ICS 35.040
L 80
备案号：

中华人民共和国密码行业标准

GM/T XXXX—XXXX

多应用智能IC卡安全中间件技术规范

Multi Application Smart Card Security Midware Technology Specification

（征求意见稿）

在提交反馈意见时，请将您知道的相关专利连同支持性文件一并附上。

××××-××-××发布　　××××-××-××实施

国家密码管理局 发布

目　次

前　　言

本规范按照GB/T 1.1-2009给出的规则起草。

本规范提供了一个通用的安全和卡片管理架构，旨在保护IC卡系统投资最重要的方面——基础架构。

本规范为首次制订。

本规范由密码行业标准化技术委员会提出并归口。

本规范起草单位：北京宏基恒信科技有限公司、北京江南天安科技有限公司、上海格尔软件股份有限公司。

本规范主要起草人：李东风、韩小军、王晓英、文楠和谭武征等人。

引　　言

本规范是一个跨行业的智能卡全局基础架构及其实现，其目标是为了减少隐藏在快速增长的跨行业、多应用的智能卡背后的障碍，使得发卡商在各种各样的卡片、终端和后台系统前，继续享有选择的自由。

本规范使用起来很灵活，使得发卡方能够创建运行单个应用或者多应用的IC卡系统来满足其不断演进的商业需求。发卡方既可以选择适合当前情况的卡片技术，也可以在将来需要的时候，迁移另一种卡片技术，而不用对其基础架构做出重大的修改。

多应用智能IC卡安全中间件技术规范

1 范围

本规范定义了存在于多应用智能 IC 卡中，用来实现智能 IC 卡片多应用机制的嵌入式软件，即安全中间件的命令接口、事务序列以及通行于诸多不同行业的接口。

本规范适用于指导多应用智能 IC 卡安全中间件的设计、开发和使用。

2 规范性引用文件

下列文件对于本部分的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本部分。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。

ISO/IEC 7816-4：2005 识别卡-接触 IC 卡第 4 部分：组织、安全和命令交换（2005）（Identification cards - Integrated circuit cards - Part 4: Organization, security and commands for interchange）

3 术语和定义

以下术语和定义适用于本规范。

3.1

应用 application

在卡片上安装后处于可选择状态的可执行模块的实例。

3.2

应用协议数据单元 application protocol data unit

读卡器和智能卡之间的标准通信消息协议。

3.3

应用会话 application session

应用与卡外世界的通过逻辑通道建立的某种联系，开始于该应用被选择，结束于另外的应用经由同一逻辑通道被选择，或者该逻辑通道被关闭，或者卡片会话被中止。

3.4

发卡方 card issuer

拥有卡片并对卡片的行为负有最终责任的实体。

3.5

可执行文件 executable file

实际存在于卡片上的包含一个或多个应用的可执行代码(可执行模块)的容器。

3.6

可执行模块 executable module

可执行加载文件中包含一个单独应用的可执行代码。

3.7

卡片注册表 card registry

一个包含卡片内容管理相关信息的容器。

3.8

生命周期 life cycle

卡上卡片内容的存在以及不同阶段，也可以指卡片本身存在的各个阶段。

3.9

生命周期状态 life cycle state

卡片或卡片内容在生命周期中的某个特定状态。

3.10

卡片运行时环境 card runtime environment

卡片运行期间的核心功能，为卡上的多个应用提供了一个安全的运行环境。

3.11

安全通道 secure channel

为卡外实体和卡片之间的信息交换提供某种安全保障的通信机制。

3.12

卡片安全域 card security field

负责对某个卡外实体(例如发卡方、应用提供方、授权管理者)的管理、安全、通信需求进行支持的卡内实体。

3.13

Native 应用 Native application

一种执行代码为机器指令的应用。

3.14

Bytecode 应用 Bytecode application

一种执行代码为字节码的应用。

4 缩略语

以下缩略语适用于本规范。

CMAP：卡多应用平台(Card Muti Application Platform)
CCORE：卡片内核模型(Card Core)
CCOM：卡片通讯模型(Card Communication)
CSTO：卡片存储模型(Card Storage)
CFW：卡片框架(Card Frame Work)
CRE：卡片运行时环境(Card Runtime Environment)
CDF：卡片数据域(Card Data Field)
CSF：卡片安全域(Card Security Field)
CMF：卡片方法域(Card Method Field)
CREG：卡片注册表(Card Registry)
CAM：卡片应用管理器(Card Application Manager)
HAL：硬件抽象层(Hardware Abstraction Layer)
CEXP：卡片异常(Card Exception)
APDU：应用协议数据单元(Application Protocol Data Unit)
SW：状态字(Status Word)
GLB：T=0 协议下应用调用的获取剩余字节的方法(Get left Bytes)
SMI：静态存储接口(Static Memory Interface)
POP：掉电保护机制(Power Off Protection)
CGS：卡片全局服务(Card Global Service)

CSS：卡片系统服务(Card System Service)
CSFW：卡片安全框架(Card Secure Frame Work)
CSMK：卡片安全管理密钥(Card Secure Management Key)
CLS：卡片生命状态(Card Life State)
CSSP：卡片安全会话包(Card Secure Session Package)
AFW：应用框架(Application Frame Work)
ASFW：应用安全框架(Application Secure Frame Work)
AID：应用标识符(Application Identifier)
CSSI：卡片系统服务标识符(Card System Service Identifier)
CASDD：卡片应用静态数据目录(Card Application Static Data Directory)
CAEMD：卡片应用可执行模块目录(Card Application Execute Module Directory)
CDPD：卡片数据包目录(Card Data Package Directory)
ADFQI：应用数据域引用标识(Application Data Field Quote Identifier)
ASFQI：应用安全域引用标识(Application Secure Field Quote Identifier)
AMFQI：应用方法域引用标识(Application Mathod Field Quote Identifier)
CDFQI：卡片数据域引用标识(Card Data Field Quote Identifier)
CSFQI：卡片安全域引用标识(Card Secure Field Quote Identifier)
CRCD：CRE 运行控制数据(CRE Run Control Data)
AI：应用实例(Application Instance)
AKSS：应用内核共享空间(Application Kernel Shared Space)
AII：应用身份信息(Application Identity Information)
ADDS：应用动态数据空间(Application Dynamic Data Space)
ADD：应用动态数据(Application Dynamic Data)
ASDS：应用静态数据空间(Application Static Data Space)
ASD：应用静态数据(应用静态数据)
ACFP：应用 COS 文件包(Application COS File Package)
ACFS：应用 COS 文件系统(Application COS File System)
API：应用编程接口(Application Programming Interface)
CVM：卡片虚拟机(Card Virtual Machine)
AEM：应用可执行模块(Application Executable Module)
CKSL：内核服务列表(Card Kernel Sevices List)
AA：应用权限(Application Authority)
FID：文件标识符(File Identifier)
ASCC：应用安全控制上下文(Application Secure Control Context)
MAC：报文鉴别码(Message Authentication Code)

5　卡片架构

5.1　概述

本规范为开发者和用户提供了一套硬件中立的卡片架构。开发者无需关注硬件细节，即可在卡片上进行应用开发。本规范将卡片通讯模型(CCOM)、卡片运行时环境(CRE)和卡片存储模型(CSTO)进行整合之后，向开发者和用户提供硬件中立的卡片应用管理器，API 和卡片的域(CDF,CMF,CSF)。

注：卡片通讯模型、卡片运行时环境、卡片存储模型、卡片应用管理器和卡片的域的定义见本章的后续章节。

卡片架构如图 1 所示。

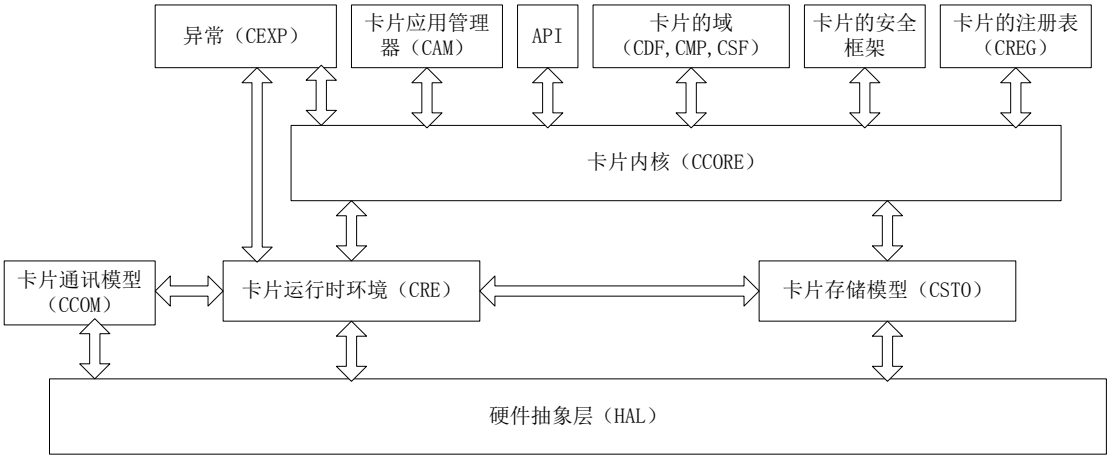

图1 卡片框架

5.2 卡片内核模型（CCORE）

卡片内核模型支持 Native 应用和 Bytecode 应用。

CCORE 有以下功能：

a） 提供 API 给应用进行调用；

b） 提供卡片应用管理器（CAM）中间件来管理多应用；

c） 提供卡片的域，要保证即使在没有任何应用的情况下，卡片也是可以被操作的，而且是安全的；

d） 能够捕捉和处理应用所有的异常，要保证卡片可以持续工作；

本规范对 CCORE 的形态不做强制性的规定，这将由卡商根据应用模型和客户需求，对 CCORE 进行灵活定制。

5.3 卡片通讯模型（CCOM）

CCOM 负责将发送给卡片的数据都封装成标准的 APDU 模型传给其他模块，并能够自动处理 GET RESPONSE 指令，应用将不负责处理该指令。

注：GET RESPONSE 指令详见 ISO/IEC 7816-4：2005 中的相关部分。

5.4 卡片存储模型（CSTO）

本规范规定，卡片存储模型 CSTO 要提供给应用动态存储和静态存储两种接口。其中，动态存储的数据断电后可以丢失。而静态存储数据在断电后必须保留，并且静态存储空间拥有掉电保护机制。

5.5 卡片运行时环境模型（CRE）

卡片运行时环境负责向所有应用提供一套硬件独立应用编程接口，一种能确保各个应用的代码和数据能相互隔离、安全地存储和执行空间分配机制，并提供服务来完成卡片和卡外实体之间的通信。

其主要功能包括：

a） 卡片安全框架的建立；

b） APDU 命令的预处理和分发；

c） 负责应用的管理，包括创建、装载和删除；

d） 负责卡片的生命周期管理；

e） 权限管理；

f） 异常管理。

注：卡片安全框架、卡片生命周期管理、权限管理和异常管理等的详细内容见本规范的第 9 章节。

5.6 卡片的域

本规范规定，卡片的域由数据域（CDF）、安全域(CSF)和方法域(CMF)组成。在一张卡片中，由 CMF 完成整个卡片的运行控制。在一个 APDU 会话建立后，CMF 在 CSF 的控制下，安全访问 CDF 资源。

卡片的方法域在卡片发卡之前即写入卡片中。卡片的数据域和安全域则在 CRE 启动的时候由 CRE 完成

初始化。

5.7 卡片的注册表（CREG）

本规范规定，卡片注册表（CREG）为卡片的可信任组件，访问权限为System。

注：权限管理详见本规范第9.7章节。

卡片注册表保存卡片框架和若干个应用框架的记录，但本规范对记录的设计不做强制规定，卡商可以根据实际情况自行实现上述记录。

注：卡片框架和应用框架详见本规范第10章节。

5.8 卡片应用管理器（CAM）

本规范规定卡片应用管理器CAM负责注册应用、加载应用和运行应用。可以看作是卡片运行时环境和卡片域的实体。

5.9 卡片虚拟机（CVM）

本规范并未对卡片的应用类型做限制性规定，一般情况下，如果卡商去定义一个纯Nativecode应用类型的卡片的时候，CVM是不需要的。但是对于Bytecode应用类型的卡片，或者Bytecode与Nativecode混合类型应用的卡片，CVM是必要的。

CVM担负的责任：装载和运行应用可执行模块，处理应用与CAM之间的IO接口。

本规范不提供任何CVM样本和实现的方法，卡商根据自己需要开发符合本规范的CVM并且移植到CAM中去。

6 卡片内核模型（CCORE）

CCORE通过对CCOM的抽象提取输入数据，通过对CRE的访问抽象应用数据IO模型，并通过对CSTO的访问完成卡片的存储访问操作。最后，将CRE中的IO模型，CSTO中的存储模型抽象并整合成API提供给应用进行调用。

CCORE应该提供卡片应用管理器CAM中间件来管理多应用。对于Native应用的管理，主要是对其数据对象的管理。但无论何种应用模型，CAM都将基于CCORE模型对其进行管理，并且安全地加载到卡片上。

卡片的域，保证了即使在没有任何应用的情况下，卡片也是可以被操作的，而且是安全的。卡片的域提供了对卡片应用的数据进行访问和生命周期控制的所有方法。

除了Native应用，卡片应该处理应用所有的异常，并返回给用户。发生异常的应用将会被CCORE强行终止，而卡片依然可以继续工作。

所有这一切都是CCORE考虑的内容，除此之外，本规范对CCORE的形态不做强制性规定，这将由卡商根据应用模型和客户需求，对CCORE进行灵活定制。

7 卡片通讯模型（CCOM）

本规范要求CCOM必须将进入卡片的数据,无论是T=0还是T=1和T=CL接口的数据都封为标准的APDU模型。在T=1和T=CL模型下，所有数据被封装为完整的APDU模型。T=0模型有所不同，APDU模型将先被填充报头，而后由应用自动获取剩余字节完成填充。APDU模型的定义如表1说明。

表1 APDU模型定义

字段标识	字段长度	字段描述
Apdu.Len	1Bytes	Apdu模型的总长度
Apdu.Header.Cla	1Bytes	Apdu字段的CLA参数
Apdu.Header.INS	1Bytes	Apdu字段的INS参数
Apdu.Header.P1	1Bytes	Apdu字段的P1参数
Apdu.Header.P2	1Bytes	Apdu字段的P2参数
Apdu.Header.LC	1Bytes	Apdu字段的LC参数
Apdu.Data	Apdu.Header.LC 中指定的字节	Apdu字段的数据

APDU模型的生成如图2所示：

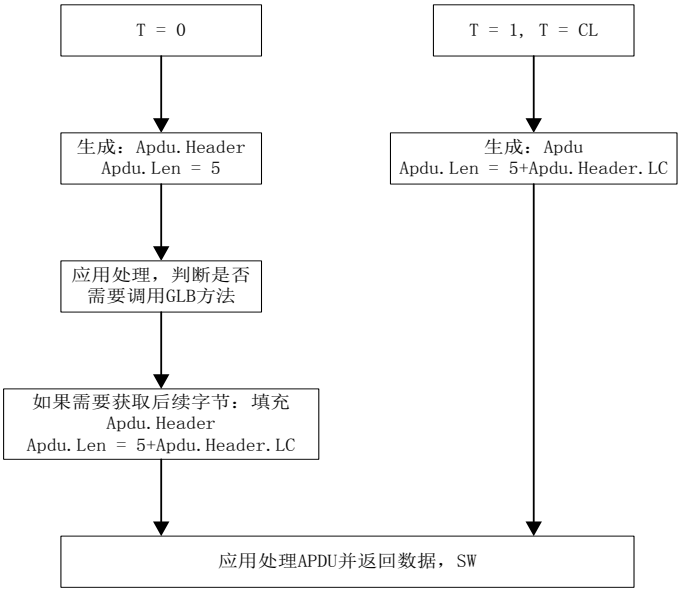

图2 APDU模型生成的流程图

其中，GLB 方法介绍见表 2 所示：

表2 GLB 的方法

函数	功能描述
receive_left_bytes	T=0 协议下，应用获取剩余字节

本规范要求 CCOM 能够自动处理 GET RESPONSE 指令，应用将不负责处理该指令。在 T=0 协议下，如果应用已经调用过 GLB 方法，那么如果用户有数据要返回，则 CCOM 自动返回 61XX(XX 为要返回的字节数)，并处理后续接收到的 C0 指令。

在 CASE2 情况的指令中，如果用户返回的数据字节数与 LE 不符合，那么本规范要求 CCOM 能够自动返回 6CXX(XX 为要返回的字节数)，并处理后续接收到的 GET RESPONSE 指令。

注：C0 指令和 CASE2 情况详见 ISO/IEC 7816-4：2005 中的相关部分。

8 卡片存储模型（CSTO）

本规范规定，卡片存储模型 CSTO 要提供给应用动态存储和静态存储两种接口。其中，动态存储的数据断电后可以丢失，而静态存储数据在断电后必须保留。

对于静态存储，本规范要求卡片存储模型 CSTO 至少要有一个提供给 CRE 的静态存储接口（SMI），以提供给 CCORE 一个静态内存 IO。而在芯片资源允许的情况下，本规范建议 CSTO 提供一个标准的 COS 文件系统接口来处理 COS 文件操作，接口见附录 A。

本规范不限制 SMI 的实现方式是否采用文件系统或者别的方法，但是要求 SMI 和应用相关。

本规范要求，静态存储空间（包括 COS 文件系统）拥有掉电保护机制（POP），最小保护周期为一个 APDU 指令周期，也就是说，在一条指令周期内的所有静态存储操作全部被保护，即要么全部成功，要么全部失败。在 APDU 指令执行期间，发生的 CEXP 事件和 SW 的错误返回事件，CSTO 应自动执行文件系统的恢复操作。在退出 POP 周期后，静态数据被真正写入静态存储（包括文件系统操作）。

POP 可以跨指令执行，也即在多条指令周期内执行如上操作。执行 POP 的方法如表 3 所示：

表3 POP 方法

POP 方法	方法描述
enter_stom	进入 POP 周期
exit_stom	退出 POP 周期

对于动态存储和静态存储的空间大小，本规范没有强制性要求，卡商可以根据行业需求和芯片资源自行定制。

9 卡片运行时环境模型（CRE）

9.1 综述

CRE 负责以下功能：

a） 卡片安全框架的建立；
b） APDU 命令的预处理和分发；
c） 应用的管理，包括创建、装载和删除；
d） 卡片的生命周期管理；
e） 权限管理；
f） 异常管理。

9.2 卡片安全框架

本规范规定每张卡片都有且只有一套安全框架，卡片安全框架负责管理卡片和安全相关的所有数据。

卡片启动时，CRE 完成卡片安全框架的初始化，更新卡片安全框架之后，卡片安全框架被激活，卡片进入安全状态 Security，可以进行安全会话。

本规范规定卡片安全框架与 CRE 的接口如表 4 所示：

表4 卡片安全框架与 CRE 的接口说明

数据类型	类型定义
Card Secure Management Key	卡片安全管理密钥
Card Life State	卡片生命状态
Card Secure Session Package	卡片安全会话包

卡片安全框架被激活后，在卡片安全会话中的运行示意图如图 3 所示：

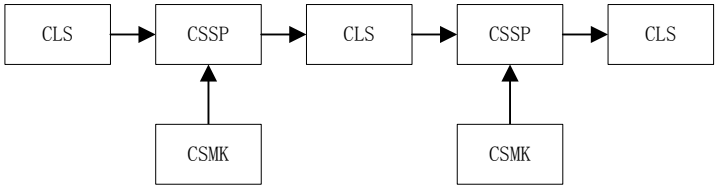

图3 卡片安全会话示意

9.3 命令分发

在 CCOM 模型封装完 APDU 模型之后，交给 CRE 处理。本规范要求 CRE 完成如下工作：

a） 对 APDU 模型进行删选，挑选出需要预处理的 APDU 模型。
b） 将需要预处理的 APDU 模型交给卡片全局服务进行处理。
c） 卡片全局服务执行完预处理指令后，若 APDU 模型选择了 CSS 服务，则后续指令 CGS 将传给 CSS 服务执行。如果 APDU 模型选择了应用，则后续指令交给应用处理。如图 4 所示：

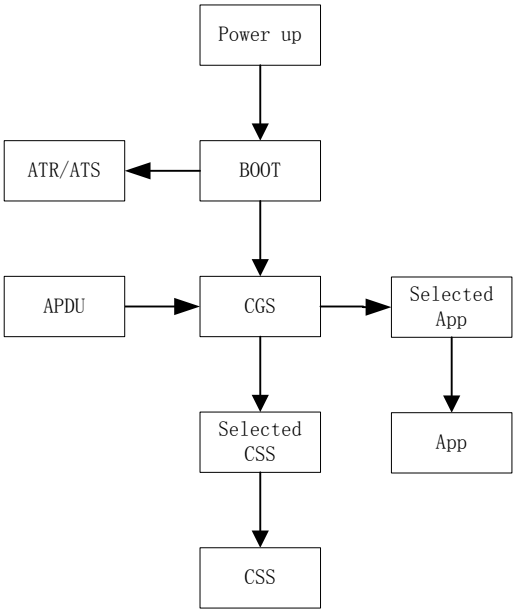

图4 CRE启动运行的示意图

注：卡片全局服务和卡片系统服务详见本规范的第 9.4 和 9.5 章节。

预处理 APDU 是指被 CRE 服务保留的 APDU 模型，这些指令用来通过会话启动卡片全局服务。

本规范规定的预处理 APDU 指令说明如表 5 所示：

表5 预处理 APDU 指令

INS	指令说明	执行方	执行完成处理
A4	选择一个应用	卡片全局服务，被选择的应用	选择成功后，交给被选择的应用进行后续处理
B1	创建 Bytecode 应用安全框架	卡片全局服务，被选择的 Bytecode 应用	执行成功后，调用被选择应用的回调函数
E0	创建 Nativecode 应用的安全框架	卡片全局服务，被选择的 Nativecode 应用	执行成功后，调用被选择应用的回调函数
84	获取随机数	卡片全局服务，被选择的应用	首先交给被选择的应用处理，如果应用不处理该指令则由卡片全局服务处理

9.4 卡片全局服务

卡片全局服务（CGS）属于 CRE 的内核程序，具有系统最高权限（Boot）。任何 APDU 模型都会被 CGS 过滤。

本规范规定卡片全局服务的功能如表 6 所示，具体的功能介绍见附录 C。

表6 卡片全局服务的功能

服务类别	服务描述
Select handler	处理全局 Select 指令，处理完成后，将指令交给应用处理
Create Native application framework	在当前任何目录下，创建 Native 应用的框架
Create Bytecode application framework	在当前任何目录下，创建 Bytecode 应用的框架
Attach a service to application	在当前任何目录下，关联一个服务到当前应用
Get random	全局获取随机数指令

9.5 卡片系统服务

卡片系统服务（CSS）是预先写入卡片的一个 Native 应用，它先于其他任何应用存在，具有应用最高权限（System）。

本规范规定了 CSS 的应用名，CSS 的功能见表 7 所示，具体的功能介绍见附录 C。

表7 卡片系统服务的功能

服务类别	服务描述
Update Card Frame Work	更新卡片安全框架
Create Bytecode Application FrameWork	在当前目录下，创建 Bytecode 应用的安全框架
Load Application Secure FrameWork	装载应用的安全框架
Load Application Executable Module	装载应用的可执行模块
Get Application Access Token	获取应用的访问令牌
Delete Application	删除应用
Promote Application Access Right	提升应用的访问权限
Card Frame Work Delete	卡片安全框架清除

9.6 卡片的生命周期管理

一张新卡上电以后，CRE 启动，创建卡片安全框架的初始模板，成功后，卡片进入 Build 状态。

一张 Build 状态的卡片，更新卡片安全框架后，卡片进入 Security 状态。

一张 Security 状态的卡片被具有 System 权限的应用锁定后，卡片进入 Blocked 状态。

一张 Security 状态的卡片被具有 Boot 权限的会话删除卡片安全框架，卡片进入 Build 状态。

注：生命周期状态详见本规范的第 14 章节。

9.7 权限管理

本规范规定卡片的权限有三大类：Boot 权限、System 权限和 App 权限。

其中，Boot 权限为系统唯一，只有卡片全局服务（CGS）和开放的 Boot 会话具有 Boot 权限。

System 权限，具有对 CSFW、CRE 以及卡片的域进行操作的全部能力。具有 System 权限的应用为：CSS、System 会话和 System 托管安全域服务。

System 权限下属子权限由相应的 Sytem 托管安全域服务来定义。

App 权限，只具有对 ASFW 和应用的域进行操作的能力。具有 App 权限的应用为：App 应用，应用会话和应用托管安全域服务。

App 权限下属子权限由相应的 App 托管安全域服务来定义。

三种权限的应用和子权限情况说明如表 8 所示。

表8 三种权限的应用和子权限情况说明

权限	是否具有子权限	具有此权限的应用
Boot 权限	N	卡片全局服务（CGS），开放的 Boot 会话
System 权限	Y	卡片系统服务（CSS），System 会话，Sytem 托管安全域服务
App 权限	Y	App 应用，应用会话，应用托管安全域服务

9.8 异常管理

本规范规定，CRE 要能够截获 Bytecode 应用的所有异常，在截获到异常后，向 CCOM 返回以 6F 开头的 SW 码。表 9 列出了本规范规定的卡片异常类型及编码范围。

表9 卡片异常类型及编码范围

异常代码范围	异常类型	发生异常后 CRE 采取的行为
6F01~6F0B	Bytecode 应用内核可屏蔽异常	应用实例关闭
6F0C~6F7F	Bytecode 应用内核自定义异常	由应用服务商定义

6F81~6F83	Bytecode 应用内核不可屏蔽异常	应用实例关闭
6F84~6FA0	CRE 异常	卡片锁死，等待重新上电。
6FB0~6FFF	CRE 扩展异常	由卡商定义

本规范对 Nativecode 应用运行过程中产生的异常不做强制规定。

9.9 卡片内核服务列表（CKSL）

卡片内核服务列表（CKSL）唯一地记录了应用标识符 AID 与服务入口之间的对应关系。
本规范对 CKSL 的实现和形式不做任何限制。
对于 Nativecode 的应用来说，CRE 从 CKSL 中取出服务入口，直接交给服务进行命令分发。

10 卡片的注册表（CREG）

本规范规定，卡片注册表（CREG）为卡片的可信任组件，访问权限为 System。CREG 的组件说明如表 10 所示。

表10 CREG 包含的组件及说明描述

类别	组件	描述
CFW（卡片框架）	CSFW	卡片安全框架
	CDFQI	卡片数据域引用标识，它记录了卡片数据域所有数据对象链表的一个索引。
	CSFQI	卡片安全域引用标识，它记录了卡片安全域，托管安全域链表的一个索引。
	CRCD	卡片运行时环境（CRE）运行数据，CRE 启动，运行中的所有静态参数记录在这里，比如 ATR/ATS 的设置参数。
AFW（应用框架）	AID	应用标识符，这里用来唯一的标识一个应用框架
	ASFW	应用安全框架，这里保存了应用的所有安全属性。
	ADFQI	应用数据域引用标识，它记录了应用数据域所有数据对象链表的一个索引。
	ASFQI	应用安全域引用标识，它记录了应用安全域，托管安全域链表的一个索引。
	AMFQI	应用方法域引用标识，它记录了应用方法的索引。

一个标准的注册表（CREG）包含了对所有卡片框架和应用框架的记录。
本规范对各组件的设计不做强制规定，卡商可以根据实际情况自行实现上述组件。

11 卡片应用管理器（CAM）

11.1 概述

本规范规定卡片应用管理器（CAM）负责将注册表中的应用框架（AFW）中记录的静态信息提取出来，生成应用实例。
注：应用实例详见本规范的第 13.4 章节。
一个符合本规范的 CAM 模型，应该承担的功能如表 11 所示。

表11 CAM 的功能说明

CAM 担负的职责	职责分析
Create，Dispatch，Receive CAM Secure Session	创建，分发，接收 CAM 安全会话
Create Application Instance 行为	在安全框架下和 CRE 环境下创建应用实例。一个应用实例的创建，要依托注册表（CREG）中记录的应用框架，在 Select 会话中创建。
Load Application To CSTO 行为	从 CRE 会话装载一个应用到 CSTO 模型。这个仅限于 Bytecode 应用
Register Application To CREG 行为	注册一个应用到 CREG 模型
Run Application From CSTO 行为	运行一个 CSTO 模型上的应用。这个仅限于 Bytecode 应用
Monitoring And Isolation Applications 行为	监控并且对应用进行隔离。这个仅限于 Bytecode 应用

11.2 Create,Dispatch,Receive CAM Secure Session 行为描述

CAM 接收 CCOM 提交的 APDU 模型，创建相应的 CAM 安全会话模型。将一个 CAM 安全会话交给当前激活的应用，并接收应用的返回，然后再将返回的数据封装为 CAM 安全会话返回，返回给 CCOM。如图 5 所示。

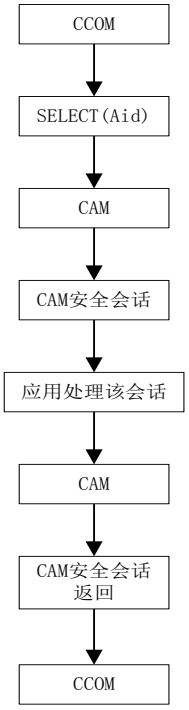

图5 创建，分发，接收CAM安全会话的流程图

11.3 Create Application Instance 行为描述

CAM 在接收到 CRE 的 Select 会话后，从 CREG 中提取出相应的 AFW，然后 CAM 会将 AFW 中记录的应用静态信息，派生出应用实例 AI。在一个标准的 AI 模型中，AKSS 为 System 权限的内核模型，只有 CAM 可以访问。其他模型都为 App 权限，应用方法域（AMF）可以直接访问。图 6 介绍了创建应用实例的流程。

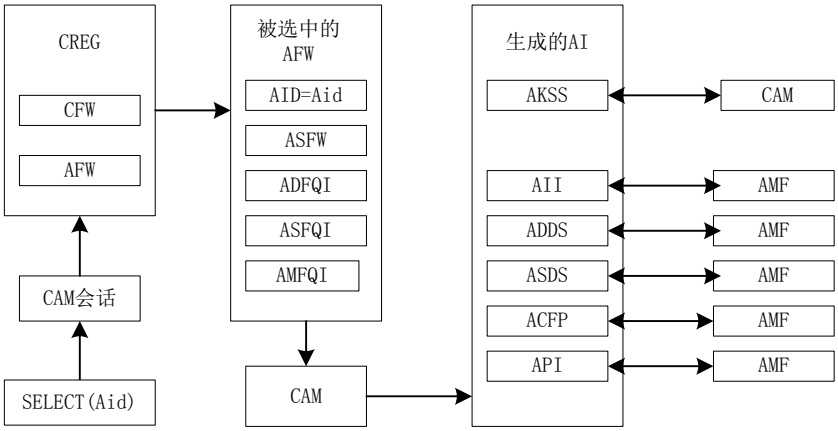

图6 CAM创建应用实例的流程图

11.4 Load Application To CSTO 和 Register Application To CREG 行为描述

Load Application To CSTO 和 Register Application To CREG 行为负责装载和注册一个应用，所以此行为必须在应用安全框架建立之后。应用安全框架是由 CGS 服务创建的。

在安全框架建立之后，CAM 接收来自 CRE（其实是 CRE 目录下的 CGS 服务）的 LOAD 安全会话，并将应

用模块数据存储在 CSTO 模型。同时，将应用模块解包然后在 CREG 中进行注册。

Load Application To CSTO 和 Register Application To CREG 行为都会产生相应的内核事件通知应用，应用方法域（AMF）可以在此阶段处理相应的初始化操作。图 7 介绍了 CAM 装载应用和注册应用的流程。

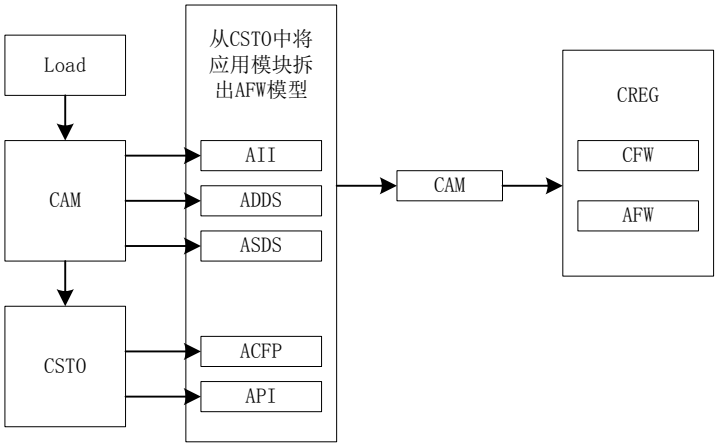

图7 CAM装载和注册应用的流程图

11.5 Run Application From CSTO 和 Monitoring And Isolation Applications 行为描述

CAM 负责在一个应用实例创建之后，将一个应用的方法域（CMF）从 CSTO 中激活并运行。在运行过程中，CAM 负责监控该应用，并保证应用的相互隔离。

对于一个 Nativecode 应用，在 CRE 启动的时候其方法域就已经被激活。而对于一个 Bytecode 应用来说，则需要 CAM 中的 CVM 来启动并运行它。本规范并不强制要求 CAM 必须携带 CVM。CVM 作为 CAM 的一个可选配置，主要用来对 Bytecode 应用的 Run Application From CSTO 和 Monitoring And Isolation Applications 行为负责。而对于一个 Nativecode 应用来说，上述两种行为无效。

一个 Bytecode 应用必须在应用实例创建后才可以被运行。如图 8 所示 CAM 负责从 CSTO 模型中提取出应用模块，并且交给 CAM 中的 CVM 执行，生成 AEM。一个激活的 AEM 只能访问本应用实例中的 AII，ADDS，ASDS，ACFP 和 API。其他应用的相关资源被保护，访问不到。而 AKSS 则由 CVM 控制，AEM 也无法访问。这也是 Monitoring And Isolation Applications 行为的主要职责。

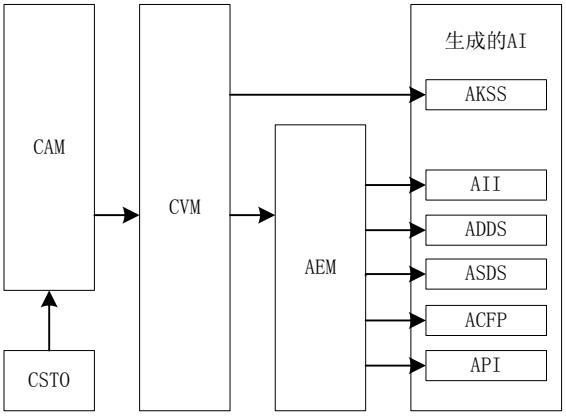

图8 CAM运行应用及监控隔离应用的流程图

12 卡片虚拟机（CVM）

CVM 由三个要素构成，有字节码规范、解释器和 IO 控制。本规范对它们的要求见表 12 所示。CVM 担负的职责如表 13 所示。

表12 CVM 的构成要素及相应要求

CVM 构成要素	CMAP 对其的要求
字节码规范	一个 CVM 的字节码规范，规定了一整套中间代码的完备性守则和实施方法。CMAP 要求 CVM 的字节码规范要有充足的完备性对应用开发的支持。并且提供可公开的相关文档。 本规范不提供任何字节码规范样本。
解释器	本规范规定,CVM 字节码解释器能够完整的额解释所有字节码分支。并可以完整的支持 CMAP 中提到的应用管理器接口（CAM）。作为 CAM 的可选组件，CVM 解释器是其与 CAM 沟通的桥梁。
IO 控制	本规范规定，CVM 要有安全的 IO 控制机制。其控制方式，要满足 CAM 接口规范和应用安全规范的相关要求。

表13 CVM 的职责

CVM 担负的责任	责任描述
Load a AEM	装载一个应用可执行模块
Initialize Application Dynamic Data Package	初始化应用动态数据包
Create ASDS instance for AI	为应用实例（AI）创建应用静态数据空间（ASDS）
Create ADDS instance ffor AI	为应用实例（AI）创建应用动态数据空间（ADDS）
Run a AEM	运行一个应用可执行模块
Handle io connect CAM	处理应用与 CAM 之间的 IO 接口

本规范不提供任何 CVM 样本和实现的方法，卡商根据自己需要开发符合本规范的 CVM 并且移植到 CAM 中去。

13 应用架构

13.1 应用的分类

本规范支持两种类型的应用：Bytecode 应用和 Nativecode 应用。

Bytecode 应用：是一种执行代码为字节码（Bytecode）的应用。运行该应用需要在卡片中由卡片虚拟机（CVM）来对 Bytecode 做解析。一个 Bytecode 应用的可执行文件包括了应用模块的静态数据和执行代码。Bytecode 应用只能由 CVM 启动，并进行运行时监控。Bytecode 应用的代码是可以被删除的。并且可以被关联到一个或者多个实例。

Nativecode 应用：是执行代码为机器指令的应用。该应用在 CRE 启动后就被激活。所有的 Nativecode 应用都是一个内核服务，其服务入口记录在 CRE 中的内核服务列表 CKSL 中。Nativecode 应用无需 CVM 解析。一个 Nativecode 的代码是无法被删除的，它可以关联到一个或者多个应用实例

13.2 应用框架（AFW）

AFW 由 CRE 中的内核服务在创建应用的过程中产生，并注册在 CREG 中。一个 Select 会话可以启动一个 AFW。

本规范规定，一个应用框架的建立，要经历 Frame，Load，Register 三个阶段。每个阶段的过程描述如表 14 所示。

表14 AFW 创建的过程和服务提供者的关系

AFW 创建的过程	过程描述	服务提供者
Frame	创建应用的安全框架，在这个阶段创建的安全框架为一个 ASFW 动态实例，并没有被静态化，也就是说此时如果断电，则此实例消失，对卡片不产生任何影响。	卡片全局服务（CGS）
Load	装载应用的模块到 CSTO 模型，此过程仅对 Bytecode 应用有效	卡片应用管理器（CAM）
Register	注册一个应用框架（AFW）到卡片注册表（CREG）	卡片应用管理器（CAM）

图 9 是对一个标准 AFW 创建过程的描述。

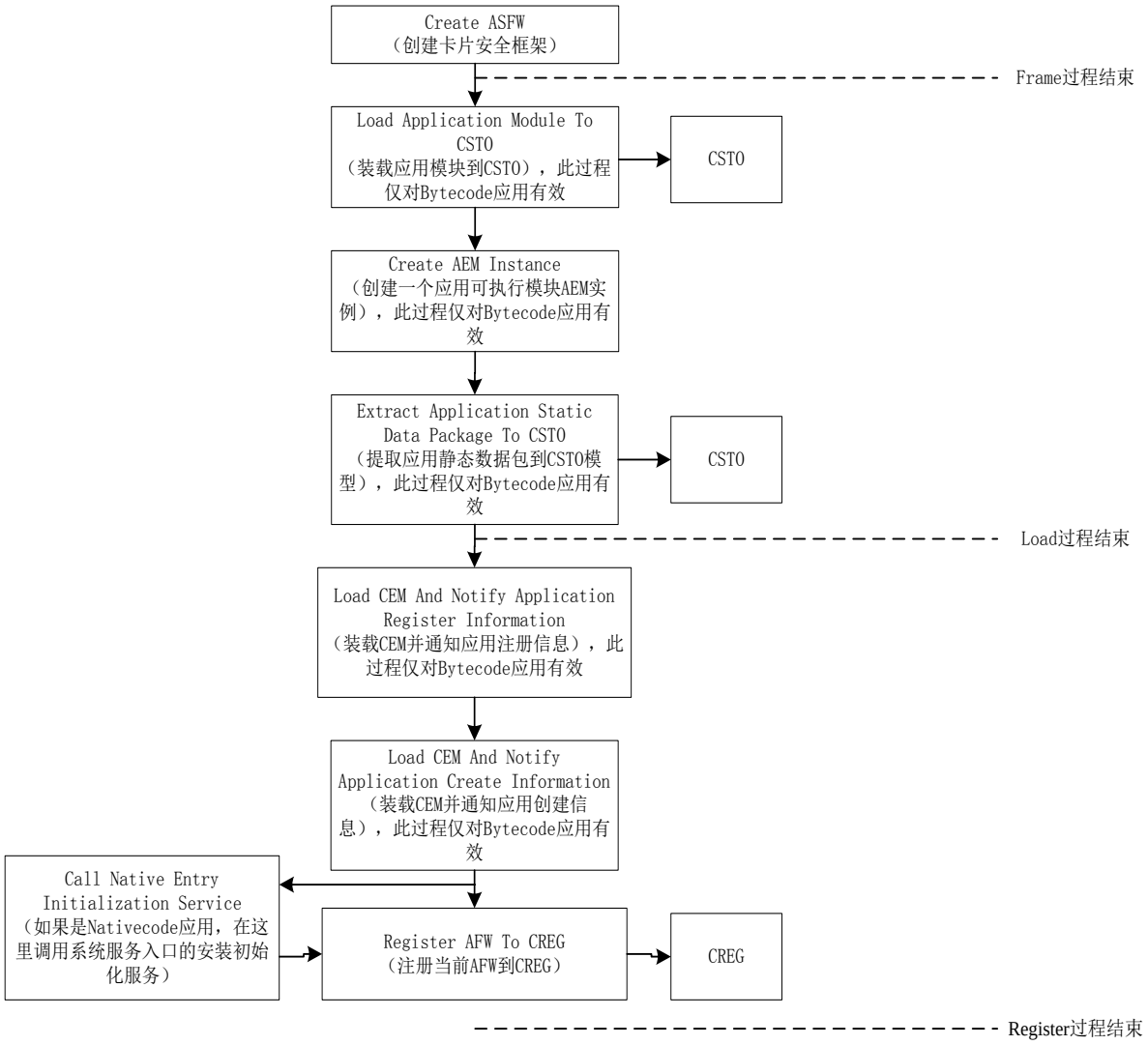

图9 标准AFW创建过程的流程图

13.3 应用的可执行模块（AEM）

本规范规定 AEM 负责应用的命令分发。AEM 由 CAM 从 CSTO 模型中存储的应用模块中提取，最后由 CVM 加载生成。

AEM 模型应承担如下责任，见表 15。

表15 AEM 模型的职责

AEM 担负的责任	责任描述
Application Apdu Dispatch	应用命令分发
Provide Application Static Data Package	提供应用静态数据包
Provide Application Dynamic Data Package	提供应用动态数据包
Provide Application Exeutable Code Package	提供应用可执行代码包
Application Dynamic Data Package Initialization	应用动态数据包初始化

当一个应用框架被装载的时候，CVM 将会将 AEM 中的应用静态数据包解析出来，在卡片应用静态数据目录（CASDD，名字空间：SCard：\Const）中创建相应的存储实例。

当一个应用被选择（Select）的时候，AEM 被 CVM 加载，CVM 将会从应用静态数据目录（CASDD，名字空间：SCard：\Const ）中打开相应的实例，并在当前应用实例(AI)中提交并创建应用静态数据空间（ASDS）实例。同时，CVM 从 AEM 提取出应用动态数据包，并在当前应用实例（AI）中提交并创建应用动态数据空间（ADDS）实例。如图 10 所示：

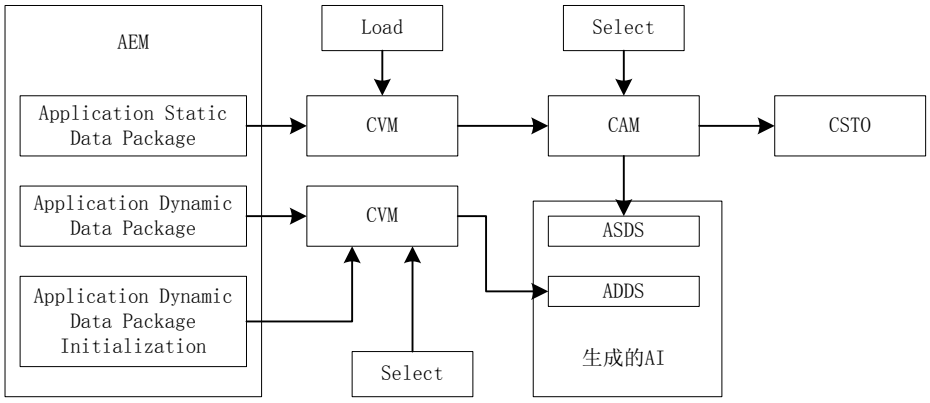

图10 应用被装载和选择过程中AEM的职责

AEM 模型只适合 Bytecode 应用模型，与 Nativecode 应用无关。对于 Nativecode 的应用来说，从 CAM 过来的会话，CRE 会从 CKSL 中取出服务入口，直接交给服务进行命令分发。而对于 Bytecode 应用来说，应用实例（AI）中的应用信息（AII）会提取出相应的应用模块交给 CVM，由此生成 AEM 实例。最后由 AEM 实例处理命令分发。流程如图 11 所示。

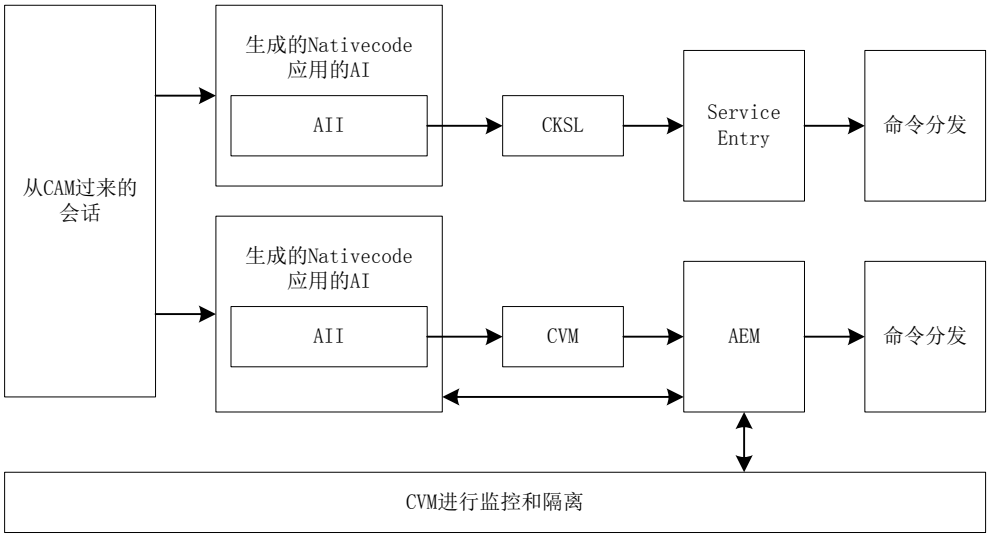

图11 应用运行的流程

13.4 应用的实例（AI）

一个应用的实例独一无二地标识了一个激活的应用。一个符合本规范的 AI 会同时兼容 Bytecode 和 Nativecode 两种应用类型。表 16 详细描述了 AI 的每个组件。

表16 AI 组件及相应提供方的关系

AI 组件	组件描述	访问权限	提供方
ASCC	应用安全控制上下文，存储应用的安全控制信息，由 CAM 管理。CAM 所有对应用的管理操作数据都存放在这里。	System	CAM

表16 AI 组件及相应提供方的关系（续）

AI 组件	组件描述	访问权限	提供方
AKSS	应用内核共享空间，这个是由 CAM 提供给每个应用的一个共享空间。	System	CAM
AII	应用身份标识，包括了应用的 AID，权限，Fid 等身份信息。	System	CAM
ADDS	应用动态数据空间，仅对 Bytecode 应用有效。由 CVM 在加载 AEM 的过程中提取应用动态数据包生成 ADDS 实例	App	CVM
ASDS	应用静态数据空间，仅对 Bytecode 应用有效。由 CVM 在加载 AEM 的过程中提取应用静态数据包生成 ASDS 实例	App	CVM
ACFP	应用 COS 文件包，对于支持 COS 文件系统的 CMAP 卡片，CAM 将一个 COS 文件目录节点关联到应用实例（AI）。	App	CAM
API	应用编程接口。在一个应用实例（AI）生成后，CAM 收集 CRE 和 CSTO 提供的 API 接口提交给应用。所有提交给应用的 API 都是具有 APP 权限的。应用可以自由调用。	App	CAM

图 12 展示了 AI 组件模型以及与其他组件的关联方式。

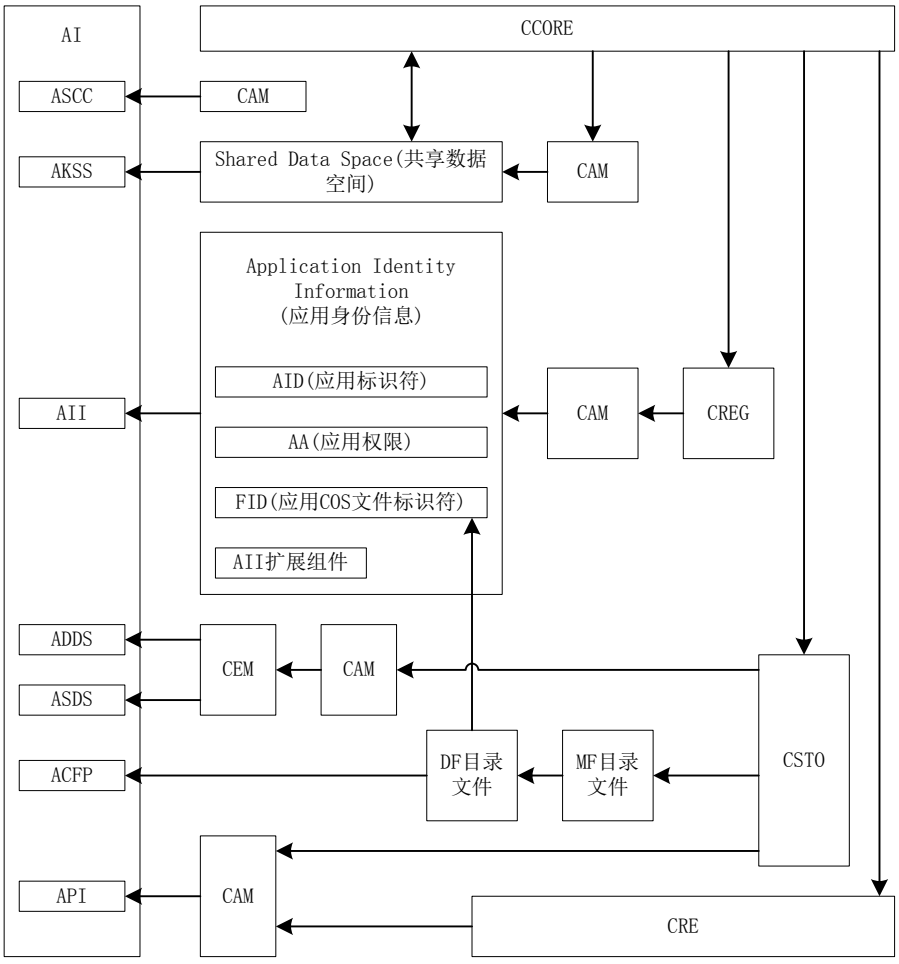

图12 AI模型与其他组件的关联方式

本规范不对 AKSS 的内容做强制规定，给应用提供多大的共享空间由卡商决定。而应用是否有权限访

问 AKSS，可以由应用的托管安全域服务提供的子权限来限定，这个也由卡商决定。默认情况下只有 System 权限的系统应用才可以访问 AKSS。

本规范规定，AII 模型，最少应该提供 AID，AA，FID 三个参数。这三个参数，应用具有只读权限，不具备改写权限。至于以什么样的方式提交给应用去读取这三个参数，也由卡商决定。

ACFP 组件为 AI 的可选组件模型，有且只有卡片支持 COS 文件系统的情况下，由 CSTO 模型提供相应的支持。

API 接口由 CRE 和 CSTO 提供，本规范只提供标准卡片 API 模型。它规范了一个卡片的最小编程接口配置。卡商可以根据实际情况添加 API 接口，并公开 API 接口模型参数和说明。

13.5 应用的隔离与共享

本规范要求每个应用实例必须互相隔离，而卡商可以根据实际情况，对应用做共享机制。共享的方式为：AI 模型中增加 AKSS 组件。

应用的隔离通过 AI 的相互隔离来实现。对于任何一个应用来说，其可以访问的资源都在 AI 中得到体现。而每个 AI 都有自己独立的 ASCC、AII、ADDS、ASDS、ACFP 和 API 组件。这些组件相互独立，分别映射不同的空间。

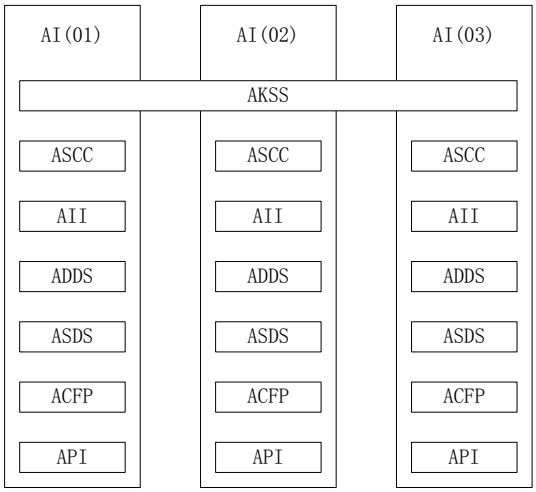

图13 应用的隔离和共享的实现方式

如图 13 所示，各 AI 的 AII 由 CAM 进行维护，CAM 为每个 AI 维护一个独立的 AII 列表。

各 AI 的 ADDS 和 ASDS 分别指向分立的内存空间，该内存空间由 CAM 维护。

各 AI 的 ACFP 指向一个独立的 COS 文件目录的位置，由 CSTO 中的 COS 文件系统维护各个目录的独立性。

各 AI 的 API 由 CAM 提供唯一的接口并保证访问的合法性。

AKSS 负责应用之间的共享，由卡商定制。卡商根据需要，确定是否需要 AKSS 模型以及提供给应用哪些 AKSS 方法去安全访问 AKSS 空间。本规范规定，合法的 AKSS 空间因为它是内核空间，所以对 AKSS 的访问要有边界控制能力，卡商自己决定提供给应用怎样的 AKSS 接口去访问 AKSS 组件。但是无论何种方式都要保证不会对 AKSS 造成越界风险。

一个定制的 AKSS 组件，卡商必须公布其实现方法和安全架构，并公布访问 AKSS 的扩展 API。

13.6 应用的权限

默认情况下，本规范认为所有的应用都属于 App 权限。但是允许应用通过应用托管安全域实现在 App 权限目录下增加其他子权限设置。

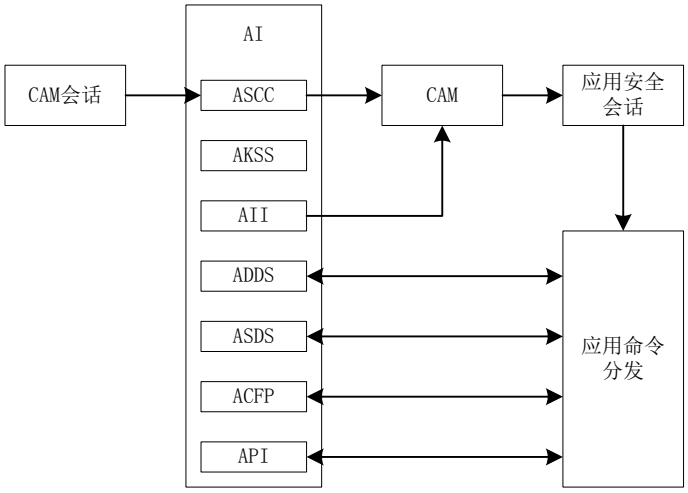

图14 没有应用托管安全域的工作模式

如图 14 所示，对于一个没有应用托管安全域的 AI 来说，CAM 会话首先访问 ASCC，并取出 AII 中的应用权限进行分析，过滤 CAM 会话。由 CAM 最终生成应用安全会话，然后进入应用的命令分发，此时控制权交给应用自己来处理 ADDS、ASDS、ACFP 和 API。

如图 15 所示，对于一个有应用托管安全域的 AI 来说，CAM 会话首先访问 ASCC，并取出 AII 中的应用权限进行分析，过滤 CAM 会话。由 CAM 最终生成应用安全会话 01。然后 CAM 访问 ASCC 取出应用托管安全域服务索引，然后访问应用托管安全域服务，并进入其会话流程，会话完成后，由 CAM 处理会话结果，生成应用安全会话 02,最后进入应用的命令分发，此时控制权交给应用自己来处理 ADDS，ASDS，ACFP，API。

在应用托管安全域服务命令分发阶段，应用托管安全域服务与 AII 中的扩展组件交互，进行相应的子权限管理操作。在应用托管安全域服务的 AFW 的安装初始化阶段，应用托管安全域服务，将子权限定义写入 AII 扩展组件字段中。

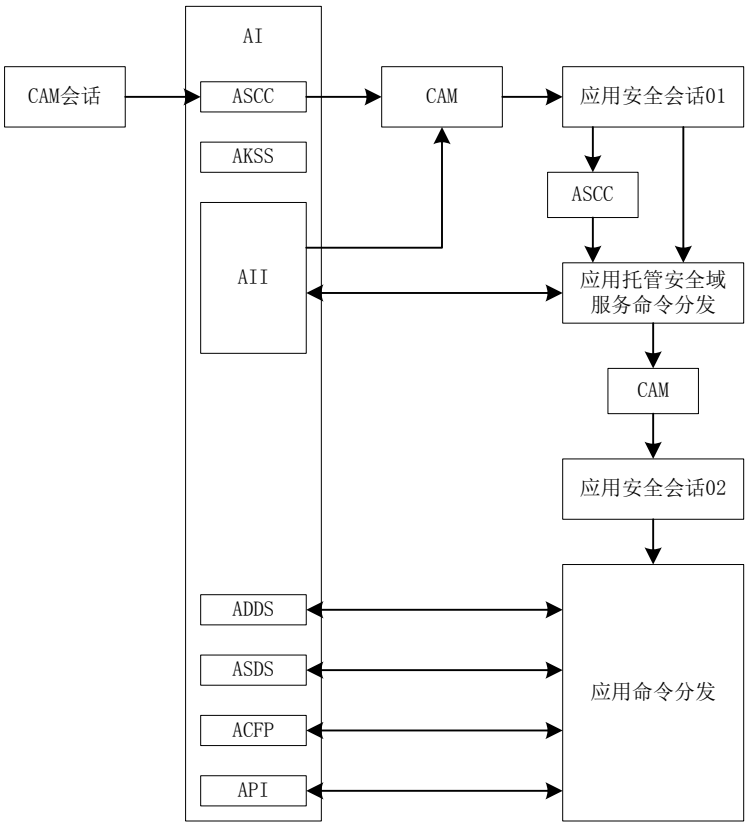

图15 有应用托管安全与的工作模式

13.7 应用的命令分发

应用的命令分发负责处理被 ASCC 处理过后的应用安全会话。应用命令分发有两个阶段，说明见表 17：

表17 应用命令分发的两个阶段说明

生命周期	命令分发	描述
Security	Create，Register，Init	对于 Bytecode 应用，在 Security 阶段需要处理，CAM 发送的 Create 和 Register 通知，来进行相应的初始化工作。 对于 Nativecode 应用来说，在服务入口处处理 Init 通知，并进行相应的初始化工作。
Actived	Select，其他指令	通过 Select 激活的应用，CAM 会把 Select 安全会话传递给应用做后续处理。在 Actived 之后，应用也应处理其他指令。

14 生命周期模型

14.1 卡片生命周期

14.1.1 概述

本规范规定的卡片生命周期状态有四种：Build、Security、Blocked 和 Deleted。具体描述如表格 18 所示。

表18 卡片生命周期状态

生命状态	状态描述
Build	卡片运行时环境建立
Security	卡片安全框架建立
Blocked	卡片被锁定
Deleted	卡片被清除

14.1.2 卡片生命周期状态 Build

在 CRE 启动后，CRE 会自动检测卡片安全框架，如果此卡为一张新卡，那么 CRE 将会自动创建卡片安全框架的初始模板，并进行其他 CRE 相关的初始化工作。创建成功后，卡片进入 Build 状态，此时的卡片具备协议分发的能力，可以运行，但此时是不安全的。

14.1.3 卡片生命周期状态 Security

用户更新卡片安全框架之后的卡片进入 Security 状态。此时的卡片具有协议分发的能力，是安全的，可以在安全框架的基础上进行安全会话，完成卡片生命周期的后续管理以及安全的应用管理服务。

14.1.4 卡片生命周期状态 Blocked

本规范允许一个具有 System 权限的应用来对卡片进行锁定操作。被锁定之后的卡片除了可以响应 System 权限的应用指令外，其他指令全部被封锁。

14.1.5 卡片生命周期状态 Delete

本规范允许一个具有 Boot 访问权限的会话来删除卡片安全框架。被删除之后的卡片在重新上电后，进入上述 Build 阶段。考虑到运营的风险性，卡片对 Boot 权限的会话不做强制要求，也就是说卡片可以视情况考虑是否支持开放 Boot 权限，以及 Boot 权限的会话。所以上述删除操作也不做强制要求，卡商根据具体情况考虑是否支持上述功能。

14.2 卡片生命周期的迁移

一张新卡上电以后，CRE 启动，创建卡片安全框架的初始模板，成功后，卡片进入 Build 状态。

一张 Build 状态的卡片，更新卡片安全框架后，卡片进入 Security 状态。

一张 Security 状态的卡片被具有 System 权限的应用锁定后，卡片进入 Blocked 状态。

一张 Security 状态的卡片被具有 Boot 权限的会话删除卡片安全框架，卡片进入 Build 状态。

卡片生命周期的迁移方法如图 16 所示。

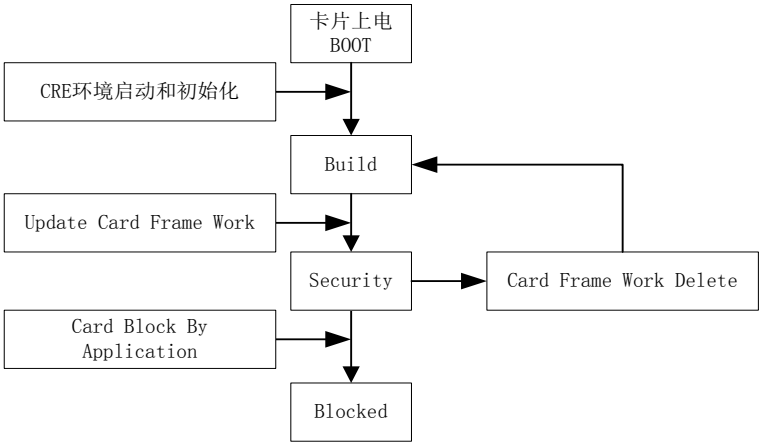

图16 卡片生命周期的迁移和系统服务的关系

14.3 应用生命周期

本规范规定，CAM 框架下应用的生命周期有五种，分别是：Ready、Build、Security、Actived 和 Unactived。具体描述如表 19 所示。

表19 应用生命状态

生命状态	状态描述
Ready	在 CFW（卡片框架）被建立的时候，卡片处于安全状态周期。此时卡片处于创建应用的就绪阶段。
Build	AFW（应用框架）被建立。此过程中，应用被安装在一个特定（由发卡人选择）的目录下。此阶段不对 CMAP 卡片产生任何静态数据，一旦掉电 Build 数据消失。
Security	应用安全框架建立，应用模块被安全装载。此阶段一个完整的 AFW 建立，并被 CMAP 卡片记录下来。
Actived	应用被激活，一个 Select 指令激活一个 AFW（应用框架），由 CAM 生成 AI（应用实例）。此条指令和后续指令将进入应用的命令分发流程。
Unactived	应用被关闭，本规范暂不支持应用逻辑通道功能

应用的生命周期被 CAM 保存在 ASCC 中。

注：这里所指的应用生命周期，是在 CAM 框架下，由 CAM 控制的应用生命周期模型。而应用本身的锁定，临时锁定等生命周期模型则由应用自己来处理。本规范对此生命周期不做限制性规定。

14.4 应用生命周期的迁移

卡片框架被建立时，卡片处于创建应用的就绪阶段，即 Ready 状态。

应用处于 Ready 状态时，CGS 创建应用框架，应用进入 Build 状态。

应用处于 Build 状态时，CSS 创建应用安全框架，装载应用模块到 CSTO，并注册应用框架到 CREG 中。应用进入 Security。

应用处于 Security 状态时，用户进行 Select 操作，应用就进入 Actived 状态。

应用处于 Actived 状态时，Power Off 后，应用进入 Security 状态。

应用处于 Actived 或 Security 状态时，用户进行 Delete 操作，应用就进入 Ready 状态。

应用处于 Build 状态时，Power Off 后，应用进入 Ready 状态。

CAM 框架下应用生命周期的迁移方法如图 17 所示。图中还加上了相关的服务提供者、卡片组件模型以及用户操作。其中，CGS 和 CSS 是服务提供者，CSTO 和 CRE 是卡片组件模型，Power Off、Select 和 Delete 是用户操作。

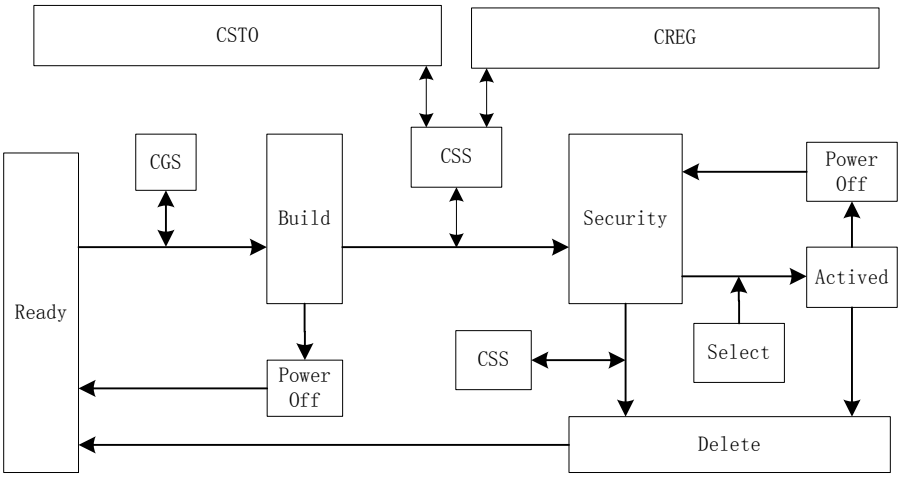

图17 应用生命周期迁移与各模块的关系

生命周期的几个主要迁移过程如表 20 所示。

表20 应用生命周期迁移与服务的关系情况

生命周期迁移	服务提供者	安全会话权限	影响的卡片组件模型
Ready to Build	CGS（卡片全局服务）	App	
Build to Security	CSS（卡片系统服务）	System	CSTO，CREG
Security to Actived	Select 会话	App	

15 域模型

15.1 综述

在本规范中，卡片是一种特殊的应用。与应用一样都是由域模型构成的。一个完整的域模型包括数据域、方法域和安全域模型。完整的域模型描述了应用所有的属性。有所不同的是，卡片的方法域和安全域是事先固化在卡片里的，而应用的所有域都是要在应用的框架建立的同时进行动态创建的。

对于 Nativecode 的应用来说，应用的方法域也是事先固化在卡片中的，其他域也是动态加载的。

一个应用的工作流程可以用图 18 来抽象说明，一个 CAM 安全会话，经过 ASF 后变成应用安全会话，并被 AMF 接收并处理。在处理期间，AMF 会通过 ASF 的安全控制去访问 ADF，最后返回 CAM 安全会话。

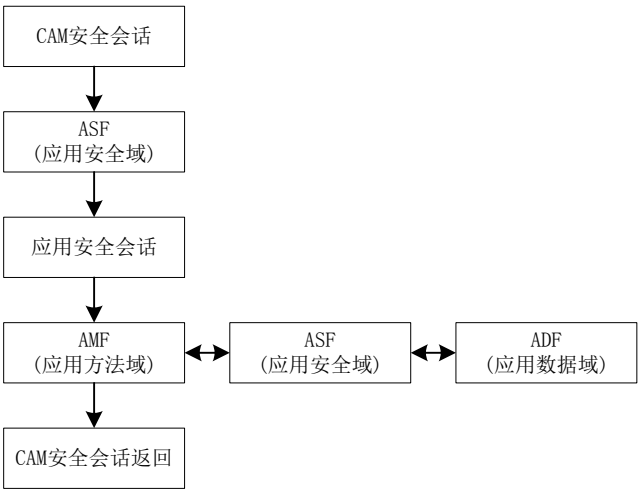

图18 应用的工作流程

15.2 数据域模型

应用/卡片的数据域，是在一个应用实例生成后，CAM 会话和实例本身可以访问的数据资源。

本规范规定，数据域包括应用动态数据（ADD），应用静态数据(ASD)和应用 COS 文件系统(ACFS)三部分组成。其中 ACFS 为应用可选配置。这取决于 CSTO 模型是否包含 COS 文件系统。

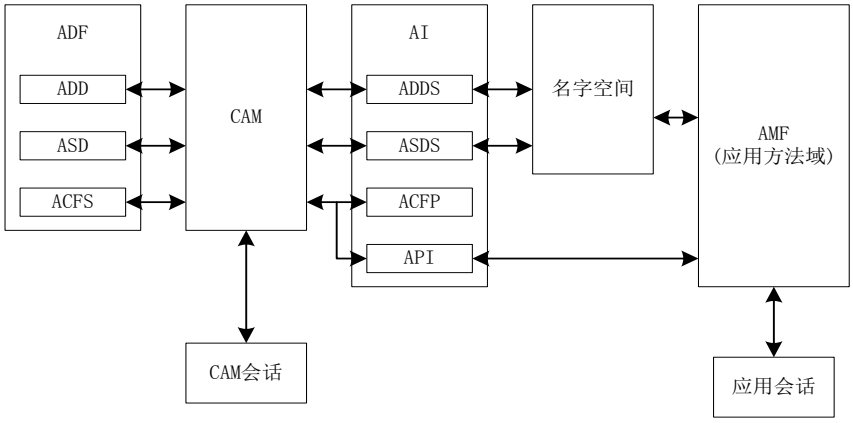

图19 应用数据域的构成及其在应用管理过程中的功能体现

如图 19 所示，在 CAM 控制下，ADD 和 ASD 被提取出来分别形成为 ADDS 和 ASDS，并在 AI 中注册。其中， AMF 通过名字空间对 ADDS 和 ASDS 进行访问。

ACFS 资源（如果存在的情况下）经过 CAM 处理后，在 AI 中生成 ACFP 和 API 接口。AMF 通过 API 接口（ACFS 接口函数）访问应用的 ACFS 资源。

CAM 会话可以直接通过 CAM 访问 ADD、ASD 和 ACFS。这将通过 CAM 进行处理。也就是说，CAM 可以不通过 AI，直接访问 ADF 的任何资源。而应用会话，必须通过 AMF，并通过 AMF 访问 AI 来最终与 ADF 进行交互。

在卡片中，ADD 与 ASD 资源都会被虚拟为虚拟内存地址，并统一编址。对于 Nativecode 的应用，由处理器内存方式决定，本规范不做强制规定。

卡片的 ADD 资源标识了应用可以访问的所有动态数据，这些数据掉电丢失，卡片不做保存。

ADD 资源除了应用访问的动态数据空间外，还包括系统调用入口参数和系统共享动态空间。其中，系统调用入口参数存储了一个激活的应用在 main 入口传入的参数数据。而系统共享动态空间则由卡商决定，本规范不做强制规定。

ASD（应用静态数据）资源除了标识应用可以访问的静态存储空间外，还包括应用代码段数据资源和应用可以访问的扩展 CRE 资源（如由卡商提交给应用的时钟模块，IO 总线模块等资源）。本规范不对扩展 CRE 资源做强制规定，由卡商自行决定提交给应用的扩展资源，并确定其名字空间。

表 21 列举了本规范强制规定的 ADD 和 ASD 资源列表。表 22 列举了本规范强制规定的卡片数据域资源。

表21 应用数据域的 ADD 和 ASD 资源

数据域组件	数据类型	名字空间
ADD	动态数据空间	“ram”
	系统调用入口参数	“lparam”
ASD	静态存储空间	“const”
	代码段资源	“code”

表22 卡片数据域资源

数据域组件	数据类型	名字空间
ASD	卡片系统服务标识（CSSI）	SCard:\Sys\Srv
	卡片管理密钥（CSMK）	SCard:\Sys\Key
	卡片静态数据目录	SCard:\Const
	卡片应用可执行模块目录	SCard:\App

打开一个 ASD 或者 ADD 资源的方法为 API 函数 open，本规范规定的系统 API 和 ACFS 的 API 在附录 A 中详细论述。

15.3 方法域模型

一个应用的方法域，应该包括该应用的入口参数、应用的执行体和提交给可执行模块的 API。

其中，应用的入口参数为由 CAM 提供的应用会话，其数据结构请参照 APDU 模型。

应用的执行体，由 CAM 从应用的可执行模块中提取，并实例化。CAM 规定，一个应用的执行体，应至少拥有一个入口（main）。其中 Nativecode 应用的入口为服务入口列表。而 Bytecode 应用的入口为 CVM 提供的函数入口。无论是哪种应用，一个标准的应用入口的 C 函数原型为：main(APDU＊pApdu, int Reason)（用 C 语法的描述）

其中，pApdu 为指向一个 APDU 模型的指针。而 Reason 指出了当前应用加载的阶段。

对于 Nativecode 应用，应用的方法域数据是事先烧录在卡里的。Nativecode 形态的可执行模块直接调用内核态的 API 函数来完成其功能。本规范对 Nativecode 的方法域不做任何限制，作为卡片的可信任组件，Nativecode 应用的方法域由卡商进行定制，并由应用检测机构负责其安全性。

本规范规定的应用 API 接口，见附录 B 的说明。

15.4 安全域模型

本规范规定，应用的安全域分为：应用 Native 安全域和托管安全域两种。与卡片安全域不同，应用安全域允许多个托管安全域同时关联到一个应用，并采取后进优先的顺序处理安全报文。

在一张标准卡片中，卡片的安全域是事先建立好的。而应用的初始安全域，也就是应用的 Native 安全域从 ASFW 中派生而来。

本规范规定，用户可以通过加载托管安全域的方法来自定义应用的安全属性。唯一的，卡片和应用都仅有一个托管安全域。

应用/卡片托管安全域被关联到应用/卡片后，用户不可以单独选择改托管安全域，用户可以直接选择应用/卡片，在一个应用/卡片被激活后，CAM 会优先加载托管安全域，并将其 AMF 安置在应用会话前端，也就是说托管安全域的 AMF 将优先处理应用会话，过滤完应用会话后，再将会话丢给该应用的 AMF 处理。

托管安全域通过负责过滤应用会话来实现应用子权限控制、应用补丁包等功能。本规范规范不对托管安全域的功能做任何规定，在基本的会话过滤功能的基础上，卡商可以根据需要开发任何应用的拓展安全功能。

托管安全域是一种特殊的应用，该应用不可以被单独选择，也不能被单独删除。在应用装载后必须关联到一个已经存在的应用，否则不会被激活。

无论是 Nativecode 应用，还是 Bytecode 应用都可以关联托管安全域。而一个托管安全域本身，它是一个 Bytecode 形态的应用。

托管安全域的创建过程和一般应用的创建过程一致。用户必须拥有该关联应用的访问权限才能将这个托管安全域关联到当前应用/卡片。

16 安全架构

16.1 概述

本规范对卡片的最主要的要求之一就是要有能力提供最起码的加密支持功能，能够为卡上的应用所使用。

发卡方安全域必须实现一种安全通道协议，能够安全地传递消息。

16.2 安全通信

本规范可以为卡片和卡外实体的信息交换提供对应的安全服务。与卡外实体的通信安全级别没有必要应用于每个单独的消息传输过程，但是可以用在进行内部消息传输的环境或上下文中。卡片生命周期的概念可用来确定卡片和卡外实体的通信安全级别。安全通信的加密算法应遵循国家密码管理局的相关规定。

16.3 安全的卡片内容管理

16.3.1 密文

更新安全框架时需要对数据进行加密，并验证数据的完整性。计算过程参见附录 C 中的指令详细说明。

16.3.2 MAC

装载可执行模块时需要附加 MAC 值，验证数据的完整性。计算过程参见附录 C 中的指令详细说明。

附 录 A
（规范性附录）
COS 文件系统的接口

A.1 COS 文件系统之 EF 文件访问方法

表 A.1 COS 文件系统之 EF 文件访问方法

EF 文件访问方法	方法描述
ef_create	在当前应用目录下，根据指定条件创建一个 EF 文件
ef_delete	在当前应用目录下，删除指定 EF 文件
ef_open	在当前应用目录下，打开指定 EF 文件，返回指向当前 EF 文件的句柄
ef_open_sfi	在当前应用目录下，根据 SFI 打开一个 EF 文件，返回指向当前 EF 文件的句柄
ef_first	在当前应用目录下，遍历第一个 EF 文件
ef_next	在当前应用目录下，遍历下一个 EF 文件
ef_sfi	在当前应用目录下，返回指定 EF 文件的 SFI
ef_type	在当前应用目录下，返回指定 EF 文件的文件类型
ef_set_info	在当前应用目录下，设置指定 EF 文件的文件头信息
ef_size	在当前应用目录下，返回指定 EF 文件的空间大小
ef_get_info	在当前应用目录下，获取指定 EF 文件的文件头信息
ef_read	在当前应用目录下，读指定的 EF 文件
ef_write	在当前应用目录下，写指定的 EF 文件
ef_record_create_linear	在当前应用目录下，创建一个线性记录文件
ef_record_create_cycle	在当前应用目录下，创建一个循环记录文件
ef_record_create_varilen	在当前应用目录下，创建一个变长记录文件
ef_record_add	在当前应用目录下，给一个指定的记录文件，添加记录
ef_record_get_from_tag	在当前应用目录下，对于指定的记录文件，根据 TAG 标志获取一条记录
ef_record_get_from_sn	在当前应用目录下，对于指定的记录文件，根据记录号获取一条记录
ef_record_linear_get_info	在当前应用目录下，获取指定线性记录文件的记录信息
ef_record_cycle_get_info	在当前应用目录下，获取指定循环记录文件的记录信息
ef_record_varilen_get_info	在当前应用目录下,，获取指定变长记录文件的记录信息
ef_record_first	在当前应用目录下，对于指定的一个记录文件，遍历第一条记录
ef_record_next	在当前应用目录下，对于指定的一个记录文件，遍历下一条记录

A.2 COS 文件系统之应用框架的访问方法

表 A.2 COS 文件系统之应用框架的访问方法

应用框架的访问方法	方法描述
df_space	获取当前应用数据域占用空间
df_set_user_info	设置当前应用框架中的个人化字段
df_get_user_info	读取当前应用框架中的个人化字段

附录 B
（规范性附录）
应用 API 接口

B.1 ADD 和 ASD 访问方法

表 B.1 ADD 和 ASD 访问方法

open	打开一个 ADD/ASD 对象，并返回其虚拟地址
size	返回一个 ADD/ASD 对象的空间大小，也就是 AI 中 ADDS/ASDS 的大小。
find	遍历当前应用可以访问的所有 ADD/ASD 对象，包括系统对象和扩展对象。

B.2 POP 方法

表 B.2 POP 方法

POP 方法	方法描述
enter_stom	进入 POP 周期
exit_stom	退出 POP 周期

B.3 ACFS 访问方法

见附录 A。

附录 C
（规范性附录）
CGS/CSS 接口协议（非安全域服务接口）

C.1 概述

CGS 属于 CRE 的内核程序，具有系统最高权限（Boot）。CSS 是预先写入卡片的一个 Native 应用，具有应用最高权限 System。

CGS 的主要服务如表 C.1 所示。CSS 的主要服务如表 C.2 所示。

表 C.1 CGS 服务列表

服务类别	服务描述
Create Native application framework	在当前任何目录下，创建 Native 应用的框架
Create Bytecode application framework	在当前任何目录下，创建 Bytecode 应用的框架
Attach a service to application	在当前任何目录下，关联一个服务到当前应用

表 C.2 CSS 的服务列表

Update Card Frame Work	更新卡片安全框架
Create Bytecode Application FrameWork	在当前目录下，创建 Bytecode 应用的安全框架
Load ApplicationSecure FrameWork	装载应用的安全框架
Load Application Executable Module	装载应用的可执行模块
Get Application Access Token	获取应用的访问令牌
Delete Application	删除应用
Promote Application Access Right	提升应用的访问权限
Card Frame Work Delete	卡片安全框架清除

下一个章节中将主要描述其服务的接口协议，本规范为卡片预先内置的 Native 服务接口协议，非安全域服务接口。如果用户已经关联了相应的卡片/应用托管安全域服务，请参考托管安全域服务提供商的接口说明。

C.2 CGS/CSS 接口协议

C.2.1 Update Card Frame Work 服务

服务描述：更新卡片的安全框架。此时，卡片的安全框架实现已经存在，但是用户需要通过更新卡片的安全框架，加载用户的安全属性，并将卡片激活。

本服务的实现由两条指令构成：

a） Prepare for LoadCardManagementKey

b） LoadCardManagementKey

指令的前后即为执行的先后顺序。指令格式如表 C.3 和 C.4 所示。

表 C.3 Prepare for LoadCardManagementKey 的指令格式

Prepare for LoadCardManagementKey	CLA	0x00
	INS	0xb2
	P1	0x01
	P2	0x00
	LC	0x04
	RESP	Random[4Bytes]
	SW	0x9000

表 C.3 Prepare for LoadCardManagementKey 的指令格式（续）

运行环境	必须在 CSS 环境下，用户必须首先选择 Aid 为"SCard:\\Sys\\Srv"的应用。
运行权限	Boot
功能说明	本指令完成更新卡片安全框架的初始化工作。
参数说明	Random：4 字节随机数。

表 C.4 LoadCardManagementKey 的指令格式

LoadCardManagementKey	CLA	0x00
	INS	0xb2
	P1	0x02
	P2	ALG
	LC	DATA 数据长度
	DATA	[Keylen， CardManagementKey（密文），MAC]
	SW	0x9000
运行环境	必须在 CSS 环境下，用户必须首先选择 Aid 为"SCard:\\Sys\\Srv"的应用。	
运行权限	Boot	
功能说明	本指令完成更新卡片安全框架的最终工作。用于最终更新卡片安全域管理密钥，此操作用来激活卡片。 默认的卡管理密钥："SCard:\\Sys\\Key" 为 16 个字节	
参数说明	Alg： 算法标识，如果 Alg==0，采用 3DES 算法，如果 Alg==1，采用 SM4 算法。 Keylen ：卡片安全域管理密钥的长度。 CardManagementKey（密文）：卡片安全域管理密钥，此处为密文。 计算方法： CardManagementKey（密文）=3des/sm4(CardManagementKey（当前），[Keylen，Key])。根据 Alg 标识，用当前卡片安全域管理密钥，对[Keylen，Key]数据进行加密，最后得到卡片安全域管理密钥的密文。 加密算法说明： 如果是 3des 算法，[KeyLen，Key]补 80,00 补足 8 的整数倍长度。 如果是 sm4 算法，[KeyLen，Key]补 80,00 补足 16 的整数倍长度。 Mac：本次安全报文的 Mac 值。 计算方法： Mac= mac_cbc(Alg，CardManagementKey（当前）， Random，mac 前的所有报文数据)。 Mac 的计算方法为用 mac_cbc 算法，采用当前卡片安全域管理密钥(补充 8000 到 16 字节)，以上一步中的 Random（后 4 字节补充 0，补足 8 字节）为初值，对 Mac 前所有报文数据计算 Mac 值。 Mac_cbc 计算说明： 如果是 3des 算法，Mac 数据以 8 字节为步长计算 Mac。 如果是 sm4 算法，Mac 数据以 16 字节为步长计算 Mac。	

C.2.2 Create bytecode application framework

服务描述：创建一个 bytecode 应用的框架。
指令格式如表 C.5 所示。

表 C.5 Create bytecode application framework 的指令格式

Create bytecode application framework	CLA	0x00
	INS	0xb1
	P1	0x01
	P2	0x00
	LC	DATA 数据长度
	DATA	[Fid(2Bytes)，AidLen(1 Byte)，AID（ AidLen Bytes)，ExeNameLen(1 Byte)，ExeName（ ExeNameLen Bytes)，UserArray]
	SW	0x9000

表 C.5 Create bytecode application framework 的指令格式（续）

运行环境	在 CGS 环境下，也就是说用户可以在任何应用环境下采用此方法。而用此方法建立的应用也将位于当前应用目录下。
运行权限	APP
功能说明	本指令完成应用框架简历的初始化工作和应用目录的隶属关系，不对 CSTO 产生任何影响，在应用方法域加载之前，所有框架掉电后消失。 该指令的使用需要卡片安全框架更新后才可以使用。
参数说明	Fid：2 字节的 COS 文件标识符，Fid。如果当前 CMAP 系统包含 COS 文件系统，此 Fid 参数被强制使用。否则，该参数被忽略。用户可以写成任意值。 AidLen：正在被创建的 bytecode 应用的 Aid 长度，其值限定在 1~16。 Aid：正在被创建的 bytecode 应用的 Aid，长度为 AidLen 标识的长度。 ExeNameLen：正在被创建的 bytecode 应用的可执行模块名长度。 ExeName：正在被创建的 bytecode 应用的可执行模块名，长度 为 ExeNameLen 中指定的长度，其标识的可执行模 块用于应用被加载后生成方法域实例。 UserArray：这个参数数组用于在应用的安装注册期间，传入应用的方法域，由方法域初始化代码处理一些应用的初始化操作使用。参数数组长度= LC 减其前面所有数据单元的长度和。

C.2.3 Load ApplicationSecure FrameWork

服务描述：装载一个应用的安全框架
指令格式如表 C.6 所示。

表 C.6 Load Application Secure FrameWork 的指令格式

Load ApplicationSecure FrameWork	CLA	0x00
	INS	0xb1
	P1	0x03
	P2	Alg
	LC	DATA 数据长度
	DATA（密文）	[AidLen(1 Byte),Aid(AidLen Bytes),IsAttach(1 Byte),Keylen（1Byte), ApplicationManagementKey (Keylen Bytes)]
	SW	0x9000
运行环境	必须在 CSS 环境下，用户必须首先选择 Aid 为"SCard:\\Sys\\Srv"的应用。	
运行权限	App	
功能说明	本指令装载应用的安全框架，不对 CSTO 产生任何影响，在应用方法域加载之前，所有框架掉电后消失。 该指令的使用需要卡片安全框架更新后才可以使用。	
参数说明	Alg： 算法标识，如果 Alg==0，采用 3DES 算法，如果 Alg==1，采用 SM4 算法。 AidLen：正在被创建的 bytecode 应用的 Aid 长度，其值限定在 1~16。 Aid：正在被创建的应用的 Aid，长度为 AidLen 标识的长度。 Keylen：应用安全域管理密钥的长度。 ApplicationManagementKey：应用安全域管理密钥，其长度为 Keylen 所标识的长度。 DATA（密文）的计算方法： DATA（密文）=3des/sm4(ApplicationSecureLoadKey, [AidLen，Aid，IsAttach，Keylen，ApplicationManagementKey])。根据 Alg 标识，用应用安全装载密钥，对[AidLen，Aid，IsAttach，Keylen， ApplicationManagementKey]数据进行加密，最后得到 DATA 密文。 加密算法说明： 如果是 3des 算法，[KeyLen，Key]补 80,00 补足 8 的整数倍长度。 如果是 sm4 算法，[KeyLen，Key]补 80,00 补足 16 的整数倍长度。 ApplicationSecureLoadKey 计算说明： ApplicationSecureLoadKey 为应用安全装载密钥，卡片管理密钥建立后，用最新的卡片管理密钥，对应用的 AID 补 8000 到 16 字节做 3DES 运算，密文即为应用安全装载密钥（ApplicationSecureLoadKey）AppLoadKey。	

C.2.4 Load Application Executable Module

服务描述：装载一个 Bytecode 应用的可执行模块。
本服务的实现由三条指令构成：
Prepare for module download
Module download
Module download terminated
指令的前后即为执行的先后顺序。
指令格式分别如表 C.7 到表 C.9 所示。

表 C.7 Prepare for module download 的指令格式

Prepare for module download	CLA	0x00
	INS	0xb1
	P1	0x05
	P2	Alg
	LC	DATA 数据长度
	DATA	[PackageNo（2Bytes），Aid]
	RSEP	[PackageSize（2Bytes），Random[4Bytes]，Mac（4Bytes）]
运行环境	必须在 CSS 环境下，用户必须首先选择 Aid 为"SCard:\\Sys\\Srv"的应用。	
运行权限	App	
功能说明	本指令完成装载可执行模块的初始化工作。	
参数说明	Alg： 算法标识，如果 Alg==0，采用 3DES 算法，如果 Alg==1，采用 SM4 算法。 Packageno：数据包序号，此处为 0。 Aid： 正在被创建的 bytecode 应用的 Aid，长度为：Lc-2。 PackageSize： 卡片返回的能够接收的最大数据包大小。 Random： 4 字节的随机数。 Mac：响应报文的 Mac 值。 Mac= mac_cbc(Alg，Seskey，0，[Aid，PackageNo])。 Mac 的计算方法为用 mac_cbc 算法，采用本次会话的过程密钥（Seskey），以 0 为初值，对[Aid，PackageNo]数据计算 Mac 值。 Mac_cbc 计算说明： 如果是 3des 算法，Mac 数据以 8 字节为步长计算 Mac。 如果是 sm4 算法，Mac 数据以 16 字节为步长计算 Mac。 Seskey（过程密钥，8/16 字节）的计算说明： Seskey=3des/sm4(ApplicationManagementKey（补 8000 到 16 字节），[Random, Package size])； 过程密钥的计算方法为，用当前的应用安全域管理密钥，对[Random, Package size]补 8000 到 8/16 字节数据做加密，密文（8/16 字节）即为过程密钥。其中如果是 3des 算法，数据为 8 字节，密文为 8 字节。如果是 sm4 算法，数据为 16 字节，密文为 16 字节。	

表 C.8 Module download 的指令格式

Module download	CLA	0x00
	INS	0xb1
	P1	0x06
	P2	Alg
	LC	DATA 数据长度
	DATA	[PackageNo（2Bytes），ApplicationCode，Mac1（4Bytes）]
	RSEP	Mac2（4Bytes）
运行环境	必须在 CSS 环境下，用户必须首先选择 Aid 为"SCard:\\Sys\\Srv"的应用。	
运行权限	App	
功能说明	本指令完成装载可执行模块数据包的下载工作。	

表 C.8 Module download 的指令格式（续）

参数说明	Alg:　　算法标识，如果 Alg==0，采用 3DES 算法，如果 Alg==1，采用 SM4 算法。 Packageno：数据包序号。 ApplicationCode ：可执行模块的下载数据，长度为：LC-6。 Mac1：发送报文的 Mac 值。 Mac2：响应报文的 Mac 值。 Seskey(过程密钥)：为 Prepare for module download 指令中计算的过程密钥。 Mac1 计算方法： Mac1=mac_cbc(Seskey,　0，[PackageNo，ApplicationCode])； Mac1 的计算方法为用 mac_cbc 算法,采用本次会话的过程密钥(Seskey),以 0 为初值,对[PackageNo（2Bytes），ApplicationCode]数据计算 Mac 值。 Mac2 计算方法： Mac2=mac_cbc(Seskey,　0，[Aid，Packageno ，　Applicationcode ，Applicationcodesize])； Mac2 的计算方法为用 mac_cbc 算法，采用本次会话的过程密钥（Seskey），以 0 为初值，对[AID，Packageno ，　Applicationcode ，Applicationcodesize]数据计算 Mac 值。 其中：Aid 为 Prepare for module download 指令中的 Aid Applicationcodesize 为实际保存的数据包长度。 Mac_cbc 计算说明： 如果是 3des 算法，Mac 数据以 8 字节为步长计算 Mac。 如果是 sm4 算法，Mac 数据以 16 字节为步长计算 Mac。

表 C.9 Module download terminated 的指令格式

Module download terminated	CLA	0x00
	INS	0xb1
	P1	0x07
	P2	Alg
	LC	DATA 数据长度
	DATA	[PackageNo（2Bytes），ApplicationCode，Mac1（4Bytes）]
	RSEP	Mac2（4Bytes）
运行环境	必须在 CSS 环境下，用户必须首先选择 Aid 为"SCard:\\Sys\\Srv"的应用。	
运行权限	App	
功能说明	本指令结束装载可执行模块的下载工作。	
参数说明	Alg:　　算法标识，如果 Alg==0，采用 3DES 算法，如果 Alg==1，采用 SM4 算法。 Packageno：数据包序号。 ApplicationCode ：最后一包数据，长度为：LC-6。 Mac1：发送报文的 Mac 值。 Mac2：响应报文的 Mac 值。 Seskey(过程密钥)：为 Prepare for module download 指令中计算的过程密钥。 Mac1 计算方法： Mac1=mac_cbc(Seskey,　0，[PackageNo，ApplicationCode])； Mac1 的计算方法为用 mac_cbc 算法，采用本次会话的过程密钥（Seskey），以 0 为初值，对[PackageNo（2Bytes），ApplicationCode]数据计算 Mac 值。 Mac2 计算方法： Mac2=mac_cbc(Seskey,　0，[Aid，Packageno ，　Applicationcode ，Applicationcodesize])；	

表 C.9 Module download terminated 的指令格式（续）

参数说明	Mac2 的计算方法为用 mac_cbc 算法，采用本次会话的过程密钥（Seskey），以 0 为初值，对[AID，Packageno ， Applicationcode ，Applicationcodesize]数据计算 Mac 值。 其中：Aid 为 Prepare for module download 指令中的 Aid。 Applicationcodesize 为实际保存的数据包长度。 Mac_cbc 计算说明： 如果是 3des 算法，Mac 数据以 8 字节为步长计算 Mac。 如果是 sm4 算法，Mac 数据以 16 字节为步长计算 Mac。

C.2.5 Get Application Access Token

服务描述：获取一个应用的访问令牌。
指令格式如表 C.10 所示。

表 C.10 Get Application Access Token 的指令格式

Get Application Access Token	CLA	0x00
	INS	0xb1
	P1	0x08
	P2	Alg
	LC	DATA 数据长度
	DATA	[AidLen（2Bytes），Aid]
	RSEP	[Random（4Bytes），Mac（4Bytes）]
运行环境	必须在 CSS 环境下，用户必须首先选择 Aid 为"SCard:\\Sys\\Srv"的应用。	
运行权限	App	
功能说明	本指令获取一个应用的访问令牌。	
参数说明	Alg： 算法标识，如果 Alg==0，采用 3DES 算法，如果 Alg==1，采用 SM4 算法。 Random：4 字节随机数。 AidLen：要访问的应用的 Aid 长度，其值限定在 1~16。 Aid：要访问的应用的 Aid，长度为 AidLen 标识的长度。 Mac：发送报文的 Mac 值。 Mac 计算方法： Mac=mac_cbc(Seskey， 0，Aid)； Mac 的计算方法为用 mac_cbc 算法，采用本次会话的过程密钥（Seskey），以 0 为初值，对 Aid 数据计算 Mac 值。 Seskey（过程密钥，8/16 字节）的计算说明： Seskey=3des/sm4(ApplicationManagementKey（补 8000 到 16 字节)，[Random，AidLen])； 过程密钥的计算方法为，用当前的应用安全域管理密钥(补 8000 到 16 字节)，对[Random，AidLen]补 8000 到 8/16 字节数据做加密，密文（8/16 字节）即为过程密钥。其中如果是 3des 算法，数据为 8 字节，密文为 8 字节。如果是 sm4 算法，数据为 16 字节，密文为 16 字节。 Mac_cbc 计算说明： 如果是 3des 算法，Mac 数据以 8 字节为步长计算 Mac。 如果是 sm4 算法，Mac 数据以 16 字节为步长计算 Mac。	

C.2.6 Delete Application

服务描述：删除一个应用的框架和所有数据。
指令格式如表 C.11 所示。

表 C.11 Delete Application 的指令格式

Delete Application	CLA	0x00
	INS	0xb1
	P1	0x0a
	P2	Alg
	LC	DATA 数据长度
	DATA	Mac（4Bytes）
	SW	0x9000
运行环境	必须在 CSS 环境下，用户必须首先选择 Aid 为"SCard:\\Sys\\Srv"的应用。	
运行权限	App	
功能说明	本指令删除一个应用的框架和所有数据。	
参数说明	Alg：　　算法标识，如果 Alg==0，采用 3DES 算法，如果 Alg==1，采用 SM4 算法。 Random：4 字节随机数。 AidLen：要访问的应用的 Aid 长度，其值限定在 1~16。 Aid：要访问的应用的 Aid，长度为 AidLen 标识的长度。 Mac：发送报文的 Mac 值。 Mac 计算方法： Mac=mac_cbc(Seskey,　0，Aid)； Mac 的计算方法为用 mac_cbc 算法，采用本次会话的过程密钥（Seskey），以 0 为初值，对 Aid 数据计算 Mac 值。 Seskey（过程密钥）： Get Application Access Token 中计算的过程密钥。 Mac_cbc 计算说明： 如果是 3des 算法，Mac 数据以 8 字节为步长计算 Mac。 如果是 sm4 算法，Mac 数据以 16 字节为步长计算 Mac。	

C.2.7　Promote Application Access Right

服务描述：提升一个应用的权限到 System。
指令格式如表 C.12 所示。

表 C.12 Promote Application Access Right 的指令格式

Promote Application Access Right	CLA	0x00
	INS	0xb1
	P1	0x0b
	P2	Alg
	LC	DATA 数据长度
	DATA	[Aid，Mac(4Bytes)]
	SW	0x9000
运行环境	必须在 CSS 环境下，用户必须首先选择 Aid 为"SCard:\\Sys\\Srv"的应用。	
运行权限	Boot	
功能说明	本指令提升一个应用的权限到 System 。	
参数说明	Alg：　　算法标识，如果 Alg==0，采用 3DES 算法，如果 Alg==1，采用 SM4 算法。 Aid：要访问的应用的 Aid，长度为 Lc-4。 Mac：发送报文的 Mac 值。 Mac 计算方法： Mac=mac_cbc(Seskey,　0，Mac 前所有报文数据)； Mac 的计算方法为用 mac_cbc 算法，采用本次会话的过程密钥（Seskey），以 0 为初值，对 Aid 数据计算 Mac 值。	

表 C.12 Promote Application Access Right 的指令格式（续）

参数说明	Seskey（过程密钥）： Get Application Access Token 中计算的过程密钥。 Mac_cbc 计算说明： 如果是 3des 算法，Mac 数据以 8 字节为步长计算 Mac。 如果是 sm4 算法，Mac 数据以 16 字节为步长计算 Mac。

C.2.8 Create nativecode application framework

服务描述：创建一个 nativecode 应用的框架。
指令格式如表 C.13 所示。

表 C.13 Create nativecode application framework 的指令格式

Create nativecode application framework	CLA	0x00
	INS	0xe0
	P1	0x00
	P2	0x00
	LC	DATA 数据长度
	DATA	[Fid(2Bytes), AidLen(1 Byte), AID(AidLen Bytes), UserArray]
	SW	0x9000
运行环境	在 CGS 环境下，也就是说用户可以在任何应用环境下采用此方法。而用此方法建立的应用也将位于当前应用目录下。	
运行权限	App	
功能说明	本指令完成 Nativecode 应用框架建立的初始化工作和应用目录的隶属关系，改指令影响 CSTO，并在 CSTO 上建立应用的 COS 文件包。 该指令的使用需要应用安全域已经固化在卡片中，且卡片安全框架更新后才可以使用。	
参数说明	Fid：2 字节的 COS 文件标识符，Fid。如果当前 CMAP 系统包含 COS 文件系统，此 Fid 参数被强制使用。否则，该参数被忽略。用户可以写成任意值。 AidLen：正在被创建的 bytecode 应用的 Aid 长度，其值限定在 1~16。 Aid：正在被创建的 bytecode 应用的 Aid，长度为 AidLen 标识的长度。 UserArray：这个参数数组用于在应用的安装注册期间，传入应用的方法域，由方法域初始化代码处理一些应用的初始化操作使用。参数数组长度= LC 减其前面所有数据单元的长度和。	

C.2.9 Attach a bytecode application to a application secure frame work

服务描述：关联一个应用到另外一个应用的安全框架，本应用将成为该应用的托管安全域服务。
指令格式如表 C.14 所示。

表 C.14 Attach a bytecode application to an application secure frame work 的指令格式

Attach a bytecode application to a application secure frame work	CLA	0x00
	INS	0xb1
	P1	0x0c
	P2	Alg
	LC	DATA 数据长度
	DATA	[Aid, Mac1(4Bytes), Mac2（4Bytes）]
	SW	0x9000
运行环境	必须在 CSS 环境下，用户必须首先选择 Aid 为"SCard:\\Sys\\Srv"的应用。	
运行权限	App/ System	
功能说明	本指令关联一个应用到另外一个应用的安全框架，本应用将成为该应用的托管安全域服务。	
参数说明	Alg： 算法标识，如果 Alg==0，采用 3DES 算法，如果 Alg==1，采用 SM4 算法。 Aid： 源应用的 Aid，长度为 Lc-8。 Mac1：目标应用的安全 Mac。 Mac2：源应用的安全 Mac。	

表 C.14 Attach a bytecode application to an application secure frame work 的指令格式（续）

参数说明	Mac2 计算方法： Mac2=mac_cbc(Seskey2, 0，Mac2 前所有报文数据)； Mac2 的计算方法为用 mac_cbc 算法，采用本次会话的过程密钥（Seskey2），以 0 为初值，对 Mac2 前所有报文数据计算 Mac 值。 Seskey2（过程密钥）： Get Application Access Token 中计算的过程密钥。 Mac_cbc 计算说明： 如果是 3des 算法，Mac 数据以 8 字节为步长计算 Mac。 如果是 sm4 算法，Mac 数据以 16 字节为步长计算 Mac。 Mac1 计算方法： Mac1=mac_cbc(Seskey1, 0，Mac1 前所有报文数据)； Mac1 的计算方法为用 mac_cbc 算法，采用本次会话的过程密钥（Seskey1），以 0 为初值，对 Mac1 前所有报文数据计算 Mac 值。 Seskey1（过程密钥，8/16 字节）的计算说明： Seskey1=3des/sm4(ApplicationManagementKey（源应用）(补 8000 到 16 字节)，[Aid（目标应用），Aid（源应用）])； 过程密钥的计算方法为，用源应用安全域管理密钥(补 8000 到 16 字节)，对[Aid（目标应用），Aid（源应用）]补 8000 到 8/16 字节数据做加密，密文（8/16 字节）即为过程密钥。其中如果是 3des 算法，数据为 8 字节，密文为 8 字节。如果是 sm4 算法，数据为 16 字节，密文为 16 字节。

附录 D
（规范性附录）
CMAP 的接口

D.1 概述

CMAP 的接口包括，内部接口和外部接口。

其中，CMAP 的内部接口为 CMAP 应用的方法域调用的接口，这些接口为 CMAP 平台的应用开发者提供了技术支持。CMAP 的外部接口为 CGS 和 CSS 服务暴露的接口以及 CGS 和 CSS 的托管方（卡片托管安全域提供者）暴露的第三方接口。本章节描述的是为 CGS 和 CSS 服务暴露的接口，CGS 和 CSS 的托管方接口请参见托管服务提供商的技术说明。

D.2 CMAP 的内部接口

CMAP 的接口，包括：应用入口、应用 API 和应用出口。

CMAP 不对应用的入口方式做强制规定，但是要求应用入口的接口要能够完整表示 CCOM 中提到 APDU 模型。所以也可以用如下方式抽象表示 CMAP 的入口：CMAP_Main_Entry（APDU）。

CMAP 规范了应用的 API，如表 D.1 到表 D.7 所示：

表 D.1 ADD 和 ASD 访问方法

ADD 和 ASD 访问方法	方法描述
open	打开一个 ADD/ASD 对象，并返回其虚拟地址
size	返回一个 ADD/ASD 对象的空间大小，也就是 AI 中 ADDS/ASDS 的大小。
find	遍历当前应用可以访问的所有 ADD/ASD 对象，包括系统对象和扩展对象。

表 D.2 GLB 方法

接口	访问权限	功能描述
receive_left_bytes	App	T=0 协议下，应用获取剩余字节

表 D.3 FS(文件系统)的方法

文件系统的访问方法	访问权限	方法描述
fs_create	System	根据名字空间，创建一个文件
fs_open	System	根据名字空间，打开一个文件
fs_delete	System	删除一个指定文件句柄的文件
fs_size	System	返回一个指定文件句柄的文件的容量
fs_first	System	在当前文件目录下，遍历第一个文件
fs_next	System	在当前文件目录下，遍历下一个文件
fs_read	System	读一个指定文件句柄的文件
fs_write	System	写一个指定文件句柄的文件

表 D.4 CFS(COS 文件系统（如果存在）)的方法

EF 文件访问方法	访问权限	方法描述
ef_create	App	在当前应用目录下，根据指定条件创建一个 EF 文件
ef_delete	App	在当前应用目录下，删除指定 EF 文件

表 D.4 CFS(COS 文件系统（如果存在）)的方法（续）

ef_open	App	在当前应用目录下，打开指定 EF 文件，返回指向当前 EF 文件的句柄
ef_open_sfi	App	在当前应用目录下，根据 SFI 打开一个 EF 文件，返回指向当前 EF 文件的句柄
ef_first	App	在当前应用目录下，遍历第一个 EF 文件
ef_next	App	在当前应用目录下，遍历下一个 EF 文件
ef_sfi	App	在当前应用目录下，返回指定 EF 文件的 SFI
ef_type	App	在当前应用目录下，返回指定 EF 文件的文件类型
ef_set_info	App	在当前应用目录下，设置指定 EF 文件的文件头信息
ef_size	App	在当前应用目录下，返回指定 EF 文件的空间大小
ef_get_info	App	在当前应用目录下，获取指定 EF 文件的文件头信息
ef_read	App	在当前应用目录下，读指定的 EF 文件
ef_write	App	在当前应用目录下，写指定的 EF 文件
ef_record_create_linear	App	在当前应用目录下，创建一个线性记录文件
ef_record_create_cycle	App	在当前应用目录下，创建一个循环记录文件
ef_record_create_varilen	App	在当前应用目录下，创建一个变长记录文件
ef_record_add	App	在当前应用目录下，给一个指定的记录文件，添加记录
ef_record_get_from_tag	App	在当前应用目录下，对于指定的记录文件，根据 TAG 标志获取一条记录
ef_record_get_from_sn	App	在当前应用目录下，对于指定的记录文件，根据记录号获取一条记录
ef_record_linear_get_info	App	在当前应用目录下，获取指定线性记录文件的记录信息
ef_record_cycle_get_info	App	在当前应用目录下，获取指定循环记录文件的记录信息
ef_record_varilen_get_info	App	在当前应用目录下,，获取指定变长记录文件的记录信息
ef_record_first	App	在当前应用目录下，对于指定的一个记录文件，遍历第一条记录
ef_record_next	App	在当前应用目录下，对于指定的一个记录文件，遍历下一条记录

表 D.5 AFW 的访问方法

应用框架的访问方法	访问权限	方法描述
df_space	App	获取当前应用数据域占用空间
df_set_user_info	App	设置当前应用框架中的个人化字段
df_get_user_info	App	读取当前应用框架中的个人化字段

表 D.6 CFW 的访问方法

卡片框架的访问方法	访问权限	方法描述
card_block	System	锁定卡片
card_is_block	System	获取卡片锁定的状态

表 D.7 POP 方法

POP 方法		方法描述
enter_stom	App	进入 POP 周期
exit_stom	App	退出 POP 周期

除以上函数之外的其他 API 函数参见卡商的说明。

CMAP 不对应用的出口做强制规定，卡商只要能够提供给开发者返回 APDU 模型的方法即可。

D.3 CMAP 的外部接口

CMAP 对外的接口如表 D.8 所示。

表 D.8 CGS/CSS 提供的接口

接口	接口描述	方法（指令的先后顺序）	关联的安全域	安全域管理密钥
Update Card Secure Frame Work	更新卡片安全框架	1Step：Select CSS 2 Step：Prepare for LoadCardManagementKey 3 Step：LoadCardManagementKey	卡片安全域	卡片安全域管理密钥
Create Bytecode Application Frame Work	创建 Bytecode 应用框架	1 Step：Create Bytecode Application Frame Work 2 Step：Select CSS 3 Step：Load Application Frame Work 4 Step：Prepare for module download 5 Step：Module download 6 Step：Module download terminated	应用安全域	应用安全域管理密钥
Create nativecode application framework	创建 Nativecode 应用框架	1 Step：Create Bytecode Application Frame Work	应用安全域	应用安全域管理密钥
Update Application Executeable Module	更新应用的可执行模块	1 Step：Select CSS 2 Step：Prepare for module download 3 Step：Module download 4 Step：Module download terminated	应用安全域	应用安全域管理密钥
Delete Application	删除应用	1 Step：Select CSS 2 Step：Get Application Access Token 3 Step：Delete Application	应用安全域	应用安全域管理密钥
Promote Application Access Right	提升应用的访问权限	1 Step：Select CSS 2 Step：Get Application Access Token 3 Step：Promote Application Access Right	卡片安全域	卡片安全域管理密钥
Attach a bytecode application to a application secure frame work	关联一个应用到另外一个应用的安全框架	1 Step：Select CSS 2 Step：Get Application Access Token 3 Step：Attach a bytecode application to a application secure frame work	应用安全域	应用安全域管理密钥

参考文献

[1] GlobalPlatform 卡片规范 v2.2
[2] ISO/IEC 7816-3：1997 识别卡-接触 IC 卡第 3 部分：电气信号和传输协议（1997）
[3] ISO/IEC 7816-6：2004 识别卡-接触 IC 卡第 6 部分：行业间的数据元素（2004）
[4] ISO/IEC 7816-15：2004 识别卡-接触 IC 卡第 15 部分：加密信息应用（2004）
[5] ISO/IEC 10116：1997 信息技术-n 位块加密算法的操作方式
[6] ISO/IEC 14443-3:2001 识别卡-非接触 IC 卡-感应卡-第 3 部分：初始化和防碰撞
[7] ISO/IEC 14443-4:2001 识别卡-非接触 IC 卡-感应卡-第 4 部分：传输协议