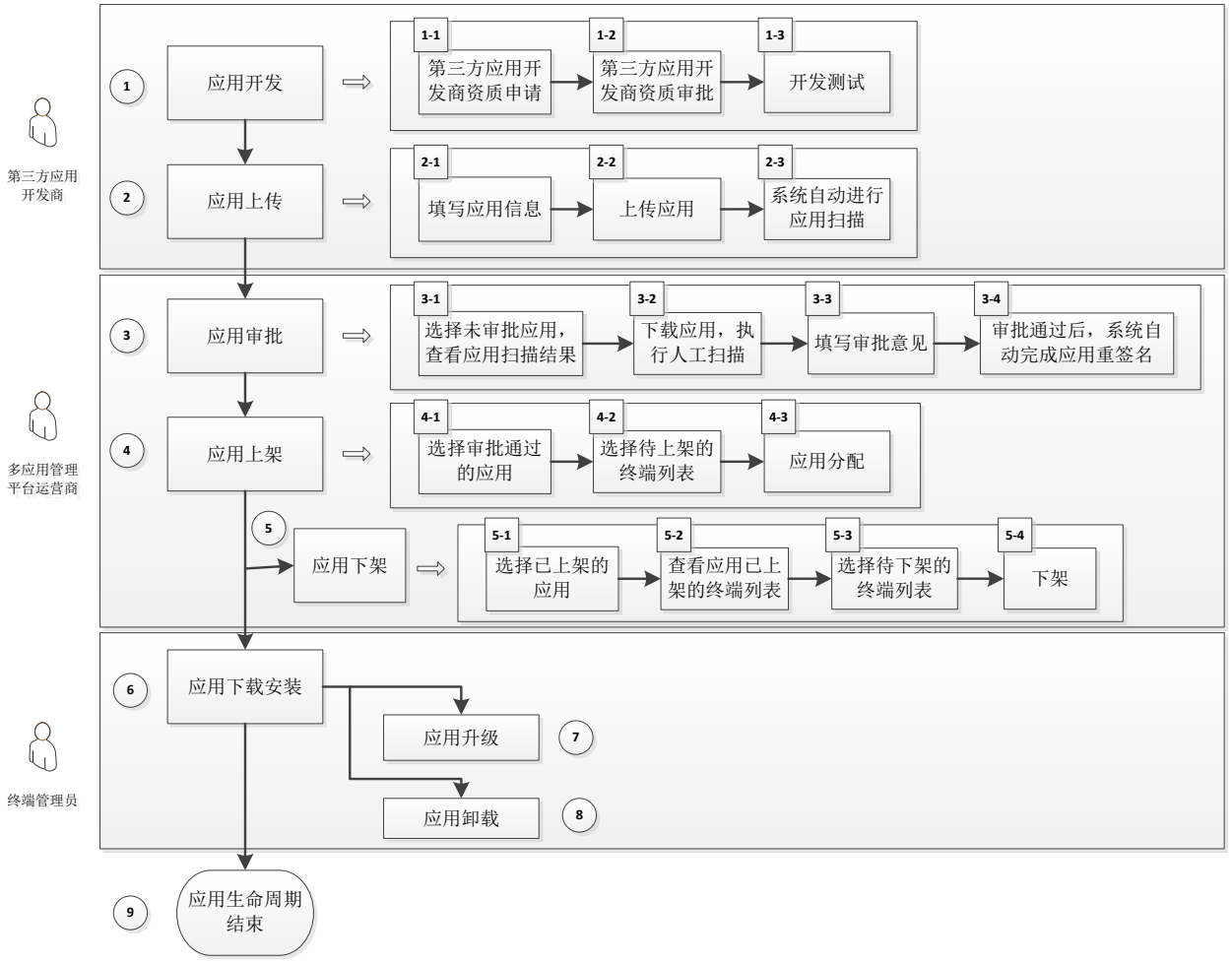


图3　业务类应用生命周期总图

第三方应用开发商，负责为机构、商户提供软件解决方案，并负责将应用上传至多应用管理平台。

多应用管理平台运营商，负责多应用管理平台的研发、运维等。多应用管理平台运营商须由机构担任。

终端管理员，负责终端运行参数的配置、终端应用的下载安装、升级、卸载等。终端管理员须由商户负责人或机构担任。

8.7.1　应用开发

第三方应用开发商遵循本规范第8章节应用开发要求，完成应用的开发及联调测试，满足机构、商户个性化需求。

第三方应用开发商登录多应用管理平台并申请应用开发者资质。申请的信息至少包括第三方应用开发商名称、企业法人营业执照、联系方式。

多应用管理平台运营商登录多应用管理平台，查看第三方应用开发商的信息并审核第三方应用开发商的资质。

8.7.2　应用上传

第三方应用开发商完成资质申请及应用开发后，登录多应用管理平台并填写应用相关信息。应用信息至少包括：应用功能描述、应用截图、终端分辨率、终端系统版本支持列表。

应用信息填写完毕后，由第三方应用开发商选择应用文件并上传至多应用管理平台。

在应用上传后，多应用管理平台应对应用进行自动扫描并获取应用基本信息。可获取的应用基本信息至少包括应用唯一标识、应用显示名称、应用显示图标、应用版本号、应用申请外设的访问权限列表。

在以下情况，多应用管理平台须拒绝该应用的上传：

——该应用第一次被上传至多应用管理平台且应用唯一标识在多应用管理平台已登记。

——该应用已有版本被上传至多应用管理平台且本次应用版本不高于该应用已有版本。

——应用显示名称包含’*’、’/’等非法字符

8.7.3 应用审批

应用上传后，多应用管理平台须采用安全的方式保存应用相关信息及文件，同时应用被置于未审批状态。

多应用管理平台应向多应用管理平台运营商提供未审批应用列表查看入口。多应用管理平台运营商查看未审批应用基本信息，以进行应用评估。评估功能点至少包括：

——应用申请外设的访问权限列表：结合应用提供商的资质及外设的安全级别，判定该应用是否被允许访问所申请的外设；

——应用显示图标：该应用显示图标不能重复。

多应用管理平台运营商宜采用安全的方式下载该应用，通过人工扫描进行应用评估。人工扫描包括以下方面：

——安全扫描：至少判断该应用是否携带病毒、木马及具备应用安装、升级及卸载等非法权限。

——业务扫描：至少判断该应用是否符合业界标准及道德标准等。

第三方应用开发商完成应用评估后，需填写应用审批意见及给出应用审批结果。

应用审核通过后，多应用管理平台应自动对应用进行重新打包，智能销售点终端只能安装运行重新打包后的应用。

8.7.4 应用上架

应用审批通过后，多应用管理平台宜采用安全的方式保存重新打包后的应用文件。

多应用管理平台应向多应用管理平台运营商提供已审批的应用列表查看入口。多应用管理平台运营商查看已审批应用的基本信息，选择该应用待上架的终端列表，完成应用的终端分配。

应用分配后，终端管理员才能通过从终端上下载安装该应用。

8.7.5 应用下架

多应用管理平台运营商可登录多应用管理平台，对已上架应用进行下架操作。应用下架后，终端不能下载安装或升级该应用。

应用下架流程如下：

——多应用管理平台运营商选择已上架的应用，查看应用信息。

——查看该应用已上架的终端列表。

——选择待下架的终端列表。

——完成应用下架。

8.7.6 应用下载安装

应用上架后，终端管理员可通过多应用管理客户端查看应用信息，对应用进行下载安装。

多应用管理客户端

应用安装、验证

安全下载

终端标识

8.7.7 应用升级

终端管理员可通过多应用管理客户端对终端上已安装的应用进行升级。多应用管理客户端以安全的方式接入多应用管理平台，获取终端已安装应用的最新版本信息，更新应用的升级状态。 多应用管理客户端应提供两种升级方式：强制升级和非强制升级。

——强制升级。多应用管理客户端判断应用有升级版本时，提供应用强制升级界面。终端管理员只有对该应用进行升级后，才能正常启动、运行该应用。

——非强制升级。多应用管理客户端判断应用有升级版本时，提供应用升级提示界面。终端管理员可选择取消该应用版本的升级。

8.7.8 应用卸载

终端管理员可通过多应用管理客户端卸载终端上已安装的应用。应用卸载后，应用从已安装状态变为可安装状态，即终端管理员依然可下载安装该应用。

多应用管理客户端卸载应用时，须从终端智能操作系统中彻底删除应用文件。

8.8 应用证书管理

显示智能销售点终端二级应用证书管理体系如图4所示：

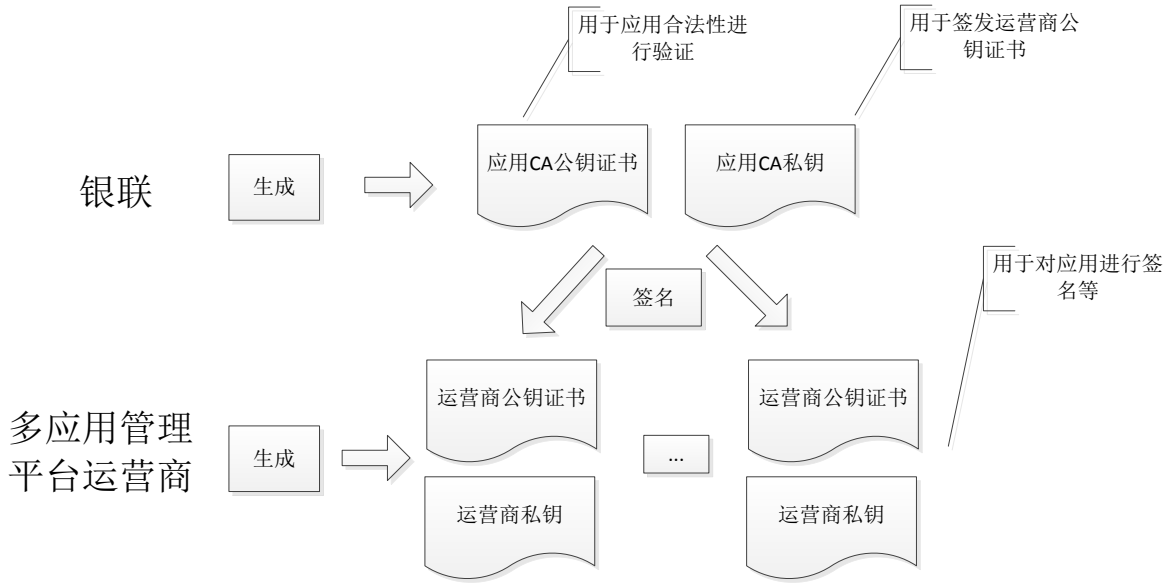


图4 二级应用证书管理

智能销售点终端二级应用证书管理体系中涉及的私钥文件和证书文件包括：应用CA私钥、应用CA公钥证书、运营商私钥、运营商公钥证书。本章节中的运营商通指多应用管理平台运营商。

8.8.1 应用 CA 私钥及公钥证书

应用CA私钥是由银联生成的,长度为2048位，用于签发运营商公钥证书。

应用CA公钥证书应由银联生产的。智能销售点终端厂商应向银联申请并获取应用CA公钥证书，在智能销售点终端出厂时，智能销售点终端厂商须将应用CA公钥证书写入终端的证书管理与加密运算模块中。

多应用管理客户端安装或升级应用时，须调用应用CA公钥证书对应用进行合法性验证。

8.8.2 运营商私钥及公钥证书

运营商私钥由多应用平台运营商生成并采用安全的方式保存，长度应不少于2048位，用于对应用进行签名。

运营商公钥证书须由银联签发，用于验证应用的完整性，防止应用被非法篡改。

8.9 业务类应用要求

8.9.1 应用文件结构要求

运行在智能销售点终端上的应用至少包含四部分：可执行文件、摘要信息、摘要数字签名信息、运营商公钥证书信息。

可执行文件至少包含应用可执行代码、资源文件。应用审批通过后，多应用管理平台对应用进行重新打包：使用SHA1散列算法对可执行文件生成摘要信息，然后使用运营商私钥对摘要信息进行签名，生成摘要数字签名信息，最后将生成的摘要信息、摘要数字签名信息、运营商公钥证书信息以及应用可执行文件一并打包成生成新的应用。

8.9.2 应用功能限制要求

业务类应用不能具备以下功能：

——不具备应用安装功能。禁止业务类应用通过任何方式安装其他应用。

——不具备应用升级功能。禁止业务类应用通过任何方式完成自升级或升级其他应用。

——不具备应用卸载功能。禁止业务类应用通过任何方式卸载其他应用。

8.9.3 银行卡支付客户端

银行卡支付客户端应能受理银行卡收单业务，并具备磁条卡阅读器、接触式IC卡阅读器、非接触式IC卡阅读器、PIN输入设备、打印机访问权限。

银行卡支付客户端应由终端管理员从多应用管理平台上下载、安装及升级。智能销售点终端至少须安装一个银行卡支付客户端。

附 录 A
（规范性附录）
智能销售点终端细化要求

A.1 基于Android系统定制的智能销售点终端的细化要求

A.1.1 系统功能要求

A.1.1.1 系统版本

Android 2.3及以上版本，建议采用Android 4.2或以上版本。

A.1.1.2 外设驱动标准C接口

基于Android系统的系统外设驱动接口均以C语言接口进行定义并提供给上层应用调用。

A.1.1.2.1 接触式IC卡阅读器

驱动接口信息

设备名称	CLOUD_DEVICE_CONTACT_IC_CARD
设备用途	智能终端上一个或多个接触式IC卡操作
设备驱动文件	/system/lib/libUnionpayCloudPos.so

驱动调用方法	void* pHandle = dlopen(“libUnionpayCloudPos.so”, RTLD_LAZY);//调用驱动
	(“函数名”)dlsym(pHandle, “函数名”);//得到函数句柄

函数列表

设备初始化	smart_card_init
设备回收	smart_card_terminate
查询设备	smart_card_query_max_number
设备打开	smart_card_open
设备关闭	smart_card_close
设备内IC卡查询	smart_card_query_presence
设备上电	smart_card_power_on
设备下电	smart_card_power_off
设置参数	smart_card_set_slot_info
数据传输	smart_card_transmit

A.1.1.2.1.1 设备初始化

设备初始化接口，主要定义为设备资源的初始功能。

接口名称	smart_card_init
功能描述	初始化接触式读卡设备
接口格式	int smart_card_init();
参数说明	无

返回值	1. 大于等于 0，初始化设备成功 2. 小于 0，错误；参考附录一：错误定义表

A.1.1.2.1.2 设备回收

设备回收接口，主要定义为设备资源的释放。

接口名称	smart_card_terminate
功能描述	回收被接触式读卡设备占用的资源
接口格式	int smart_card_terminate();
参数说明	无
返回值	1. 等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表

A.1.1.2.1.3 查询设备

硬件支持多个卡槽的IC卡设备，提供查询功能，后续可以根据查询的卡槽数控制具体操作的设备。

接口名称	smart_card_query_max_number
功能描述	查询本设备中有多少槽（slot）可用
接口格式	int smart_card_query_max_number();
参数说明	无
返回值	1.等于 0，表示没有槽（slot） 2. 大于 0，表示槽（slot）数 3. 小于 0，表示错误；参考附录一：错误定义表

A.1.1.2.1.4 设备打开

设备打开，主要定义为设备打开，对于硬件支持多卡槽的设备，需要指定卡槽编号。

接口名称	smart_card_open			
功能描述	打开指定槽（slot）中的设备			
接口格式	int smart_card_open(int nSlotIndex, SMART_CARD_NOTIFIER pNotify, void* pUserData);			
参数说明	nSlotIndex	int	必选	槽的索引
	pNotify	SMART_CARD_NOTIFYER	必选	回调函数，类型定义为 typedef void (*SMART_CARD_NOTIFIER)(void* pUserData, int nCardIndex, int nEvent); nEvent:可以有如下选项： SMART_CARD_EVENT_INSERT_CARD SMART_CARD_EVENT_REMOVE_CARD SMART_CARD_EVENT_POWER_ON SMART_CARD_EVENT_POWER_OFF

	pUserData	void*	必选	用户回调函数的参数
返回值	1. 大于等于 0，设备句柄 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.1.5 设备关闭

关闭设备，主要定义为关闭已经打开的设备。

接口名称	smart_card_close			
功能描述	关闭 smart_card_open 函数打开的设备			
接口格式	int smart_card_close(inthandle);			
参数说明	handle	int	必选	设备句柄
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.1.6 查询卡在位

查询相应卡槽内的IC卡是否在位。

接口名称	smart_card_query_presence			
功能描述	查询指定卡槽中卡（和该句柄相关联的卡槽）是否存在			
接口格式	int smart_card_query_presence(int nSlotIndex);			
参数说明	nSlotIndex	int	必选	槽的索引
返回值	1.大于 0，表示存在 2.等于 0，表示不存在 3.小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.1.7 设备上电

根据设备特性，给予上电操作，可根据实际情况设置上电参数。

接口名称	smart_card_power_on			
功能描述	给指定卡槽中的卡上电			
接口格式	int smart_card_power_on(int Handle, unsigned char* pATR, unsigned int* pATRBufferLength, SMART_CARD_SLOT_INFO*pSlotInfo);			
参数说明	Handle	int	必选	设备句柄
	pATR	unsigned char*	必选	用来存放 ATR 的缓冲区
	pATRBufferLength	unsigned int*	必选	在输入的时候，是 pATR 缓冲区的长度。在输出的时候，是存放在 pATR 缓冲区中实际数据的长度。
	pSlotInfo	SMART_CARD_SLOT	必选	存放指定槽的相关信息。

		_INFO*		
返回值	1. 大于等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表			

——数据结构 SMART_CARD_SLOT_INFO.结构的定义如下：

字段名	类型	说明
FIDI	unsigned char	参见标准 ISO 7816-3 中关于 TA1 的说明
EGT	unsigned char	额外警戒时间（ Extra Guard Time） (ISO 7816-3 标准中参数 TC1 或 N)
WI	unsigned char	该参数的定义和使用协议相关，如果协议为 T=0, 该参数的含义为等待时间（Waiting Integer）（缺省值是 10，参见 ISO 7816-3 标准中关于 TC2 的说明） *如果协议为 T=1，该参数的含义为块和字符等待时间（Block and Character Waiting Time Integer）。参见 ISO 7816-3 标准中关于 TB3 的说明
WTX	unsigned char	如果协议是 T=1，该参数表示等待时间延长（Waiting Time Extention）
EDC	unsigned char	如果协议为 T=1，该参数表示校验计算方式（mode of EDC）HAL_SCS_EDC_LRC 或者 HAL_SCS_EDC_CRC（缺省为 LRC）
protocol	unsigned char	协议类型，可以是：* HAL_SCS_PROTOCOL_T0 ：用于 CPU 卡 * HAL_SCS_PROTOCOL_T1 ：用于 CPU 卡 * HAL_SCS_PROTOCOL_RAW ：用于 memory 类卡
power	unsigned char	供电电压 ，可以是：* HAL_SCS_POWER_1_8V * HAL_SCS_POWER_3V * HAL_SCS_POWER_5V
conv	unsigned char	传输字节的方式，可以是：* HAL_SCS_CONV_DIRECT * HAL_SCS_CONV_INVERSE
IFSC	unsigned char	如果协议是 T=1，该参数表示卡的字段大小（Field Size for the Card），可以参看 ISO 7816-3 标准中关于 TA3 的说明
reserved	unsigned char [3]	
cwt	unsigned int	可以用来设置字符等待时间（Character Waiting Time）
bwt	unsigned int	可以用来设置块等待时间（Block Waiting Time）
nSlotInfoItem	unsigned int	由以下常量通过或（OR）操作构成，用来指定目前结构中哪些位得到了更新，或哪些项需要设置。 SMART_CARD_SLOT_INFO_FIDI SMART_CARD_SLOT_INFO_EGT SMART_CARD_SLOT_INFO_WI SMART_CARD_SLOT_INFO_WTX SMART_CARD_SLOT_INFO_EDC SMART_CARD_SLOT_INFO_PROTOCOL SMART_CARD_SLOT_INFO_POWER SMART_CARD_SLOT_INFO_CONV SMART_CARD_SLOT_INFO_IFSC

		SMART_CARD_SLOT_INFO_CWT SMART_CARD_SLOT_INFO_BWT

A.1.1.2.1.8 设备下电

设备下电，下电后不可进行通信，需要关闭后释放资源。

接口名称	smart_card_power_off			
功能描述	给指定卡槽（和该句柄相关联的卡槽）中的卡下电			
接口格式	int smart_card_power_off(int Handle);			
参数说明	Handle	Int	必选	设备句柄
返回值	1.大于等于 0，表示成功下电 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.1.9 设置参数

定义为IC卡参数设置，具体是硬件情况而定。

接口名称	smart_card_set_slot_info			
功能描述	设置相关信息			
接口格式	int smart_card_set_slot_info(int Handle, SMART_CARD_SLOT_INFO* pSlotInfo);			
参数说明	Handle	int	必选	设备句柄
	pSlotInfo	SMART_CARD_SLOT_INFO*	必选	存放相关参数的最新值，参数类型已经在上文（2.2.6）中说明
返回值	1.大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.1.10 数据传输

IC卡数据通信接口，数据格式基于ISO 7816-4关于APDU/TPDU的定义。

接口名称	smart_card_transmit			
功能描述	数据传输			
接口格式	int smart_card_transmit(int Handle, unsigned char* pAPDU, unsigned int nAPDULength, unsigned char* pResponse, unsigned int *pResponseLength);			
参数说明	Handle	int	必选	设备句柄
	pAPDU	unsigned char*	必选	存放 APDU 命令

	nAPDULength	unsigned int	必选	APDU 命令的长度
	pResponse	unsigned char*	必选	用来存放响应的缓冲区。
	pResonseLength	unsigned int*	必选	输入时作为 pResponse 缓冲区的长度，输出的时候，是响应数据的长度。
返回值	1.大于等于 0，表示通讯成功 2.小于 0，通讯失败；参考附录一：错误定义表			

——数据缓冲区的数据格式基于 ISO 7816-4 关于 APDU/TPDU 的定义

A.1.1.2.2 非接触式IC卡阅读器

驱动接口信息

设备名称	CLOUD_DEVICE_CONTACTLESS_IC_CARD
设备用途	智能终端上一个或多个非接触式IC卡操作
设备驱动文件	/system/lib/libUnionpayCloudPos.so

驱动调用方法	void* pHandle = dlopen(“libUnionpayCloudPos.so”, RTLD_LAZY);//调用驱动
	(“函数名”)dlsym(pHandle, “函数名”);//得到函数句柄

函数列表

设备打开	contactless_card_open
设备关闭	contactless_card_close
搜索卡片	contactless_card_search_target_begin
结束搜索	contactless_card_search_target_end
连接卡片	contactless_card_attach_target
断开连接	contactless_card_detach_target
命令交互	contactless_card_transmit
卡片控制	contactless_card_send_control_command

A.1.1.2.2.1 设备打开

打开非接卡设备，定义为打开设备，初始化资源。

接口名称	contactless_card_open			
功能描述	创建或者打开一个已经存在的非接触读卡设备			
接口格式	int contactless_card_open(CONTACTLESS_CARD_NOTIFIER fNotifier, void* pUserData,int* pErrorCode);			
参数说明	fNotifier	CONTACTLESS_CARD_NOTIFIER	必选	回调函数 CONTACTLESS_CARD_NOTIFIER 被定义为：typedef void

				(*CONTACTLESS_CARD_NOTIFIER)(void* pUserData, int nEvent, unsigned char* pEventData, int nDataLength)
	pUserData	void	必选	用户数据，会作为回调函数的参数，传入
	pErrorCode	int	必选	错误代码。当返回值等于 0 的时候设置
返回值	1. 等于 0，错误 2. 不等于 0，设备句柄			

A.1.1.2.2.2 设备关闭

定义为关闭设备，释放资源。

接口名称	contactless_card_close			
功能描述	关闭非接触读卡设备			
接口格式	int contactless_card_close(int nHandle);			
参数说明	nHandle	int	必选	设备句柄
返回值	1. 大于等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表			

A.1.1.2.2.3 搜索设备

搜索卡片，定义为卡片检测，检测非接卡有效区域内的卡片。

接口名称	contactless_card_search_target_begin			
功能描述	开始搜索非接触 IC 卡			
接口格式	int contactless_card_search_target_begin(int nHandle, int nCardMode, int nFlagSearchAll,int nTimeout_MS);			
参数说明	nHandle	int	必选	设备句柄
	nCardMode	int	必选	卡类型[0,1,2,3]，分别为： CONTACTLESS_CARD_MODE_AUTO， CONTACTLESS_CARD_MODE_TYPE_A， CONTACTLESS_CARD_MODE_TYPE_A， CONTACTLESS_CARD_MODE_TYPE_B， CONTACTLESS_CARD_MODE_TYPE_C

	nFlagSearchAll	int	必选	0：找一张卡，1：找所有的卡
	nTimeout_MS	int	必选	超时，单位为毫秒。小于 0 时，将一直搜索，直到使用方法：contactless_card_search_target_end 打断。
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.2.4 停止搜索

停止当前的搜索，停止后，与搜索接口成对出现。

接口名称	contactless_card_search_target_end			
功能描述	停止搜索非接触 IC 卡			
接口格式	int contactless_card_search_target_end(int nHandle);			
参数说明	nHandle	int	必选	设备句柄
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.2.5 连接卡片

卡片连接函数，当存在多张卡片时，只能与一张卡片连接，在传递APDU命令前，必须要连接非接触IC卡

接口名称	contactless_card_attach_target			
功能描述	在传递 APDU 命令前，链接非接触 IC 卡			
接口格式	int contactless_card_attach_target(int nHandle, unsigned char* pATRBuffer, unsigned int nATRBufferLength);			
参数说明	nHandle	int	必选	设备句柄
	pATRBuffer	unsigned char	必选	ATR 缓冲区
	nATRBufferLength	unsigned int	必选	ATR 缓冲区长度
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.2.6 断开连接

解除和当前卡片的连接状态，解除连接状态后，将不能继续进行APDU命令的传输。

接口名称	contactless_card_detach_target
功能描述	和非接触 IC 卡解除链接

接口格式	int contactless_card_detach_target(int nHandle);			
参数说明	nHandle	int	必选	设备句柄
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.2.7 APDU 命令交互

非接卡APDU交互接口。

接口名称	contactless_card_transmit			
功能描述	发送 APDU 命令并接收响应			
接口格式	int contactless_card_transmit(int nHandle, unsigned char* pAPDU, unsigned int nAPDULength, unsigned char* pResponse, unsigned int* pResponseLength);			
参数说明	nHandle	int	必选	设备句柄
	pAPDU	unsigned char	必选	APDU 命令
	nAPDULength	int	必选	APDU 命令长度
	pResponse	unsigned char	必选	存储 APDU 命令响应的缓冲区
	pResponseLength	unsigned int	必选	输入时，为响应 APDU 命令缓冲区长度。输出时，为响应数据长度
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.2.8 IC 卡控制

有些硬件在使用时，需要进行特殊指令的传输，该接口定义为提供这种指令的调用。

接口名称	contactless_card_send_control_command			
功能描述	发送非接触 IC 卡控制命令			
接口格式	int contactless_card_send_control_command(int nHandle, unsigned int nCmdID, unsigned char* pCmdData, unsigned int nDataLength);			
参数说明	nHandle	int	必选	设备句柄
	nCmdID	unsigned int	必选	控制命令 ID 号
	pCmdData	unsigned char	必选	和命令有关的数据。输入时，是和命令有关的数据，如此命令没有数据，可设置为 NULL。输出时，为响应数据。
	nDataLength	unsigned int	必选	命令的数据长度

返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表

A.1.1.2.3 磁条卡阅读器

驱动接口信息

设备名称	CLOUD_DEVICE_MSR
设备用途	控制智能终端上磁条卡读卡器
设备驱动文件	/system/lib/libUnionpayCloudPos.so

驱动调用方法	void* pHandle = dlopen(“libUnionpayCloudPos.so”, RTLD_LAZY);//调用驱动
	(“函数名”)dlsym(pHandle, “函数名”);//得到函数句柄

函数列表

设备打开	MSR_OPEN
设备关闭	MSR_CLOSE
注册刷卡	MSR_REGISTER_NOTIFIER
取消刷卡	MSR_UNREGISTER_NOTIFIER
读取磁道错误	MSR_GET_TRACK_ERROR
获取磁道数据长度	MSR_GET_TRACK_DATA_LENGTH
获取磁道数据	MSR_GET_TRACK_DATA

A.1.1.2.3.1 打开设备

打开磁条卡设备

接口名称	MSR_OPEN
功能描述	打开一个已经存在的磁条卡阅读器
接口格式	int MSR_OPEN ();
参数说明	无
返回值	1. 大于等于 0，成功打开设备 2. 小于 0，错误；参考附录一：错误定义表

A.1.1.2.3.2 关闭设备

关闭磁条卡设备设备未打开关闭无效。

接口名称	MSR_CLOSE
功能描述	关闭已打开的磁条卡阅读器
接口格式	int MSR_CLOSE ();
参数说明	无
返回值	1. 大于等于 0，成功关闭设备 2. 小于 0，错误；参考附录一：错误定义表

A.1.1.2.3.3 注册刷卡

打开设备后，通过该函数进行刷卡注册，刷卡注册后，磁条卡设备将启动监听，直到等到刷卡或者主动取消。

接口名称	MSR_REGISTER_NOTIFIER			
功能描述	登记回调函数，回调函数登记好之后，如果用户刷卡，回调函数会被调用，用来通知用户来取数据			
接口格式	MSR_REGISTER_NOTIFIER (MSR_NOTIFIER notifier, void* pUserData);			
参数说明	notifier	MSR_NOTIFIER	必选	回调函数。MSR_NOTIFIER 被定义为：typedef void (*MSR_NOTIFIER)(void* pUserData);
	pUserData	void	必选	用户数据，会作为回调函数的参数，传入
返回值	1. 大于等于 0，表示成功。 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.3.4 取消刷卡

取消刷卡，只在调用msr_register_notifier之后有用，取消刷卡后，磁条卡设备恢复正常模式。

接口名称	MSR_UNREGISTER_NOTIFIER
功能描述	撤销登记的函数，撤销之后，刷卡后上层程序不会收到通知
接口格式	int MSR_UNREGISTER_NOTIFIER ();
参数说明	无
返回值	1. 大于等于 0，表示成功。 2. 小于 0，表示错误；参考附录一：错误定义表

A.1.1.2.3.5 读取磁道错误

读取磁道错误，监听刷卡后，调用该函数确定磁道数据是否正常。

接口名称	MSR_GET_TRACK_ERROR			
功能描述	查询对应磁道在解码过程中发生的错误			
接口格式	int MSR_GET_TRACK_ERROR (int nTrackIndex);			
参数说明	nTrackIndex	int	必选	磁道索引，可以是 0，1，2
返回值	1. 大于等于 0，表示成功。 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.3.6 获取磁道数据长度

获取磁道数据长度。

接口名称	MSR_GET_TRACK_DATA_LENGTH
功能描述	查询对应磁道数据的长度
接口格式	int MSR_GET_TRACK_DATA_LENGTH (int nTrackIndex);

参数说明	nTrackIndex	int	必选	磁道索引，可以是0，1，2
返回值	1. 大于等于0，表示磁道数据的长度。 2. 小于0，表示错误；参考附录一：错误定义表			

A.1.1.2.3.7 获取磁道数据

获取磁道数据明文。

接口名称	MSR_GET_TRACK_DATA			
功能描述	获取对应磁道数据的数据			
接口格式	int MSR_GET_TRACK_DATA (int nTrackIndex, unsigned char* pTrackData, int nLength);			
参数说明	nTrackIndex	int	必选	磁道索引，可以是0，1，2
	pTrackData	unsigned char*	必选	存放磁道数据的缓冲区。
	nLength	int	必选	数据缓冲区的长度。由于有接口查询长度信息，所以调用者可以避免缓冲区长度不够的错误。
返回值	1. 大于等于0，表示磁道数据的长度。 2. 小于0，表示错误；参考附录一：错误定义表			

A.1.1.2.4 PIN输入设备

驱动接口信息

设备名称	CLOUD_DEVICE_PINPAD
设备用途	控制智能终端上PIN输入设备
设备驱动文件	/system/lib/libUnionpayCloudPos.so

驱动调用方法	void* pHandle = dlopen(“libUnionpayCloudPos.so”, RTLD_LAZY);//调用驱动
	(“函数名”)dlsym(pHandle, “函数名”);//得到函数句柄

函数列表

设备初始化	pinpad_init
打开设备	pinpad_open
关闭设备	pinpad_close
PIN输入设备显示	pinpad_show_text
选择密钥	pinpad_select_key
数据加密	pinpad_encrypt_string
更新工作密钥	pinpad_update_user_key
计算PINBLOCK	pinpad_calculate_pin_block
计算Mac	pinpad_calculate_mac
设置Pin长度	pinpad_set_pin_length

更新密钥	pinpad_update_master_key
初始化	pinpad_init
pinblock回调	pinpad_set_pinblock_callback

A.1.1.2.4.1 密码初始化

密码初始化

接口名称	pinpad_init			
功能描述	密码键盘初始化接口，用于供系统区分内置/外置密码键盘			
接口格式	int pinpad_init (int id)			
参数说明	id	int	必选	1.等于 0，使用外置 2.等于 1，使用内置
返回值	1. 等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表			

A.1.1.2.4.2 打开设备

打开PIN输入设备。

接口名称	pinpad_open
功能描述	打开存在的 PIN 输入设备
接口格式	int pinpad_open();
参数说明	无
返回值	1. 等于 0，打开成功 2. 小于 0，错误；参考附录一：错误定义表

A.1.1.2.4.3 关闭设备

关闭PIN输入设备。

接口名称	pinpad_close
功能描述	关闭已打开的 PIN 输入设备
接口格式	int pinpad_close();
参数说明	无
返回值	1. 等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表

A.1.1.2.4.4 PIN 输入设备显示

PIN输入设备显示接口，将数据显示在PIN输入设备LCD上。

接口名称	pinpad_show_text
功能描述	写入显示数据同时显示在 PIN 输入设备 LCD 上

接口格式	int pinpad_show_text(int nLineIndex, char* strText, int nLength, int nFlagSound);			
参数说明	nLineIndex	int	必选	行标，从上到下，依次从 0 开始
	strText	char	必选	写入的显示数据
	nLength	int	必选	写入数据的字节长度，为 0 时，清 PIN 输入设备 LED
	nFlagSound	int	必选	声音提示，0 不提示，1 提示
返回值	1. 大于等于 0，写入显示成功 2. 小于 0，写入显示失败			

A.1.1.2.4.5 选择密钥

选择当前操作的PIN输入设备主密钥和工作密钥（用户密钥）。

接口名称	pinpad_select_key			
功能描述	选择一个 PIN 输入设备主密钥和用户密钥			
接口格式	int pinpad_select_key(int nKeyType, int nMasterKeyID, int nUserKeyID, int nAlgorith);			
参数说明	nKeyType	int	必选	主密钥类型。0：dukpt，1：Tdukpt，2：master key，3：public key，4：fix key
	nMasterKeyID	int	必选	主密钥 ID，[0x00, …,0x09]。当 nKeyType 为 master key 时使用
	nUserKeyID	int	必选	用户密钥 ID，[0x00,0x01]。当 nKeyType 为 master key 时使用
	nAlgorith	int	必选	算法选择。1：3DES，2：DES
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.4.6 数据加密

使用当前选择的主密钥和工作密钥进行数据加密。

接口名称	pinpad_encrypt_string			
功能描述	使用用户密钥加密数据			
接口格式	int pinpad_encrypt_string(unsigned char* pPlainText, int nTextLength, unsigned char* pCipherTextBuffer,int nCipherTextBufferLength);			
参数说明	pPlainText	unsigned char	必选	待加密字符串
	nTextLength	int	必选	待加密字符串长度
	pCipherTextBuffer	unsigned char	必选	用于存储密文的缓冲区
	nCipherTextBufferLength	int	必选	缓冲区长度
返回值	1. 大于等于 0，表示成功			

	2. 小于 0，表示错误；参考附录一：错误定义表

A.1.1.2.4.7 更新工作密钥

更新工作密钥，工作密钥为密文下载。

接口名称	pinpad_update_user_key			
功能描述	更新用户密钥			
接口格式	int pinpad_update_user_key(int nMasterKeyID,int nUserKeyID, unsigned char* pCipherNewUserKey,int nCipherNewUserKeyLength);			
参数说明	nMasterKeyID	int	必选	PIN 输入设备主密钥 ID
	nUserKeyID	int	必选	PIN 输入设备用户密钥 ID
	pCipherNewUserKey	unsigned char	必选	待更细的用户密钥，已加密过
	nCipherNewUserKeyLength	int	必选	待更细用户密钥的长度
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.4.8 更新密钥

更新密钥

接口名称	pinpad_update_master_key			
功能描述	更新密码键盘主密钥，旧密钥与密码键盘一致，方能调用成功			
接口格式	int pinpad_update_master_key (int nMasterKeyID, unsigned char* pOldKey, int nOldKeyLength, unsigned char* pNewKey, int nNewKeyLength)			
参数说明	nMasterKeyID	int	必选	PIN 输入设备主密钥 ID
	pOldKey	unsigned char	必选	旧密钥
	nOldKeyLength	int	必选	旧密钥长度
	pNewKey	unsigned char	必选	新密钥
	nNewKeyLength	int	必选	新密钥长度
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；			

A.1.1.2.4.9 计算 PINBLOCK

计算Pinblock, 键盘输入PIN处理方式为ISO9564-1格式0.

接口名称	pinpad_calculate_pin_block			
功能描述	计算 Pin Block			
接口格式	int pinpad_calculate_pin_block(unsigned char* pASCIICardNumber, int nCardNumberLength, unsigned char* pPinBlockBuffer, int nPinBlockBufferLength, int nTimeout_MS, int nFlagSound)			
参数说明	pASCIICardNumber	unsigned char	必选	银联卡卡号
	nCardNumberLength	int	必选	银联卡卡号长度
	pPinBlockBuffer	unsigned char	必选	保存 Pin Block 的缓冲区
	nPinBlockBufferLength	int	必选	缓冲区长度
	nTimeout_MS	int	必选	输入超时，单位毫秒，当小于零时，一直等待输入
	nFlagSound	int	必选	是否需要声音提示，0 否，1 是
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.1.1.2.4.10 计算 Mac

根据指定的用户密钥计算MAC。

接口名称	pinpad_calculate_mac			
功能描述	使用用户密钥计算 MAC			
接口格式	int pinpad_calculate_mac(unsigned char* pData, int nDataLength, int nMACFlag, unsigned char* pMACOutBuffer, int nMACOutBufferLength)			
参数说明	pData	unsigned char	必选	数据
	nDataLength	int	必选	数据长度
	nMACFlag	int	必选	0：X99，1：ECB
	pMACOutBuffer	unsigned char	必选	保存 MAC 的缓冲区

	nMACOutBufferLength	int	必选	缓冲区长度
返回值	1. 大于等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表			

A.1.1.2.4.11 设置 Pin 长度

指定Pin输入的长度。

接口名称	pinpad_set_pin_length			
功能描述	设置 Pin 的最大限制长度			
接口格式	int pinpad_set_pin_length(int nLength, int nFlag)			
参数说明	nLength	int	必选	长度范围［0x00,0x0D］
	nFlag	int	必选	0：最小长度，1：最大长度
返回值	1. 大于等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表			

A.1.1.2.4.12 设置 Pin 回调

Pinblock回调

接口名称	pinpad_set_pinblock_callback			
功能描述				
接口格式	int pinpad_set_pinblock_callback (KEYEVENT_NOTIFIER notifier)			
参数说明	notifier	KEYEVENT_NOTIFIER	必选	
返回值	针对内置密码键盘的情况。此接口设置内置密码键盘按键事件回调			

A.1.1.2.5 打印机

驱动接口信息

设备名称	CLOUD_DEVICE_PRINTER
设备用途	控制智能终端上打印机设备
设备驱动文件	/system/lib/libUnionpayCloudPos.so

驱动调用方法	void* pHandle = dlopen(“libUnionpayCloudPos.so”, RTLD_LAZY);//调用驱动
	(“函数名”)dlsym(pHandle, “函数名”);//得到函数句柄

函数列表

打开设备	printer_open
关闭设备	printer_close
传输打印数据	printer_write
开启打印	printer_begin
结束打印	printer_end

A.1.1.2.5.1 打开设备

打开打印机设备。

接口名称	printer_open
功能描述	打开一个已经存在的打印设备
接口格式	int printer_open();
参数说明	无
返回值	1. 等于 0，打开成功 2. 小于 0，错误；参考附录一：错误定义表

A.1.1.2.5.2 关闭设备

关闭打印机设备。

接口名称	printer_close
功能描述	关闭已打开的打印设备
接口格式	int printer_close();
参数说明	无
返回值	1. 等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表

A.1.1.2.5.3 传输打印数据

写入打印数据并开启打印。

接口名称	printer_write			
功能描述	从当前点连续打印，打印的数据存放在缓冲区			
接口格式	int printer_write(unsigned char* pData, int nDataLength);			
参数说明	pData	unsigned char	必选	打印数据或打印控制命令
	nDataLength	int	必选	打印数据或控制命令长度

返回值	1. 等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表

A.1.1.2.5.4 开启打印

开启打印，并分配资源，在printer_write之前调用。

接口名称	printer_begin
功能描述	准备开始打印
接口格式	int printer_begin();
参数说明	无
返回值	1. 等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表

A.1.1.2.5.5 结束打印

结束打印，并释放资源。在所有数据写完后调用。

接口名称	printer_end
功能描述	结束打印
接口格式	int printer_end();
参数说明	无
返回值	1. 等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表

A.1.1.2.6 证书管理与加密运算模块

驱动接口信息

设备名称	CLOUD_DEVICE_PKCS
设备用途	控制智能终端上打印机设备
设备驱动文件	/system/lib/libUnionpayCloudPos.so

驱动调用方法	void* pHandle = dlopen(“libPKCS11Wrapper.so”, RTLD_LAZY);//调用驱动
	(“函数名”)dlsym(pHandle, “函数名”);//得到函数句柄

函数列表

打开硬件证书管理与加密运算模块设备	HSM_OSM_OPEN
关闭硬件证书管理与加密运算模块设备	HSM_OSM_CLOSE
保存安全证书	HSM_OSM_SAVE_OBJECT
删除指定证书	HSM_OSM_DELETE_OBJECT
删除所有证书	HSM_OSM_DELETE_ALL

读取证书	HSM_OSM_LOAD_OBJECT

A.1.1.2.6.1 打开硬件证书管理与加密运算模块设备

函数	int (*HSM_OSM_OPEN)()
描述	打开硬件证书管理与加密运算模块设备
参数	无
返回值	1. 等于 0，打开成功 2. 小于 0，错误

A.1.1.2.6.2 关闭硬件证书管理与加密运算模块设备

函数	int (*HSM_OSM_CLOSE)()
描述	关闭硬件证书管理与加密运算模块设备
参数	无
返回值	1. 等于 0，关闭成功 2. 小于 0，错误

A.1.1.2.6.3 保存安全证书

函数	int (*HSM_OSM_SAVE_OBJECT)(HSM_OBJECT_PROPERTY* pObjectProperty, unsigned char* pObjectData, unsigned int nDataLength, HSM_OBJECT_DATA_TYPE nDataType)			
描述	保存安全证书至硬件证书管理与加密运算模块设备			
参数	pObjectProperty	HSM_OBJECT_PROPERTY	必选	证书属性。
	pObjectData	unsigned char	必选	证书数据
	nDataLength	unsigned int	必选	证书数据长度
	nDataType	HSM_OBJECT_DATA_TYPE	必选	证书格式。
返回值	1. 等于 0，保存成功。 2. 小于 0，保存失败。			

A.1.1.2.6.4 删除指定证书

函数	int (*HSM_OSM_DELETE_OBJECT)(HSM_OBJECT_PROPERTY* pObjectProperty,

	char* pPIN, unsigned int nPINLength)			
描述	从硬件证书管理与加密运算模块设备中删除指定证书			
参数	pObjectProperty	HSM_OBJECT_PROPERTY	必选	证书属性。
	pPIN	char	必选	PIN 值。
	nPINLength	unsigned int	必选	PIN 值长度。
返回值	1. 等于 0，删除成功 2. 小于 0，错误			

A.1.1.2.6.4.1 删除所有证书

函数	int (*HSM_OSM_DELETE_ALL)(char* pPIN, unsigned int nPINLength)			
描述	从硬件证书管理与加密运算模块中设备删除所有证书			
参数	pPIN	char	必选	保留的 Pin 值
	nPINLength	unsigned int	必选	PIN 值长度
返回值	1. 等于 0，删除成功 2. 小于 0，错误			

A.1.1.2.6.5 读取证书

函数	int (*HSM_OSM_LOAD_OBJECT)(unsigned int nIndex,HSM_OBJECT_PROPERTY* ObjectProperty, unsigned char* byteData, unsigned int nDataLength, HSM_OBJECT_DATA_TYPE nDataType)			
描述	从硬件证书管理与加密运算模块内读取证书			
参数	nIndex	unsigned int	必选	证书索引号。
	pObjectProperty	HSM_OBJECT_PROPERTY	必选	证书属性。
	byteData	unsigned char	必选	证书数据。
	nDataLength	unsigned int	必选	证书数据长度。
	nDataType	HSM_OBJECT_DATA_TYPE	必选	证书格式。
返回值	1. 等于 0，读取成功 2. 小于 0，读取失败			

A.1.1.2.6.6 参数类型说明

银联基于 Android 系统智能销售点终端支持三种证书类型，定义如下：

支持三种证书类型：

证书类型	定义(ENUM HSM_OBJECT_TYPE)
私钥证书	HSM_OBJECT_TYPE_private_key
公钥证书	HSM_OBJECT_TYPE_public_key
CERT证书	HSM_OBJECT_TYPE_cert

其中证书文件格式支持四种格式，定义如下：

文件证书格式	定义(ENUM HSM_OBJECT_DATA_TYPE)
pem证书格式	HSM_OBJECT_DATA_TYPE_pem
der编码证书格式	HSM_OBJECT_DATA_TYPE_der
PKCS #7证书格式	HSM_OBJECT_DATA_TYPE_p7b
PKCS #12证书格式	HSM_OBJECT_DATA_TYPE_pfx

——证书数据结构 HSM_OBJECT_PROPERTY.结构的定义如下：

字段名	类型	说明
strID	unsigned char[32]	证书结构体 ID 号
strLabel	unsigned char[32]	证书名称
strPassword	unsigned char[32]	获取正式密码
nObjectType	HSM_OBJECT_TYPE	证书类型标识

A.1.1.2.7 错误定义表

错误名	错误值	说明
EDEVICENAME	-1	设备名称不存在
EIO	-2	底层 I/O 错误
EUNKOWN	-10	未知错误
E_HAL_UNKOWN	-100	未知错误
E_HAL_ARG	-101	输入参数错误
E_HAL_IO	-102	底层I/O错误

A.1.1.3 IC卡内核C接口

银联基于Android的智能销售点终端要求终端厂商预制支持处理中国金融集成电路卡（PBOC）3.0内核处理接口。以下以C语言来描述此类接口定义。

A.1.1.3.1 PBOC tag存取功能

A.1.1.3.1.1 检查 Tag 的值是否存在

函数	int (EMV_IS_TAG_PRESENT)(int tagName)
描述	判断 EMV tag 的值是否存在

参数	tagName	int	必选	tag 名
返回值	1. 小于 0，不存在 2. 大于等于 0， 存在。长度值			

A.1.1.3.1.2 读取 Tag 的值

函数	int (*EMV_GET_TAG_DATA)(int tagName, unsigned char *pTagValue, int tagValueLen)			
描述	读取EMV tag的值			
参数	tagName	int	必选	tag 名
	pTagValue	unsigned char	必选	tag 值
	tagValueLen	int	必选	pTagValue 最大长度
返回值	1. 小于 0， 获取失败 2. 大于等于 0 获取 tag 的值长度			

A.1.1.3.1.3 设置 Tag 的值

函数	int (*EMV_SET_TAG_DATA)(int tagName, unsigned char *pTagValue, int tagValueLength)			
描述	设置EMV tag的值,如果该tag的值不存在，则返回 -1，否则返回tag的长度			
参数	tagName	int	必选	tag 名
	pTagValue	unsigned char	必选	tag 值（TLV 的 V）
	tagValueLength	int	必选	tag 值的长度
返回值	1. 大于等于 0，设置成功 2. 小于 0，设置失败			

A.1.1.3.1.4 读取 EMV tag 的值

函数	int (*EMV_GET_TAG_LIST_DATA)(int* tagNames, int tagCount, unsigned char *pTagsValue, int tagsValueLen)			
描述	读取EMV tag的值			
参数	tagNames	int	必选	一组tag名（TLV格式中的T）
	tagCount	int	必选	tag数量
	pTagsValue	unsigned char	必选	一组tag值（每一个tag值的格式是TLV）
	tagsValueLen	int	必选	pTagsValue最大长度

返回值	1.小于0，获取失败 2.大于等于0，获取成功。pTagsValue值长度

A.1.1.3.1.5 非接预处理

函数	int (*EMV_PREPROCESS_QPBOC)()			
描述	非接预处理：AID 非接限额处理（只有在qPBOC交易的时候需要在EMV_INITIALIZE、SetAmount之后调用）			
参数	无			
返回值	1. 小于 0， 失败，金额受限，提示不能使用非接界面； 2.等于-1， 金额超限；-2 金额为 0 3. 大于等于 0， 可以使用非接界面交易			

A.1.1.3.1.6 侦听卡片事件

函数	void (*CARD_EVENT_OCCURED)(int eventType)			
描述	侦听卡片事件，当有卡片靠近或插入的时候，触发该事件 即 当调用OPEN_READER后，如果有卡片靠近或插入时，回调该函数			
参数	eventType	int	必选	1.SMART_CARD_EVENT_INSERT_CARD = 0; 2.SMART_CARD_EVENT_REMOVE_CARD = 1;
返回值	无			

A.1.1.3.1.7 EMV kernel 处理回调函数

函数	void (*EMV_PROCESS_CALLBACK)(unsigned char *pData)			
描述	EMV kernel处理回调函数，pData包含2个字段（status，desc）			

参数	pData	unsigned char	必选	1.STATUS_ERROR = 0; //执行报错
				2. STATUS_CONTINUE = 1; //还未完成
				3. STATUS_COMPLETION = 2; //完成

				4.desc 取值如下表所示
返回值	无			

stastus取值说明：

status	desc取值

status = STATUS_COMPLETION	APPROVE_OFFLINE = 1; //Transaction approved Offline
	APPROVE_ONLINE = 2; //Transaction approved Online
	DECLINE_OFFLINE = 3; //Transaction declined Offline
	DECLINE_ONLINE = 4; //Transaction declined Online

status = STATUS_ERROR	SUCCESS = 0; //SUCCESS
	ERROR_NO_APP = 1; //No Application Selected when Application Select
	ERROR_APP_BLOCKED = 2; //card return 6A81 when Application Select
	ERROR_APP_SELECT = 3; //Error when Application Select
	ERROR_INIT_APP = 4; //Error when Initialize Application Data
	ERROR_EXPIRED_CARD = 5 //Any error after initialApplication should halt the process
	ERROR_APP_DATA = 6; //Error when Read Application Data
	ERROR_DATA_INVALID = 7; // 卡中有非法数据
	ERROR_DATA_AUTH = 8; // 脱机认证失败
	ERROR_GEN_AC = 9; //Generate AC error when Transaction Process
	ERROR_PROCESS_CMD = 10; //Process Command ERROR
	ERROR_SERVICE_NOT_ALLOWED = 11; //Service not Allowed
	ERROR_PINENTERY_TIMEOUT = 12; //PIN Entry timeout
	ERROR_OFFLINE_VERIFY = 13; //Check Offline PIN Error when Cardholder Verify
	ERROR_NEED_ADVICE = 14; //Communication Error with Host, but the card need advice, halted the transaction
	ERROR_USER_CANCELLED = 15
	ERROR_AMOUNT_OVER_LIMIT = 16; // 金额超限 used for qpboc
	ERROR_AMOUNT_ZERO = 17; // 金额不能为0 used for qpboc

status = STATUS_CONTINUE	EMV_CANDIDATE_LIST = 1; //notify Application show Application Candidate List
	EMV_APP_SELECTED = 2; //Application Select Completed
	EMV_READ_APP_DATA = 3; //Read Application Data Completed
	EMV_DATA_AUTH = 4; //Data Authentication Completed
	EMV_OFFLINE_PIN = 5; // notify Application need to enter offline PIN, 但应用只提示“请输入密码”界面，处理逻辑还是由EMV内核处理
	EMV_ONLINE_ENC_PIN = 6; //notify Application prompt Caldholder enter Online PIN
	EMV_PIN_BYPASS_CONFIRM = 7; //notify Application confirm to Accepted PIN Bypass or not
	EMV_PROCESS_ONLINE = 8; //notify Application to Process Online

	EMV_ID_CHECK = 9; //notify Application Check Cardholder's Identification

A.1.1.3.1.8 设置 kernerlType

函数	int (*EMV_SET_KERNERL_TYPE)(unsigned char kernerlType)			
描述	设置kernerlType			

参数	kernerlType	unsigned char	必选	1.PBOC_KERNAL = 1; //借贷记
				2.QPBOC_KERNAL = 2; //QPBOC
				3.UPCASH_KERNAL = 3; //电子现金

返回值	1. 大于等于 0， 返回成功 2. 小于 0，返回失败			

A.1.1.3.1.9 打开读卡器

函数	int (*OPEN_READER)(int reader)			
描述	打开读卡器，并等待读卡（开始轮询卡片）			
参数	reader	int	必选	阅读器类型
返回值	1. 等于-1， 打开读卡器失败 （如果需要打开两个读卡器，只要有一个成功，就报成功）			

A.1.1.3.1.10 关闭读卡器

函数	void (*CLOSE_READER)(int reader)			
描述	关闭读卡器，并释放资源			
参数	reader	int	必选	阅读器类型 1. 0 关闭所有读卡器 2. 1 只关闭接触式读卡器 3. 2 只关闭非接触式读卡器
返回值	无			

A.1.1.3.1.11 获取卡片类型

函数	int (*GET_CARD_TYPE)()
描述	获取卡片类型，当前内核正在处理的卡片类型：接触or非接
参数	无
返回值	1. CARD_CONTACT = 1 2. CARD_CONTACTLESS = 2 3. NO_CARD = -1

A.1.1.3.1.12 获取卡片 ATR 值

函数	int (*GET_CARD_ATR)(unsigned char * pAtr)			
描述	获取卡片ATR值			
参数	pAtr	unsigned char	必选	
返回值	1. CARD_CONTACT = 1 2. CARD_CONTACTLESS = 2 3. NO_CARD = -1			

A.1.1.3.1.13 调用 EMV kernal 处理宿主函数

函数	int (*EMV_PROCESS_NEXT)()
描述	调用EMV kernal处理宿主函数
参数	无
返回值	1. 大于等于 0， 返回成功 2. 小于 0，返回失败

A.1.1.3.1.14 读取 EMV Kernal 版本

函数	int (*EMV_GET_VERSION_STRING)(unsigned char* buffer, int bufferLen)			
描述	读取EMV Kernal版本，返回值为版本长度			
参数	buffer	unsigned char	必选	版本值
	bufferLen	int	必选	版本值的最大长度
返回值	版本实际长度			

A.1.1.3.2 EMV交易处理功能

A.

——

——smart_card_transmit 函数定义如下：

函数	typedef int (*smart_card_transmit)(int handle, unsigned char *p_APDU, unsigned short APDU_length, unsigned char *p_response, unsigned short *p_response_length);			
描述	IC卡APDU交互函数			
参数	p_APDU	unsigned char*	必选	输入的 APDU 指针
	APDU_length	unsigned short	必选	APDU 长度
	p_response	unsigned char*	必选	处理结果的指针
	p_response_length	unsigned short*	必选	处理结果的长度
返回值	1. 等于 1，设置成功 2. 小于 0，设置失败			

——emv_process_next_completed 函数定义如下：

函数	typedef void (*emv_process_next_completed)(unsigned char status, unsigned char info);			
描述	IC卡APDU交互函数，状态与信息对应关系请参照对应表			
参数	status	unsigned char	必选	状态
	Info	unsigned char	必选	信息
返回值	无			

A.1.1.3.3 其他功能

B.初始化

void (*EMV_TRANS_INITIALIZE)()

函数	void (*EMV_TRANS_INITIALIZE)()
描述	初始化
参数	无
返回值	无

C.设置交易金额

int (*EMV_SET_TRANS_AMOUNT)(unsigned char* amount)

函数	int (*EMV_SET_TRANS_AMOUNT)(unsigned char* amount);
描述	设置交易金额，以分为单位

参数	amount	unsigned char	必选	向内核输入金额，以分为单位， 以'\0'作为结束符，例如"9999"表示 99 元 9 毛 9 分
返回值	1.大于等于 0，设置成功 2.小于 0，设置失败			

C.设置交易类型

int (*EMV_SET_TRANS_TYPE)(unsigned char transType)

函数	int (*EMV_SET_TRANS_TYPE)(unsigned char transType)			
描述	设置交易金额			
参数	transType	unsigned char	必选	向内核输入交易类型,类型参看交易类型表
返回值	无			

交易类型表	
类型名	类型值
TRANS_GOODS_SERVICE	0x00
TRANS_CASH	0x01
TRANS_INQUIRY	0x31
TRANS_TRANSFER	0x05
TRANS_PAYMENT	0x06
TRANS_ADMIN	0x07
TRANS_CASHBACK	0x09
TRANS_CARD_RECORD	0x0A
TRANS_EC_BALANCE	0x0B电子现金余额查询
TRANS_PRE_AUTH	0x03预授权
TRANS_CASH_LOAD	0x63现金圈存
TARNS_DIRECT_LOAD	0x60指定账户圈存
TRANS_UNDIRECT_LOAD	0x62非指定账户圈存

D.设置其他金额

int (*EMV_SET_OTHER_AMOUNT)(unsigned char* amount)

函数	int (*EMV_SET_OTHER_AMOUNT)(unsigned char* amount)			
描述	设置其他金额，以分为单位，一般默认为0，用于国外某种交易类型(TRANS_CASHBACK 0x09)			
参数	amount	unsigned int	必选	向内核输入金额，以分为单位，以'\0'作为结束符，例如

				"9999"表示 99 元 9 毛 9 分
返回值	1.大于等于 0，设置成功 2.小于 0，设置失败			

E.清除AID参数

int (*EMV_AIDPARAM_CLEAR)()

函数	int (*EMV_AIDPARAM_CLEAR)()
描述	清除AID参数
参数	无
返回值	1.大于等于 0，清除成功 2.小于 0，清除失败

F.增加AID参数

int (*EMV_AIDPARAM_ADD)(unsigned char* AIDParam, int dataLength)

函数	int (*EMV_AIDPARAM_ADD)(unsigned char* AIDParam, int dataLength)			
描述	增加AID参数			
参数	AIDParam	unsigned char*	必选	POS 参数传递报文中的 62 域值
	dataLength	Int	必选	长度
返回值	1.小于 0，添加失败 2.大于等于 0，添加成功			

--结构体 AIDPARAM定义如下：

字段名	类型	说明
aid_length	unsigned char	Aid' length
aid	unsigned char[16]	[BCD] Application Identifier
app_lable	unsigned char[17]	[ASC] Application Label
app_preferred_name	unsigned char[17]	[ASC] Application preferred name
app_priority	unsigned char	Application Priority
term_floor_limit	unsigned char[4]	[HEX]Terminal Floor Limit
term_action_code_default	unsigned char[5]	[BCD]Terminal Action Code - Default
term_action_code_denial	unsigned char[5]	[BCD]Terminal Action Code - Denial
term_action_code_online	unsigned char[5]	[BCD]Terminal Action Code - Online
target_percentage	unsigned char	[HEX] Target Percentage
threshold Value	unsigned char[4]	[HEX] threshold Value
max_target_percentage	unsigned char	[HEX] Maximum Target Percentage
acquirer_id	unsigned char[6]	[BCD] Acquirer Identifier
mechant_category_code	unsigned char[2]	[BCD]Merchant Category Code

merchant_id	unsigned char[15]	[ASC] Merchant Identifier
app_version_number	unsigned char[2]	[BCD]Application Version Number
pos_entry_mode	unsigned char	[HEX]Point-of-Service(POS) Entry Mode
trans_refer_currency_code	unsigned char[2]	Transaction Reference Currency Code
trans_Refer_Currency_Exponent	unsigned char	Transaction Reference Currency Exponent
default_ddol_length	unsigned char	Default Dynamic Data Authentication Data Object List(DDOL)
default_ddol	unsigned char[128]	Default Dynamic Data Authentication Data Object List(DDOL)
support_online_pin	unsigned char	supportOnlinePin[0] = 0 means the Application unsupported , others means supported
support_aid_partial	unsigned char	supportAIDPartial[0] = 0 means the Application unsupported AID Partial, others means supported
contactless_limit	unsigned char[4]	[HEX] unsigned char
contactless_floor_limit	unsigned char[4]	[HEX] QPBOC Contactless Floor Limit
cvm_limit	unsigned char[4]	[HEX]QPBOC CVM Limit
ec_term_trans_limit	unsigned char[6]	[BCD] 电子现金终端交易限额

G.清除CAPK参数

函数	int (*EMV_CAPKPARAM_CLEAR)()
描述	清除CAPK参数
参数	无
返回值	1.小于 0，清除失败 2.大于等于 0，清除成功

H.增加CAPK参数

函数	int (*EMV_CAPKPARAM_ADD)(unsigned char* CAPKParam, int dataLength)			
描述	增加CAPK参数			
参数	CAPKParam	unsigned char	必选	
	dataLength	int	必选	
返回值	1.小于 0，添加失败 2.大于等于 0，添加成功			

--结构体 CAPKPARAM定义如下：

字段名	类型	说明
rid	unsigned char[5]	[BCD]Registered Application Provider Identifier
capki	unsigned char	[BCD] Certificate Authority Public Key Index

hash_ind	unsigned char	[ASC] Application Label
arith_ind	unsigned char	[ASC] Certificate Authority Public Key Algorithm Indicator
modul_len	unsigned char	The Length of Certificate Authority Public Key Modulu
modul	unsigned char[248]	[BCD]
exponent_len	unsigned char	The Length of Certificate Authority Public Key Exponent
exponent	unsigned char[3]	Certificate Authority Public Key Exponent
check_sum	unsigned char[20]	[BCD]Certificate Authority Public Key Check Sum
expiry	unsigned char[8]	[ASC]Certificate Expiration Date

I.设置EMV终端参数

函数	int (*EMV_TERMINAL_PARAM_SET)(IC_TERMINAL_INFO * terminalParam)			
描述	设置IC相关的终端参数参数			
参数	terminal_param	IC_TERMINAL_INFO	必选	字符串，参看结构体定义
返回值	1.小于 0，添加失败 2.大于等于 0，添加成功			

--结构体 CAPKPARAM定义如下：

字段名	类型	说明
terminal_country_code	unsigned char[2]	9F1A [BCD]:Terminal Country
Tid	unsigned char[8]	9F1C [ASC]
Ifd	unsigned char[8]	9F1E [ASC] : IFD Serial Number
transaction_currency_code	unsigned char[2]	5F2A [BCD]
terminal_capabilities	unsigned char[3]	9F33 [BIN]
terminal_type	unsigned char	9F35 [BCD]
transaction_currency_exponent	unsigned char	5F36 [BCD]
additional_terminal_capabilities	unsigned char[3]	9F40 [BIN]
merchant_name_length;	unsigned char	
merchant_name	unsigned char[20]	9F4E [ASC]
status_check_support	unsigned char	qpbc : 是否支持状态检查 0-不支持； 1-支持

L.清除证书回收名单

函数	int (*EMV_REVOKED_CERT_CLEAR)()
描述	清除证书回收名单
参数	无
返回值	1. 等于 1，设置成功 2. 小于 0，设置失败

M.增加证书回收名单

函数	int (*EMV_REVOKED_CERT_ADD)(unsigned char* BlackCert)			
描述	增加证书回收名单			
参数	blackcert	unsigned char	必选	参看回收证书结构体定义
返回值	1. 等于 1，设置成功 2. 小于 0，设置失败			

--结构体 BLACKCARD定义如下：

字段名	类型	说明
Rid	unsigned char[5]	[ASC]RegisteredApplication Provider Identifier
Capki	unsigned char	[HEX] Certificate Authority Public Key Index

N.该EMV交易是否需要上送Advice

函数	int (*EMV_IS_NEED_ADVICE)()
描述	该EMV交易是否需要上送Advice
参数	无
返回值	返回1, 需要上送Advice; 返回0,不需要上送Advice.

O.该EMV交易是否需要签名

函数	int (*EMV_IS_NEED_SIGNATURE)()
描述	设置其他金额
参数	无
返回值	返回1, 需要签名; 返回0,不需要签名.

P.设置是否强制联机

函数	int (*EMV_SET_FORCE_ONLINE)(int flag)			
描述	设置是否强制联机			
参数	flag	int	必选	1：是 0： 否
返回值	无			

Q.读取卡片交易记录

函数	int (*EMV_GET_CARD_RECORD)(unsigned char *data, int dataLen)			
描述	读取卡片交易记录，返回交易记录数			
参数	dataLen	int	必选	交易记录最大长度
	data	unsigned char	必选	交易记录, [TO DO]：每一条记录的格式定义
返回值	1. 大于等于 0，交易记录数 2. 小于 0，获取失败			

R.获取应用选择列表

int (*EMV_GET_CANDIDATE_LIST)(unsigned char* data, int dataLen)

函数	int (*EMV_GET_CANDIDATE_LIST)(unsigned char* data, int dataLen)			
描述	获取应用选择列表，返回列表数目，在EMV_CANDIDATE_LIST这个状态的时候调用			
参数	data	unsigned char*	必选	aid 显示信息列表，每一个 aid 显示信息以"LV"编码，V 的编码格式是 gb2312 编码
	dataLen	int	必选	data 最大长度
返回值	1.小于 0，获取失败 2.大于等于 0，aid 列表数			

--结构体 应用信息 定义如下：

字段名	类型	说明
aid_length	unsigned char	[BCD]
aid	unsigned char[16]	[HEX] Certificate Authority Public Key Index
app_label	unsigned char[17]	Application Label
app_preferred_name	unsigned char[17]	Application Preferred Name
priority_exist	unsigned char	Application Priority exist flag
app_priority	unsigned char	Application Priority

S.设置应用选择结果

函数	int (*EMV_SET_CANDIDATE_LIST_RESULT)(int index)			
描述	设置应用选择结果			
参数	index	unsigned char	必选	应用序号，从 0 开始
返回值	1.小于 0，设置失败 2.大于等于 0，设置成功			

T.设置持卡人证件检查结果
根据IDType（9F62）、IDNumber(9F61)，设置证件检查结果

函数	int (*EMV_SET_ID_CHECK_RESULT)(int result)			
描述	设置持卡人证件验证结果			
参数	result	int	必选	检查结果，0 表示不成功，1 表示成功
返回值	1.小于 0，设置失败 2.大于等于 0，设置成功			

U.设置OnlinePIN是否输入

函数	int (*EMV_SET_ONLINE_PIN_ENTERED)(int result)			
描述	设置OnlinePIN是否输入			
参数	result	unsigned int	必选	1，用户已输入 PIN; 0，用户未输入 PIN
返回值	1.小于 0，设置失败 2.大于等于 0，设置成功			

V.确认是否允许持卡人Bypass PIN（不输入PIN）

函数	int (*EMV_SET_PIN_BYPASS_CONFIRMED)(int result)			
描述	确认是否允许持卡人Bypass PIN（不输入PIN）			
参数	result	int	必选	BYPASS PIN 输入结果，0 表示未输入，1 表示已输入
返回值	1.小于 0，设置失败 2.大于等于 0，设置成功			

W.设置联机认证结果

int emv_set_online_result(unsigned char result,

unsigned char * is_suer_resp_data,
unsigned char is_suer_resp_data_length)

result:
0，联机通讯成功，并收到交易成功应答
1，联机通讯失败
2，联机通讯成功，但收到交易拒绝应答（Response Code != "00"）

函数	int (*EMV_SET_ONLINE_RESULT)(int result, unsigned char*respCode, unsigned char* issuerRespData, int issuerRespDataLength)			
描述	设置联机认证结果			
参数	result	int	必选	1.-1 通讯失败； 2.0 拒绝； 3.1 接受
	respCode	unsigned char	必选	两位的应答码
	issuerRespData	unsigned char	必选	55 域数据
	issuerRespDataLength	int	必选	数据长度
返回值				

A.1.1.4 外设访问权限控制模块

1、访问证书管理与加密运算模块权限
<uses-permission android:name="android.permission.CLOUDPOS_SAFE_MODULE" />
2、访问磁条卡读卡器设备权限
<uses-permission android:name="android.permission.CLOUDPOS_MSR" />
3、访问接触式IC卡阅读器权限
<uses-permission android:name="android.permission.CLOUDPOS_SMARTCARD" />
4、访问非接触IC卡读卡设备权限
<uses-permission android:name="android.permission.CLOUDPOS_CONTACTLESS_CARD" />
5、访问打印机设备权限
<uses-permission android:name="android.permission.CLOUDPOS_PRINTER" />
6、访问PIN输入设备权限
<uses-permission android:name="android.permission.CLOUDPOS_PINPAD" />
7、PIN输入设备计算Pinblock权限
<uses-permission android:name="android.permission.CLOUDPOS_PIN_GET_PIN_BLOCK" />
8、PIN输入设备计算Mac权限
<uses-permission android:name="android.permission.CLOUDPOS_PIN_MAC" />
9、PIN输入设备加密数据权限
<uses-permission android:name="android.permission.CLOUDPOS_PIN_ENCRYPT_DATA" />
10、PIN输入设备更新终端主密钥权限
<uses-permission android:name="android.permission.CLOUDPOS_PIN_UPDATE_MASTER_KEY" />
11、PIN输入设备更新用户密钥权限
<uses-permission android:name="android.permission.CLOUDPOS_PIN_UPDATE_USER_KEY" />

A.1.2 多应用管理要求

A.1.3 银行卡支付服务及客户端要求

A.1.3.1 银行卡支付调用接口

A.1.3.1.1 调用交易接口

该接口定义为调用收单应用，完成包括磁条卡和PBOC交易。

发起调用接口：通过调用该接口，跳转到银联交易支付模块进行交易。
接口类型：Android IPC通信AIDL接口
接口权限：com.cloudpos.CloudPosPaymentClient.permission.CLOUDPAY
接口方法：String payCash(String jsonData);
输入参数：
String类型，JSON数据格式，属性说明参照文档
{
“AppID”：“”,
“AppName”:“”,
“TransType”:“”,
“TransAmount”:“”,
“TransIndexCode”:“”,
“ReqTransDate”:“”,
“ReqTransTime”:“”,
“cardInfo2”：“”，
“cardInfo3”：“”
}
输出参数：
String类型，JSON数据格式，属性说明参照文档附录
消费返回参数集
{
“AppID”：“”,
“AppName”:“”,
“TransType”:“”,
“TransAmount”:“”,
“TransIndexCode”:“”,
“ReqTransDate”:“”
“ReqTransTime”:“”,
“TransDate”:“”,
“TransTime”:“”,
“RespCode”:“”,
“RespDesc”:“”,
（以下参数可选）
“OptCode”:“”,
“CardNum”:“”,
“BatchNum”:“”,
“CertNum”:“”,
“ReferCode”:“”,
“Reference”:“”,

“FeeAmount”:“”
}

A.1.3.1.2 调用查询接口

接口类型：Android IPC 通信AIDL接口
接口权限：com.cloudpos.CloudPosPaymentClient.permission.CLOUDPAY
接口方法：String getPayInfo(String jsonData)
输入参数：
String类型，JSON数据格式，属性说明参照
{
“AppID”：“”,
“AppName”：“”,
“TransType”:“”,
“TransIndexCode”：“”,
“ReqTransDate”:“”
}
输出参数：
String类型，JSON数据格式，属性说明参照
查询返回参数集
{
“AppId”:“”,
“AppName”:“”,
“TransType”:“”,
“TransIndexCode”:“”,
“ReqTransDate”:“”,
“ReqTransTime”:“”,
“TransDate”:“”,
“TransTime”:“”,
“RespCode”:“”,
“RespDesc”:“”,
(以下参数可选)
“TransAmount”:“”
}

A.1.3.1.3 终端信息查询接口

接口类型：Android IPC 通信AIDL接口
接口权限：com.cloudpos.CloudPosPaymentClient.permission.CLOUDPAY
接口方法：String getPOSInfo(String jsonData)
输入参数：
String类型，JSON数据格式，属性说明参照
{
“AppID”：“”,
“AppName”：“”,
“TransType”:“”,

“ReqTransDate”:“”
}
输出参数：
String类型，JSON数据格式，属性说明参照
查询返回参数集
{
“MerchantID”:“”,
“TerminaID”:“”,
“OperatorID”:“”,
“RespCode”:“”,
“RespDesc”:“”,
}

A.1.3.1.4 参数解释

注：这里的请求是指第三方收银应用向银联支付应用发送请求报文； 应答是指银联支付应用向第三方收银应用发送响应报文。

第三方应用与支付之间的交换消息中，各数据元类型如下所列：
A 字母。
AN 字母和/或数字
ANS 字母、数字和/或特殊符号
AS 字母和/或特殊符号
N 数字
YY 年
MM 月
DD 日。
hh 时。
mm 分
ss 秒
注：可选的意思是有值返回，第三方可以根据需求选择是否获取，此类参数非必须参数。

A.1.3.1.5 消费

表2 消费请求参数列表

字段名	变量名	是否必填	类型(长度)	说明
应用 ID	AppId	是	N	第三方应用的唯一标识，由银联指定。
应用名称	AppName	是	AN 或中文	第三方应用名称，由第三方应用自行定义。
交易类型	TransType	是	N(2)	交易类型可参考 A.1.3.1.8 章节的详述。
交易金额	TransAmount	是	N(12)	交易金额的币种均为人民币,保留两位小数，示例：消费金额为 100 元，则交易金额应为 00000010000。
交易索引号	TransIndexCode	是	N(最大 16)	由第三方应用提供，此值用

				来标识一次交易请求，一天内不能重复。
交易请求日期	ReqTransDate	是	YYMMDD(6)	第三方交易发起请求的日期
交易请求时间	ReqTransTime	是	Hhmmss(6)	第三方交易发起请求的具体时间。
磁道 2 信息	cardInfo2	可选	NS	磁条卡第二磁道信息
磁道 3 信息	cardInfo3	可选	NS	磁条卡第三磁道信息

表3　消费返回通知参数列表

字段名	变量名	是否返回	类型	说明
应用 ID	AppId	是	N	第三方应用的唯一标识，由银联指定。
应用名称	AppName	是	AN 或中文	第三方应用名称，由第三方应用自行定义。
交易类型	TransType	是	N(2)	返回的交易类型同请求的交易类型，中文显示。
交易金额	TransAmount	是	N(12)	交易金额的币种均为人民币,保留两位小数，示例：消费金额为 100 元，则交易金额应为 00000010000。
交易索引号	TransIndexCode	是	N(最大 16)	此值同第三方请求的交易索引号，返回值和请求值相同。
交易请求日期	ReqTransDate	是	YYMMDD(6)	第三方交易发起请求的日期
交易请求时间	ReqTransTime	是	Hhmmss(6)	第三方交易发起请求的时间
交易应答日期	TransDate	是	MMDD(4)	支付系统的处理交易成功的日期。
交易应答时间	TransTime	是	Hhmmss(6)	支付系统的处理交易成功的时间。
返回码	RespCode	是	AS(最大 6)	支付系统返回给第三方应用的应答码
返回码中文解释	RespDesc	是	中文	支付系统返回给第三方应用的应答码的对应中文解释，请参照所述。
操作员号	OptCode	可选	N(2)	操作员号只可能是 01-98，普通操作员登陆成功签到后才能使用第三方应用，请参照
卡号	CardNum	可选	N(最大 19)	前 5 位数字和末 4 位数字显示，其他位数显示为*。
批次号	BatchNum	可选	N(6)	交易中心返回的 6 位批次号，每批结算一次批次号加一。

凭证号	CertNum	可选	N(6)	6 位凭证号，相当于交易流水号，每一批次凭证号从 000000 开始递增。
参考号	ReferCode	可选	N(12)	12 位交易参考号是交易中心返回的随机编码，不会重复。
备注信息	Reference	可选	ANS	交易中心返回，可能为空值。
小费金额	FeeAmount	可选		备用

A.1.3.1.6 查询

表4 查询参数列表

字段名	变量名	是否必填	类型(长度)	说明
应用 ID	AppId	是	N	第三方应用的唯一标识，由银联指定。
应用名称	AppName	是	AN 或中文	第三方应用名称，由第三方应用自行定义。
交易类型	TransType	是	N(2)	交易类型可参考章的详述。
交易索引号	TransIndexCode	是	N(最大 16)	此值填写交易发起时的索引号，用来查询一次交易请求，和交易日期配合使用。
交易请求日期	ReqTransDate	是	YYMMDD(6)	第三方交易发起请求的日期，此值限制交易索引。

表5 查询返回参数列表

字段名	变量名	是否必填	类型	说明
应用 ID	AppId	是	N	第三方应用的唯一标识，由银联指定。
应用名称	AppName	是	AN 或中文	第三方应用名称，由第三方应用自行定义。
交易类型	TransType	是	N(2)	返回的交易类型同请求的交易类型，中文显示。
交易索引号	TransIndexCode	是		此值同第三方请求的交易索引号，返回值和请求值相同
交易请求日期	ReqTransDate	是	YYMMDD(6)	第三方交易发起请求的日期
交易请求时间	ReqTransTime	是	Hhmmss(6)	第三方交易发起请求的时间
交易应答日期	TransDate	是	YYMMDD(6)	支付系统的处理交易成功的日期。
交易应答时间	TransTime	是	Hhmmss(6)	支付系统的处理交易成功的时间。
返回码	RespCode	是	AS(最大 6)	支付系统返回给第三方应用的应答码，请参照本文 4.3

				章所述。
返回码中文解释	RespDesc	是	中文	支付系统返回给第三方应用的应答码的对应中文解释，请参照本文 4.3 章所述。
交易金额	TransAmount	可选	N(12)	查询的交易请求的金额，保留两位小数，示例：消费金额为 100 元，则交易金额应为 00000010000。

A.1.3.1.7 POS信息查询

表6 POS 信息查询请求参数列表

字段名	变量名	是否必填	类型(长度)	说明
应用 ID	AppID	是	N	第三方应用的唯一标识，由银联指定。
应用名称	AppName	是	AN 或中文	第三方应用名称，由第三方应用自行定义。
交易类型	TransType	是	N(2)	交易类型可参考本文 A.1.3.1.8 章节的详述。
交易请求日期	ReqTransDate	是	YYMMDD(6)	第三方发起请求的日期。

表7 POS 信息查询返回通知参数列表

字段名	变量名	是否返回	类型(长度)	说明
商户号	MerchantID	是	N	商户 ID，唯一标志商户
终端号	TerminaID	是	N	终端 ID，唯一标志终端
操作员号	OperatorID	是	N	操作员 ID，唯一标志操作员
返回码	RespCode	是	AS(最大 6)	支付系统返回给第三方应用的应答码的对应中文解释，请参照本文 A.1.3.1.8 章节所述。
返回码中文解释	RespDesc	是	中文	支付系统返回给第三方应用的应答码的对应中文解释，请参照本文 4.3 章所述。

A.1.3.1.8 交易类型

接口类型参数值	描述
1	消费
2	查询消费信息
3	查询 POS 信息

A.1.3.1.9 返回应答码表

交易返回 POS 终端时都有应答码，可以把操作分为以下几类：

A：交易成功

B：交易失败，可重试

C：交易失败，不需要重试

D：交易失败，终端操作员处理

E：交易失败，系统故障，不需要重试

F：交易失败，支付客户端未处理

1. 1：如果返回应答码不能在下表中找到，中文解释显示“交易失败”

2. 2：如果POS交易的批次号和网络中心批次号不一致时应答码会填 “77”，此时POS机应当提示操作员重新签到，再作交易。

代码	意义	类别	原因/采取的措施	POS 显示的内容
00	承兑或交易成功	A	承兑或交易成功	交易成功
01	查发卡行	C	查发卡行	请持卡人与发卡银行联系
03	无效商户	C	商户需要在银行或中心登记	无效商户
04	没收卡	D	操作员没收卡	此卡被没收
05	身份认证失败	C	发卡不予承兑，预约信息匹配失败	持卡人认证失败
10	部分承兑	A	部分金额批准，请收取余额	显示部分批准金额，提示操作员
11	重要人物批准（VIP）	A	此为 VIP 客户	成功，VIP 客户
12	无效的关联交易	C	发卡行不支持的交易	无效交易
13	无效金额	B	金额为 0 或其他非法值	无效金额
14	无效卡号（无此账号）	B	卡种未在中心登记或读卡号有误	无效卡号
15	无此发卡方	C	此发卡行未与中心开通业务	此卡无对应发卡方
21	卡未初始化	C	1、该卡未激活、开卡； 2、该卡初始密码未变更； 3、初始密码限制的交易； 4、长期未使用而冻结或状态为“睡眠”的卡。	该卡未初始化或睡眠卡
22	故障怀疑，关联交易错误	C	POS 状态与中心不符，可重新签到	操作有误，或超出交易允许天数
25	找不到原始交易	C	发卡行未能找到有关记录	没有原始交易，请联系发卡方
30	报文格式错误	C	格式错误（不符合磁道预校验规则）	请重试
34	有作弊嫌疑	D	有作弊嫌疑的卡，操作员可以没收	作弊卡,吞卡
38	超过允许的 PIN 试输入	D	密码错次数超限，操作员可以没收	密码错误次数超限，请与发卡方联系
40	请求的功能尚不支持	C	发卡行不支持的交易类型	发卡方不支持的交易类型

代码	意义	类别	原因/采取的措施	POS 显示的内容
41	挂失卡	D	挂失的卡， 操作员可以没收	挂失卡，请没收（POS）
43	被窃卡	D	被窃卡， 操作员可以没收	被窃卡，请没收
51	资金不足	C	账户内余额不足	可用余额不足
54	过期的卡	C	过期的卡	该卡已过期
55	不正确的 PIN	C	密码输错	密码错
57	不允许持卡人进行的交易	C	不允许持卡人进行的交易	不允许此卡交易
58	不允许终端进行的交易	C	该商户不允许进行的交易	发卡方不允许该卡在本终端进行此交易
59	有作弊嫌疑	C	CVN 验证失败	卡片校验错
61	超出金额限制	C	一次交易的金额太大	交易金额超限
62	受限制的卡	C		受限制的卡
64	原始金额错误	C	原始金额不正确	交易金额与原交易不匹配
65	超出消费次数限制	C	超出消费次数限制	超出消费次数限制
68	发卡行响应超时	C	发卡行规定时间内没有回答	交易超时，请重试
75	允许的输入 PIN 次数超限	C	允许的输入 PIN 次数超限	密码错误次数超限
90	正在日终处理	C	日期切换正在处理	系统日切，请稍后重试
91	发卡方不能操作	C	电话查询发卡方或银联，可重作	发卡方状态不正常，请稍后重试
92	金融机构或中间网络设施找不到或无法达到	C	电话查询发卡方或网络中心，可重作	发卡方线路异常，请稍后重试
94	重复交易	C	查询网络中心，可能是一笔已经成功上送的交易	拒绝，重复交易，请稍后重试
96	银联处理中心系统异常、失效	C	发卡方或网络中心出现故障	拒绝，交换中心异常，请稍后重试
97	POS 终端号找不到	D	终端未在中心或银行登记	终端未登记
98	银联处理中心收不到发卡方应答	E	银联收不到发卡行应答	发卡方超时
99	PIN 格式错	B	可重新签到作交易	PIN 格式错，请重新签到
A0	MAC 鉴别失败	B	可重新签到作交易	MAC 校验错，请重新签到
A1	转账货币不一致	C	转账货币不一致	转账货币不一致
A2	有缺陷的成功	A	银联处理中心转发了原充值交易请求，但未收到发卡方应答时，银联处理中心直接向受理方应答为有缺陷的成功交易	交易成功，请向发卡行确认

代码	意义	类别	原因/采取的措施	POS 显示的内容
A3	资金到账行无此账户	C	资金到账行账号不正确	账户不正确
A4	有缺陷的成功	A	未收到原充值交易请求时，对关联的确认交易的承兑为有缺陷的成功交易	交易成功，请向发卡行确认
A5	有缺陷的成功	A	原充值交易为拒绝时，对关联的确认交易的承兑为有缺陷的成功交易	交易成功，请向发卡行确认
A6	有缺陷的成功	A	银联处理中心转发了原充值交易请求，但未收到发卡方应答时，对受理方发来的关联的确认交易的承兑为有缺陷的成功交易	交易成功，请向发卡行确认
A7	安全处理失败	C	1、调用 MAC 校验程序失败 2、调用 PIN 校验程序失败 3、MAC 处理失败 4、密钥处理失败	拒绝，交换中心异常，请稍后重试
100	上一笔交易未完成	F	等待上一笔交易完成后重试	上一笔交易未完成
101	请求报文数据格式错误	F	重新组装正确的报文	请求报文数据格式错误
102	支付调用取消	F	第三方应用收到支付响应报文之前，支付界面关闭	支付调用取消
103	支付应用未启动或未登录	F	启动支付客户端登陆后重试	支付应用未启动或未登录
104	未找到交易类型	F	支付后台不支持的交易类型	未找到交易类型
105	未知错误，交易失败	F	未知原因引起的错误	未知错误，交易失败
RS	冲正成功	B	交易超时无应答，冲正成功	冲正成功
RF	冲正失败	B	交易超时无应答，冲正无应答	冲正失败

A.1.3.2 银行卡支付客户端

在基于Android操作系统定制智能系统中，银联已发布了银联支付客户端。基于Android操作系统定制的智能销售点终端应能正常运行银联支付客户端。

A.2 基于TEEI系统定制的智能销售点终端的细化要求

本部分描述基于TEEI系统的智能销售点终端应用管理规范。

A.2.1 TEEI系统概述

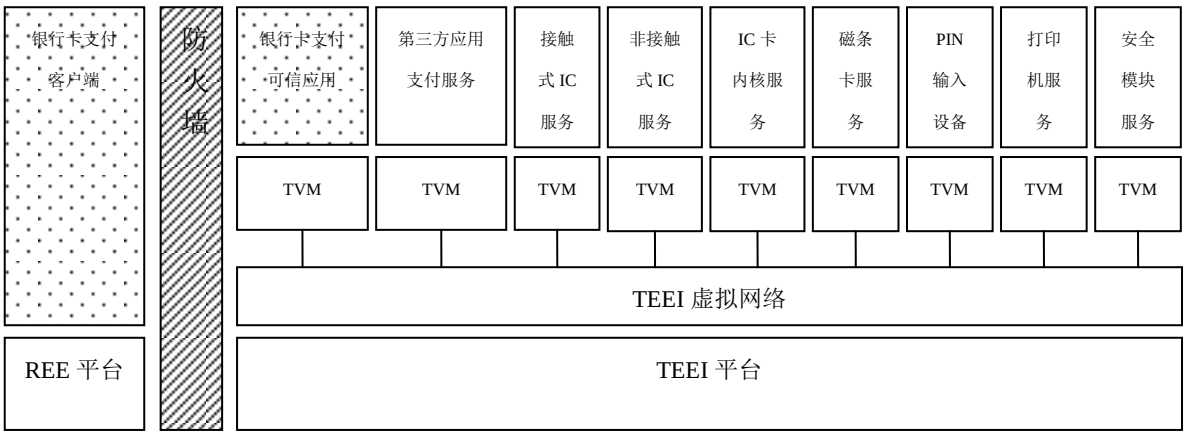
TEEI（Trusted Execution Environment Integration，可信执行环境集成）。TEEI要求智能设备具备两种执行环境，其中TEEI运行在安全执行环境下，移动操作系统如Android等运行在多媒体执行环境下。在TEEI内提供一种执行机器（TVM，Trusted Virtual Machine），多个用户通过TEEI系统可以在

同一个智能设备上创建出多个这样的执行机器，并拥有各自创建的执行机器的控制权，用户在这些执行机器上安装部署各自的软件后就拥有了各自类型的TEE。这些TEE在TEEI系统内以虚拟网络形式连接在一起，可以互相之间通信，也可以与TEEI系统外的网络实体互相通信。对于这些TEE中所用到的公共功能服务，TEEI系统以一种内置服务器（也是部署在执行机器上）的方式连接在虚拟网络上，供各个TEE访问。

部署了软件之后的各个执行机器在TEEI内部是以虚拟网络的形式连接在一起，其中对于可信虚拟机这种由TEEI系统创建的执行机器是以TEEI虚拟网络连接。

智能设备运行TEEI系统所需要具备的支撑架构，这个支撑架构可以基于安全硬件实现，也可以是纯软件实现，但必须保证安全系统（可信执行环境）和非安全系统（多媒体执行环境）之间的完全隔离，同时，二者之间能够通过安全可靠的方式互相访问。就目前来看，主要有三种实现方式：基于单一处理器的硬件防火墙及安全通道的纯硬件实现方式、基于独立安全处理器的硬件防火墙及安全通道的纯硬件实现方式、基于虚拟化技术的实现方式。

A.2.2 智能销售点终端系统架构



如图所示，在基于TEEI系统的智能销售点终端的系统架构中，和金融支付相关的外设服务以及金融支付服务本身都在TEEI平台上以可信虚拟机（TVM）的形式存在，并且彼此之间通过TEEI虚拟网络进行连接。由于这些服务都是金融支付应用相关的公共服务，所以在TEEI平台中都以内置服务的形式存在。

A.2.3 系统功能要求

A.2.3.1 系统版本

运行TEEI平台的智能设备需要支持硬件安全技术或硬件虚拟化技术。

A.2.3.2 屏幕分辨率

A.2.3.3 系统参数设置

针对WIFI、3G、以太网口应具备安全访问控制。

A.2.3.4 恢复出厂设置

系统设置中增加恢复出厂设置选项，可由终端维护人员进行恢复到终端的出厂状态。

A.2.3.5 内置服务

TEEI的内置服务是仅可被TEEI系统内可信应用或其他内置服务访问的对象，其标准访问接口为TEEI虚拟网络协议，以网络数据包形式进行消息的传递。为了方便第三方可信应用开发，TEEI平台还提供了

封装后的内置服务访问接口，即TEEI虚拟机固件API，本部分内容提供的均为TEEI虚拟机固件API形式的接口，开发者只要将对应的头文件引入即可实现对内置服务的访问。

A.2.3.5.1 共通定义

A.2.3.5.1.1 返回状态定义表

错误名	错误值	说明
EDEVICENAME	-1	设备名称不存在
EIO	-2	底层 I/O 错误
EUNKOWN	-10	未知错误
E_HAL_UNKOWN	-100	未知错误
E_HAL_ARG	-101	输入参数错误
E_HAL_IO	-102	底层I/O错误
TEEI_SUCCESS	0x00000000	操作成功
TEEI_ERROR_ACCESS_CONFLICT	0xFFFF0003	访问冲突
TEEI_ERROR_OUT_OF_MEMORY	0xFFFF000C	内存溢出
TEEI_ERROR_ITEM_NOT_FOUND	0xFFFF0008	条目未发现
TEEI_ERROR_SHORT_BUFFER	0xFFFF0010	缓冲区不足
TEEI_ERROR_COMMUNICATION	0xFFFF000E	通信错误
TEEI_ERROR_BAD_STATE	0xFFFF0007	状态错误
TEEI_ERROR_BAD_PARAMETERS	0xFFFF0006	参数错误
TEEI_ERROR_NOT_SUPPORTED	0xFFFF000A	不支持
TEEI_ERROR_SECURITY	0xFFFF000F	安全错误
TEEI_ERROR_NO_DATA	0xFFFF000B	无数据

A.2.3.5.2 接触式IC服务

A.2.3.5.2.1 头文件

使用接触式IC服务需要引入头文件：

```
#include "teei_platform_se_api.h";
```

A.2.3.5.2.2 数据类型

A.2.3.5.2.2.1 基本类型

本规范利用C99规范（ISO/IEC 9899:1999）中的C语言类型来进行整数和布尔类型的定义，使用基本数据类型有以下几种：

- uint32_t: 无符号的32位整数
- int32_t: 有符号的32位整数
- uint16_t: 无符号的16位整数
- int16_t: 有符号的16位整数
- uint8_t: 无符号的8位整数
- int8_t: 有符号的8位整数
- bool: true和false的布尔类型
- char: 0结尾、UTF-8编码的一个字符

- size_t：大到足够容纳对象在内存中大小的无符号整数
- 本规范中，所有的数值都是小端序。

A.2.3.5.2.2.2 TEEI_Result

```
typedef uint32_t TEEI_Result;
```

A.2.3.5.2.2.3 SE_ReaderProperties

读卡器的属性信息。

```
typedef struct __SE_ReaderProperties
{
    bool sePresent; //读卡器中是否存在SE
    bool teeOnly; //读卡器是否是TEE专有设备
    bool selectResponseEnable; // 是否存在SELECT命令的响应
} SE_ReaderProperties;
```

A.2.3.5.2.2.4 SE_AID

SE应用的AID信息。

```
typedef struct __TEEI_SEAID
{
    uint8_t *buffer //应用的AID值
    size_t bufferLen //应用AID的长度
} SE_AID;
```

A.2.3.5.2.2.5 句柄

不透明的句柄类型，代表安全元件服务、读卡器、会话和通道。

```
typedef struct __SE_ServiceHandle* SE_ServiceHandle;
typedef struct __SE_ReaderHandle* SE_ReaderHandle;
typedef struct __SE_SessionHandle* SE_SessionHandle;
typedef struct __SE_ChannelHandle* SE_ChannelHandle;
```

A.2.3.5.2.2.6 SE_Service

A.2.3.5.2.2.6.1 SE_ServiceOpen

```
TEEI_Result SE_ServiceOpen(
    [out] SE_ServiceHandle *seServiceHandle
);
```

描述：

打开安全元件服务。

参数：

seServiceHandle: 安全元件服务。

返回值：
函数成功完成后，返回值为TEEI_SUCCESS。
如果已经打开服务，返回TEEI_ERROR_ACCESS_CONFLICT。
如果资源不足以执行操作，返回TEEI_ERROR_OUT_OF_MEMORY。

A.2.3.5.2.2.6.2 SE_ServiceClose

```
void SE_ServiceClose(SE_ServiceHandle seServiceHandle);
```

描述：
关闭安全元件服务。

参数：
seServiceHandle: 安全软件服务。

A.2.3.5.2.2.6.3 SE_ServiceGetReaders

```
TEEI_Result SE_ServiceGetReaders(
    SE_ServiceHandle seServiceHandle,
    [out] SE_ReaderHandle* seReaderHandleList,
    [inout] size_t* seReaderHandleListLen
);
```

描述：
获得可用的读卡器列表。

参数：
seServiceHandle: 安全元件服务。
seReaderHandleList: 读卡器句柄列表。
seReaderHandleListLen: 读卡器句柄列表长度。

返回值：
函数成功完成后，返回值为TEEI_SUCCESS。
如果未找到读卡器，返回TEEI_ERROR_ITEM_NOT_FOUND。
如果资源不足以执行操作，返回TEEI_ERROR_OUT_OF_MEMORY。
如果缓冲区大小不足，返回TEEI_ERROR_SHORT_BUFFER。

A.2.3.5.2.2.7 SE_Reader

A.2.3.5.2.2.7.1 SE_ReaderGetProperties

```
void SE_ReaderGetProperties(
    SE_ReaderHandle seReaderHandle,
    [out] SE_ReaderProperties* readerProperties
);
```

描述：
获得读卡器属性信息。

参数：
seReaderHandle: 读卡器句柄。
readerProperties: 读卡器属性。

A.2.3.5.2.2.7.2 SE_ReaderGetName

```
TEEI_Result SE_ReaderGetName(
    SE_ReaderHandle seReaderHandle,
    [outstring] char* readerName,
    size_t* readerNameLen
);
```

描述：
获得读卡器的名字。

参数：
seReaderHandle: 读卡器句柄。
readerName: 读卡器名字。
readerNameLen: 读卡器名字长度。

返回值：
函数成功完成后，返回值为TEEI_SUCCESS。
如果缓冲区大小不足，返回TEEI_ERROR_SHORT_BUFFER。

A.2.3.5.2.2.7.3 SE_ReaderOpenSession

```
TEEI_Result SE_ReaderOpenSession(
    SE_ReaderHandle seReaderHandle,
    [out] SE_SessionHandle* seSessionHandle
);
```

描述：
打开安全元件会话。

参数：
seReaderHandle: 读卡器句柄。
seSessionHandle: 读卡器会话句柄。

返回值：
函数成功完成后，返回值为TEEI_SUCCESS。
如果无法建立会话，返回TEEI_ERROR_COMMUNICATION。
如果资源不足以执行操作，返回TEEI_ERROR_OUT_OF_MEMORY。

A.2.3.5.2.2.7.4 SE_ReaderCloseSessions

```
void SE_ReaderCloseSessions(SE_ReaderHandle seReaderHandle);
```

描述：
关闭所有安全元件会话。

参数：
seReaderHandle: 读卡器句柄。

A.2.3.5.2.2.8 SE_Session

A.2.3.5.2.2.8.1 SE_SessionGetATR

```
TEEI_Result SE_SessionGetATR(
    SE_SessionHandle seSessionHandle,
    [outbuf] void* atr, size_t* atrLen
);
```

描述：
获得安全元件的ATR信息。

参数：
seSessionHandle: 安全元件会话句柄。
atr: ATR信息。
atrLen: ATR信息长度

返回值：
函数成功完成后，返回值为TEEI_SUCCESS。
如果ATR不可用或I/O错误，返回TEEI_ERROR_COMMUNICATION。
如果缓冲区大小不足，返回TEEI_ERROR_SHORT_BUFFER。

A.2.3.5.2.2.8.2 SE_SessionIsClosed

```
TEEI_Result SE_SessionIsClosed(SE_SessionHandle seSessionHandle);
```

描述：
判断安全元件会话是否关闭。

参数：
seSessionHandle: 安全元件会话句柄。

返回值：
函数成功完成后，返回值为TEEI_SUCCESS。
如果ATR不可用或I/O错误，返回TEEI_ERROR_COMMUNICATION。

A.2.3.5.2.2.8.3 SE_SessionClose

```
void SE_SessionClose(SE_SessionHandle seSessionHandle);
```

描述：
关闭安全元件会话。

参数：
seSessionHandle: 安全元件会话句柄。

A.2.3.5.2.2.8.4 SE_SessionCloseChannels

```
void SE_SessionCloseChannels(SE_SessionHandle seSessionHandle);
```

描述：
关闭安全元件会话下打开的所有通道。

参数：
seSessionHandle: 安全元件会话句柄。

A.2.3.5.2.2.8.5 SE_SessionOpenBasicChannel

```
TEEI_Result SE_SessionOpenBasicChannel(
    SE_SessionHandle seSessionHandle,
    [in] SE_AID *seAID,
    [out] SE_ChannelHandle *seChannelHandle
);
```

描述：
打开安全元件的基本通道。

参数：
seSessionHandle: 安全元件会话句柄。
seAID: 应用的AID信息。
seChannelHandle: 安全元件通道句柄。

返回值：
函数成功完成后，返回值为TEEI_SUCCESS。
如果ATR不可用或I/O错误，返回TEEI_ERROR_COMMUNICATION。
如果安全元件会话已经关闭，返回TEEI_ERROR_BAD_STATE。
如果参数格式有误，返回TEEI_ERROR_BAD_PARAMETERS。
如果指定的AID不支持，返回TEEI_ERROR_NOT_SUPPORTED。
如果访问没有得到授权，返回TEEI_ERROR_SECURITY。

A.2.3.5.2.2.8.6 SE_SessionOpenLogicalChannel

```
TEEI_Result SE_SessionOpenLogicalChannel(
    SE_SessionHandle seSessionHandle,
    [in] SE_AID *seAID,
    [out] SE_ChannelHandle *seChannelHandle
);
```

描述：
打开安全元件的逻辑通道。

参数：
seSessionHandle: 安全元件会话句柄。
seAID: 应用的AID信息。
seChannelHandle: 安全元件通道句柄。

返回值：
函数成功完成后，返回值为TEEI_SUCCESS。
如果ATR不可用或I/O错误，返回TEEI_ERROR_COMMUNICATION。
如果安全元件会话已经关闭，返回TEEI_ERROR_BAD_STATE。
如果参数格式有误，返回TEEI_ERROR_BAD_PARAMETERS。
如果指定的AID不支持，返回TEEI_ERROR_NOT_SUPPORTED。
如果访问没有得到授权，返回TEEI_ERROR_SECURITY。

A.2.3.5.2.2.9 SE_Channel

A.2.3.5.2.2.9.1 SE_ChannelClose

```
void SE_ChannelClose(SE_ChannelHandle seChannelHandle);
```

描述：
关闭安全元件通道。

参数：
seChannelHandle: 安全元件通道句柄。

A.2.3.5.2.2.9.2 SE_ChannelSelectNext

```
TEEI_Result SE_ChannelSelectNext(SE_ChannelHandle seChannelHandle);
```

描述：
选择当前通道上的下一个应用。

参数：
seChannelHandle: 安全元件通道句柄。

返回值：
函数成功完成后，返回值为TEEI_SUCCESS。
如果ATR不可用或I/O错误，返回TEEI_ERROR_COMMUNICATION。
如果安全元件会话已经关闭，返回TEEI_ERROR_BAD_STATE。
如果安全元件不支持本功能，返回TEEI_ERROR_NOT_SUPPORTED。
如果未找到下一个应用，返回TEEI_ERROR_ITEM_NOT_FOUND。

A.2.3.5.2.2.9.3 SE_ChannelGetSelectResponse

```
TEEI_Result SE_ChannelGetSelectResponse(
```

```
    SE_ChannelHandle seChannelHandle,
    [outbuf] void* response, size_t *responseLen
);
```

描述：
获得当前通道的SELECT命令的响应。

参数：
seChannelHandle: 安全元件通道句柄。
response: 响应消息。
responseLen: 响应消息长度

返回值：
函数成功完成后，返回值为TEEI_SUCCESS。
如果ATR不可用或I/O错误，返回TEEI_ERROR_COMMUNICATION。
如果安全元件会话已经关闭，返回TEEI_ERROR_BAD_STATE。
如果数据不存在，返回TEEI_ERROR_NO_DATA。

A.2.3.5.2.2.9.4 SE_ChannelTransmit

```
TEEI_Result SE_ChannelTransmit(
    SE_ChannelHandle seChannelHandle,
    [inbuf] void* command, size_t commandLen,
    [outbuf] void* response, size_t *responseLen
);
```

描述：
收发安全元件APDU命令。

参数：
seChannelHandle: 安全元件通道句柄。
command: 命令。
commandLen: 命令长度。
response: 响应消息。
responseLen: 响应消息长度

返回值：
函数成功完成后，返回值为TEEI_SUCCESS。
如果ATR不可用或I/O错误，返回TEEI_ERROR_COMMUNICATION。
如果安全元件会话已经关闭，返回TEEI_ERROR_BAD_STATE。
如果参数格式错误，返回TEEI_ERROR_BAD_PARAMETERS。
如果访问没有得到授权，返回TEEI_ERROR_SECURITY。

A.2.3.5.3 非接触式IC服务

A.2.3.5.3.1 头文件

使用非接触式IC服务需要引入头文件：

```
#include "teei_platform_contactless_api.h";
```

A.2.3.5.3.2 设备打开

打开非接卡设备，定义为打开设备，初始化资源。

接口名称	contactless_card_open			
功能描述	创建或者打开一个已经存在的非接触读卡设备			
接口格式	int contactless_card_open(CONTACTLESS_CARD_NOTIFIER fNotifier, void* pUserData,int* pErrorCode);			
参数说明	fNotifier	CONTACTLESS_CARD_NOTIFIER	必选	回调函数 CONTACTLESS_CARD_NOTIFIER 被定义为：typedef void (*CONTACTLESS_CARD_NOTIFIER)(void* pUserData, int nEvent, unsigned char* pEventData, int nDataLength)
	pUserData	void*	必选	用户数据，会作为回调函数的参数，传入
	pErrorCode	int*	必选	错误代码。当返回值等于 0 的时候设置
返回值	1. 等于 0，错误 2. 不等于 0，设备句柄			

A.2.3.5.3.3 设备关闭

定义为关闭设备，释放资源。

接口名称	contactless_card_close			
功能描述	关闭非接触读卡设备			
接口格式	int contactless_card_close(int nHandle);			
参数说明	nHandle	int	必选	设备句柄
返回值	1. 大于等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表			

A.2.3.5.3.4 搜索设备

搜索卡片，定义为卡片检测，检测非接卡有效区域内的卡片。

接口名称	contactless_card_search_target_begin			
功能描述	开始搜索非接触 IC 卡			
接口格式	int contactless_card_search_target_begin(int nHandle, int nCardMode, int nFlagSearchAll,int nTimeout_MS);			
参数说明	nHandle	int	必选	设备句柄
	nCardMode	int	必选	卡类型[0,1,2,3]，分别为： CONTACTLESS_CARD_MODE_AUTO，

				CONTACTLESS_CARD_MODE_TYPE_A ，CONTACTLESS_CARD_MODE_TYPE_A ，CONTACTLESS_CARD_MODE_TYPE_B ，CONTACTLESS_CARD_MODE_TYPE_C
	nFlagSearchAll	int	必选	0：找一张卡，1：找所有的卡
	nTimeout_MS	int	必选	超时，单位为毫秒。小于 0 时，将一直搜索，直到使用方法：contactless_card_search_target_end 打断。
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.3.5 停止搜索

停止当前的搜索，停止后，与搜索接口成对出现。

接口名称	contactless_card_search_target_end			
功能描述	停止搜索非接触 IC 卡			
接口格式	int contactless_card_search_target_end(int nHandle);			
参数说明	nHandle	int	必选	设备句柄
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.3.6 连接卡片

卡片连接函数，当存在多张卡片时，只能与一张卡片连接，在传递APDU命令前，必须要连接非接触IC卡

接口名称	contactless_card_attach_target			
功能描述	在传递 APDU 命令前，链接非接触 IC 卡			
接口格式	int contactless_card_attach_target(int nHandle, unsigned char* pATRBuffer, unsigned int nATRBufferLength);			
参数说明	nHandle	int	必选	设备句柄
	pATRBuffer	int	必选	ATR 缓冲区
	data	void*	必选	ATR 缓冲区长度
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.3.7 断开连接

解除和当前卡片的连接状态，解除连接状态后，将不能继续进行APDU命令的传输。

接口名称	contactless_card_detach_target			
功能描述	和非接触 IC 卡解除链接			
接口格式	int contactless_card_detach_target(int nHandle);			
参数说明	nHandle	int	必选	设备句柄
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.3.8 APDU 命令交互

非接卡APDU交互接口。

接口名称	contactless_card_transmit			
功能描述	发送 APDU 命令并接收响应			
接口格式	int contactless_card_transmit(int nHandle, unsigned char* pAPDU, unsigned int nAPDULength, unsigned char* pResponse, unsigned int* pResponseLength);			
参数说明	nHandle	int	必选	设备句柄
	pAPDU	char*	必选	APDU 命令
	nAPDULength	int	必选	APDU 命令长度
	pResponse	char*	必选	存储 APDU 命令响应的缓冲区
	pResponseLength	Int*	必选	输入时，为响应 APDU 命令缓冲区长度。输出时，为响应数据长度
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.3.9 IC 卡控制

有些硬件在使用时，需要进行特殊指令的传输，该接口定义为提供这种指令的调用。

接口名称	contactless_card_send_control_command			
功能描述	发送非接触 IC 卡控制命令			
接口格式	int contactless_card_send_control_command(int nHandle, unsigned int nCmdID, unsigned char* pCmdData, unsigned int nDataLength);			
参数说明	nHandle	int	必选	设备句柄
	nCmdID	char*	必选	控制命令 ID 号
	pCmdData	int	必选	和命令有关的数据。输入时，是和命令有关的数据，如此命令没有数据，可设置为 NULL。输出时，为响应数据。
	nDataLength	char*	必选	命令的数据长度

返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表

A.2.3.5.4 磁条卡服务

A.2.3.5.4.1 头文件

使用磁条卡服务需要引入头文件：

```
#include "teei_platform_msr_api.h";
```

A.2.3.5.4.2 打开设备

打开磁条卡设备

接口名称	msr_open
功能描述	打开一个已经存在的磁条卡读卡设备
接口格式	int msr_open();
参数说明	无
返回值	1. 大于等于 0，成功打开设备 2. 小于 0，错误；参考附录一：错误定义表

A.2.3.5.4.3 关闭设备

关闭磁条卡设备。设备未打开关闭无效。

接口名称	msr_close
功能描述	关闭已打开的磁条卡读卡设备
接口格式	int msr_close();
参数说明	无
返回值	1. 大于等于 0，成功关闭设备 2. 小于 0，错误；参考附录一：错误定义表

A.2.3.5.4.4 注册刷卡

打开设备后，通过该函数进行刷卡注册，刷卡注册后，磁条卡设备将启动监听，直到等到刷卡或者主动取消。

接口名称	msr_register_notifier			
功能描述	登记回调函数，回调函数登记好之后，如果用户刷卡，回调函数会被调用，用来通知用户来取数据			
接口格式	int msr_register_notifier(MSR_NOTIFIER notifier, void* pUserData);			
参数说明	notifier	MSR_NOTIFIER	必选	回调函数。MSR_NOTIFIER 被定义为： typedef void (*MSR_NOTIFIER)(void* pUserData);
	pUserData	void*	必选	用户数据，会作为回调函数的参数，传入
返回值	1. 大于等于 0，表示成功。 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.4.5 取消刷卡

取消刷卡，只在调用msr_register_notifier之后有用，取消刷卡后，磁条卡设备恢复正常模式。

接口名称	msr_unregister_notifier
功能描述	撤销登记的函数，撤销之后，刷卡后上层程序不会收到通知
接口格式	int msr_unregister_notifier();
参数说明	无
返回值	1. 大于等于 0，表示成功。 2. 小于 0，表示错误；参考附录一：错误定义表

A.2.3.5.4.6 读取磁道错误

读取磁道错误，监听刷卡后，调用该函数确定磁道数据是否正常。

接口名称	msr_get_track_error			
功能描述	查询对应磁道在解码过程中发生的错误			
接口格式	int msr_get_track_error(int nTrackIndex);			
参数说明	nTrackIndex	int	必选	磁道索引，可以是 0，1，2
返回值	1. 大于等于 0，表示成功。 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.4.7 获取磁道数据长度

获取磁道数据长度。

接口名称	msr_get_track_data_length			
功能描述	查询对应磁道数据的长度			
接口格式	int msr_get_track_data_length(int nTrackIndex);			
参数说明	nTrackIndex	int	必选	磁道索引，可以是 0，1，2
返回值	1. 大于等于 0，表示磁道数据的长度。 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.4.8 获取磁道数据

获取磁道数据明文。

接口名称	msr_get_track_data			
功能描述	获取对应磁道数据的数据			
接口格式	int msr_get_track_data(int nTrackIndex, unsigned char* pTrackData, int nLength);			
参数说明	nTrackIndex	int	必选	磁道索引，可以是 0，1，2
	pTrackData	unsigned char*	必选	存放磁道数据的缓冲区。
	nLength	int	必选	数据缓冲区的长度。由于有接口查询长度信息，所以调用者可

				以避免缓冲区长度不够的错误。
返回值	1. 大于等于 0，表示磁道数据的长度。 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.5 PIN输入设备服务

A.2.3.5.5.1 头文件

使用PIN输入设备服务需要引入头文件：

```
#include "teei_platform_pinpad_api.h";
```

A.2.3.5.5.2 打开设备

打开PIN输入设备。

接口名称	pinpad_open
功能描述	打开存在的 PIN 输入设备
接口格式	int pinpad_open();
参数说明	无
返回值	1. 等于 0，打开成功 2. 小于 0，错误；参考附录一：错误定义表

A.2.3.5.5.3 关闭设备

关闭PIN输入设备。

接口名称	pinpad_close
功能描述	关闭已打开的 PIN 输入设备
接口格式	int pinpad_close();
参数说明	无
返回值	1. 等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表

A.2.3.5.5.4 PIN 输入设备显示

PIN输入设备显示接口，将数据显示在PIN输入设备LCD上。

接口名称	pinpad_show_text			
功能描述	写入显示数据同时显示在 PIN 输入设备 LCD 上			
接口格式	int pinpad_show_text(int nLineIndex, char* strText, int nLength, nFlagSound);			
参数说明	nLineIndex	int	必选	行标，从上到下，依次从 0 开始
	strText	char*	必选	写入的显示数据
	nLength	int	必选	写入数据的字节长度，为 0 时，清 PIN

				输入设备 LED
	nFlagSound	int	必选	声音提示，0 不提示，1 提示
返回值	1. 大于等于 0，写入显示成功 2. 小于 0，写入显示失败			

A.2.3.5.5.5 选择密钥

选择当前操作的PIN输入设备主密钥和工作密钥（用户密钥）。

接口名称	pinpad_select_key			
功能描述	选择一个 PIN 输入设备主密钥和用户密钥			
接口格式	int pinpad_select_key(int nKeyType, int nKeyMasterKeyID, int nUserKeyID, int aAlgorith);			
参数说明	nKeyType	int	必选	主密钥类型。0：dukpt，1：Tdukpt，2：master key，3：public key，4：fix key
	nKeyMasterKeyID	int	必选	主密钥 ID，[0x00, …,0x09]。当 nKeyType 为 master key 时使用
	nUserKeyID	int	必选	用户密钥 ID，［0x00,0x01］。当 nKeyType 为 master key 时使用
	aAlgorith	int	必选	算法选择。1：3DES，2：DES
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.5.6 数据加密

使用当前选择的主密钥和工作密钥进行数据加密。

接口名称	pinpad_encrypt_string			
功能描述	使用用户密钥加密数据			
接口格式	int pinpad_encrypt_string(unsigned char* pPlainText, int nTextLength, unsigned char* pCipherTextBuffer,int nCipherTextBufferLength);			
参数说明	pPlainText	char*	必选	待加密字符串
	nTextLength	int	必选	待加密字符串长度
	pCipherTextBuffer	char*	必选	用于存储密文的缓冲区
	nCipherTextBufferLength	int	必选	缓冲区长度
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.5.7 更新工作密钥

更新工作密钥，工作密钥为密文下载。

接口名称	pinpad_update_user_key

功能描述	更新用户密钥			
接口格式	int pinpad_update_user_key(int nMasterKeyID,int nUserKeyID, unsigned char* pCipherNewUserKey,int nCipherNewUserKeyLength);			
参数说明	nMasterKeyID	int	必选	PIN 输入设备主密钥 ID
	nUserKeyID	int	必选	PIN 输入设备用户密钥 ID
	pCipherNewUserKey	char*	必选	待更细的用户密钥，已加密过
	nCipherNewUserKeyLength	int	必选	待更细用户密钥的长度
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.5.8 计算 PINBLOCK

计算Pinblock, 键盘输入PIN处理方式为ISO9564-1格式0.

接口名称	pinpad_calculate_pin_block			
功能描述	计算 Pin Block			
接口格式	int pinpad_calculate_pin_block(unsigned char* pASCIICardNumber, int nCardNumberLength, unsigned char* pPinBlockBuffer, int nPinBlockBufferLength, int nTimeout_MS, int nFlagSound)			
参数说明	pASCIICardNumber	char*	必选	银联卡卡号
	nCardNumberLength	int	必选	银联卡卡号长度
	pPinBlockBuffer	char*	必选	保存 Pin Block 的缓冲区
	nPinBlockBufferLength	int	必选	缓冲区长度
	nTimeout_MS	int	必选	输入超时，单位毫秒，当小于零时，一直等待输入
	nFlagSound	int	必选	是否需要声音提示，0 否，1 是
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.5.9 计算 Mac

根据指定的用户密钥计算MAC。

接口名称	pinpad_calculate_mac
功能描述	使用用户密钥计算 MAC

接口格式	int pinpad_calculate_mac(unsigned char* pData, int nDataLength, int nMACFlag, unsigned char* pMACOutBuffer, int nMACOutBufferLength)			
参数说明	pData	char*	必选	数据
	nDataLength	int	必选	数据长度
	nMACFlag	int	必选	0：X99，1：ECB
	pMACOutBuffer	int	必选	保存 MAC 的缓冲区
	nMACOutBufferLength	int	必选	缓冲区长度
返回值	1. 大于等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表			

A.2.3.5.5.10 设置 Pin 长度

指定Pin输入的长度。

接口名称	pinpad_set_pin_length			
功能描述	设置 Pin 的最大限制长度			
接口格式	int pinpad_set_pin_length(int nLength, int nFlag)			
参数说明	nLength	int	必选	长度范围［0x00,0x0D］
	nFlag	int	必选	0：最小长度，1：最大长度
返回值	1. 大于等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表			

A.2.3.5.6 打印机服务

A.2.3.5.6.1 头文件

使用打印机服务需要引入头文件：

```
#include "teei_platform_printer_api.h";
```

A.2.3.5.6.2 打开设备

打开打印机设备。

接口名称	printer_open
功能描述	打开一个已经存在的打印设备
接口格式	int printer_open();

参数说明	无
返回值	1. 等于 0，打开成功 2. 小于 0，错误；参考附录一：错误定义表

A.2.3.5.6.3 关闭设备

关闭打印机设备。

接口名称	printer_close
功能描述	关闭已打开的打印设备
接口格式	int printer_close();
参数说明	无
返回值	1. 等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表

A.2.3.5.6.4 传输打印数据

写入打印数据并开启打印。

接口名称	printer_write			
功能描述	从当前点连续打印，打印的数据存放在缓冲区			
接口格式	int printer_write(unsigned char* pData, int nDataLength);			
参数说明	pData	char*	必选	打印数据或打印控制命令
	nDataLength	int	必选	打印数据或控制命令长度
返回值	1. 等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.2.3.5.6.5 开启打印

开启打印，并分配资源，在printer_write之前调用。

接口名称	printer_begin
功能描述	准备开始打印
接口格式	int printer_begin();
参数说明	无
返回值	1. 等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表

A.2.3.5.6.6 结束打印

结束打印，并释放资源。在所有数据写完后调用。

接口名称	printer_end
功能描述	结束打印
接口格式	int printer_end();

参数说明	无
返回值	1. 等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表

A.2.3.5.7 安全模块服务

A.2.3.5.7.1 头文件

使用安全模块服务需要引入头文件：

```
#include "teei_platform_securemodule_api.h";
```

A.2.3.5.7.2 打开硬件安全模块设备

函数	int hsm_osm_open();
描述	打开硬件安全模块设备
参数	无
返回值	1. 等于 0，打开成功 2. 小于 0，错误

A.2.3.5.7.3 关闭硬件安全模块设备

函数	int hsm_osm_close();
描述	关闭硬件安全模块设备
参数	无
返回值	1. 等于 0，关闭成功 2. 小于 0，错误

A.2.3.5.7.4 保存安全证书

函数	int hsm_osm_save_object(HSM_OBJECT_PROPERTY* pObjectProperty, unsigned char* pObjectData, unsigned int nDataLength, HSM_OBJECT_DATA_TYPE nDataType);			
描述	保存安全证书至硬件安全模块设备			
参数	pObjectProperty	HSM_OBJECT_PROPERTY*	必选	证书属性。
	pObjectData	unsigned char*	必选	证书数据
	nDataLength	unsigned int	必选	证书数据长度
	nDataType	HSM_OBJECT_DATA_TYPE	必	证书格式。

			选	
返回值	1. 等于 0，保存成功。 2. 小于 0，保存失败。			

A.2.3.5.7.5 删除指定证书

函数	int hsm_osm_delete_object(HSM_OBJECT_PROPERTY* pObjectProperty, char* pPIN, unsigned int nPINLength);			
描述	从硬件安全模块设备中删除指定证书			
参数	pObjectProperty	HSM_OBJECT_PROPERTY*	必选	证书属性。
	pPIN	char*	必选	PIN 值。
	nPINLength	unsigned int	必选	PIN 值长度。
返回值	1. 等于 0，删除成功 2. 小于 0，错误			

A.2.3.5.7.6 删除所有证书

函数	int hsm_osm_delete_all(char* pPIN, unsigned int nPINLength);			
描述	从硬件安全模块中设备删除所有证书			
参数	pPIN	char*	必选	保留的 Pin 值
	nPINLength	unsigned int	必选	PIN 值长度
返回值	1. 等于 0，删除成功 2. 小于 0，错误			

A.2.3.5.7.7 查询证书数量

函数	int HSM_OSM_QUERY_OBJECT_COUNT(HSM_OBJECT_PROPERTY* pObjectProperty);			
描述	从硬件安全模块内查询证书数量			
参数	pObjectProperty	HSM_OBJECT_PROPERTY*	必选	证书属性。
返回值	1. 小于 0，查询失败。 2. 大于等于 0，证书数量。			

A.2.3.5.7.8 读取证书

函数	int hsm_osm_load_object(unsigned int nIndex, HSM_OBJECT_PROPERTY* pObjectProperty, unsigned char* pDataBuffer, unsigned int nDataBufferLength, HSM_OBJECT_DATA_TYPE nDataType);			
描述	从硬件安全模块内读取证书			
参数	nIndex	unsigned int	必选	证书索引号。
	pObjectProperty	HSM_OBJECT_PROPERTY*	必选	证书属性。
	pDataBuffer	unsigned char*	必选	证书数据。
	nDataBufferLength	unsigned int	必选	证书数据长度。
	nDataType	HSM_OBJECT_DATA_TYPE	必选	证书格式。
返回值	1. 等于 0，读取成功 2. 小于 0，读取失败			

A.2.3.5.7.9 参数类型说明

证书类型类型数据结构：pem 证书格式；der 编码证书；PKCS #7 证书；PKCS#12 证书；

```
typedef enum Object_data_type{
    HSM_OBJECT_DATA_TYPE_pem,
    HSM_OBJECT_DATA_TYPE_der,
    HSM_OBJECT_DATA_TYPE_p7b,
    HSM_OBJECT_DATA_TYPE_pfx
}HSM_OBJECT_DATA_TYPE
```

证书种类：私钥；公钥；cert 证书；

```
typedef enum object_type_{
    HSM_OBJECT_TYPE_private_key,
    HSM_OBJECT_TYPE_public_key,
    HSM_OBJECT_TYPE_cert
}HSM_OBJECT_TYPE
```

证书属性：证书 ID；证书 Label；密码；

```
typedef struct object_proptery_{

    unsigned char strID[32];

    unsigned char strLabel[32];

    unsigned char strPassword[32];

    HSM_OBJECT_TYPE nObjectType;
    }HSM_OBJECT_PROPERTY
```

A.2.3.5.8 IC卡内核服务

A.2.3.5.8.1 头文件

使用IC内核服务需要引入头文件：

```
#include "teei_platform_ickernel_api.h";
```

A.2.3.5.8.2 EMV tag 存取功能

A. EMV tag的值是否存在

```
int emv_is_tag_present(unsigned short tag)
```

B.读取EMV tag的值,如果该tag的值不存在，则返回false

```
int emv_get_tag_data(unsigned char *data, unsigned short *length, unsigned short tag)
```

C.设置EMV tag的值

```
int emv_set_tag_data (unsigned char *data, unsigned short length, unsigned short tag)
```

A.2.3.5.8.3 EMV 交易处理功能

A. EMV参数初始化

```
void emv_initialize(unsigned char kernel_type, void *parameter)

    kernerlType:
    #define PBOC_KERNAL           1 // 借贷记
    #define QPBOC_KERNAL          2 // QPBOC
    #define UPCASH_KERNAL         3 // 电子现金

    typedef struct
    {
    smart_card_transmit p_transmit_pointer; // Transmit函数指针
    int readerHandle;                       // 读卡器的句柄
        unsigned char  card_type;               // CARD_CONTACT: 1; CARD_CONTACTLESS: 2
    unsigned short atr_length;
    unsigned char atr[30];
    emv_process_next_completed p_emv_process_next_completed;    // 回调函数指针
    }EMVFUNC_PARAMETER;
```

```
typedef int (*smart_card_transmit)(int   handle,
unsigned char  *p_APDU,
unsigned short  APDU_length,
unsigned char  *p_response,
unsigned short *p_response_length);

typedef void (*emv_process_next_completed)(unsigned char status, unsigned char info);
```

B.调用EMV kernal处理宿主函数emvProcess.

```
int emv_process_next()
```

C. EMV kernel处理回调函数

```
void emv_process_next_completed(unsigned char status, unsigned char info)

status:
      STATUS_ERROR = 0; //执行报错
STATUS_CONTINUE= 1; //还未完成
STATUS_COMPLETION = 2; //完成

1) If status == STATUS_COMPLETION,
       Info :
APPROVE_OFFLINE            1    /** Transaction approved Offline */
APPROVE_ONLINE             2    /** Transaction approved Online */
DECLINE_OFFLINE            3    /** Transaction declined Offline */
DECLINE_ONLINE             4    /** Transaction declined Online */
2) If status == STATUS_ERROR,
Info:
SUCCESS   = 0;
/** Selected Application do not in the Candidate List when Application Select */
ERROR_APP_NO_INFO        =  1;
/** No Application Selected when Application Select */
ERROR_NO_APP             =  2;
/** Parse Card Returned Data Error when Application Select */
ERROR_APP_ANALYSIS       =  3;
/** card return 6A81 when Application Select */
ERROR_APP_BLOCKED        =  4;
/** Error when Application Select */
ERROR_APP_SELECT         =  5;
    /** Application Interchange Profile(AIP) and Application File Locator(AFL) not exist
when Initialize Application */
ERROR_NO_AIPAFL          =  6;
/** Error when Initialize Application Data */
ERROR_INIT_APP           =  7;
ERROR_OTHER_CARD         =  8;
ERROR_EXPIRED_CARD       =  9;
```

```
    // Any error after initialApplication should halt the process
    /** Error when Read Application Data */
    ERROR_APP_DATA           = 10;
        /** card return 6983, command not allowed, authentication method blocked */
    ERROR_AUTH_METHOD_BLOCKED= 11;
        /** card return 6984, command not allowed, referenced data invalidated */
    ERROR_REFDATA_INVALIDATED= 12;
        /** card return 6985, command not allowed, conditions of use not satisfied */
    ERROR_COND_NOT_SATISFIED = 13;
    /** card return 6A81, wrong parameter p1 p2, function not supported */
    ERROR_FUNC_NOT_SUPPORTED = 14;
    /** card return 6A82, wrong parameter p1 p2, file not found */
    ERROR_FILE_NOT_FOUND     = 15;
    /** card return 6A83, wrong parameter p1 p2, record not found */
    ERROR_RECORD_NOT_FOUND   = 16;
    /** card return 6A88, referenced data (data objects) not found */
    ERROR_REFDATA_NOT_FOUND  = 17;
        /** card return 6283, state of non-volatile memory unchanged, selected file
invalidated */
    ERROR_SELFILE_INVALIDATED= 18;
        /** card return 6300, state of non-volatile memory changed, authentication failed */
    ERROR_AUTH_FAILED        = 19;
        /** card return 63Cx, state of non-volatile memory changed, counter provided by
'x'(from 0-15) */
    ERROR_COUNTER_X          = 20;
    /** card blocked or select command not supported */
    ERROR_BLOCKED_NOSEL      = 21;
    /** Parse Card Returned Data Error */
    ERROR_ANALYSIS           = 22;
    /** Error when Processing Restrict */
    ERROR_READ_DATA          = 23;
    /** Card Returned Data for SDA Overflow when Read Application Data */
    ERROR_GEN_RANDOM         = 24;
    /** Generate DOL Block error when Data Authentication */
    ERROR_GEN_DOLBLOCK       = 25;
    /** Generate AC error when Transaction Process */
    ERROR_GEN_AC             = 26;
    /** CDOL1 not exist when Transaction Process */
    ERROR_NO_CDOL1           = 27;
    /** CDOL2 not exist when Transaction Process */
    ERROR_NO_CDOL2           = 28;
    /** Logic Error when Transaction Process */
    ERROR_LOGIC              = 29;
    //  public final static short EMV_CVM_METHOD_NOT_AVAILABLE = 36;
```

```
    /** Card Returned Unknown Response Code */
    ERROR_CHIP_CANNOT_BE_READ= 30;
    /** Process Command ERROR */
    ERROR_PROCESS_CMD        = 31;
    /** Card decision is AAR when Transaction Process */
    ERROR_AAR_ABORTED        = 32;
    /** Log File Error */
    ERROR_LOG_FILE           = 33;
    /** Service not Allowed */
    ERROR_SERVICE_NOT_ALLOWED= 34;
    /** PIN Entry timeout */
    ERROR_PINENTERY_TIMEOUT  = 35;
    /** Check Offline PIN Error when Cardholder Verify */
    ERROR_OFFLINE_VERIFY     = 36;
        /** Communication Error with Host, but the card need advice, halted the transaction
*/
    ERROR_NEED_ADVICE        = 37;
    ERROR_USER_CANCELLED     = 38;
    3) If status == STATUS_CONTINUE,
    Info:
    /** EMV Transaction Started*/
    EMV_START = 0;
    /** notify Application show Application Candidate List */
    EMV_CANDIDATE_LIST = 1;    // do 3.18, 3.19
    /** Application Select Completed */
    EMV_APP_SELECTED= 2;
    /** Get Process Option Completed */
    EMV_GET_PROC_OPTION= 3;
    /** Read Application Data Completed */
    EMV_READ_APP_DATA= 4;
    /** Data Authentication Completed */
    EMV_DATA_AUTH= 5;
    /** Process Restrict Completed */
    EMV_PROCESS_RESTRICT = 6;
    /** notify Application prompt Caldholder enter Online PIN */
    EMV_ONLINE_ENC_PIN = 7;   // do 3.21
    /** notify Application confirm to Accepted PIN Bypass or not */
    EMV_PIN_BYPASS_CONFIRM = 8;   // do 3.22
    /** Cardholder Verify Completed */
    EMV_CARDHOLDER_VERIFY = 9;
    /** Terminal Risk Management Completed */
    EMV_TERMINAL_RISK_MANAGEMENT    = 10;
    /** notify Application to Process Online */
    EMV_PROCESS_ONLINE= 11;   // do 3.23
```

```
/** notify Application Check Cardholder's Identification */
EMV_ID_CHECK= 12;   // do 3.20
```

A.2.3.5.8.4 其他功能

A.读取EMV Kernal版本

```
void emv_get_version_string(unsigned char *buffer)
```

B.设置交易金额

```
void emv_set_trans_amount(unsigned int amount) // 以分为单位
```

C.设置交易类型

```
void emv_set_trans_type(unsigned char trans_type)

#define TRANS_GOODS_SERVICE      0x00
#define TRANS_CASH               0x01
#define TRANS_INQUIRY            0x04
#define TRANS_TRANSFER           0x05
#define TRANS_PAYMENT            0x06
#define TRANS_ADMIN              0x07
#define TRANS_CASHBACK           0x09
#define TRANS_CARD_RECORD        0x0A
#define TRANS_EC_BALANCE         0x0B
```

D.设置其他金额

```
void emv_set_other_amount(unsigned int amount) // 以分为单位
```

E.清除AID参数

```
int emv_aid_param_clear()
```

F.增加AID参数

```
int emv_aid_param_add(unsigned char *aid_param)


typedef struct{
unsigned char aid_length;
    unsigned char aid[16];                      // [BCD] Application Identifier
unsigned char app_lable[16+1];              // [ASC] Application Label
/** Application Preferred Name */
unsigned char app_preferred_name[16+1];     // [ASC]
unsigned char app_priority;                 // Application Priority
/** Terminal Floor Limit */
unsigned char term_floor_limit[4];          // [HEX]
/** Terminal Action Code - Default */
unsigned char term_action_code_default[5];  // [BCD]
/** Terminal Action Code - Denial */
unsigned char term_action_code_denial[5];   // [BCD]
```

```
/** Terminal Action Code - Online */
unsigned char term_action_code_online[5];   // [BCD]
/** Target Percentage */
unsigned char target_percentage;
/** threshold Value */
unsigned char threshold_value[4];           // [HEX]
/** Maximum Target Percentage */
unsigned char max_target_percentage;
/** Acquirer Identifier */
unsigned char acquirer_id[6];               // [BCD]
/** Merchant Category Code */
unsigned char mechant_category_code[2];     // [BCD]
/** Merchant Identifier */
unsigned char merchant_id[15];              // [ASC]
/** Application Version Number */
unsigned char app_version_number[2];        // [BCD]
/** Point-of-Service(POS) Entry Mode */
unsigned char pos_entry_mode;
/** Transaction Reference Currency Code */
unsigned char trans_refer_currency_code[2]; // [BCD]
/** Transaction Reference Currency Exponent */
unsigned char trans_Refer_Currency_Exponent;
/** Default Dynamic Data Authentication Data Object List(DDOL) */
unsigned char default_ddol_length;
unsigned char default_ddol[128];            // [BIN]
/** Default Transaction Certificate Data Object List(TDOL) */
unsigned char default_tdol_length
unsigned char default_tdol[128];            // [BIN]
// supportOnlinePin[0] = 0 means the Application unsupported
// online PIN, any other value means the Application supported
// online PIN
unsigned char support_online_pin;
// supportAIDPartial[0] = 0 means the Application unsupported AID
// Partial, any other value means the Application supported AID
// Partial
unsigned char support_aid_partial;
/** QPBOC Contactless Limit */
unsigned char contactless_limit[4];         // [HEX]
/** QPBOC Contactless Floor Limit */
unsigned char contactless_floor_limit[4];   // [HEX]
/** QPBOC CVM Limit */
unsigned char cvm_limit[4];                 // [HEX]
/** 电子现金终端交易限额 */
unsigned char ec_term_trans_limit[6];       // [BCD]
```

```
}AIDPARAM
```

G.清除CAPK参数

```
int emv_capk_param_clear()
```

H.增加CAPK参数

```
int emv_capk_param_add(unsigned char *capk_param)

typedef struct{
/** Registered Application Provider Identifier */
    unsigned char rid[5];                              // [BCD]
/** Certificate Authority Public Key Index */
unsigned char capki;
/** Hash Algorithm Indicator */
unsigned char hash_ind;
/** Certificate Authority Public Key Algorithm Indicator */
unsigned char arith_ind;
/** The Length of Certificate Authority Public Key Modulus */
unsigned int modul_len;
/** Certificate Authority Public Key Modulus */
unsigned char modul[248];                         // [BIN]
/** The Length of Certificate Authority Public Key Exponent */
unsigned char exponent_len;
/** Certificate Authority Public Key Exponent */
unsigned char exponent[3];                        // [BIN]
/** Certificate Authority Public Key Check Sum */
unsigned char check_sum[20];                      // [BIN]
/** Certificate Expiration Date */
unsigned char expiry[8];                           // [ASC] “YYYYMMDD”
}CAPKPARAM
```

I.设置EMV终端参数

```
int emv_set_terminal_param(unsigned char *terminal_param)


typedef struct{
unsigned char terminal_country_code[2];             // 9F1A [BCD] : Terminal Country
Code
unsigned char tid[8];   // 9F1C [ASC]
unsigned char ifd[8];           // 9F1E [ASC] : IFD Serial Number
```

```
    unsigned char transaction_currency_code[2]; // 5F2A [BCD]
    unsigned char terminal_capabilities[3];             // 9F33 [BIN]
    unsigned char terminal_type[1]; // 9F35 [BCD]
    unsigned char transaction_currency_exponent[1];     // 5F36 [BCD]
    unsigned char additional_terminal_capabilities[5];  // 9F40 [BIN]
    unsigned char merchant_name_length;
    unsigned char merchant_name[20];                    // 9F4E [ASC]
    unsigned char status_check_support;           // qpbc : 是否支持状态检查 0-不支持； 1-
支持
    }TERMINAL_INFO;
```

J.清除卡片黑名单

```
int emv_blackcard_clear()
```

K.增加卡片黑名单

```
int emv_blackcard_add(unsigned char *blackcard)


typedef struct{
/** PAN */
unsigned char card_no[19];
/** PAN Sequence Number */
unsigned char pan_sequence;
}BLACKCARD
```

L.清除证书回收名单

```
int emv_blackcert_clear()
```

M.增加证书回收名单

```
int emv_blackcert_add(unsigned char *blackcert)


typedef struct{
/** Registered Application Provider Identifier */
unsigned char rid[5];
/** Certificate Authority Public Key Index */
unsigned char capki;
}BLACKCERT
```

N.该EMV交易是否需要上送Advice

```
int emv_is_need_advice()
返回1, 需要上送Advice;
返回0,不需要上送Advice.

O.该EMV交易是否需要签名

int emv_is_need_signature()
返回1, 需要签名;
返回0,不需要签名.

P.设置是否强制联机

void emv_set_force_online(int flag)
flag  == 1 强制联机
== 0 不强制

Q.读取卡片交易记录

int emv_get_card_record(unsigned char *record_no, unsigned char *data)

data 长度 = 45 * 10, 即每条记录45字节，最多10条记录。
R.获取应用选择列表

int emv_get_candidate_list(unsigned char *aid_number, unsigned char *aid_list)
typedef struct
{
unsigned char aid_length;
unsigned char aid[16];                          // Application Identifier
unsigned char app_label[16+1];  // Application Label
unsigned char app_preferred_name[16+1]; // Application Preferred Name
unsigned char priority_exist; // Application Priority exist flag
unsigned char app_priority; // Application Priority
}CANDIDATE

S.设置应用选择结果

int emv_set_candidate_list_result(unsigned char index);

T.设置持卡人证件检查结果
根据IDType（9F62）、IDNumber(9F61)，设置证件检查结果

int emv_set_id_check_result(int result)
result  == 1，证件和出示的一致
```

== 0，证件和出示的不符

U.设置OnlinePIN是否输入

```
int emv_set_online_pin_entered(int result)
result  == 1，用户已输入PIN
== 0，用户未输入PIN
```

V.确认是否允许持卡人Bypass PIN（不输入PIN）

```
int emv_set_pin_bypass_confirmed(int result)
result == 1，用户确认不输入脱机PIN
== 0，用户不确认，即需要重新输入脱机PIN
```

W.设置联机认证结果

```
int emv_set_online_result(unsigned char result,
unsigned char * is_suer_resp_data,
unsigned char  is_suer_resp_data_length)
```

result:
0，联机通讯成功，并收到交易成功应答
1，联机通讯失败
2，联机通讯成功，但收到交易拒绝应答（Response Code != "00"）

A.2.3.5.9 第三方应用支付服务

A.2.3.5.9.1 头文件

使用第三方应用支付可信应用需要引入头文件：

```
#include "teei_platform_payment_api.h";
```

A.2.3.5.9.2 数据类型

A.2.3.5.9.2.1 PAY_PayCashRequest

启动收单服务请求。

```
typedef struct
{
    char* AppId, //第三方应用的唯一标识，由银联指定。
    char* AppName, //第三方应用名称，由第三方应用自行定义。
    int TransType, //交易类型可参考本文4.2章的详述。
    long TransAmount, //交易金额的币种均为人民币,保留两位小数，
                      //示例：消费金额为100元，则交易金额应为00000010000。
    long TransIndexCode, //由第三方应用提供，此值用来标识一次交易请求，一天内不能重复。
    char* ReqTransDate, //第三方交易发起请求的日期
    char* ReqTransTime, //第三方交易发起请求的具体时间。
    char* cardInfo2, //[可选]磁条卡第二磁道信息
```

```
    char* cardInfo3 //[可选]磁条卡第三磁道信息
} PAY_PayCashRequest;
```

A.2.3.5.9.2.2 PAY_PayCashResponse

启动收单服务响应。

```
typedef struct
{
    char* AppId, //第三方应用的唯一标识，由银联指定。
    char* AppName, //第三方应用名称，由第三方应用自行定义。
    int TransType, //交易类型可参考本文4.2章的详述。
    long TransAmount, //交易金额的币种均为人民币,保留两位小数，
                      //示例：消费金额为100元，则交易金额应为00000010000。
    long TransIndexCode, //由第三方应用提供，此值用来标识一次交易请求，一天内不能重复。
    char* ReqTransDate, //第三方交易发起请求的日期
    char* ReqTransTime, //第三方交易发起请求的具体时间。
    char* TransDate, //支付系统的处理交易成功的日期。
    char* TransTime, //支付系统的处理交易成功的时间。
    char* RespCode, //支付系统返回给第三方应用的应答码
    char* RespDesc, //支付系统返回给第三方应用的应答码的对应中文解释。
    int OptCode, //[可选] 操作员号只可能是01-98，普通操作员登陆成功签到后才能使用第三方应用。
    char* CardNum, //[可选] 前5位数字和末4位数字显示，其他位数显示为*。
    int BatchNum, //[可选] 交易中心返回的6位批次号，每批结算一次批次号加一。
    int CertNum, //[可选] 6位凭证号，相当于交易流水号，每一批次凭证号从000000开始递增。
    char* ReferCode, //[可选] 12位交易参考号是交易中心返回的随机编码，不会重复。
    char* Reference, //[可选] 交易中心返回，可能为空值。
    int FeeAmount //[可选] 备用
} PAY_PayCashResponse;
```

A.2.3.5.9.2.3 PAY_GetPayInfoRequest

查询支付信息请求。

```
typedef struct
{
    char* AppId, //第三方应用的唯一标识，由银联指定。
    char* AppName, //第三方应用名称，由第三方应用自行定义。
    int TransType, //交易类型可参考本文4.2章的详述。
    long TransIndexCode, //此值填写交易发起时的索引号，用来查询一次交易请求，和交易日期配合使用。
    char* ReqTransDate, //第三方交易发起请求的日期，此值限制交易索引。
} PAY_GetPayInfoRequest;
```

A.2.3.5.9.2.4 PAY_GetPayInfoResponse

查询支付信息响应。

```
typedef struct
{
    char* AppId, //第三方应用的唯一标识，由银联指定。
    char* AppName, //第三方应用名称，由第三方应用自行定义。
```

```
    int TransType, //返回的交易类型同请求的交易类型，中文显示。
    long TransIndexCode, //此值同第三方请求的交易索引号，返回值和请求值相同
    char* ReqTransDate, //第三方交易发起请求的日期
    char* ReqTransTime, //第三方交易发起请求的具体时间。
    char* TransDate, //支付系统的处理交易成功的日期。
    char* TransTime, //支付系统的处理交易成功的时间。
    char* RespCode, //支付系统返回给第三方应用的应答码
    char* RespDesc, //支付系统返回给第三方应用的应答码的对应中文解释。
    long TransAmount, //交易金额的币种均为人民币,保留两位小数，
                      //示例：消费金额为100元，则交易金额应为00000010000。
} PAY_GetPayInfoResponse;
```

A.2.3.5.9.2.5 PAY_GetPOSInfoRequest

查询POS信息请求。

```
typedef struct
{
    char* AppId, //第三方应用的唯一标识，由银联指定。
    char* AppName, //第三方应用名称，由第三方应用自行定义。
    int TransType, //交易类型可参考本文4.2章的详述。
    char* ReqTransDate, //第三方交易发起请求的日期。
} PAY_GetPayInfoRequest;
```

A.2.3.5.9.2.6 PAY_GetPOSInfoResponse

查询POS信息响应。

```
typedef struct
{
    long MerchantID, //商户ID，唯一标志商户。
    long TerminaID, //终端ID，唯一标志终端。
    long OperatorID, //操作员ID，唯一标志操作员。
    char* RespCode, //支付系统返回给第三方应用的应答码
    char* RespDesc, //支付系统返回给第三方应用的应答码的对应中文解释。
} PAY_GetPayInfoResponse;
```

A.2.3.5.9.3 调用交易接口

该接口定义为调用收单应用，完成包括磁条卡和PBOC交易。

接口名称	PAY_PayCash			
功能描述	发起银联交易支付交易			
接口格式	int PAY_PayCash(PAY_PayCashRequest* request, PAY_PayCashResponse* response);			
参数说明	Request	PAY_PayCashRequest*	必选	请求消息体
	response	PAY_PayCashResponse*	必选	响应消息体
返回值	1. 等于0，打开成功 2. 小于0，错误；参考附录一：错误定义表			

A.2.3.5.9.4 调用查询接口

该接口定义为调用收单应用，完成支付信息查询处理。

接口名称	PAY_GetPayInfo			
功能描述	支付信息查询处理			
接口格式	int PAY_GetPayInfo(PAY_GetPayInfoRequest* request, PAY_GetPayInfoResponse* response);			
参数说明	request	PAY_GetPayInfoRequest*	必选	请求消息体
	response	PAY_GetPayInfoResponse*	必选	响应消息体
返回值	1. 等于 0，打开成功 2. 小于 0，错误；参考附录一：错误定义表			

A.2.3.5.9.5 终端信息查询接口

查询终端信息。

接口名称	PAY_GetPOSInfo			
功能描述	查询终端信息			
接口格式	int PAY_GetPayInfo(PAY_GetPOSInfoRequest* request, PAY_GetPOSInfoResponse* response);			
参数说明	request	PAY_GetPOSInfoRequest*	必选	请求消息体
	response	PAY_GetPOSInfoResponse*	必选	响应消息体
返回值	1. 等于 0，打开成功 2. 小于 0，错误；参考附录一：错误定义表			

A.2.3.5.9.6 交易类型

接口类型参数值	描述
1	消费
2	查询消费信息
3	查询 POS 信息

A.2.3.5.9.7 返回应答码表

交易返回 POS 终端时都有应答码，可以把操作分为以下几类：

A：交易成功

B：交易失败，可重试

C：交易失败，不需要重试

D：交易失败，终端操作员处理

E：交易失败，系统故障，不需要重试

F：交易失败，支付客户端未处理

3. 1：如果返回应答码不能在下表中找到，中文解释显示“交易失败”

4. 2：如果POS交易的批次号和网络中心批次号不一致时应答码会填 “77”，此时POS机应当提示操作员重新签到，再作交易。

代码	意义	类别	原因/采取的措施	POS 显示的内容

代码	意义	类别	原因/采取的措施	POS 显示的内容
00	承兑或交易成功	A	承兑或交易成功	交易成功
01	查发卡行	C	查发卡行	请持卡人与发卡银行联系
03	无效商户	C	商户需要在银行或中心登记	无效商户
04	没收卡	D	操作员没收卡	此卡被没收
05	身份认证失败	C	发卡不予承兑，预约信息匹配失败	持卡人认证失败
10	部分承兑	A	部分金额批准，请收取余额	显示部分批准金额，提示操作员
11	重要人物批准（VIP）	A	此为 VIP 客户	成功，VIP 客户
12	无效的关联交易	C	发卡行不支持的交易	无效交易
13	无效金额	B	金额为 0 或其他非法值	无效金额
14	无效卡号（无此账号）	B	卡种未在中心登记或读卡号有误	无效卡号
15	无此发卡方	C	此发卡行未与中心开通业务	此卡无对应发卡方
21	卡未初始化	C	1、该卡未激活、开卡； 2、该卡初始密码未变更； 3、初始密码限制的交易； 4、长期未使用而冻结或状态为“睡眠”的卡。	该卡未初始化或睡眠卡
22	故障怀疑，关联交易错误	C	POS 状态与中心不符，可重新签到	操作有误，或超出交易允许天数
25	找不到原始交易	C	发卡行未能找到有关记录	没有原始交易，请联系发卡方
30	报文格式错误	C	格式错误（不符合磁道预校验规则）	请重试
34	有作弊嫌疑	D	有作弊嫌疑的卡，操作员可以没收	作弊卡,吞卡
38	超过允许的 PIN 试输入	D	密码错次数超限，操作员可以没收	密码错误次数超限，请与发卡方联系
40	请求的功能尚不支持	C	发卡行不支持的交易类型	发卡方不支持的交易类型
41	挂失卡	D	挂失的卡， 操作员可以没收	挂失卡，请没收（POS）
43	被窃卡	D	被窃卡， 操作员可以没收	被窃卡，请没收
51	资金不足	C	账户内余额不足	可用余额不足
54	过期的卡	C	过期的卡	该卡已过期
55	不正确的 PIN	C	密码输错	密码错
57	不允许持卡人进行的交易	C	不允许持卡人进行的交易	不允许此卡交易

代码	意义	类别	原因/采取的措施	POS 显示的内容
58	不允许终端进行的交易	C	该商户不允许进行的交易	发卡方不允许该卡在本终端进行此交易
59	有作弊嫌疑	C	CVN 验证失败	卡片校验错
61	超出金额限制	C	一次交易的金额太大	交易金额超限
62	受限制的卡	C		受限制的卡
64	原始金额错误	C	原始金额不正确	交易金额与原交易不匹配
65	超出消费次数限制	C	超出消费次数限制	超出消费次数限制
68	发卡行响应超时	C	发卡行规定时间内没有回答	交易超时，请重试
75	允许的输入 PIN 次数超限	C	允许的输入 PIN 次数超限	密码错误次数超限
90	正在日终处理	C	日期切换正在处理	系统日切，请稍后重试
91	发卡方不能操作	C	电话查询发卡方或银联，可重作	发卡方状态不正常，请稍后重试
92	金融机构或中间网络设施找不到或无法达到	C	电话查询发卡方或网络中心，可重作	发卡方线路异常，请稍后重试
94	重复交易	C	查询网络中心，可能是一笔已经成功上送的交易	拒绝，重复交易，请稍后重试
96	银联处理中心系统异常、失效	C	发卡方或网络中心出现故障	拒绝，交换中心异常，请稍后重试
97	POS 终端号找不到	D	终端未在中心或银行登记	终端未登记
98	银联处理中心收不到发卡方应答	E	银联收不到发卡行应答	发卡方超时
99	PIN 格式错	B	可重新签到作交易	PIN 格式错，请重新签到
A0	MAC 鉴别失败	B	可重新签到作交易	MAC 校验错，请重新签到
A1	转账货币不一致	C	转账货币不一致	转账货币不一致
A2	有缺陷的成功	A	银联处理中心转发了原充值交易请求，但未收到发卡方应答时，银联处理中心直接向受理方应答为有缺陷的成功交易	交易成功，请向发卡行确认
A3	资金到账行无此账户	C	资金到账行账号不正确	账户不正确
A4	有缺陷的成功	A	未收到原充值交易请求时，对关联的确认交易的承兑为有缺陷的成功交易	交易成功，请向发卡行确认
A5	有缺陷的成功	A	原充值交易为拒绝时，对关联的确认交易的承兑为有缺陷的成功交易	交易成功，请向发卡行确认

代码	意义	类别	原因/采取的措施	POS 显示的内容
A6	有缺陷的成功	A	银联处理中心转发了原充值交易请求，但未收到发卡方应答时，对受理方发来的关联的确认交易的承兑为有缺陷的成功交易	交易成功，请向发卡行确认
A7	安全处理失败	C	1、调用 MAC 校验程序失败 2、调用 PIN 校验程序失败 3、MAC 处理失败 4、密钥处理失败	拒绝，交换中心异常，请稍后重试
100	上一笔交易未完成	F	等待上一笔交易完成后重试	上一笔交易未完成
101	请求报文数据格式错误	F	重新组装正确的报文	请求报文数据格式错误
102	支付调用取消	F	第三方应用收到支付响应报文之前，支付界面关闭	支付调用取消
103	支付应用未启动或未登录	F	启动支付客户端登陆后重试	支付应用未启动或未登录
104	未找到交易类型	F	支付后台不支持的交易类型	未找到交易类型
105	未知错误，交易失败	F	未知原因引起的错误	未知错误，交易失败
RS	冲正成功	B	交易超时无应答，冲正成功	冲正成功
RF	冲正失败	B	交易超时无应答，冲正无应答	冲正失败

A.2.3.6 银行卡支付客户端

应能受理银联线下收单业务，并声明磁条卡阅读器、接触式IC卡阅读器、非接触式IC卡阅读器、PIN输入设备、打印机访问权限。银行卡支付客户端应由终端管理员从多应用管理平台上下载、安装及升级。

银行卡支付客户端中非敏感数据和业务处理在REE端存在和处理，敏感数据和业务处理在TEEI端存在和处理，在TEEI平台中存在对应的银行卡支付可信应用（TA，Trusted Application）与之通信，REE系统和TEEI系统之间的通信可通过REE系统和TEEI系统之间的通信接口进行调用。

A.2.3.7 系统基本能力

能正常运行银联颁布的银联支付客户端、银联应用商店。

A.2.4 多应用管理要求

在基于TEEI系统定制智能系统中，银联已发布了银联应用商店客户端（即上文所述的多应用管理客户端）、银联应用商店平台（即上文所述的多应用管理平台）。基于TEEI系统定制的智能销售点终端应能正常运行银联应用商店客户端。

A.3 基于N3 TEE系统定制的智能销售点终端的细化要求

A.3.1 N3 TEE系统概述

N3 TEE（N3 Trusted Execution Environment，N3可信执行环境），是一种以Java技术为基础的创新型可信操作系统，它主要使用以类Java虚拟机技术为核心构建的TEE内核来实现Java可信应用的搭载

和运行，较好地实现了TEE可信应用的兼容性和跨平台可移植性。另外，N3 TEE与其他TEE一样都是需要安全硬件模块保障的可信操作系统。因此，与TEEI系统运行环境类似，N3 TEE也要求智能设备具备两种执行环境，其中N3 TEE运行在安全执行环境下，而移动操作系统如Android等则运行在多媒体执行环境下。N3 TEE可信应用程序执行时的一些敏感信息操作都将受到该安全硬件模块的隔离保护从而保证操作的安全性。

A.3.2 系统功能要求

A.3.2.1 系统版本

运行N3 TEE系统的智能设备需要支持硬件安全技术技术。

A.3.2.2 外设驱动标准接口

以下基于N3 TEE系统的系统外设驱动接口均以Java语言规范描述。银联的第三方智能销售点终端生产商可以通过提供java jar包以便应用开发商开发应用。

而如果第三方智能销售点终端生产商提供的是C语言的动态库文件，则需要考虑采用JNI本地抽象化的方式予以实现。

A.3.2.2.1 接触式IC卡阅读器

驱动接口信息

设备名称	CLOUD_DEVICE_CONTACT_IC_CARD
设备用途	智能终端上一个或多个接触式IC卡操作
设备驱动文件	/system/lib/libCloudPos.jar

驱动调用方法	预先将设备驱动文件对应的jar包导入设备中，再通过该驱动文件所对应的ContactICReader对象进行调用。
	//调用接触式IC卡阅读器类
	URL url = new URL("/system/lib/libCloudPos.jar ");
	URLClassLoader urlCL = new URLClassLoader(new URL[]{url});
	Class clazz = urlCL.loadClass("CloudDeviceContactICCard ");
	CloudDeviceContactICCard cloudDeviceContactICCard = (CloudDeviceContactICCard) clazz.newInstance();
	//得到函数
	cloudDeviceContactICCard .("函数名")();

函数列表

设备初始化	smart_card_init
设备回收	smart_card_terminate
查询设备	smart_card_query_max_number
设备打开	smart_card_open
设备关闭	smart_card_close
设备内IC卡查询	smart_card_query_presence
设备上电	smart_card_power_on
设备下电	smart_card_power_off
设置参数	smart_card_set_slot_info
数据传输	smart_card_transmit

A.3.2.2.1.1 设备初始化

设备初始化接口，主要定义为设备资源的初始功能。

接口名称	smart_card_init
功能描述	初始化接触式读卡设备
接口格式	int smart_card_init();
参数说明	无
返回值	1. 大于等于 0，初始化设备成功 2. 小于 0，错误；参考附录一：错误定义表

A.3.2.2.1.2 设备回收

设备回收接口，主要定义为设备资源的释放。

接口名称	smart_card_terminate
功能描述	回收被接触式读卡设备占用的资源
接口格式	int smart_card_terminate();
参数说明	无
返回值	1. 等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表

A.3.2.2.1.3 查询设备

硬件支持多个卡槽的IC卡设备，提供查询功能，后续可以根据查询的卡槽数控制具体操作的设备。

接口名称	smart_card_query_max_number
功能描述	查询本设备中有多少槽（slot）可用
接口格式	int smart_card_query_max_number();
参数说明	无
返回值	1.等于 0，表示没有槽（slot） 2. 大于 0，表示槽（slot）数 3. 小于 0，表示错误；参考附录一：错误定义表

A.3.2.2.1.4 设备打开

设备打开，主要定义为设备打开，对于硬件支持多卡槽的设备，需要指定卡槽编号。

接口名称	smart_card_open			
功能描述	打开指定槽（slot）中的设备			
接口格式	int smart_card_open(int nSlotIndex, SMART_CARD_NOTIFIER pNotify, Object pUserData);			
参数说明	nSlotIndex	int	必选	槽的索引

	pNotify	SMART_CARD_NOTIFYER	必选	回调函数，类型定义为：
				Public interface SMARE_CARD_NOTIFIER {
				abstract void doExec (Object pUserData, int nCardIndex, int nEvent);
				}
				nEvent:可以有如下选项：
				SMART_CARD_EVENT_INSERT_CARD
				SMART_CARD_EVENT_REMOVE_CARD
				SMART_CARD_EVENT_POWER_ON
				SMART_CARD_EVENT_POWER_OFF

	pUserData	Object	必选	用户回调函数的参数
返回值	1. 大于等于 0，设备句柄 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.1.5 设备关闭

关闭设备，主要定义为关闭已经打开的设备。

接口名称	smart_card_close			
功能描述	关闭 smart_card_open 函数打开的设备			
接口格式	int smart_card_close(int Handle);			
参数说明	Handle	int	必选	设备句柄
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.1.6 查询卡在位

查询相应卡槽内的IC卡是否在位。

接口名称	smart_card_query_presence			
功能描述	查询指定卡槽中卡（和该句柄相关联的卡槽）是否存在			
接口格式	int smart_card_query_presence(int Handle);			
参数说明	Handle	int	必选	设备句柄
返回值	1.大于 0，表示存在 2.等于 0，表示不存在 3. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.1.7 设备上电

根据设备特性，给予上电操作，可根据实际情况设置上电参数。

接口名称	smart_card_power_on			
功能描述	给指定卡槽中的卡上电			
接口格式	int smart_card_power_on(int Handle, char[] pATR, int[] pATRBufferLength, SMART_CARD_SLOT_INFO[] pSlotInfo);			
参数说明	Handle	int	必选	设备句柄
	pATR	char[]	必选	用来存放 ATR 的缓冲区
	pATRBufferLength	int[]	必选	在输入的时候，是 pATR 缓冲区的长度。在输出的时候，是存放在 pATR 缓冲区中实际数据的长度。
	pSlotInfo	SMART_CARD_SLOT_INFO[]	必选	存放指定槽的相关信息。
返回值	1. 大于等于 0，成功 2.小于 0，错误；参考附录一：错误定义表			

——数据结构 SMART_CARD_SLOT_INFO.结构的定义如下：

字段名	类型	说明
FIDI	char	参见标准 ISO 7816-3 中关于 TA1 的说明
EGT	char	额外警戒时间（ Extra Guard Time） (ISO 7816-3 标准中参数 TC1 或 N)
WI	char	该参数的定义和使用协议相关，如果协议为 T=0, 该参数的含义为等待时间（Waiting Integer）（缺省值是 10，参见 ISO 7816-3 标准中关于 TC2 的说明） *如果协议为 T=1，该参数的含义为块和字符等待时间（Block and Character Waiting Time Integer）。参见 ISO 7816-3 标准中关于 TB3 的说明
WTX	char	如果协议是 T=1，该参数表示等待时间延长（Waiting Time Extention）
EDC	char	如果协议为 T=1，该参数表示校验计算方式（mode of EDC）HAL_SCS_EDC_LRC 或者 HAL_SCS_EDC_CRC（缺省为 LRC）
protocol	char	协议类型，可以是：* HAL_SCS_PROTOCOL_T0 ：用于 CPU 卡 * HAL_SCS_PROTOCOL_T1 ：用于 CPU 卡 * HAL_SCS_PROTOCOL_RAW ：用于 memory 类卡
power	char	供电电压 ，可以是：* HAL_SCS_POWER_1_8V * HAL_SCS_POWER_3V * HAL_SCS_POWER_5V
conv	char	传输字节的方式，可以是：* HAL_SCS_CONV_DIRECT * HAL_SCS_CONV_INVERSE
IFSC	char	如果协议是 T=1，该参数表示卡的字段大小（Field Size for the Card），可以参看 ISO 7816-3 标准中关于 TA3 的说明

reserved	char [3]	
cwt	int	可以用来设置字符等待时间（Character Waiting Time）
bwt	int	可以用来设置块等待时间（Block Waiting Time）
nSlotInfoItem	int	由以下常量通过或（OR）操作构成，用来指定目前结构中哪些位得到了更新，或哪些项需要设置。 SMART_CARD_SLOT_INFO_FIDI SMART_CARD_SLOT_INFO_EGT SMART_CARD_SLOT_INFO_WI SMART_CARD_SLOT_INFO_WTX SMART_CARD_SLOT_INFO_EDC SMART_CARD_SLOT_INFO_PROTOCOL SMART_CARD_SLOT_INFO_POWER SMART_CARD_SLOT_INFO_CONV SMART_CARD_SLOT_INFO_IFSC SMART_CARD_SLOT_INFO_CWT SMART_CARD_SLOT_INFO_BWT

A.3.2.2.1.8 设备下电

设备下电，下电后不可进行通信，需要关闭后释放资源。

接口名称	smart_card_power_off			
功能描述	给指定卡槽（和该句柄相关联的卡槽）中的卡下电			
接口格式	int smart_card_power_off(int Handle);			
参数说明	Handle	int	必选	设备句柄
返回值	1.大于等于 0，表示成功下电 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.1.9 设置参数

定义为IC卡参数设置，具体是硬件情况而定。

接口名称	smart_card_set_slot_info			
功能描述	设置相关信息			
接口格式	int smart_card_set_slot_info(int Handle, SMART_CARD_SLOT_INFO[] pSlotInfo);			
参数说明	Handle	int	必选	设备句柄
	pSlotInfo	SMART_CARD_SLOT_INFO[]	必选	存放相关参数的

				最新值，参数类型已经在上文（2.2.6）中说明
返回值	1.大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.1.10 数据传输

IC卡数据通信接口，数据格式基于ISO 7816-4关于APDU/TPDU的定义。

接口名称	smart_card_transmit			
功能描述	数据传输			
接口格式	int smart_card_transmit(int Handle, char[] pAPDU, int nAPDULength, char[] pResponse, int[] pResponseLength);			
参数说明	Handle	int	必选	设备句柄
	pAPDU	char[]	必选	存放 APDU 命令,无符号字符型数组
	nAPDULength	int	必选	APDU 命令的长度
	pResponse	char[]	必选	用来存放响应的缓冲区。
	pResonseLength	int[]	必选	输入时作为 pResponse 缓冲区的长度，输出的时候，是响应数据的长度。
返回值	1.大于等于 0，表示通讯成功 2.小于 0，通讯失败；参考附录一：错误定义表			

——数据缓冲区的数据格式基于 ISO 7816-4 关于 APDU/TPDU 的定义

A.3.2.2.2 非接触式IC卡阅读器

驱动接口信息

设备名称	CLOUD_DEVICE_CONTACTLESS_IC_CARD
设备用途	智能终端上一个或多个非接触式IC卡操作
设备驱动文件	/system/lib/libCloudPos.jar
驱动调用方法	预先将设备驱动文件对应的jar包导入设备中，再通过该驱动文件所对应的CloudDeviceContactlessICCard对象进行调用。 //调用接触式IC卡阅读器类 URL url = new URL("/system/lib/libCloudPos.jar "); URLClassLoader urlCL = new URLClassLoader(new URL[]{url}); Class clazz =

	urlCL.loadClass("CloudDeviceContactlessICCard ");
	CloudDeviceContactlessICCard cloudDeviceContactlessICCard = (CloudDeviceContactlessICCard) clazz.newInstance();
	//得到函数
	cloudDeviceContactlessICCard.("函数名")();

函数列表

设备打开	contactless_card_open
设备关闭	contactless_card_close
搜索卡片	contactless_card_search_target_begin
结束搜索	contactless_card_search_target_end
连接卡片	contactless_card_attach_target
断开连接	contactless_card_detach_target
命令交互	contactless_card_transmit
卡片控制	contactless_card_send_control_command

A.3.2.2.2.1 设备打开

打开非接卡设备，定义为打开设备，初始化资源。

接口名称	contactless_card_open			
功能描述	创建或者打开一个已经存在的非接触读卡设备			
接口格式	int contactless_card_open(CONTACTLESS_CARD_NOTIFIER fNotifier, Object pUserData,int[] pErrorCode);			

参数说明	fNotifier	CONTACTLESS_CARD_NOTIFIER	必选	回调函数 CONTACTLESS_CARD_NOTIFIER 被定义为：
				Public interface CONTACTLESS_CARD_NOTIFIER {
				abstract void doExec (Object pUserData, int nEvent, char[] pEventData, int nDataLength);
				}

	pUserData	Object	必选	用户数据，会作为回调函数的参数，传入
	pErrorCode	int[]	必选	错误代码。当返回值等于 0 的时候设置
返回值	1. 等于 0，错误 2. 不等于 0，设备句柄			

A.3.2.2.2.2 设备关闭

定义为关闭设备，释放资源。

接口名称	contactless_card_close			
功能描述	关闭非接触读卡设备			
接口格式	int contactless_card_close(int nHandle);			
参数说明	nHandle	int	必选	设备句柄
返回值	1. 大于等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表			

A.3.2.2.2.3 搜索设备

搜索卡片，定义为卡片检测，检测非接卡有效区域内的卡片。

接口名称	contactless_card_search_target_begin			
功能描述	开始搜索非接触 IC 卡			
接口格式	int contactless_card_search_target_begin(int nHandle, int nCardMode, int nFlagSearchAll,int nTimeout_MS);			
参数说明	nHandle	int	必选	设备句柄
	nCardMode	int	必选	卡类型[0,1,2,3]，分别为： CONTACTLESS_CARD_MODE_AUTO， CONTACTLESS_CARD_MODE_TYPE_A ， CONTACTLESS_CARD_MODE_TYPE_A ， CONTACTLESS_CARD_MODE_TYPE_B ， CONTACTLESS_CARD_MODE_TYPE_C
	nFlagSearchAll	int	必选	0：找一张卡，1：找所有的卡
	nTimeout_MS	int	必选	超时，单位为毫秒。小于 0 时，将一直搜索，直到使用方法： contactless_card_search_target_end 打断。
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.2.4 停止搜索

停止当前的搜索，停止后，与搜索接口成对出现。

接口名称	contactless_card_search_target_end			
功能描述	停止搜索非接触 IC 卡			
接口格式	int contactless_card_search_target_end(int nHandle);			
参数说明	nHandle	int	必选	设备句柄
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.2.5 连接卡片

卡片连接函数，当存在多张卡片时，只能与一张卡片连接，在传递APDU命令前，必须要连接非接触IC卡

接口名称	contactless_card_attach_target			
功能描述	在传递 APDU 命令前，链接非接触 IC 卡			
接口格式	int contactless_card_attach_target(int nHandle, char[] pATRBuffer, int nATRBufferLength);			
参数说明	nHandle	int	必选	设备句柄
	pATRBuffer	char[]	必选	ATR 缓冲区
	nATRBufferLength	int	必选	ATR 缓冲区长度
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.2.6 断开连接

解除和当前卡片的连接状态，解除连接状态后，将不能继续进行APDU命令的传输。

接口名称	contactless_card_detach_target			
功能描述	和非接触 IC 卡解除链接			
接口格式	int contactless_card_detach_target(int nHandle);			
参数说明	nHandle	int	必选	设备句柄
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.2.7 APDU 命令交互

非接卡APDU交互接口。

接口名称	contactless_card_transmit			
功能描述	发送 APDU 命令并接收响应			
接口格式	int contactless_card_transmit(int nHandle, char[] pAPDU, int nAPDULength, char[] pResponse, int[] pResponseLength);			
参数说明	nHandle	int	必选	设备句柄
	pAPDU	char[]	必选	APDU 命令
	nAPDULength	int	必选	APDU 命令长度
	pResponse	char[]	必选	存储 APDU 命令响应的缓冲区
	pResponseLength	int[]	必选	输入时，为响应APDU 命令缓冲区长度。输出时，为响应数据长度
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.2.8 IC 卡控制

有些硬件在使用时，需要进行特殊指令的传输，该接口定义为提供这种指令的调用。

接口名称	contactless_card_send_control_command			
功能描述	发送非接触 IC 卡控制命令			
接口格式	int contactless_card_send_control_command(int nHandle, int nCmdID, char[] pCmdData, int nDataLength);			
参数说明	nHandle	int	必选	设备句柄
	nCmdID	int	必选	控制命令 ID 号
	pCmdData	char[]	必选	和命令有关的数据。输入时，是和命令有关的数据，如此命令没有数据，可设置为NULL。输出时，为响应数据。
	nDataLength	int	必选	命令的数据长度
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.3 磁条卡阅读器

驱动接口信息

设备名称	CLOUD_DEVICE_MSR
设备用途	控制智能终端上磁条卡读卡器
设备驱动文件	/system/lib/libCloudPos.jar

驱动调用方法	预先将设备驱动文件对应的jar包导入设备中，再通过该驱动文件所对应的CloudDeviceMSR对象进行调用。
	//调用接触式IC卡阅读器类
	URL url = new URL("/system/lib/libCloudPos.jar ");
	URLClassLoader urlCL = new URLClassLoader(new URL[]{url});
	Class clazz = urlCL.loadClass("CloudDeviceMSR");
	CloudDeviceMSRcontactICReader cloudDeviceMSR = (CloudDeviceMSR) clazz.newInstance();
	//得到函数
	cloudDeviceMSR.("函数名")();

函数列表

设备打开	msr_open
设备关闭	msr_close
注册刷卡	contactless_card_search_target_begin
取消刷卡	msr_unregister_notifier
读取磁道错误	msr_get_track_error
获取磁道数据长度	msr_get_track_data_length
获取磁道数据	msr_get_track_data

A.3.2.2.3.1 打开设备

打开磁条卡设备

接口名称	msr_open
功能描述	打开一个已经存在的磁条卡阅读器
接口格式	int msr_open();
参数说明	无
返回值	1. 大于等于 0，成功打开设备 2. 小于 0，错误；参考附录一：错误定义表

A.3.2.2.3.2 关闭设备

关闭磁条卡设备设备未打开关闭无效。

接口名称	msr_close
功能描述	关闭已打开的磁条卡阅读器
接口格式	int msr_close();
参数说明	无

返回值	1. 大于等于 0，成功关闭设备 2. 小于 0，错误；参考附录一：错误定义表

A.3.2.2.3.3 注册刷卡

打开设备后，通过该函数进行刷卡注册，刷卡注册后，磁条卡设备将启动监听，直到等到刷卡或者主动取消。

接口名称	msr_register_notifier			
功能描述	登记回调函数，回调函数登记好之后，如果用户刷卡，回调函数会被调用，用来通知用户来取数据			
接口格式	int msr_register_notifier(MSR_NOTIFIER notifier, Object pUserData);			

参数说明	notifier	MSR_NOTIFIER	必选	回调函数。MSR_NOTIFIER 被定义为：
				Public interface MSR_NOTIFIER {
				abstract void doExec (Object pUserData);
				}

	pUserData	Object	必选	用户数据，会作为回调函数的参数，传入
返回值	1. 大于等于 0，表示成功。 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.3.4 取消刷卡

取消刷卡，只在调用msr_register_notifier之后有用，取消刷卡后，磁条卡设备恢复正常模式。

接口名称	msr_unregister_notifier
功能描述	撤销登记的函数，撤销之后，刷卡后上层程序不会收到通知
接口格式	int msr_unregister_notifier();
参数说明	无
返回值	1. 大于等于 0，表示成功。 2. 小于 0，表示错误；参考附录一：错误定义表

A.3.2.2.3.5 读取磁道错误

读取磁道错误，监听刷卡后，调用该函数确定磁道数据是否正常。

接口名称	msr_get_track_error
功能描述	查询对应磁道在解码过程中发生的错误
接口格式	int msr_get_track_error(int nTrackIndex);

参数说明	nTrackIndex	int	必选	磁道索引，可以是0，1，2
返回值	1. 大于等于0，表示成功。 2. 小于0，表示错误；参考附录一：错误定义表			

A.3.2.2.3.6 获取磁道数据长度

获取磁道数据长度。

接口名称	msr_get_track_data_length			
功能描述	查询对应磁道数据的长度			
接口格式	int msr_get_track_data_length(int nTrackIndex);			
参数说明	nTrackIndex	int	必选	磁道索引，可以是0，1，2
返回值	1. 大于等于0，表示磁道数据的长度。 2. 小于0，表示错误；参考附录一：错误定义表			

A.3.2.2.3.7 获取磁道数据

获取磁道数据明文。

接口名称	msr_get_track_data			
功能描述	获取对应磁道数据的数据			
接口格式	int msr_get_track_data(int nTrackIndex, char[] pTrackData, int nLength);			
参数说明	nTrackIndex	int	必选	磁道索引，可以是0，1，2
	pTrackData	char[]	必选	存放磁道数据的缓冲区。
	nLength	int	必选	数据缓冲区的长度。由于有接口查询长度信息，所以调用者可以避免缓冲区长度不够的错误。
返回值	1. 大于等于0，表示磁道数据的长度。 2. 小于0，表示错误；参考附录一：错误定义表			

A.3.2.2.4 PIN输入设备

驱动接口信息

设备名称	CLOUD_DEVICE_PINPAD

设备用途	控制智能终端上PIN输入设备
设备驱动文件	/system/lib/libCloudPos.jar

驱动调用方法	预先将设备驱动文件对应的jar包导入设备中，再通过该驱动文件所对应的CloudDevicePinpad对象进行调用。
	//调用接触式IC卡阅读器类
	URL url = new URL("/system/lib/libCloudPos.jar ");
	URLClassLoader urlCL = new URLClassLoader(new URL[]{url});
	Class clazz = urlCL.loadClass("ContactICReader");
	CloudDevicePinpad cloudDevicePinpad = (CloudDevicePinpad) clazz.newInstance();
	//得到函数
	cloudDevicePinpad .("函数名")();

函数列表

打开设备	pinpad_open
关闭设备	pinpad_close
PIN输入设备显示	pinpad_show_text
选择密钥	pinpad_select_key
数据加密	pinpad_encrypt_string
获取磁道数据长度	msr_get_track_data_length
获取磁道数据	msr_get_track_data

A.3.2.2.4.1 打开设备

打开PIN输入设备。

接口名称	pinpad_open
功能描述	打开存在的 PIN 输入设备
接口格式	int pinpad_open();
参数说明	无
返回值	1. 等于 0，打开成功 2. 小于 0，错误；参考附录一：错误定义表

A.3.2.2.4.2 关闭设备

关闭PIN输入设备。

接口名称	pinpad_close
功能描述	关闭已打开的 PIN 输入设备
接口格式	int pinpad_close();
参数说明	无

返回值	1. 等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表

A.3.2.2.4.3 PIN 输入设备显示

PIN输入设备显示接口，将数据显示在PIN输入设备LCD上。

接口名称	pinpad_show_text			
功能描述	写入显示数据同时显示在 PIN 输入设备 LCD 上			
接口格式	int pinpad_show_text(int nLineIndex, char[] strText, int nLength, nFlagSound);			
参数说明	nLineIndex	int	必选	行标，从上到下，依次从 0 开始
	strText	char[]	必选	写入的显示数据
	nLength	int	必选	写入数据的字节长度，为 0 时，清 PIN 输入设备 LED
	nFlagSound	int	必选	声音提示，0 不提示，1 提示
返回值	1. 大于等于 0，写入显示成功 2. 小于 0，写入显示失败			

A.3.2.2.4.4 选择密钥

选择当前操作的PIN输入设备主密钥和工作密钥（用户密钥）。

接口名称	pinpad_select_key			
功能描述	选择一个 PIN 输入设备主密钥和用户密钥			
接口格式	int pinpad_select_key(int nKeyType, int nKeyMasterKeyID, int nUserKeyID, int aAlgorith);			
参数说明	nKeyType	int	必选	主密钥类型。0：dukpt，1：Tdukpt，2：master key，3：public key，4：fix key
	nKeyMasterKeyID	int	必选	主密钥 ID，[0x00, …,0x09]。当 nKeyType 为 master key 时使用
	nUserKeyID	int	必选	用户密钥 ID，[0x00,0x01]。当 nKeyType 为 master key 时使用
	aAlgorith	int	必选	算法选择。1：3DES，2：DES
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.4.5 数据加密

使用当前选择的主密钥和工作密钥进行数据加密。

接口名称	pinpad_encrypt_string

功能描述	使用用户密钥加密数据			
接口格式	int pinpad_encrypt_string(char[] pPlainText, int nTextLength, char[] pCipherTextBuffer,int nCipherTextBufferLength);			
参数说明	pPlainText	char[]	必选	待加密字符串
	nTextLength	int	必选	待加密字符串长度
	pCipherTextBuffer	char[]	必选	用于存储密文的缓冲区
	nCipherTextBufferLength	int	必选	缓冲区长度
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.4.6 更新工作密钥

更新工作密钥，工作密钥为密文下载。

接口名称	pinpad_update_user_key			
功能描述	更新用户密钥			
接口格式	int pinpad_update_user_key(int nMasterKeyID,int nUserKeyID, char[] pCipherNewUserKey,int nCipherNewUserKeyLength);			
参数说明	nMasterKeyID	int	必选	PIN 输入设备主密钥 ID
	nUserKeyID	int	必选	PIN 输入设备用户密钥 ID
	pCipherNewUserKey	char[]	必选	待更细的用户密钥，已加密过
	nCipherNewUserKeyLength	int	必选	待更细用户密钥的长度
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.4.7 计算 PINBLOCK

计算Pinblock, 键盘输入PIN处理方式为ISO9564-1格式0.

接口名称	pinpad_calculate_pin_block
功能描述	计算 Pin Block
接口格式	int pinpad_calculate_pin_block(char[] pASCIICardNumber, int nCardNumberLength, char[] pPinBlockBuffer, int nPinBlockBufferLength, int nTimeout_MS, int nFlagSound)

参数说明	pASCIICardNumber	char[]	必选	银联卡卡号
	nCardNumberLength	int	必选	银联卡卡号长度
	pPinBlockBuffer	char[]	必选	保存 Pin Block 的缓冲区
	nPinBlockBufferLength	int	必选	缓冲区长度
	nTimeout_MS	int	必选	输入超时，单位毫秒，当小于零时，一直等待输入
	nFlagSound	int	必选	是否需要声音提示，0 否，1 是
返回值	1. 大于等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.4.8 计算 Mac

根据指定的用户密钥计算MAC。

接口名称	pinpad_calculate_mac			
功能描述	使用用户密钥计算 MAC			
接口格式	int pinpad_calculate_mac(char[] pData, int nDataLength, int nMACFlag, char[] pMACOutBuffer, int nMACOutBufferLength)			
参数说明	pData	char[]	必选	数据
	nDataLength	int	必选	数据长度
	nMACFlag	int	必选	0：X99，1：ECB
	pMACOutBuffer	char[]	必选	保存 MAC 的缓冲区
	nMACOutBufferLength	int	必选	缓冲区长度
返回值	1. 大于等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表			

A.3.2.2.4.9 设置 Pin 长度

指定Pin输入的长度。

接口名称	pinpad_set_pin_length
功能描述	设置 Pin 的最大限制长度

接口格式	int pinpad_set_pin_length(int nLength, int nFlag)			
参数说明	nLength	int	必选	长度范围[0x00,0x0D]
	nFlag	int	必选	0：最小长度，1：最大长度
返回值	1. 大于等于0，成功 2. 小于0，错误；参考附录一：错误定义表			

A.3.2.2.5 打印机

驱动接口信息

设备名称	CLOUD_DEVICE_PRINTER
设备用途	控制智能终端上打印机设备
设备驱动文件	/system/lib/libCloudPos.jar

驱动调用方法	预先将设备驱动文件对应的jar包导入设备中，再通过该驱动文件所对应的CloudDevicePrinter对象进行调用。
	//调用打印机类
	URL url = new URL("/system/lib/libCloudPos.jar ");
	URLClassLoader urlCL = new URLClassLoader(new URL[]{url});
	Class clazz = urlCL.loadClass("CloudDevicePrinter");
	CloudDevicePrinter cloudDevicePrinter = (CloudDevicePrinter) clazz.newInstance();
	//得到函数
	cloudDevicePrinter.(“函数名”)();

函数列表

打开设备	printer _open
关闭设备	printer _close
传输打印数据	printer _write
开启打印	printer _begin
关闭打印	printer _end

A.3.2.2.5.1 打开设备

打开打印机设备。

接口名称	printer_open
功能描述	打开一个已经存在的打印设备
接口格式	int printer_open();

参数说明	无
返回值	1. 等于 0，打开成功 2. 小于 0，错误；参考附录一：错误定义表

A.3.2.2.5.2 关闭设备

关闭打印机设备。

接口名称	printer_close
功能描述	关闭已打开的打印设备
接口格式	int printer_close();
参数说明	无
返回值	1. 等于 0，成功 2. 小于 0，错误；参考附录一：错误定义表

A.3.2.2.5.3 传输打印数据

写入打印数据并开启打印。

接口名称	printer_write			
功能描述	从当前点连续打印，打印的数据存放在缓冲区			
接口格式	int printer_write(char[] pData, int nDataLength);			
参数说明	pData	char[]	必选	打印数据或打印控制命令
	nDataLength	int	必选	打印数据或控制命令长度
返回值	1. 等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表			

A.3.2.2.5.4 开启打印

开启打印，并分配资源，在printer_write之前调用。

接口名称	printer_begin
功能描述	准备开始打印
接口格式	int printer_begin();
参数说明	无
返回值	1. 等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表

A.3.2.2.5.5 结束打印

结束打印，并释放资源。在所有数据写完后调用。

接口名称	printer_end
功能描述	结束打印
接口格式	int printer_end();
参数说明	无
返回值	1. 等于 0，表示成功 2. 小于 0，表示错误；参考附录一：错误定义表

A.3.2.2.6 证书管理与加密运算模块

驱动接口信息

设备名称	CLOUD_DEVICE_PINPAD
设备用途	控制智能终端上证书管理与加密运算模块
设备驱动文件	/system/lib/ libPKCS11Wrapper.jar
驱动调用方法	预先将设备驱动文件对应的jar包导入设备中，再通过该驱动文件所对应的PKCS11Wrapper对象进行调用。 //调用证书管理与加密运算模块类 URL url = new URL("/system/lib/ libPKCS11Wrapper.jar"); URLClassLoader urlCL = new URLClassLoader(new URL[]{url}); Class clazz = urlCL.loadClass("PKCS11Wrapper"); PKCS11Wrapper pKCS11Wrapper = (PKCS11Wrapper) clazz.newInstance();

函数列表

打开硬件证书管理与加密运算模块设备	hsm_osm_open
关闭硬件证书管理与加密运算模块设备	int hsm_osm_close
保存安全证书	hsm_osm_save_object
删除指定证书	hsm_osm_delete_object
删除所有证书	hsm_osm_delete_all
查询证书数量	hsm_osm_query_object_count
读取证书	hsm_osm_load_object

A.3.2.2.6.1 打开硬件证书管理与加密运算模块设备

函数	int hsm_osm_open();
描述	打开硬件证书管理与加密运算模块设备
参数	无
返回值	1. 等于 0，打开成功 2. 小于 0，错误

A.3.2.2.6.2 关闭硬件证书管理与加密运算模块设备

函数	int hsm_osm_close();
描述	关闭硬件证书管理与加密运算模块设备
参数	无
返回值	1. 等于 0，关闭成功 2. 小于 0，错误

A.3.2.2.6.3 保存安全证书

函数	int hsm_osm_save_object(HSM_OBJECT_PROPERTY[] pObjectProperty, char[] pObjectData, int nDataLength, HSM_OBJECT_DATA_TYPE nDataType);			
描述	保存安全证书至硬件证书管理与加密运算模块设备			
参数	pObjectProperty	HSM_OBJECT_PROPERTY[]	必选	证书属性。
	pObjectData	char[]	必选	证书数据
	nDataLength	int	必选	证书数据长度
	nDataType	HSM_OBJECT_DATA_TYPE	必选	证书格式。
返回值	1. 等于 0，保存成功。 2. 小于 0，保存失败。			

A.3.2.2.6.4 删除指定证书

函数	int hsm_osm_delete_object(HSM_OBJECT_PROPERTY[] pObjectProperty, char[] pPIN, int nPINLength);			
描述	从硬件证书管理与加密运算模块设备中删除指定证书			
参数	pObjectProperty	HSM_OBJECT_PROPERTY[]	必选	证书属性。
	pPIN	char[]	必选	PIN 值。

	nPINLength	int	必选	PIN 值长度。
返回值	1. 等于 0，删除成功 2. 小于 0，错误			

A.3.2.2.6.5 删除所有证书

函数	int hsm_osm_delete_all(char[] pPIN, int nPINLength);			
描述	从硬件证书管理与加密运算模块中设备删除所有证书			
参数	pPIN	char[]	必选	保留的 Pin 值
	nPINLength	int	必选	PIN 值长度
返回值	1. 等于 0，删除成功 2. 小于 0，错误			

A.3.2.2.6.6 查询证书数量

函数	int hsm_osm_query_object_count (HSM_OBJECT_PROPERTY[] pObjectProperty);			
描述	从硬件证书管理与加密运算模块内查询证书数量			
参数	pObjectProperty	HSM_OBJECT_PROPERTY[]	必选	证书属性。
返回值	1. 小于 0，查询失败。 2. 大于等于 0，证书数量。			

A.3.2.2.6.7 读取证书

函数	int hsm_osm_load_object(int nIndex, HSM_OBJECT_PROPERTY[] pObjectProperty, char[] pDataBuffer, int nDataBufferLength, HSM_OBJECT_DATA_TYPE nDataType);			
描述	从硬件证书管理与加密运算模块内读取证书			
参数	nIndex	int	必选	证书索引号。
	pObjectProperty	HSM_OBJECT_PROPERTY[]	必选	证书属性。
	pDataBuffer	char[]	必选	证书数据。
	nDataBufferLength	int	必选	证书数据长度。
	nDataType	HSM_OBJECT_DATA_TYPE	必选	证书格式。

返回值	1. 等于 0，读取成功 2. 小于 0，读取失败

A.3.2.2.6.8 参数类型说明

证书类型数据结构：pem 证书格式；der 编码证书；PKCS #7 证书；PKCS#12 证书；
——数据结构 Object_data_type 结构的定义如下：

字段名	类型	说明
HSM_OBJECT_DATA_TYPE	int[]	以下为四个枚举值： HSM_OBJECT_DATA_TYPE_pem HSM_OBJECT_DATA_TYPE_der HSM_OBJECT_DATA_TYPE_p7b HSM_OBJECT_DATA_TYPE_pfx

证书种类数据结构：私钥；公钥；cert 证书；
——数据结构 object_type 结构的定义如下：

字段名	类型	说明
HSM_OBJECT_TYPE	int[]	以下为四个枚举值： HSM_OBJECT_TYPE_private_key HSM_OBJECT_TYPE_public_key HSM_OBJECT_TYPE_cert

证书属性数据结构：证书 ID；证书 Label；密码；
——数据结构 object_proptery 结构的定义如下：

字段名	类型	说明
HSM_OBJECT_PROPERTY	int[32]	以下为四个枚举值： int strID[32]; int strLabel[32]; int strPassword[32]; HSM_OBJECT_TYPE nObjectType;

A.3.2.2.7 错误定义表

错误名	错误值	说明
EDEVICENAME	-1	设备名称不存在
EIO	-2	底层 I/O 错误
EUNKOWN	-10	未知错误
E_HAL_UNKOWN	-100	未知错误

E_HAL_ARG	-101	输入参数错误
E_HAL_IO	-102	底层I/O错误

A.3.2.3 IC卡内核接口

A.3.2.3.1 EMV tag存取功能

A. EMV tag的值是否存在

接口名称	emvIsTagPresent
功能描述	判断 Tag 值存在与否
接口格式	boolean emvIsTagPresent(short tag);
参数说明	Tag 值
返回值	1. true，成功打开设备 2. false

B. 读取EMV tag的值,如果该tag的值不存在，则返回false

接口名称	emvGetTagData
功能描述	读取 Tag 值
接口格式	String emvGetTagData(short tag)
参数说明	Tag 值
返回值	以 Tag 表示的内容

C. 设置EMV tag的值

接口名称	emvSetTagData
功能描述	设置 Tag 值
接口格式	void emvSetTagData (String data, short tag)
参数说明	1. Tag 的内容 2. Tag
返回值	无

A.3.2.3.2 EMV交易处理功能

A. EMV参数初始化

void emvInitialize(char kernelType, void *parameter)

接口名称	emvInitialize
功能描述	设置 Tag 值
接口格式	void emvInitialize(char kernelType, EMVFUNC_PARAMETER parameter)

参数说明	1.内核类型 kernelType 2. EMVFUNC_PARAMETER 类引用
返回值	无

内核类型kernerlType说明:

内核类型	值	说明
PBOC_KERNAL	1	借贷记
QPBOC_KERNAL	2	QPBOC
UPCASH_KERNAL	3	电子现金

--EMVFUNC_PARAMETER类结构定义如下：

字段名	类型	说明
readerHandle	int	读卡器的句柄或类型
cardType	int	交互的卡片类型： CARD_CONTACT: 1; CARD_CONTACTLESS: 2
atrLength	int	卡片复位时，ATR 长度
atr[30]	char	卡片复位时，响应 ATR
smartCardTransmit 方法	返回：int	Transmit 接口，见以下介绍
emvProcessNextCompleted	返回：char[]	回调函数接口，见以下介绍

方法介绍:

1）Transmit函数,命令发送及响应

接口名称	smartCardTransmit
功能描述	APDU 命令发送及响应
接口格式	int smartCardTransmit(int handle, char[] pAPDU, short APDULength, char[] pResponse, short pResponseLength);
参数说明	1. 读卡器的句柄或类型 2. APDU 命令 3. APDU 长度 4.响应数据 5.响应数据长度
返回值	1. 返回为 0，执行成功 2. 返回非 0，执行失败

2) 回调函数

接口名称	emvProcessNextCompleted

功能描述	回调函数，指明某个业务处理执行完成后，正常或异常情况
接口格式	void emvProcessNextCompleted(char status, char info);
参数说明	1. 表明业务处理执行状态，见 2. 各自状态下，对应的信息
返回值	无

B.调用EMV kernal处理宿主函数emvProcess.

接口名称	emvProcessNext
功能描述	EMV 业务处理入口
接口格式	int emvProcessNext()
参数说明	无
返回值	1.返回为 0，执行成功 2.返回非 0，执行失败

C. EMV kernel处理回调函数，见A中回调函数

void emvProcessNextCompleted(char status, char info)

status说明，见下表:

状态 status	值	说明
STATUS_ERROR	0	执行报错
STATUS_CONTINUE	1	还未完成
STATUS_COMPLETION	2	完成

Info说明：

1）If status == STATUS_COMPLETION, Info见下表:

信息 Info	值	说明
APPROVE_OFFLINE	1	Transaction approved Offline
APPROVE_ONLINE	2	Transaction approved Online
DECLINE_OFFLINE	3	Transaction declined Offline
DECLINE_ONLINE	4	Transaction declined Online

2）If status == STATUS_ERROR, Info见下表:

信息 Info	值	说明
SUCCESS	0	成功
ERROR_APP_NO_INFO	1	Selected Application do not in the Candidate List when Application Select
ERROR_NO_APP	2	No Application Selected when Application Select
ERROR_APP_ANALYSIS	3	Parse Card Returned Data Error when Application Select

ERROR_APP_BLOCKED	4	card return 6A81 when Application Select
ERROR_APP_SELECT	5	Error when Application Select
ERROR_NO_AIPAFL	6	Application Interchange Profile(AIP) and Application File Locator(AFL) not exist when Initialize Application
ERROR_INIT_APP	7	Error when Initialize Application Data
ERROR_OTHER_CARD	8	
ERROR_EXPIRED_CARD	9	
ERROR_APP_DATA	10	Error when Read Application Data
ERROR_AUTH_METHOD_BLOCKED	11	card return 6983, command not allowed, authentication method blocked
ERROR_REFDATA_INVALIDATED	12	card return 6984, command not allowed, referenced data invalidated
ERROR_COND_NOT_SATISFIED	13	card return 6985, command not allowed, conditions of use not satisfied
ERROR_FUNC_NOT_SUPPORTED	14	card return 6A81, wrong parameter p1 p2, function not supported
ERROR_FILE_NOT_FOUND	15	card return 6A82, wrong parameter p1 p2, file not found
ERROR_RECORD_NOT_FOUND	16	card return 6A83, wrong parameter p1 p2, record not found
ERROR_REFDATA_NOT_FOUND	17	card return 6A88, referenced data (data objects) not found
ERROR_SELFILE_INVALIDATED	18	card return 6283, state of non-volatile memory unchanged, selected file invalidated
ERROR_AUTH_FAILED	19	card return 6300, state of non-volatile memory changed, authentication failed
ERROR_COUNTER_X	20	card return 63Cx, state of non-volatile memory changed, counter provided by 'x'(from 0-15)
ERROR_BLOCKED_NOSEL	21	card blocked or select command not supported
ERROR_ANALYSIS	22	Parse Card Returned Data Error
ERROR_READ_DATA	23	Error when Processing Restrict
ERROR_GEN_RANDOM	24	Card Returned Data for SDA Overflow when Read Application Data

ERROR_GEN_DOLBLOCK	25	Generate DOL Block error when Data Authentication
ERROR_GEN_DOLBLOCK	26	Generate AC error when Transaction Process
ERROR_NO_CDOL1	27	CDOL1 not exist when Transaction Process
ERROR_NO_CDOL2	28	CDOL2 not exist when Transaction Process
ERROR_LOGIC	29	Logic Error when Transaction Process
ERROR_CHIP_CANNOT_BE_READ	30	Card Returned Unknown Response Code
ERROR_PROCESS_CMD	31	Process Command ERROR
ERROR_AAR_ABORTED	32	Card decision is AAR when Transaction Process
ERROR_LOG_FILE	33	Log File Error
ERROR_SERVICE_NOT_ALLOWED	34	Service not Allowed
ERROR_PINENTERY_TIMEOUT	35	PIN Entry timeout
ERROR_OFFLINE_VERIFY	36	Check Offline PIN Error when Cardholder Verify
ERROR_NEED_ADVICE	37	Communication Error with Host, but the card need advice, halted the transaction
ERROR_USER_CANCELLED	38	

3）If status == STATUS_CONTINUE, Info，见下表:

信息 Info	值	说明
EMV_START	0	EMV Transaction Started
EMV_CANDIDATE_LIST	1	notify Application show Application Candidate List
EMV_APP_SELECTED	2	Application Select Completed
EMV_GET_PROC_OPTION	3	Get Process Option Completed
EMV_READ_APP_DATA	4	Read Application Data Completed
EMV_DATA_AUTH	5	Data Authentication Completed
EMV_PROCESS_RESTRICT	6	Process Restrict Completed
EMV_ONLINE_ENC_PIN	7	notify Application prompt Caldholder enter Online PIN
EMV_PIN_BYPASS_CONFIRM	8	notify Application confirm to Accepted PIN Bypass or not

EMV_CARDHOLDER_VERIFY	9	Cardholder Verify Completed
EMV_TERMINAL_RISK_MANAGEMENT	10	Terminal Risk Management Completed
EMV_PROCESS_ONLINE	11	notify Application to Process Online
EMV_ID_CHECK	12	notify Application Check Cardholder's Identification

A.3.2.3.3 其他功能

A.读取EMV Kernal版本

接口名称	emvGetVersionString
功能描述	读取EMV Kernal版本
接口格式	String emvGetVersionString()
参数说明	无
返回值	Kernal 版本数据

B.设置交易金额

接口名称	emvSetTransAmount
功能描述	设置交易金额
接口格式	void emvSetTransAmount(int amount)
参数说明	金额，以分为单位
返回值	无

C. 设置交易类型

接口名称	emvSetTransType
功能描述	设置交易类型
接口格式	void emvSetTransType(char transType)
参数说明	交易类型，见下表
返回值	无

交易类型说明：

交易类型 transType	值	说明
TRANS_GOODS_SERVICE	0x00	
TRANS_CASH	0x01	
TRANS_INQUIRY	0x04	
TRANS_TRANSFER	0x05	
TRANS_PAYMENT	0x06	
TRANS_ADMIN	0x07	

TRANS_CASHBACK	0x09	
TRANS_CARD_RECORD	0x0A	
TRANS_EC_BALANCE	0x0B	

D.设置其他金额

接口名称	emvSetOtherAmount
功能描述	设置其他金额
接口格式	void emvSetOtherAmount(int amount)
参数说明	金额，以分为单位
返回值	无

E.清除AID参数

接口名称	emvAidParamClear
功能描述	清除AID参数
接口格式	int emvAidParamClear()
参数说明	无
返回值	1. 返回 0，成功 2. 返回非 0，失败

F.增加AID参数

接口名称	emvAidParamClear
功能描述	增加AID参数
接口格式	int emvAidParamAdd(AIDPARAM aidParam)
参数说明	AIDPARAM 对象引用
返回值	3. 返回 0，成功 4. 返回非 0，失败

--AIDPARAM类结构定义：

字段名	类型	说明
aidLength	int	AID length
aid[16]	char	Application Identifier, [BCD]
appLable[16+1]	char	Application Label, [ASC]
appPreferredName[16+1]	char	Application Preferred Name, [ASC]
appPriority	char	Application Priority
termFloorLimit[4]	char	Terminal Floor Limit, [HEX]
termActionCodeDefault[5]	char	Terminal Action Code - Default, [BCD]

termActionCodeDenial[5]	char	Terminal Action Code - Denial, [BCD]
termActionCodeOnline[5]	char	Terminal Action Code - Online, [BCD]
targetPercentage	char	Target Percentage
thresholdValue[4]	char	threshold Value, [HEX]
maxTargetPercentage	char	Maximum Target Percentage
acquirer_id[6]	char	Acquirer Identifier, [BCD]
mechantCategoryCode[2]	char	Merchant Category Code, [BCD]
merchantID[15]	char	Merchant Identifier, [ASC]
appVersionNumber[2]	char	Application Version Number[BCD]
posEntryMode	char	Point-of-Service(POS) Entry Mode
transReferCurrencyCode	char	Transaction Reference Currency Code, [BCD]
transReferCurrencyExponent	Char	Transaction Reference Currency Exponent
defaultDDOLLength	Char	Default DDOL length
defaultDDOL[128]	Char	Default Dynamic Data Authentication Data Object List(DDOL), [BIN]
defaultTDOLLength	Char	Default TDOL length
defaultTDOL[128]	Char	Default Transaction Certificate Data Object List(TDOL) , [BIN]
supportOnlinePin	char	supportOnlinePin[0] = 0 means the Application unsupported online PIN, any other value means the Application supported online PIN

supportAidPartial	char	// supportAIDPartial[0] = 0 means the Application unsupported AID
		// Partial, any other value means the Application supported AID
		// Partial

contactlessLimit[4]	char	QPBOC Contactless Limit, [HEX]
contactlessFloorLimit[4]	char	QPBOC Contactless Floor Limit
cvmLimit[4]	char	QPBOC CVM Limit, [HEX]
ecTermTransLimit[6]	char	电子现金终端交易限额, [BCD]

G.清除CAPK参数

接口名称	emvCapkParamClear
功能描述	清除CAPK参数
接口格式	int emvCapkParamClear()

参数说明	无
返回值	5. 返回 0，成功 6. 返回非 0，失败

H.增加CAPK参数

接口名称	emvAidParamClear
功能描述	增加CAPK参数
接口格式	int emvCapkParamAdd(CAPKPARAM capkParam)
参数说明	CAPKPARAM 对象引用
返回值	7. 返回 0，成功 8. 返回非 0，失败

--CAPKPARAM类定义：

字段名	类型	说明
rid[5]	Char	Registered Application Provider Identifier, [BCD]
capki	char	Certificate Authority Public Key Index
hashInd	char	Hash Algorithm Indicator
arithInd	char	Certificate Authority Public Key Algorithm Indicator
modulLen	int	The Length of Certificate Authority Public Key Modulus
modul[248]	char	Certificate Authority Public Key Modulus
exponentlen	int	The Length of Certificate Authority Public Key Exponent
exponent[3]	char	Certificate Authority Public Key Exponent
checkSum[20	char	Certificate Authority Public Key Check Sum
expiry[8]	char	Certificate Expiration Date, [ASC] “YYYYMMDD”

I.设置EMV终端参数

接口名称	emvSetTerminalParam
功能描述	设置EMV终端参数
接口格式	int emvSetTerminalParam(TERMINAL_INFO terminalParam)
参数说明	TERMINAL_INFO 对象引用
返回值	1.返回 0，成功 2.返回非 0，失败

TERMINAL_INFO类定义：

字段名	类型	说明
terminalCountryCode[2]	Char	Terminal Country Code，9F1A [BCD]
tid[8]	char	Terminal ID，9F1C [ASC]
ifd[8]	char	IFD Serial Number，9F1E [ASC]
transactionCurrencyCode[2]	char	Transaction Currency Code，5F2A [BCD]
terminalCapabilities[3]	char	Terminal Capabilities ，9F33 [BIN]
terminal_type[1]	char	Terminal Type，[9F35 [BCD]
transactionCurrencyExponent[1]	char	Transaction Currency Exponent， 5F36 [BCD]
additionalTerminalCapabilities[5]	char	Additional Terminal Capabilities ，9F40 [BIN]
merchantNameLength	char	
merchantName[20]	char	Additional Terminal Capabilities ，9F4E [ASC]
statusCheckSupport	char	qpbc： 是否支持状态检查 0-不支持； 1-支持

J.清除卡片黑名单

接口名称	emvBlackcardClear
功能描述	清除卡片黑名单
接口格式	int emvBlackcardClear()
参数说明	无
返回值	1.返回 0，成功 2.返回非 0，失败

K.增加卡片黑名单

接口名称	emvBlackcardAdd
功能描述	增加卡片黑名单
接口格式	int emvBlackcardAdd(BLACKCARD blackcard)
参数说明	BLACKCARD 对象引用
返回值	1.返回 0，成功 2.返回非 0，失败

--BLACKCARD类定义：

字段名	类型	说明
cardNo[19]	Char	PAN

panSequence	char	PAN Sequence Number

L.清除证书回收名单

接口名称	emvBlackcertClear
功能描述	清除证书回收名单
接口格式	int emvBlackcertClear()
参数说明	无
返回值	1.返回 0，成功 2.返回非 0，失败

M.增加证书回收名单

接口名称	emvBlackcertAdd
功能描述	增加证书回收名单
接口格式	int emvBlackcertAdd(BLACKCERT blackcert)
参数说明	BLACKCERT 对象引用
返回值	1.返回 0，成功 2.返回非 0，失败

--BLACKCERT类定义：

字段名	类型	说明
rid[5]	Char	Registered Application Provider Identifier
capki	char	Certificate Authority Public Key Index

N.该EMV交易是否需要上送Advice

接口名称	emvIsNeedAdvice
功能描述	该EMV交易是否需要上送Advice
接口格式	int emvIsNeedAdvice()
参数说明	无
返回值	返回1, 需要上送Advice; 返回0,不需要上送Advice.

O.该EMV交易是否需要签名

接口名称	emvIsNeedSignature
功能描述	该EMV交易是否需要签名
接口格式	int emvIsNeedSignature()
参数说明	无

返回值	返回1, 需要签名; 返回0,不需要签名.

P.设置是否强制联机

接口名称	emvSetForceOnline
功能描述	设置是否强制联机
接口格式	void emvSetForceOnline(int flag)
参数说明	flag == 1 强制联机 == 0 不强制
返回值	无

Q.读取卡片交易记录

接口名称	emvGetCardRecord
功能描述	读取卡片交易记录
接口格式	int emvGetCardRecord(char[] record_no, char[] data)
参数说明	1. 记录条数：10 2. 存放记录 buffer 注：data 长度 = 45 * 10， 即每条记录 45 字节，最多 10 条记录
返回值	1.返回 0，成功 2.返回非 0，失败

R.获取应用选择列表

接口名称	emvGetCandidateList
功能描述	获取应用选择列表
接口格式	int emvGetCandidateList(char[] aidNumber, CANDIDATE aidList)
参数说明	1. AID number 2. CANDIDATE 对象引用
返回值	1.返回 0，成功 2.返回非 0，失败

--CANDIDATE类定义：

字段名	类型	说明
aid_length	char	Application Identifier
aid[16]	char	Certificate Authority Public Key Index
appLabel[16+1]	char	Application Label
appPreferredName[16+1]	char	Application Preferred Name

priorityExist	char	Application Priority exist flag
appPriority	char	Application Priority

S.设置应用选择结果

接口名称	emvSetCandidateListResult
功能描述	设置应用选择结果
接口格式	int emvSetCandidateListResult(char index);
参数说明	应用选择时的应用索引
返回值	1.返回 0，成功 2.返回非 0，失败

T.设置持卡人证件检查结果

根据IDType（9F62）、IDNumber(9F61)，设置证件检查结果

接口名称	emvSetIDCheckResult
功能描述	设置持卡人证件检查结果
接口格式	int emvSetIDCheckResult(int result)
参数说明	result == 1，证件和出示的一致 == 0，证件和出示的不符
返回值	1.返回 0，成功 2.返回非 0，失败

U.设置OnlinePIN是否输入

接口名称	emvSetOnlinePinEntered
功能描述	设置OnlinePIN是否输入
接口格式	int emvSetOnlinePinEntered(int result)
参数说明	result == 1，用户已输入PIN == 0，用户未输入PIN
返回值	1.返回 0，成功 2.返回非 0，失败

V.确认是否允许持卡人Bypass PIN（不输入PIN）

接口名称	emvSetPinBypassConfirmed
功能描述	确认是否允许持卡人Bypass PIN（不输入PIN）
接口格式	int emvSetPinBypassConfirmed(int result)
参数说明	result == 1，用户确认不输入脱机PIN == 0，用户不确认，即需要重新输入脱机PIN
返回值	1.返回 0，成功

	2.返回非 0，失败

W.设置联机认证结果

接口名称	emvSetOnlineSesult
功能描述	设置联机认证结果
接口格式	int emvSetOnlineSesult(char result,char[] is_suer_resp_data, char is_suer_resp_data_length)
参数说明	result:0，联机通讯成功，并收到交易成功应答 1，联机通讯失败 2，联机通讯成功，但收到交易拒绝应答（Response Code != "00"）
返回值	1.返回 0，成功 2.返回非 0，失败

A.3.2.4 外设访问权限控制模块

1、访问证书管理与加密运算模块权限
<uses-permission N3 TEE:name="N3 TEE.permission.CLOUDPOS_SAFE_MODULE" />
2、访问磁条卡读卡器设备权限
<uses-permission N3 TEE:name="N3 TEE.permission.CLOUDPOS_MSR" />
3、访问接触式IC卡阅读器权限
<uses-permission N3 TEE:name="N3 TEE.permission.CLOUDPOS_SMARTCARD" />
4、访问非接触IC卡读卡设备权限
<uses-permission N3 TEE:name="N3 TEE.permission.CLOUDPOS_CONTACTLESS_CARD" />
5、访问打印机设备权限
<uses-permission N3 TEE:name="N3 TEE.permission.CLOUDPOS_PRINTER" />
6、访问PIN输入设备权限
<uses-permission N3 TEE:name="N3 TEE.permission.CLOUDPOS_PINPAD" />
7、PIN输入设备计算Pinblock权限
<uses-permission N3 TEE:name="N3 TEE.permission.CLOUDPOS_PIN_GET_PIN_BLOCK" />
8、PIN输入设备计算Mac权限
<uses-permission N3 TEE:name="N3 TEE.permission.CLOUDPOS_PIN_MAC" />
9、PIN输入设备加密数据权限
<uses-permission N3 TEE:name="N3 TEE.permission.CLOUDPOS_PIN_ENCRYPT_DATA" />
10、PIN输入设备更新终端主密钥权限
<uses-permission N3 TEE:name="N3 TEE.permission.CLOUDPOS_PIN_UPDATE_MASTER_KEY" />
11、PIN输入设备更新用户密钥权限
<uses-permission N3 TEE:name="N3 TEE.permission.CLOUDPOS_PIN_UPDATE_USER_KEY" />

A.3.3 多应用管理要求

在基于 N3 TEE 操作系统定制智能系统中，银联已发布了银联应用商店客户端（即上文所述的多应用管理客户端）、银联应用商店平台（即上文所述的多应用管理平台）。基于 N3 TEE 操作系统定制的智能销售点终端应能正常运行银联应用商店客户端。

A.3.4 银行卡支付服务及客户端要求

A.3.4.1 银行卡支付调用接口

A.3.4.1.1 调用交易接口

该接口定义为调用收单应用，完成包括磁条卡和PBOC交易。

接口名称	payCash			
功能描述	通过调用该接口，跳转到银联交易支付模块进行交易			
接口格式	String payCash(String jsonData);			
接口权限	com.cloudpos.CloudPosPaymentClient.permission.CLOUDPAY			
参数说明	jsonData 为 JSON 数据格式，属性说明参照			
	AppID	String	必选	第三方应用的唯一标识，由银联指定。
	AppName	String	必选	第三方应用名称，由第三方应用自行定义。
	TransType	String	必选	交易类型可参考本文 4.2 章的详述。
	TransAmount	String	必选	交易金额的币种均为人民币,保留两位小数，示例：消费金额为100元，则交易金额应为00000010000。
	TransIndexCode	String	必选	由第三方应用提供，此值用来标识一次交易请求，一天内不能重复。
	ReqTransDate	String	必选	第三方交易发起请求的日期
	ReqTransTime	String	必选	第三方交易发起请求的具体时间。
	cardInfo2	String	必选	磁条卡第二磁道信息
	cardInfo3	String	必选	磁条卡第三磁道信息
返回值	String类型，JSON数据格式，属性说明参照文档附录,消费返回参数集			
	AppID	String	必选	第三方应用的唯一标识，由银联指定。
	AppName	String	必选	第三方应用名称，由第三方应用自行定义。
	TransType	String	必选	返回的交易类型同请求的交易类型，中文显示。
	TransAmount	String	必选	交易金额的币种均为人民币,保留两位小数，示例：消费金额为100元，则交易金额应为00000010000。
	TransIndexCode	String	必选	此值同第三方请求的交易索引号，返回值和请求值相同。
	ReqTransDate	String	必选	第三方交易发起请求的日期
	ReqTransTime	String	必选	第三方交易发起请求的时间
	TransDate	String	必选	支付系统的处理交易成功的日期。
	TransTime	String	必选	支付系统的处理交易成功的时间。
	RespCode	String	必选	支付系统返回给第三方应用的应答码
	RespDesc	String	必选	支付系统返回给第三方应用的应答码的对应中文解释，请参照所述。

	OptCode	String	可选	操作员号只可能是 01-98，普通操作员登陆成功签到后才能使用第三方应用，请参照
	CardNum	String	可选	前 5 位数字和末 4 位数字显示，其他位数显示为*。
	BatchNum	String	可选	交易中心返回的 6 位批次号，每批结算一次批次号加一。
	CertNum	String	可选	6 位凭证号，相当于交易流水号，每一批次凭证号从 000000 开始递增。
	ReferCode	String	可选	12 位交易参考号是交易中心返回的随机编码，不会重复。
	Reference	String	可选	交易中心返回，可能为空值。
	FeeAmount	String	可选	备用

A.3.4.1.2 调用查询接口

接口名称	getPayInfo			
功能描述	通过调用该接口进行交易信息的查询			
接口格式	String getPayInfo(String jsonData)			
接口权限	com.cloudpos.CloudPosPaymentClient.permission.CLOUDPAY			
参数说明	jsonData 为 JSON 数据格式，属性说明参照			
	AppId	String	必选	第三方应用的唯一标识，由银联指定。
	AppName	String	必选	第三方应用名称，由第三方应用自行定义。
	TransType	String	必选	交易类型可参考章的详述。
	TransIndexCode	String	必选	此值填写交易发起时的索引号，用来查询一次交易请求，和交易日期配合使用。
	ReqTransDate	String	必选	第三方交易发起请求的日期，此值限制交易索引。
返回值	String类型，JSON数据格式，属性说明参照文档附录,查询返回参数集			
	AppId	String	必选	第三方应用的唯一标识，由银联指定。
	AppName	String	必选	第三方应用名称，由第三方应用自行定义。
	TransType	String	必选	返回的交易类型同请求的交易类型，中文显示。
	TransIndexCode	String	必选	此值同第三方请求的交易索引号，返回值和请求值相同
	ReqTransDate	String	必选	第三方交易发起请求的日期
	ReqTransTime	String	必选	第三方交易发起请求的时间
	TransDate	String	必选	支付系统的处理交易成功的日期。
	TransTime	String	必选	支付系统的处理交易成功的时间。
	RespCode	String	必选	支付系统返回给第三方应用的应答码，请

				参照本文 A.1.3.1.8 章节所述。
	RespDesc	String	必选	支付系统返回给第三方应用的应答码的对应中文解释，请参照本文 A.1.3.1.8 章节所述。
	TransAmount	String	可选	查询的交易请求的金额，保留两位小数，示例：消费金额为 100 元，则交易金额应为 00000010000。

A.3.4.1.3 终端信息查询接口

接口名称	getPOSInfo			
功能描述	通过调用该接口查询终端信息			
接口格式	String getPOSInfo(String jsonData)			
接口权限	com.cloudpos.CloudPosPaymentClient.permission.CLOUDPAY			
参数说明	jsonData 为 JSON 数据格式，属性说明参照			
	AppID	String	必选	第三方应用的唯一标识，由银联指定。
	AppName	String	必选	第三方应用名称，由第三方应用自行定义。
	TransType	String	必选	交易类型可参考本文 4.2 章的详述。
	ReqTransDate	String	必选	第三方发起请求的日期。
返回值	String类型，JSON数据格式，属性说明参照文档附录,查询返回参数集			
	MerchantID	String	必选	商户 ID，唯一标志商户
	TerminaID	String	必选	终端 ID，唯一标志终端
	OperatorID	String	必选	操作员 ID，唯一标志操作员
	RespCode	String	必选	支付系统返回给第三方应用的应答码的对应中文解释，请参照本文 A.1.3.1.8 章节所述。
	RespDesc	String	必选	支付系统返回给第三方应用的应答码的对应中文解释，请参照本文 A.1.3.1.8 章节所述。

A.3.4.1.4 参数解释

注：这里的请求是指第三方收银应用向银联支付应用发送请求报文； 应答是指银联支付应用向第三方收银应用发送响应报文。

第三方应用与支付之间的交换消息中，各数据元类型如下所列：

数据元类型	说明
A	字母
AN	字母和/或数字
ANS	字母、数字和/或特殊符号
AS	字母和/或特殊符号

N	数字
YY	年
MM	月
DD	日
hh	时
mm	分
ss	秒

注：可选的意思是有值返回，第三方可以根据需求选择是否获取，此类参数非必须参数。

A.3.4.1.5 消费

表8 消费请求参数列表

字段名	变量名	是否必填	类型(长度)	说明
应用 ID	AppId	是	N	第三方应用的唯一标识，由银联指定。
应用名称	AppName	是	AN 或中文	第三方应用名称，由第三方应用自行定义。
交易类型	TransType	是	N(2)	交易类型可参考 A.1.3.1.8 章节的详述。
交易金额	TransAmount	是	N(12)	交易金额的币种均为人民币,保留两位小数，示例：消费金额为 100 元，则交易金额应为 00000010000。
交易索引号	TransIndexCode	是	N(最大 16)	由第三方应用提供，此值用来标识一次交易请求，一天内不能重复。
交易请求日期	ReqTransDate	是	YYMMDD(6)	第三方交易发起请求的日期
交易请求时间	ReqTransTime	是	Hhmmss(6)	第三方交易发起请求的具体时间。
磁道 2 信息	cardInfo2	可选	NS	磁条卡第二磁道信息
磁道 3 信息	cardInfo3	可选	NS	磁条卡第三磁道信息

表9 消费返回通知参数列表

字段名	变量名	是否返回	类型	说明
应用 ID	AppId	是	N	第三方应用的唯一标识，由银联指定。
应用名称	AppName	是	AN 或中文	第三方应用名称，由第三方应用自行定义。
交易类型	TransType	是	N(2)	返回的交易类型同请求的交易类型，中文显示。
交易金额	TransAmount	是	N(12)	交易金额的币种均为人民币,保留两位小数，示例：

				消费金额为 100 元，则交易金额应为 00000010000。
交易索引号	TransIndexCode	是	N(最大 16)	此值同第三方请求的交易索引号，返回值和请求值相同。
交易请求日期	ReqTransDate	是	YYMMDD(6)	第三方交易发起请求的日期
交易请求时间	ReqTransTime	是	Hhmmss(6)	第三方交易发起请求的时间
交易应答日期	TransDate	是	MMDD(4)	支付系统的处理交易成功的日期。
交易应答时间	TransTime	是	Hhmmss(6)	支付系统的处理交易成功的时间。
返回码	RespCode	是	AS(最大 6)	支付系统返回给第三方应用的应答码
返回码中文解释	RespDesc	是	中文	支付系统返回给第三方应用的应答码的对应中文解释，请参照所述。
操作员号	OptCode	可选	N(2)	操作员号只可能是 01-98，普通操作员登陆成功签到后才能使用第三方应用，请参照
卡号	CardNum	可选	N(最大 19)	前 5 位数字和末 4 位数字显示，其他位数显示为*。
批次号	BatchNum	可选	N(6)	交易中心返回的 6 位批次号，每批结算一次批次号加一。
凭证号	CertNum	可选	N(6)	6 位凭证号，相当于交易流水号，每一批次凭证号从 000000 开始递增。
参考号	ReferCode	可选	N(12)	12 位交易参考号是交易中心返回的随机编码，不会重复。
备注信息	Reference	可选	ANS	交易中心返回，可能为空值。
小费金额	FeeAmount	可选		备用

A.3.4.1.6 查询

表10 查询参数列表

字段名	变量名	是否必填	类型(长度)	说明
应用 ID	AppId	是	N	第三方应用的唯一标识，由银联指定。
应用名称	AppName	是	AN 或中文	第三方应用名称，由第三方应用自行定义。
交易类型	TransType	是	N(2)	交易类型可参考章的详述。

交易索引号	TransIndexCode	是	N(最大 16)	此值填写交易发起时的索引号，用来查询一次交易请求，和交易日期配合使用。
交易请求日期	ReqTransDate	是	YYMMDD(6)	第三方交易发起请求的日期，此值限制交易索引。

表11　查询返回参数列表

字段名	变量名	是否必填	类型	说明
应用 ID	AppId	是	N	第三方应用的唯一标识，由银联指定。
应用名称	AppName	是	AN 或中文	第三方应用名称，由第三方应用自行定义。
交易类型	TransType	是	N(2)	返回的交易类型同请求的交易类型，中文显示。
交易索引号	TransIndexCode	是		此值同第三方请求的交易索引号，返回值和请求值相同
交易请求日期	ReqTransDate	是	YYMMDD(6)	第三方交易发起请求的日期
交易请求时间	ReqTransTime	是	Hhmmss(6)	第三方交易发起请求的时间
交易应答日期	TransDate	是	YYMMDD(6)	支付系统的处理交易成功的日期。
交易应答时间	TransTime	是	Hhmmss(6)	支付系统的处理交易成功的时间。
返回码	RespCode	是	AS(最大 6)	支付系统返回给第三方应用的应答码，请参照本文 4.3 章所述。
返回码中文解释	RespDesc	是	中文	支付系统返回给第三方应用的应答码的对应中文解释，请参照本文 4.3 章所述。
交易金额	TransAmount	可选	N(12)	查询的交易请求的金额，保留两位小数，示例：消费金额为 100 元，则交易金额应为 00000010000。

A.3.4.1.7　POS信息查询

表12　POS 信息查询请求参数列表

字段名	变量名	是否必填	类型(长度)	说明
应用 ID	AppID	是	N	第三方应用的唯一标识，由银联指定。
应用名称	AppName	是	AN 或中文	第三方应用名称，由第三方应用自行定义。
交易类型	TransType	是	N(2)	交易类型可参考本文 A.1.3.1.8 章节的详述。
交易请求日期	ReqTransDate	是	YYMMDD(6)	第三方发起请求的日期。

表13　POS 信息查询返回通知参数列表

字段名	变量名	是否返回	类型(长度)	说明
商户号	MerchantID	是	N	商户 ID，唯一标志商户
终端号	TerminaID	是	N	终端 ID，唯一标志终端
操作员号	OperatorID	是	N	操作员 ID，唯一标志操作员
返回码	RespCode	是	AS(最大 6)	支付系统返回给第三方应用的应答码的对应中文解释，请参照本文 A.1.3.1.8 章节所述。
返回码中文解释	RespDesc	是	中文	支付系统返回给第三方应用的应答码的对应中文解释，请参照本文 4.3 章所述。

A.3.4.1.8　交易类型

接口类型参数值	描述
1	消费
2	查询消费信息
3	查询 POS 信息

A.3.4.1.9　返回应答码表

交易返回 POS 终端时都有应答码，可以把操作分为以下几类：

A：交易成功

B：交易失败，可重试

C：交易失败，不需要重试

D：交易失败，终端操作员处理

E：交易失败，系统故障，不需要重试

F：交易失败，支付客户端未处理

1.1：如果返回应答码不能在下表中找到，中文解释显示“交易失败”

1.2：如果POS交易的批次号和网络中心批次号不一致时应答码会填 “77”，此时POS机应当提示操作员重新签到，再作交易。

代码	意义	类别	原因/采取的措施	POS 显示的内容
00	承兑或交易成功	A	承兑或交易成功	交易成功
01	查发卡行	C	查发卡行	请持卡人与发卡银行联系
03	无效商户	C	商户需要在银行或中心登记	无效商户
04	没收卡	D	操作员没收卡	此卡被没收
05	身份认证失败	C	发卡不予承兑，预约信息匹配失败	持卡人认证失败
10	部分承兑	A	部分金额批准，请收取余额	显示部分批准金额，提示操作员

代码	意义	类别	原因/采取的措施	POS 显示的内容
11	重要人物批准（VIP）	A	此为 VIP 客户	成功，VIP 客户
12	无效的关联交易	C	发卡行不支持的交易	无效交易
13	无效金额	B	金额为 0 或其他非法值	无效金额
14	无效卡号（无此账号）	B	卡种未在中心登记或读卡号有误	无效卡号
15	无此发卡方	C	此发卡行未与中心开通业务	此卡无对应发卡方
21	卡未初始化	C	1、该卡未激活、开卡； 2、该卡初始密码未变更； 3、初始密码限制的交易； 4、长期未使用而冻结或状态为“睡眠”的卡。	该卡未初始化或睡眠卡
22	故障怀疑，关联交易错误	C	POS 状态与中心不符，可重新签到	操作有误，或超出交易允许天数
25	找不到原始交易	C	发卡行未能找到有关记录	没有原始交易，请联系发卡方
30	报文格式错误	C	格式错误（不符合磁道预校验规则）	请重试
34	有作弊嫌疑	D	有作弊嫌疑的卡，操作员可以没收	作弊卡,吞卡
38	超过允许的 PIN 试输入	D	密码错次数超限，操作员可以没收	密码错误次数超限，请与发卡方联系
40	请求的功能尚不支持	C	发卡行不支持的交易类型	发卡方不支持的交易类型
41	挂失卡	D	挂失的卡， 操作员可以没收	挂失卡，请没收（POS）
43	被窃卡	D	被窃卡， 操作员可以没收	被窃卡，请没收
51	资金不足	C	账户内余额不足	可用余额不足
54	过期的卡	C	过期的卡	该卡已过期
55	不正确的 PIN	C	密码输错	密码错
57	不允许持卡人进行的交易	C	不允许持卡人进行的交易	不允许此卡交易
58	不允许终端进行的交易	C	该商户不允许进行的交易	发卡方不允许该卡在本终端进行此交易
59	有作弊嫌疑	C	CVN 验证失败	卡片校验错
61	超出金额限制	C	一次交易的金额太大	交易金额超限
62	受限制的卡	C		受限制的卡
64	原始金额错误	C	原始金额不正确	交易金额与原交易不匹配
65	超出消费次数限制	C	超出消费次数限制	超出消费次数限制
68	发卡行响应超时	C	发卡行规定时间内没有回答	交易超时，请重试

代码	意义	类别	原因/采取的措施	POS 显示的内容
75	允许的输入 PIN 次数超限	C	允许的输入 PIN 次数超限	密码错误次数超限
90	正在日终处理	C	日期切换正在处理	系统日切，请稍后重试
91	发卡方不能操作	C	电话查询发卡方或银联，可重作	发卡方状态不正常，请稍后重试
92	金融机构或中间网络设施找不到或无法达到	C	电话查询发卡方或网络中心，可重作	发卡方线路异常，请稍后重试
94	重复交易	C	查询网络中心，可能是一笔已经成功上送的交易	拒绝，重复交易，请稍后重试
96	银联处理中心系统异常、失效	C	发卡方或网络中心出现故障	拒绝，交换中心异常，请稍后重试
97	POS 终端号找不到	D	终端未在中心或银行登记	终端未登记
98	银联处理中心收不到发卡方应答	E	银联收不到发卡行应答	发卡方超时
99	PIN 格式错	B	可重新签到作交易	PIN 格式错，请重新签到
A0	MAC 鉴别失败	B	可重新签到作交易	MAC 校验错，请重新签到
A1	转账货币不一致	C	转账货币不一致	转账货币不一致
A2	有缺陷的成功	A	银联处理中心转发了原充值交易请求，但未收到发卡方应答时，银联处理中心直接向受理方应答为有缺陷的成功交易	交易成功，请向发卡行确认
A3	资金到账行无此账户	C	资金到账行账号不正确	账户不正确
A4	有缺陷的成功	A	未收到原充值交易请求时，对关联的确认交易的承兑为有缺陷的成功交易	交易成功，请向发卡行确认
A5	有缺陷的成功	A	原充值交易为拒绝时，对关联的确认交易的承兑为有缺陷的成功交易	交易成功，请向发卡行确认
A6	有缺陷的成功	A	银联处理中心转发了原充值交易请求，但未收到发卡方应答时，对受理方发来的关联的确认交易的承兑为有缺陷的成功交易	交易成功，请向发卡行确认
A7	安全处理失败	C	1、调用 MAC 校验程序失败 2、调用 PIN 校验程序失败 3、MAC 处理失败 4、密钥处理失败	拒绝，交换中心异常，请稍后重试
100	上一笔交易未完成	F	等待上一笔交易完成后重试	上一笔交易未完成

代码	意义	类别	原因/采取的措施	POS 显示的内容
101	请求报文数据格式错误	F	重新组装正确的报文	请求报文数据格式错误
102	支付调用取消	F	第三方应用收到支付响应报文之前，支付界面关闭	支付调用取消
103	支付应用未启动或未登录	F	启动支付客户端登陆后重试	支付应用未启动或未登录
104	未找到交易类型	F	支付后台不支持的交易类型	未找到交易类型
105	未知错误，交易失败	F	未知原因引起的错误	未知错误，交易失败
RS	冲正成功	B	交易超时无应答，冲正成功	冲正成功
RF	冲正失败	B	交易超时无应答，冲正无应答	冲正失败

A.3.4.2 银行卡支付客户端

在基于N3TEE系统中，银联已发布了银联支付客户端。基于N3TEE系统定制的智能销售点终端应能正常运行银联支付客户端。

附 录 B
（资料性附录）
补充说明

B.1 银联支付服务层

银联支付服务层是由银联开发的。在终端布放时，终端厂商将银联支付服务层安装于终端智能操作系统之上。银联支付服务层包括磁条卡消费服务等。

B.2 银联支付客户端

银联支付客户端是由银联开发的。在终端布放时，终端管理员可通过银联应用商店客户端下载安装银联支付客户端。银联支付客户端包括银联传统POS终端的所有支付功能，并提供一套标准的、图形化的线下收单操作界面。

B.3 银联支付平台

银联支付平台是银联收单平台的前置平台，由银联开发和运维，仅为银联支付客户端提供安全的支付服务接口。该支付服务接口采用HTTPS POST协议，JSON报文格式，包括消费、消费撤销、预授权、余额查询等所有传统POS支付接口。

B.4 银联应用商店客户端

银联应用商店客户端是由银联开发的。在终端布放前，终端厂商应从银联获取银联应用客户端最新版本，并安装在终端智能操作系统之上。

B.5 银联应用商店平台

银联应用商店平台是由银联开发和运维，负责智能销售点终端应用整个生命周期的管理，面向的使用对象包括：银联、机构、开发者。开发者是应用的提供者和上传者，负责应用的上传和维护。开发者角色可以是具体开发应用的应用开发单位或个人，也可以是机构或银联委托的应用管理人员。机构是终端的拥有方，负责应用的业务审核，终端分组和应用在终端分组的上架等。机构角色可以是银行、第三方专业化服务机构、银联分公司等。