1 特性

1.1 RF 接口：ISO 14443 类型 A

- 通过 RF 场实现数据非接触式传输和供电（无需电池）
- 操作距离：最大 100mm（取决于天线的形状）
- 操作频率：13.56MHz
- 快速数据传输：106kbit/s，424kbit/s
- 高数据完整性：4 字节 MAC、16 位 CRC、奇偶校验、位编码、位计数
- 真实可确定的防冲突性
- 7 字节的唯一标识符（符合 ISO 14443-3 的级联第二层）
- 使用 ISO 14443-4 传输协议

1.2 非易失性（NV）存储器

- 4k 字节 NV-存储器
- NV-存储器写时间 2ms（1ms 擦除，1ms 编程）
- 数据可保存 10 年
- 可写 100, 000 次

1.3 NV-存储器组织

- 灵活的文件系统
- 一个 PICC 最多可同时存在 28 个应用
- 每个应用程序多达 16 个文件

1.4 安全性

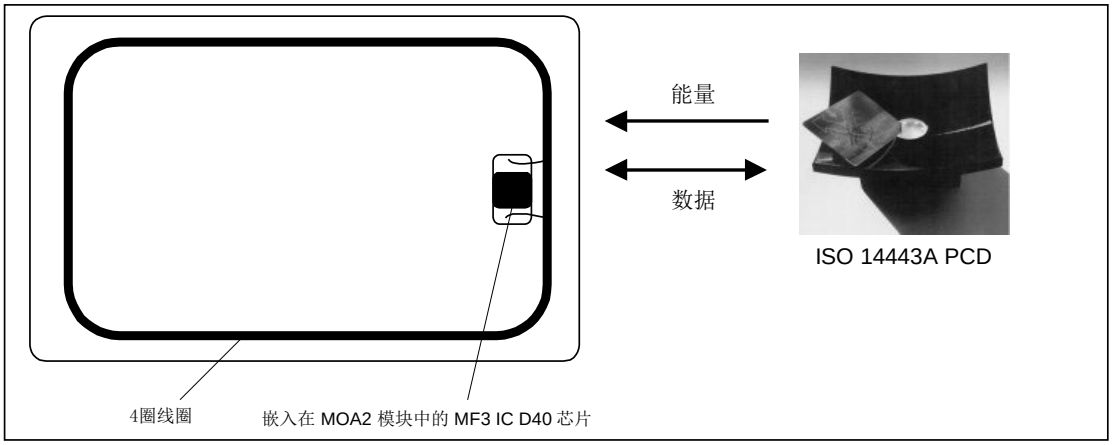
- 每个器件有唯一的 7 字节序列号
- 交互的 3 段验证
- RF 信道的硬件 DES/3DES 数据加密，可防止重放攻击。
- 通过 4 字节 MAC 进行数据验证
- 应用级验证

2 概述

PHILIPS 开发了 MIFARE DESFire (MF3 IC D40)可用于符合 ISO 14443 类型 A 的近耦合设备（PCD）。传输协议遵循 ISO 14443-4 部分。MF3 IC D40 主要是为安全的非接触式传输应用和相关的 loyalty 程序而设计的。

2.1 非接触式能量和数据传输

在 MIFARE 系统中，MF3 IC D40 与一个嵌入在标准 ISO 智能卡中的线圈相连接。不需要电池。当卡接近 PCD 天线时，高速通信接口可实现高达 424kbit/s 的数据传输。



2.2 交付类型

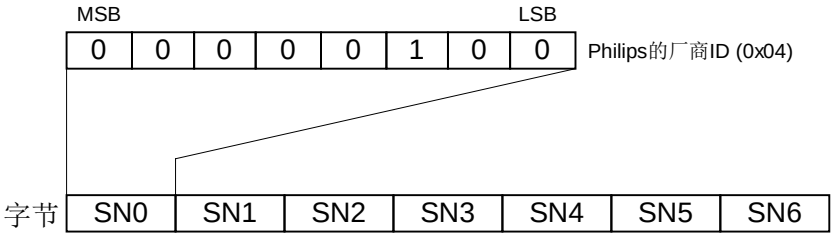
- 8”圆晶上的管芯，在 FFC 上切割，厚度 150um
- 8”圆晶上带金凸点的管芯，在 FFC 上切割，厚度 150um
- MOA2 非接触式芯片卡模块

2.3 防冲突

智能的防冲突机制允许同时处理多个 PICC。防冲突算法单独选择每个 PICC 并确保与选择的 PICC 正确执行处理，不会因为场中的其它 PICC 而导致数据被破坏。

2.4 UID/序列号

唯一的 7 字节序列号（UID）编程到厂商保留的一个 NV 存储器锁定部分。由于安全性和系统的要求，这些字节在生产线编程之后处于写保护状态。



7 字节的唯一序列号

根据 ISO14443-3，第一个防冲突循环（见 4.1.3）将返回级联标签 0x88 和 UID 的前 3 个字节、SN0~SN2 和 BCC。第二个防冲突循环（见 4.1.4）将返回字节 SN3~SN6 和 BCC。

根据 ISO14443-3 和 ISO 7816-6 AM1，SN0 保存厂商 ID，PHILIPS 为 04h。

2.5 存储器组织

4k 字节 NV 存储器通过灵活的文件系统来组织。文件系统允许单个 PICC 上存在最多 28 个不同的应用。每个应用最多提供 16 个文件。每个应用由它的 3 字节应用标识符（AID）表示。

支持 5 种不同的文件类型，见 3.1。

每个文件可在 PICC 初始化时（卡的生产/印刷）或 PICC 终端（自动售货机）或场中创建。

如果文件或应用在操作中被废弃，可使它永久无效。

对文件结构本身产生影响的命令（例如，建立或删除一个应用，改变密钥）会启动一个自动的恢复（rollback）机制，它可保护文件结构不会被破坏。

如果需要使用恢复机制，它在下一个命令执行之前完成，不需要用户的交互操作。

为了确保应用级的数据完整性，所有 5 个带备份的文件类型都实现了面向处理的备份。在一个应用中，可以混合带备份和不带备份的文件类型，因此备份只用于文件 0~7，而文件 8~15 不支持备份机制。

2.6 安全性

7 字节 UID 在生产线编程之后就不能再进行更改，这样就确保了每个器件的唯一性。

UID 可用于为每个卡生成多种密钥。多种 PICC 密钥可用于实现有效的防冲突机制。

在数据发送之前，PICC 和 PCD 之间可完成 3 路验证，这取决于配置使用 DES 还是 3DES。

PICC 和 PCD 之间的数据传输可在 3 层安全性下完成：

- 直接的数据传输
- 带 DES/3DES 加密校验和（MAC）的直接数据传输
- DES/3DES 加密的数据传输（在加密之前进行 CRC 校验）

应用层允许对用户数据进行访问。每个应用最多有 14 个不同的用户密钥用于控制对 PICC 中保存的数据进行访问。

3 MF3 IC D40 – 安全编码、应用和文件相关特性

3.1 文件类型编码

应用中的文件可具有下列不同的类型：

- 标准数据文件 （编码为 0x00）
- 备份数据文件 （编码为 0x01）
- 带备份的值文件 （编码为 0x02）
- 带备份的线性记录文件 （编码为 0x03）
- 带备份的循环记录文件 （编码为 0x04）

3.2 通信设定的编码－加密等级

通信设定定义了 PCD 与 PICC 之间通信的安全等级。通信设定总是适用于文件级。

设定编码为 1 个字节，可设定为：

通信模式	**bit7-bit2**	**bit1**	**bit0**
明码通信	RFU=0	忽略	0
DES/3DES MAC 所确保的明码通信	RFU=0	0	1
完全 DES/3DES 加密的通信	RFU=0	1	1

DES 和 3DES 密钥都保存在 16 个字节的字符串中：

如果密钥串的后半部分等于前半部分，密钥作为单个 DES 密钥进行处理。

如果密钥串的后半部分不等于前半部分，密钥作为 1 个 3DES 密钥进行处理。

所有基于密钥的操作（验证，MACing，加密）分别使用 DES 或 3DES 方法进行进一步处理。

单个DES密钥的例子： 0x00 11 22 33 44 55 66 77 00 11 22 33 44 55 66 77

0x00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *

3DES密钥的例子： 0x00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF

0x00 11 22 33 44 55 66 77 00 00 00 00 00 00 01 00

0x00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00

* 所有 0x00 字节都是 MF3 IC D40 的默认密钥，而单个 DES 操作定义为默认操作。

3.3 访问权限编码

应用中的每个文件都有 4 种不同的访问权限（每个文件 2 字节）：

－读访问

－写访问

－读&写访问

－改变访问权限

每种访问权限都编码为 4 位，半字节。每个半字节代表与对应的应用密钥文件中的一个密钥的连接。

半字节可实现 16 个不同值的编码。如果设定为 0~13 之间的数(最多 14 个密钥)，它对应于应用的密钥文件中一个特定的密钥，前提是该密钥存在（不允许配置一个不存在的密钥）。

如果编码为 14（0xE），则表示“自由的”访问。在选择应用之后，不管之前是否进行验证，都允许进行相关的访问。

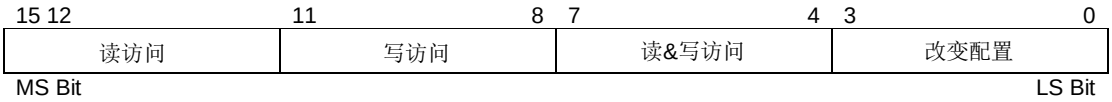
如果只有“读”和“读&写”访问（或“写”和“读&写”访问）密钥中的一个设定为 0xE，其它密钥的值都不同于 0xE，在有效验证的情况下通信以 MAC/加密的方式执行，在没有有效验证的情况下通信以明码的方式执行。在第二种情况下，通信的设定被 PICC 所忽略。

如果编码为 15（0xF），表示访问永远被禁止。因此各个连接的访问权总是被拒绝。

必须知道两字节参数代码的最高 4 位以获取读访问权（可使用 GetValue 和 Debit 命令）。接下来的 4 位用于获取写访问权（可使用 GetValue, Debit 和 LimitedCredit 命令）。低字节的高半部分用于获取读&写访问权。在值文件中，该权限允许完全访问（可使用 GetValue, Debit, LimitedCredit 和 Credit 命令）

最低 4 位保存密钥的参考号码，当改变文件的访问权限和将每个访问权限连接到密钥号时，需要对它进行验证。

访问权限的编码如下所示：



读访问和读&写访问权限允许执行读操作。

写访问和读&写访问权限允许执行写操作。

如果访问一个文件时没有有效的验证，但至少可通过一个相关的访问权限实现自由访问（0xE），那么通信模式被强制为明码通信。

3.4 状态编码和错误代码

Hex 代码	状态	描述
0x00	OPERATION_OK	成功的操作
0x0C	NO_CHANGES	备份文件不改变，不需要 CommitTransaction 和 AbortTransaction
0x0E	OUT_OF_EEPROM_ERROR	完成命令所需的 NV 存储器不足
0x1C	ILLEGAL_COMMAND_CODE	不支持的命令代码
0x1E	INTEGRITY_ERROR	CRC 或 MAC 与数据不匹配，填充字节无效
0x40	NO_SUCH_KEY	指定的密钥无效
0x7E	LENGTH_ERROR	命令串长度无效
0x9D	PERMISSION_DENIED	当前的配置/状态拒绝执行所请求的命令
0x9E	PARAMETER_ERROR	参数值无效
0xA0	APPLICATION_NOT_FOUND	请求的 AID 不存在
0xA1	APPL_INTEGRITY_ERROR	应用中不可恢复的错误，应用将被禁止*

0xAE	AUTHENTICATION_ERROR	当前验证状态不允许执行请求的命令
0xAF	ADDITIONAL_FRAME	期待发送额外的数据帧
0xBE	BOUNDARY_ERROR	试图读取/写入的数据超出文件/记录的边界。
0xC1	PICC_INTEGRITY_ERROR	PICC 内不可恢复的错误，PICC 将被禁止*
0xCD	PICC_DISABLED_ERROR	PICC 因为一个不可恢复的错误而被禁止*
0xCE	COUNT_ERROR	应用的数目限制为 28，CreateApplication 不再可用
0xDE	DUPLICATE_ERROR	因为已经存在相同编号的文件/应用，因此文件/应用的创建失败。
0xEE	EEPROM_ERROR	因为电源故障而无法完成 NV 写操作，启动内部备份/恢复机制。
0xF0	FILE_NOT_FOUND	指定的文件名不存在。
0xF1	FILE_INTEGRITY_ERROR	文件中不可恢复的错误，文件将被禁止*

* 这些错误在正常操作中是不期望出现的。

4 DESFIRE 命令集

4.1 命令集 ISO 14443-3：

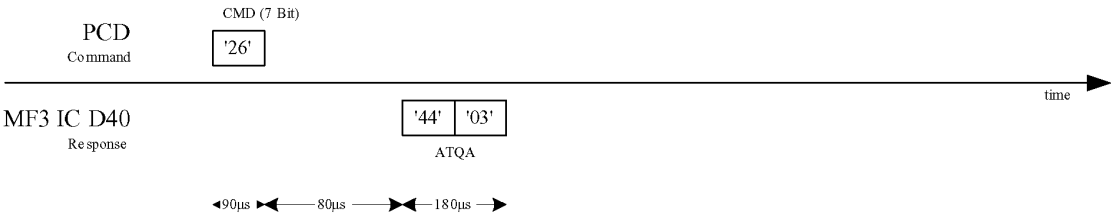
MF3 IC D40 提供下列符合 ISO 14443-3 的命令：

4.1.1 请求类型 A（REQA）

代码	参数	数据	完整性机制	响应
0x26（7 位）	—	—	—	0x0344

描述：MF3 IC D40 只在空闲状态下接受 REQA 命令。响应为 2 字节 ATQA(0x0344)。REQA 和 ATQA 的实现完全符合 ISO 14443-3。

REQA:

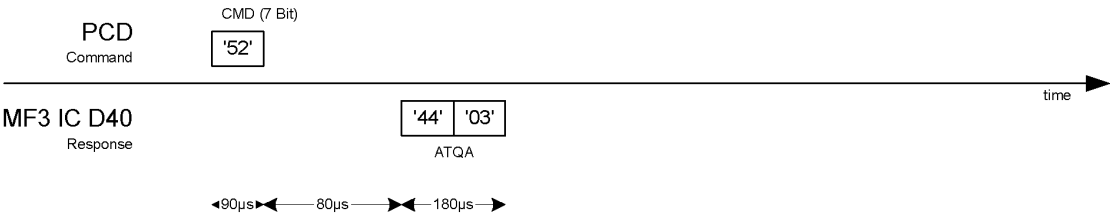


4.1.2 唤醒（WUPA）

代码	参数	数据	完整性机制	响应
0x52（7 位）	—	—	—	0x0344

描述：MF3 IC D40 只在空闲和暂停状态下接受 WUPA 命令。响应为 2 字节 ATQA(0x0344)。唤醒的实现完全符合 ISO 14443-3。

WAKE-UP:

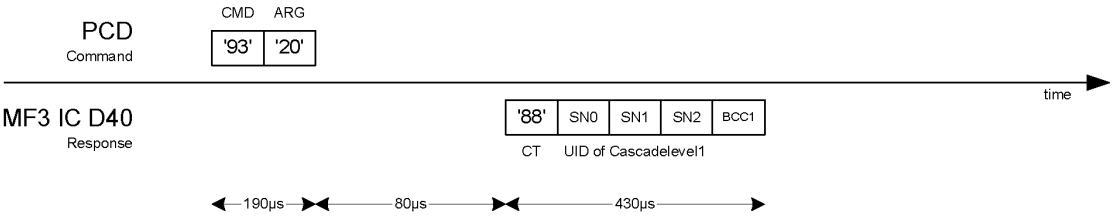


4.1.3 级联第一级的防冲突和选择

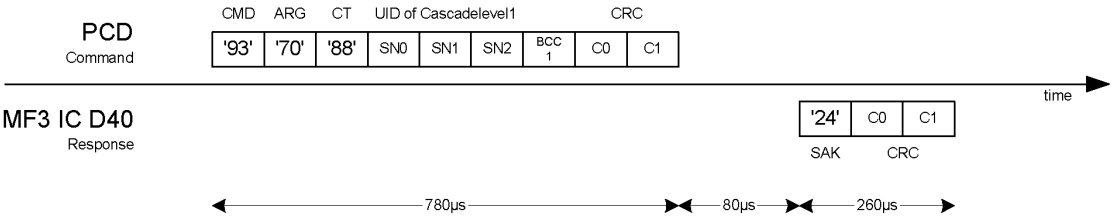
代码	参数	数据	完整性机制	响应
防冲突：0x93	0x20-0x67	部分 UID	奇偶校验	部分 UID
选择：0x93	0x70	级联标签，UID 的前 3 个字节	奇偶校验,BCC, CRC	SAK(0x24)

描述：防冲突命令和选择命令基于相同的命令代码。它们唯一的区别在于参数字节。该字节在选择时定义为 0x70。MF3 IC D40 只在 Ready1 状态接受这两个命令。响应是 UID 的第 1 部分。

防冲突：



选择：

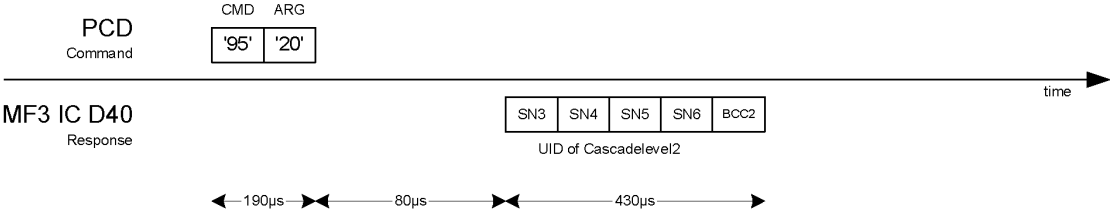


4.1.4 级联第 2 级的防冲突和选择

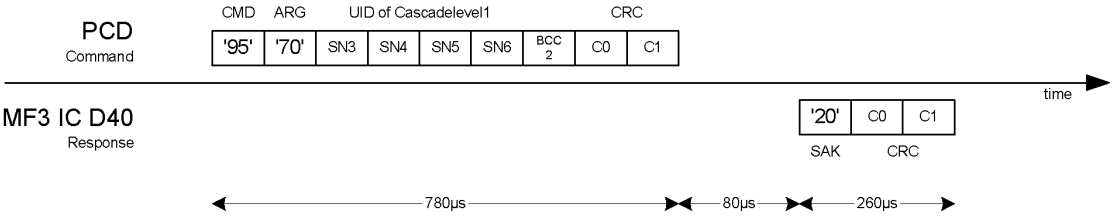
代码	参数	数据	完整性机制	响应
防冲突：0x95	0x20-0x67	部分 UID	奇偶校验	部分 UID
选择：0x95	0x70	级联标签，UID 的后 4 个字节	奇偶校验,BCC, CRC	SAK(0x20)

描述：防冲突命令和选择命令基于相同的命令代码。它们唯一的区别在于参数字节。该字节在选择时定义为 0x70。MF3 IC D40 只在 Ready2 状态接受这两个命令。响应是 UID 的第 2 部分。

防冲突：



选择：



4.2 命令集 ISO 14443-4：

MF3 IC D40 提供下列符合 ISO 14443-4 的命令集：

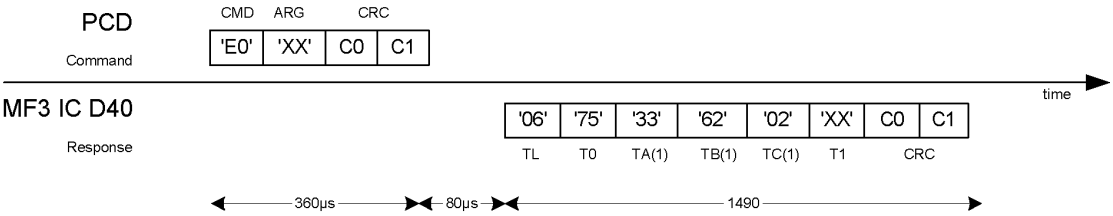
4.2.1 请求应答以选择（RATS）

代码	参数	数据	完整性机制	响应
0xE0	高半字节：FSDI 低半字节：CID	—	CRC	16 字节数据

描述：RATS 命令的响应提供 PICC 的类型使 PCD 进行识别。参数字节使用两个不同的参数编码进行通信：

- FSDI：参数字节的高半字节编码为 PCD 与 PICC 通信所支持的最大帧规格。
- CID：参数字节的低半字节编码为被寻址的 PICC 的逻辑编号。该逻辑编号范围为 0x00~0x0E。该 CID 用于区分单个 PCD 同时选择的几个 PICC。DESFire PICC 完全支持 CID，它使用 TC(1)进行编码。

RATS



MF3 IC D40 对 RATS 命令的响应就是“应答以选择”（ATS）。ATS 包含下面的字节：

TL：“长度字节”，长度字节 TL 指定传输的 ATS 的长度（包括它自身）。TL 不包括 2 个 CRC 字节。对于 MF3 IC D40 芯片来说，TL 设置为 0x06。

T0：“格式字节”，T0 定义随后的字节 TA(1), TB(1)和 TC(1)是否存在。3 个字节都存在，那么高半字节为 0x7。低半字节（FSCI）指定帧的最大规格，MF3 IC D40 接受的最大帧为 64 字节，编码为 0x5。因此格式字节为 0x75。

TA(1)：“接口字节 TA(1)”编码 PICC 所支持的最大数据传输速率。MF3 IC D40 支持的双向传输速率达到 424kbaud，因此 TA(1)字节编码为 0x33。

TB(1)：高半字节编码帧等待时间（FWT），对于 DESFire，设为 0x6，表示 19.33ms。低半字节编码启动帧保护时间（SFGT）。设置为 0x2，表示 1.21ms。因此 TB(1)字节编码为 0x62。

TC(1)：支持 CID，不支持 NAD，因此 TC(1)设置为 0x02。

T1：DESFire PICC 发送一个字节作为历史的字符，该字符应被应用软件所忽略。

4.2.2 协议和参数选择请求（PPS）

代码	参数	数据	完整性机制	响应
‘PPSS: ‘DX	PPS0 PPS1	—	CRC	PPSS: ‘D0’

描述：PPS 命令允许独立选择 PCD 和 PICC 之间的通信波特率。对于 DESFire，可以在两个方向独立设置波特率。也就是说，DESFire 允许非对称的数据交换速度。

- PPSS：PPSS 的高半字节为 RFU（保留将来之用），因此设置为’D’。低半字节指示所选择的 PICC 的 CID，范围为 0x00~0x0E。
- PPS0：PPS 参数 0 字节指示 PPS1 的存在并因此设置为 0x11。
- PPS1：PPS 参数 1 字节定义 PCD 和 PICC 之间定时的分频系数，它直接定义每个方向的波特率。PPS1 的高半字节为 RFU，必须设置为 0。bit b3 和 b2 编码 PICC 到 PCD 方向的分频系数，称为

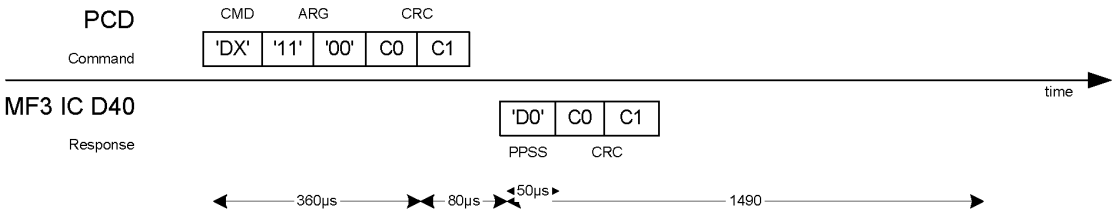
DSI，bit b1 和 b0 编码 PCD 到 PICC 方向的分频系数，称为 DRI。

DRI 和 DSI 的编码定义如下：

DRI, DSI 位编码	00	01	10
分频系数	1	2	4
波特率	106k	212k	424k

DESFire 支持双向高达 424kbaud 的波特率。请注意两个方向可以分别设置通信速度，例如，允许 PCD 到 PICC 方向使用 106kbaud(DRI=00)进行通信，而 PICC 到 PCD 使用 424kbaud 进行通信（DSI=10）。如果双向都使用 106kbaud 进行通信，PPS1 应当设置为 00。如果 PICC 没有接收到 PPS 命令，这将是它的默认值。

PPS



MF3 IC D40 对 PPS 命令的响应是 PPS 的起始字节（PPSS，0xD0）。无效的 PPS 请求将被忽略（按照 ISO 的定义）。

4.2.3 帧等待延长（WTX）

如果 PICC 需要比定义的 FWT 更长的时间来响应 PCD 的命令，它会发送一个等待时间延长的请求。

PICC 发送一个 14443-4 S-块。根据 ISO，INF 场包含 0x01，请求另一个 FWT 间隔。

PCD 必须通过发送另一个 INF 场为 0x01 的 S-块来确认该请求。

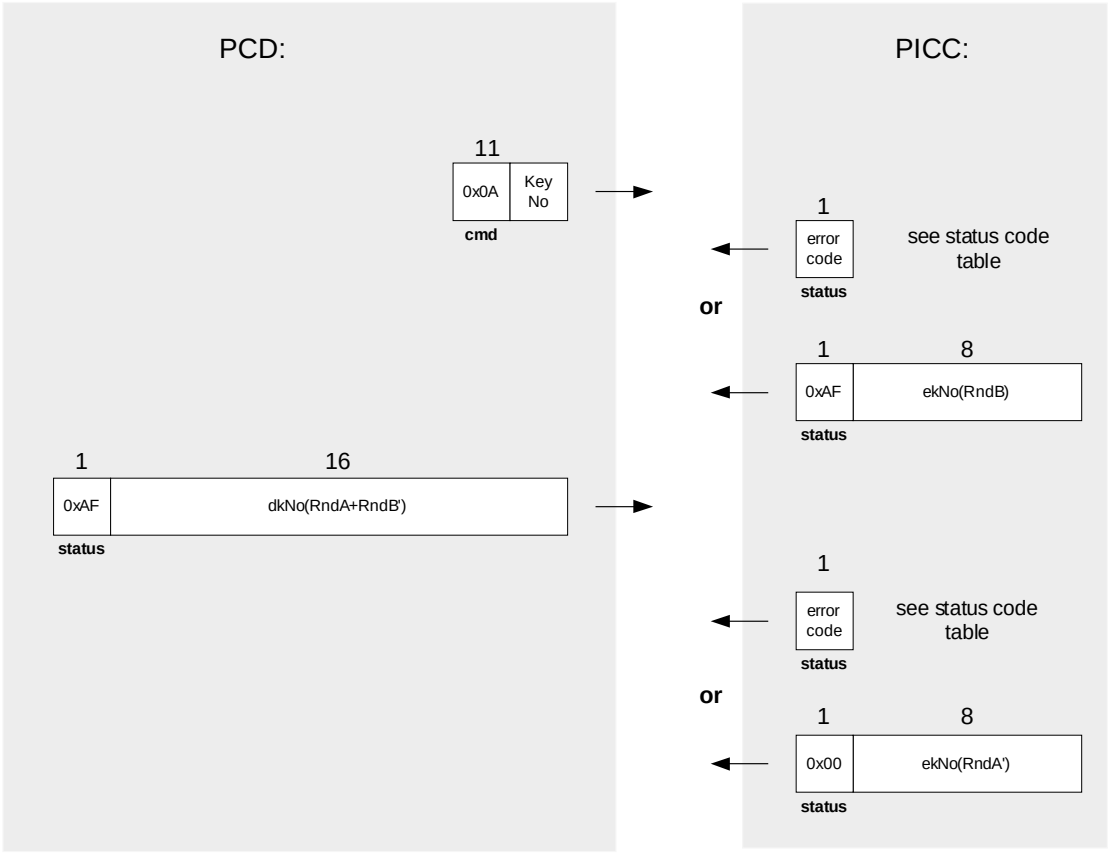
4.3 MF3 IC D40 命令集－安全性相关的命令

MF3 IC D40 提供下列用于安全性相关功能的命令集：

4.3.1 Authenticate

在这个过程当中，PICC 和读卡器都以加密的方式显示它们拥有相同的密钥。该过程不仅确认两个允许互相操作的实体，而且还创建一个会话密钥用于保持进一步的安全通信。正如“会话密钥”这个名字所明确表明的，每次一个新的验证过程成功结束后，会获得一个新的密钥用于下一步加密的操作。

Authenticate(keyNo) [2字节]



下面详细讲述交换的 3 段验证过程是如何完成的，以及会话密钥的产生：

#	PCD	数据交换	PICC
1	读卡器总是启动验证过程的实体。该过程从发送命令 Authenticate 开始，将参数密钥号发送给 PICC 用来和 PICC 保存的密钥进行验证。如果密钥无效，PICC 响应一个错误代码。 根据之前选择的 PICC 上的 AID，随后的验证过程根据这一指定的 AID 执行。 如果之前选择的 AID 为 0x00，验证过程使用 PICC 主密钥来完成。这时，参数密钥号必须设置为 0x00。PICC 主密钥的使用在后面的章节讲述。PICC 上电后 AID 为 0x00，表示上电后的 Authenticate 命令总是参考 PICC 的主密钥。	→ Authenticate (KeyNo)	
2		← 8 字节 ek_{keyNo}(RndB)	在选择指定的密钥后，PICC 产生一个 8 字节随机数 RndB。该随机数使用所选择的密钥进行 DES/3DES 加密，用 ek_{keyNo}(RndB)表示。然后发送到 PCD。

3	PCD 对接收到的 ek_{keyNo}(RndB) 执行 DES/3DES 译码操作并获得 RndB。使用的密钥与之前 PICC 加密所用的密钥相同。 在下一个步骤中，将译码得到的 RndB 循环左移 8 位(第一个字节移到 RndB 的最后)，得到 RndB'。 PCD 自身产生一个 8 字节随机数 RndA。RndA 与 RndB'串联并使用 DES/3DES 进行译码（两个块的译码通过 Cipher Block Chaining(CBC)发送模式连接在一起）。然后令牌 dk_{keyNo}(RndA+ RndB')发送到 PICC。	→ 16 字节 dk_{keyNo}(RndA+ RndB')	
4		← 8 字节 ek_{keyNo}(RndA')	PICC 运行 DES/3DES 对接收到的令牌进行加密并得到 RndA+RndB'。PICC 可将 RndB 在内部循环左移 8 位得到 RndB'，并将其与收到的 RndB'进行比较验证。 如果验证通过，证明 PICC 与 PCD 拥有相同的密钥。 如果验证失败，PICC 停止验证过程并返回一个出错信息。 由于 PICC 也接收了由 PCD 产生的随机数 RndA，可以对 RndA 执行循环左移 8 位得到 RndA'并再次加密得到 ek_{keyNo}(RndA')，将该令牌发送到 PCD。
5	PCD 对收到的 ek_{keyNo}(RndA') 执行 DES/3DES 译码获得 RndA'，将它与 PCD 内部移位的 RndA'比较。 如果结果不匹配，PCD 退出验证过程并可停止 PICC。		
6			PICC 为当前选择的应用或 PICC 自身（AID=0x00 时）设置验证状态。
7	如果所有的比较都匹配，通过 RndA 和 RndB 获得 16 字节会话密钥。会话密钥根据下列公式得到： 会话密钥＝$RndA_{1st\ half}$ + $RndB_{1st\ half}$ + $RndA_{2nd\ half}$ + $RndB_{2nd\ half}$ RndB 和 RndA 的不规则性避免了恶意的 PCD 通过强制 RndA=RndB，使 3DES 加密算法退化为单 DES 操作。 如果需要使用单 DES 操作（密钥的前 8 个字节和后 8 个字节相同），只使用会话密钥的前 8 个字节（$RndA_{1st\ half}$ + $RndB_{1st\ half}$）执行进一步的加密操作，后 8 个字节必须禁用。 随着会话密钥的产生，交互的 3 段验证过程也成功结束。		

根据应用的配置（通过 AID 表示），必须完成验证才能执行下列特定的操作：

- 收集关于应用的信息
- 改变应用的密钥
- 在应用中创建和删除文件
- 改变访问权限

- 访问已验证应用中的数据文件

根据 PICC 的安全性配置，下列命令可能要求使用 PICC 主密钥的验证：

- 收集关于 PICC 上应用的信息
- 改变 PICC 主密钥本身
- 改变 PICC 密钥设定
- 创建一个新的应用
- 删除一个现有的应用

验证状态在下列情况下无效：

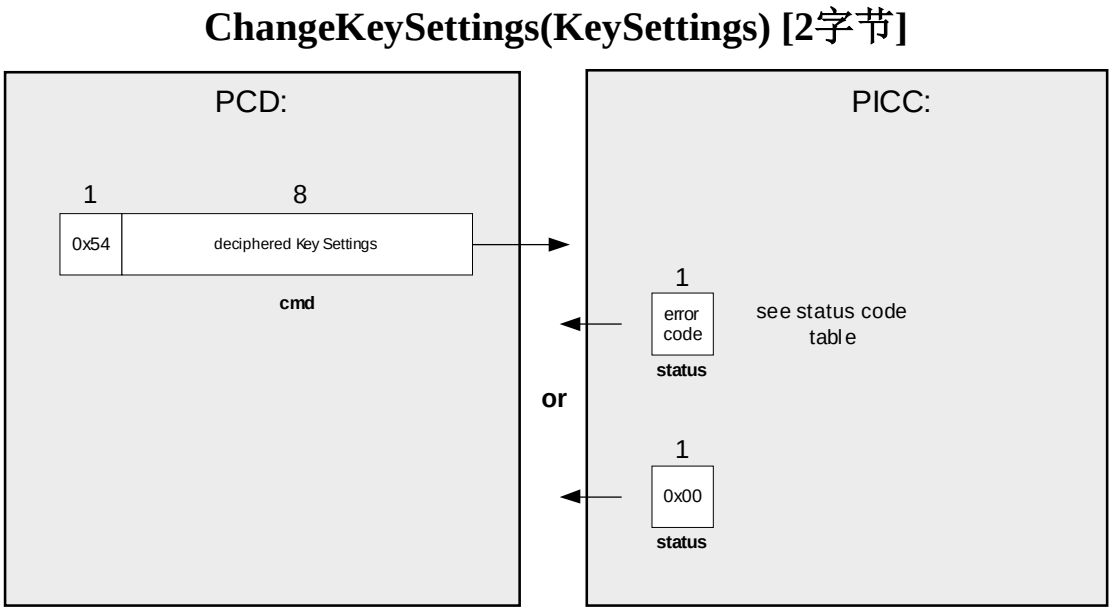
- 选择一个应用
- 用于进入当前有效验证状态的密钥被更改
- 验证失败

注：主密钥通过密钥号 0x00 来识别。这在 PICC 级（选择的 AID=0x0）和应用级（选择的 AID≠0x0）上有效。

4.3.2 ChangeKeySettings

该命令根据当前选择的 AID 改变主密钥的设定。

如果之前已经选择 AID=0x00，则将更改应用到 PICC 密钥设定，否则（AID≠0x0）应用到当前选择的应用的密钥设定。



该命令只有在当前密钥设定中的“配置可更改”位置位的情况下才会成功执行。

另外还需要在之前有一次成功的主密钥验证（如果 AID=0x00，则为 PICC 主密钥，否则为应用主密钥）。

该命令带由一个字节的参数，它编码新的主密钥设定。

为了保证 ChangeKeySettings 命令由执行之前验证命令的同一个 PCD 发送，因此应用的安全机制必须与 ChangeKey 命令所使用的安全机制相同，见 4.3.4。

对参数字节进行 CRC 计算并附在参数字节之后。由于修改的数据流现在的长度为 3 个字节因此在后面加上 5 个值为 0x00 的字节，这样得到 8 字节长度的数据流，正好适合 DES/3DES 的译码操作。

PICC 可以通过运行 DES/3DES 加密操作来验证接收到的数据并恢复的 CRC 和填充的空数据。

PICC 主密钥设定：

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RFU	RFU	RFU	RFU	配置可更改	无需主密钥的自由创建/删除	无需主密钥的自由目录列表访问	允许更改主密钥

PICC 级（所选 AID=0x00）对该编码的解释：

Bit7-Bit4：RFU，必须设置为 0。

Bit3：编码是否允许更改 PICC 主密钥的设定：

- 0：配置不可更改（冻结）
- 1：如果通过 PICC 主密钥的验证，该配置可更改（默认设定）

Bit2：在执行 CreateApplication/DeleteApplication 之前是否需要主密钥验证：

- 0：只有通过 PICC 主密钥验证才允许执行 CreateApplication/DeleteApplication
- 1：执行 CreateApplication 无需 PICC 主密钥的验证（默认设定）
  DeleteApplication 需要通过应用主密钥或 PICC 主密钥的验证。

Bit1：应用目录的访问是否需要 PICC 主密钥的验证：

- 0：执行 GetApplicationIDs 和 GetKeySettings 命令之前需要通过 PICC 主密钥验证。
- 1：执行 GetApplicationIDs 和 GetKeySettings 命令之前不需要 PICC 主密钥验证（默认设定）

Bit0：PICC 主密钥是否可更改：

- 0：PICC 主密钥不可更改
- 1：PICC 主密钥可更改（必须通过当前 PICC 主密钥验证，默认设定）

应用主密钥设定：

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ChangeKey 访问权限 bit3	ChangeKey 访问权限 bit2	ChangeKey 访问权限 bit1	ChangeKey 访问权限 bit0	配置可更改	无需主密钥的自由创建/删除	无需主密钥的自由目录列表访问	允许更改主密钥

应用级（所选 AID≠0x00）对该编码的解释：

Bit7-Bit4：保存更改应用密钥的访问权限（ChangeKey 命令）。

- 0x0：更改任何密钥都必须通过应用主密钥的验证（默认设定）
- 0x1..0xD：更改任何密钥必须通过指定密钥的验证
- 0xE：要更改某个密钥，必须通过该密钥的验证
- 0xF：应用中的所有密钥（应用主密钥除外，见 Bit0）都被冻结。

Bit3：是否允许更改应用主密钥的设定：

- 0：配置不可更改（冻结）
- 1：如果通过应用主密钥的验证，该配置可更改（默认设定）

Bit2：在执行 CreateFile/DeleteFile 之前是否需要主密钥验证：

- 0：只有通过应用主密钥验证才允许执行 CreateFile/DeleteFile
- 1：执行 CreateFile/DeleteFile 无需应用主密钥的验证（默认设定）

Bit1：应用目录的访问是否需要应用主密钥的验证：

- 0：执行 GetFileIDs 和 GetKeySettings 命令之前需要通过应用主密钥验证。
- 1：执行 GetFireIDs 和 GetKeySettings 命令之前不需要应用主密钥验证（默认设定）

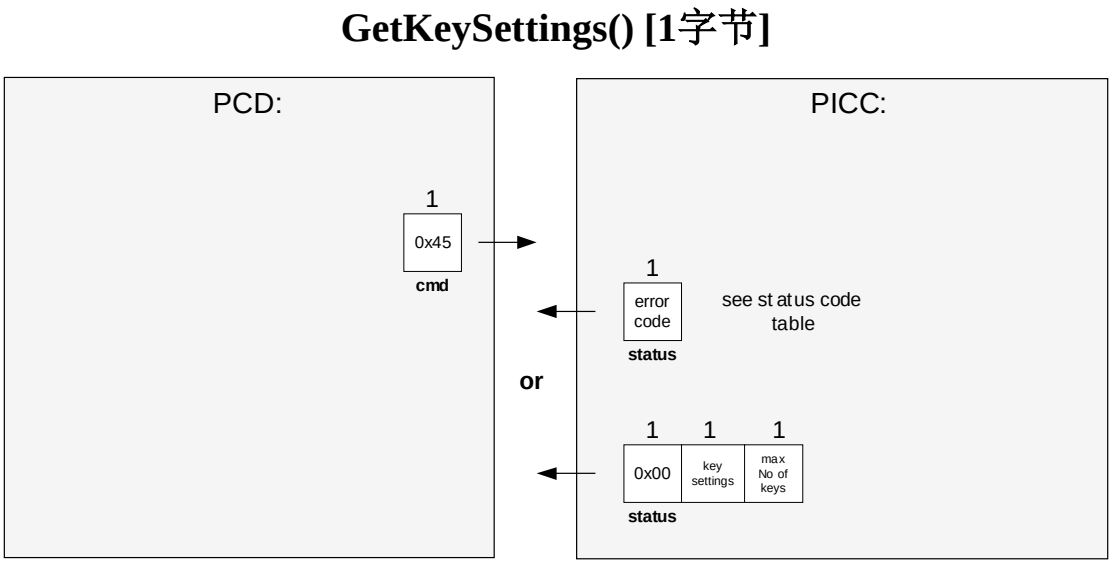
Bit0：应用主密钥是否可更改：

- 0：应用主密钥不可更改

- 1：应用主密钥可更改（必须通过当前应用主密钥验证，默认设定）

4.3.3 GetKeySettings

GetKeySettings 命令可获取 PICC 和应用主密钥设定的信息。此外，它还返回所选应用中可保存的最大密钥数。



该命令不传递参数。

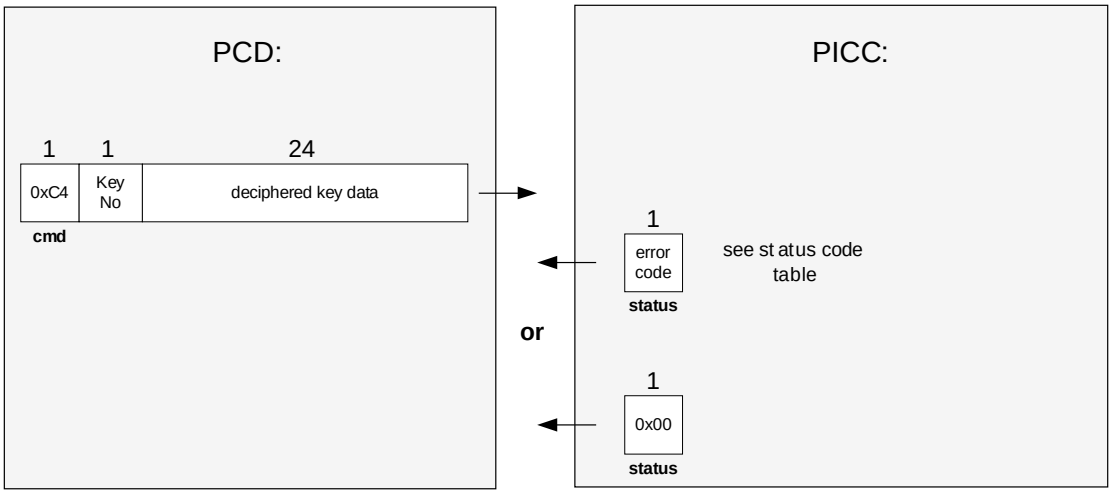
根据主密钥的设定（见 4.3.2），需要在该命令执行之前通过主密钥的验证。

如果查询 PICC 主密钥的设定（当前选择的 AID=0x00），密钥数返回 0x01，因为 PICC 上只存在一个 PICC 主密钥。

4.3.4 ChangeKey

该命令允许更改 PICC 上保存的任何密钥。

如果选择 AID=0x00，则更改 PICC 主密钥，只有 KeyNo＝0x00 有效（PICC 上只存在一个 PICC 主密钥）。在所有其它情况下（AID≠0x00），更改应用于当前所选应用中的指定 KeyNo。



KeyNo 是该命令的一个参数，长度为 1 字节，取值范围必须从 0x00 到（应用密钥数-1）。

第二个参数保存经过加密的新密钥的信息。

各自的属性设定定义了是否允许更改密钥，另外还显示哪一个密钥在执行 ChangeKey 命令之前需要进行验证。

- 如果 ChangeKey 密钥设置为 0x0..0xD，那么更改应用中的任何密钥（主密钥和 ChangeKey 密钥本身除外）都必须根据 ChangeKey 密钥设定中定义的 KeyNo 进行验证。

在这种情况下，PCD 以下列方式产生数据帧“译码的密钥数据”：

新的密钥和当前的密钥按位异或（16 字节）。对异或的数据进行 CRC 计算（2 字节）并将结果附在结尾。另外还要附上新密钥的 CRC（2 字节）。之后填充 0（4 字节）以符合帧的规格（总共 24 字节）。最后对整个密钥数据场执行 DES/3DES 译码操作（使用从 ChangeKey 密钥得到的当前会话密钥）。3 个加密块通过 CBC 发送模式连接起来。

改变 ChangeKey 密钥本身必须使用同样的方法：这时必须在之前通过应用主密钥的验证。

- 如果 ChangeKey 密钥设置为 0xE，必须对将要更改的 KeyNo 进行验证。该模式也适用于更改一个主密钥。

在这种情况下，PCD 以下列方式产生数据帧“译码的密钥数据”：

对新的密钥数据（16 字节）进行 CRC 计算并将结果（2 字节）附在后面。之后填充 6 个 0 字节以符合帧的规格（总共 24 字节）。最后对整个密钥数据场执行 DES/3DES 译码操作（使用当前会话密钥）。3 个加密块通过 CBC 发送模式连接起来。

- 如果 ChangeKey 密钥设置为 0xF，除了主密钥之外的所有密钥都被冻结。因此，当 ChangeKey 命令试图更改一个密钥（主密钥除外）时，将返回一个出错代码。

为了支持密钥版本，DESFire 允许在密钥中保存一个字节（8 位）的密钥版本。DES 和 3DES 的一个固有特性是只使用每个密钥字节的 7 个位作为有效的密钥信息。第 8 位（最低位）只保存奇偶校验信息。由于 DESFire 加密硬件不使用奇偶信息，因此可使用该位来保存密钥版本。DES/3DES 密钥前 8 个字节的奇偶位用来保存密钥版本。

为了将密钥版本保存到 PICC 中，在执行 ChangeKey 命令之前，必须通过应用软件将新的版本号编码到密钥信息中。PICC 在任何情况下都不会检查密钥的版本。

从 PICC 读出当前密钥的版本可使用 GetKeyVersion 命令。

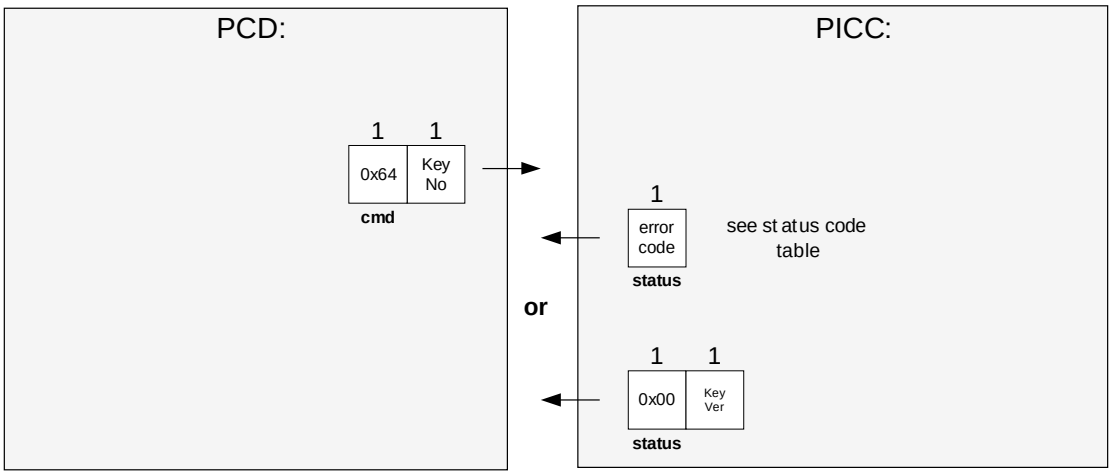
注：在成功修改了用于到达当前验证状态的密钥之后，该验证失效。也就是说，随后的操作需要使用新的密钥进行验证。

4.3.5 GetKeyVersion

GetKeyVersion 命令允许读出任何保存在 PICC 上的密钥的当前版本。

如果选择 AID=0x0，该命令返回 PICC 主密钥的版本，并且只有 KeyNo＝0x00 有效（PICC 上只存在一个 PICC 主密钥）。其它任何情况下（AID≠0x0），返回当前所选应用中指定的密钥的版本。

GetKeyVersion(KeyNo)[2 字节]



该命令带有一个参数，用于编码密钥号。

命令返回指定密钥的当前版本，为单字节无符号数。

要改变任何密钥的密钥版本，可使用 ChangeKey 命令，见 4.3.4。

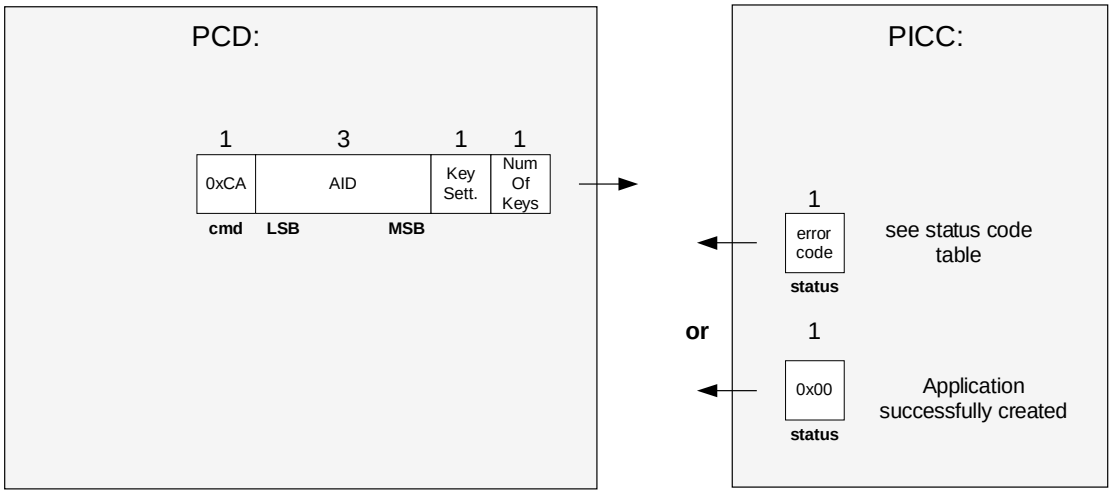
该命令的执行不需要进行验证。

4.4 MF3 IC D40 命令集－PICC 级命令：

4.4.1 CreateApplication

CreateApplication 命令允许在 PICC 上建立新的应用。

CreateApplication(AID,KeySettings,NumOfKeys)[6 字节]



应用通过“应用标识符”（AID）来识别，AID 长度为 24 位。应用标识符 0x00 00 00 保留给 PICC 自身。

根据 PICC 主密钥的设定，见 4.3.2，该命令在执行之前可能需要进行验证。

一个 PICC 最多可保存 28 个应用。每个应用最多可有 14 个由用户定义的访问密钥。要在应用中保存数据，必须在该应用中创建文件，详见 4.5.4~4.5.8。在每个文件中最多可创建 16 个不同规格和类型的文件。可以将单个文件的不同访问权限级别与应用密钥相连接。

24 位 AID 是命令的第一个参数。

第二个参数是应用主密钥的设定，见 4.3.2。

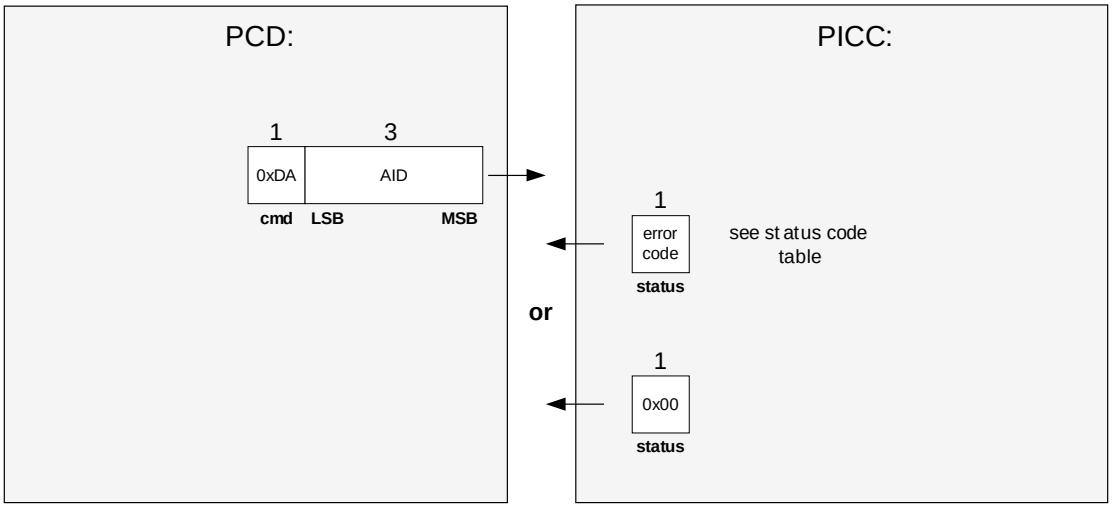
最后一个参数“密钥数”定义一个应用中可以保存多少个用于加密的密钥。所有密钥都使用包含 16 个 0x00 字节的字符串进行初始化，这是默认的单 DES 密钥值，见 3.2。

注：这些密钥在最终发行之前必须通过 ChangeKey 命令修改成其它值。

4.4.2 DeleteApplication

DeleteApplication 命令允许使 PICC 上的应用永久失效。

DeleteApplication(AID)[4 字节]



将被删除的应用由其应用标识符（AID）表示，这是该命令的唯一参数

根据 PICC 主密钥的设定，见 4.3.2，该命令在执行之前可能需要进行 PICC 主密钥的验证或应用主密钥的验证。

在后一种情况下，必须是对将被该命令删除的应用进行主密钥验证。

AID 的分配被撤销，因此可以使用被删除应用的 AID 创建一个新的应用。但是被删除的存储器块只能通过 FormatPICC 命令（该命令将整个 PICC 用户存储器擦除）恢复，见 4.4.5。

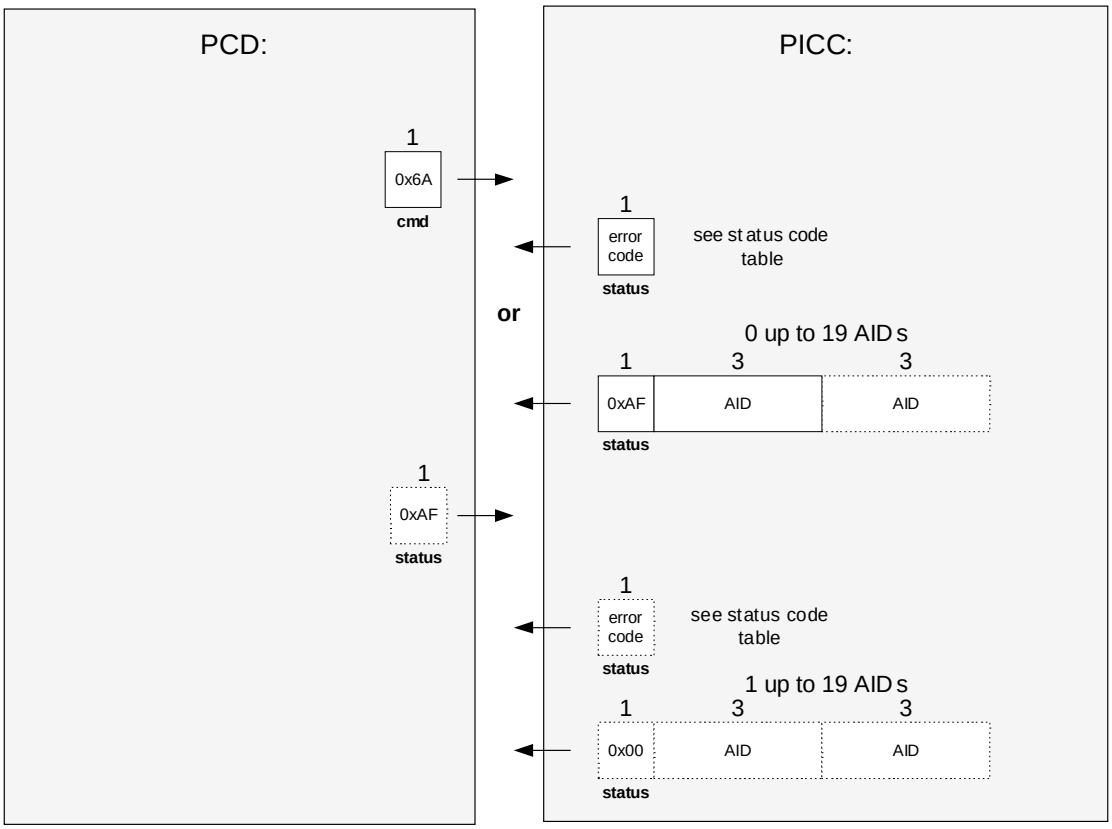
注：即使 PICC 主密钥包含默认值 0，并且 bit2－“无需主密钥的自由创建/删除”位置位，也必须使用为 0 的 PICC 主密钥或相关的应用主密钥进行验证。

注：如果当前选择的应用被删除，该命令自动选择 PICC 级，所选 AID＝0x00 00 00。

4.4.3 GetApplicationIDs

GetApplicationIDs 命令返回 PICC 上所有有效应用的 AID。

GetApplicationIDs()[1 字节]



该命令不接受任何参数。

根据 PICC 主密钥的设定，见 4.3.2，该命令在执行之前可能需要进行 PICC 主密钥的验证。

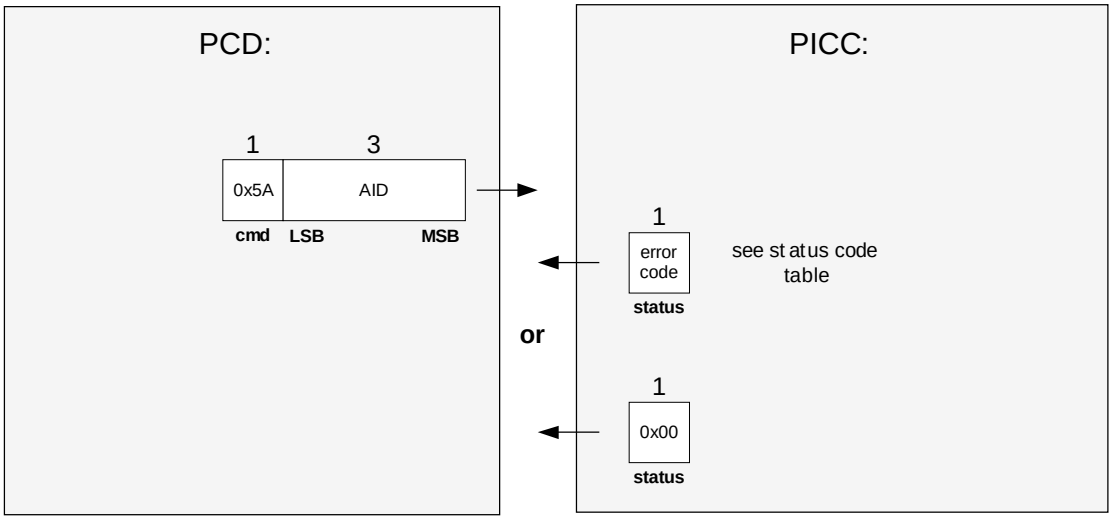
该命令要求当前选择的 AID 为 0x00 00 00 代表 PICC 级。

作为响应，PICC 发送所有已安装的 AID 序列。如果单个帧容不下该序列，则由 PICC 额外发送一帧，所有帧的状态字节都为 0xAF。

4.4.4 SelectApplication

SelectApplication 命令用于选择一个指定的应用作进一步的访问。在正常操作中，这通常是一个随后的验证命令。

SelectApplication(AID)[4 字节]



该命令所带参数为 3 个字节，用于编码 AID。

如果该参数为 0x00 00 00，PICC 级被选，并且任何进一步操作（例如，CreateApplication, DeleteApplication）都与该级别相关联。

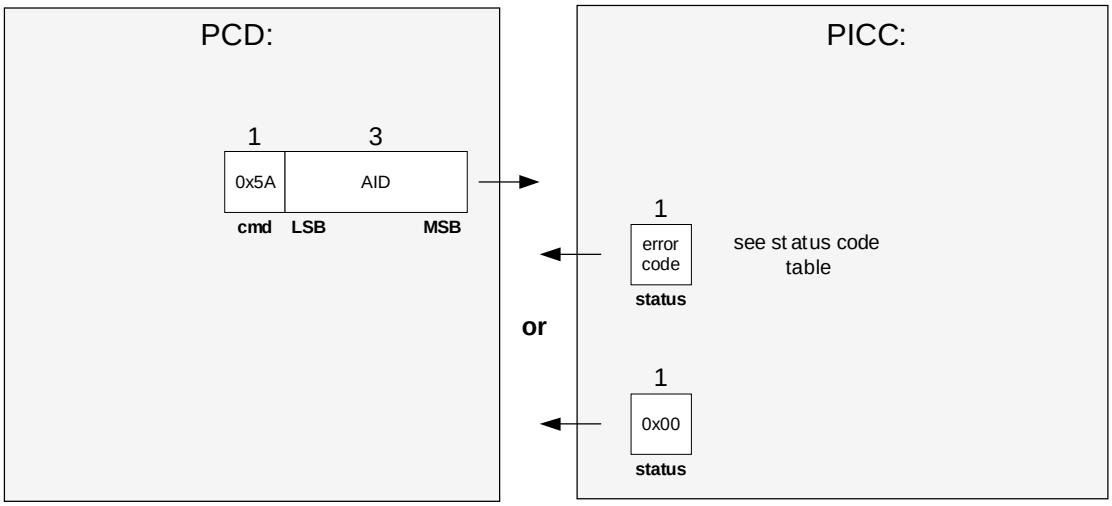
如果在 PICC 的应用目录中找到具有指定 AID 的应用，那么之后的命令都对该应用进行操作。

正如 Authenticate 命令一节（见 4.3.1）中提到的，每个 SelectApplication 命令都会使当前的验证状态失效。

4.4.5 FormatPICC

该命令释放 PICC 用户存储器。

FormatPICC()[1 字节]



该命令不传递参数。

FormatPICC 命令释放 PICC 上所有已分配的用户存储器。

所有应用都被删除，这些应用中的所有文件也被删除。

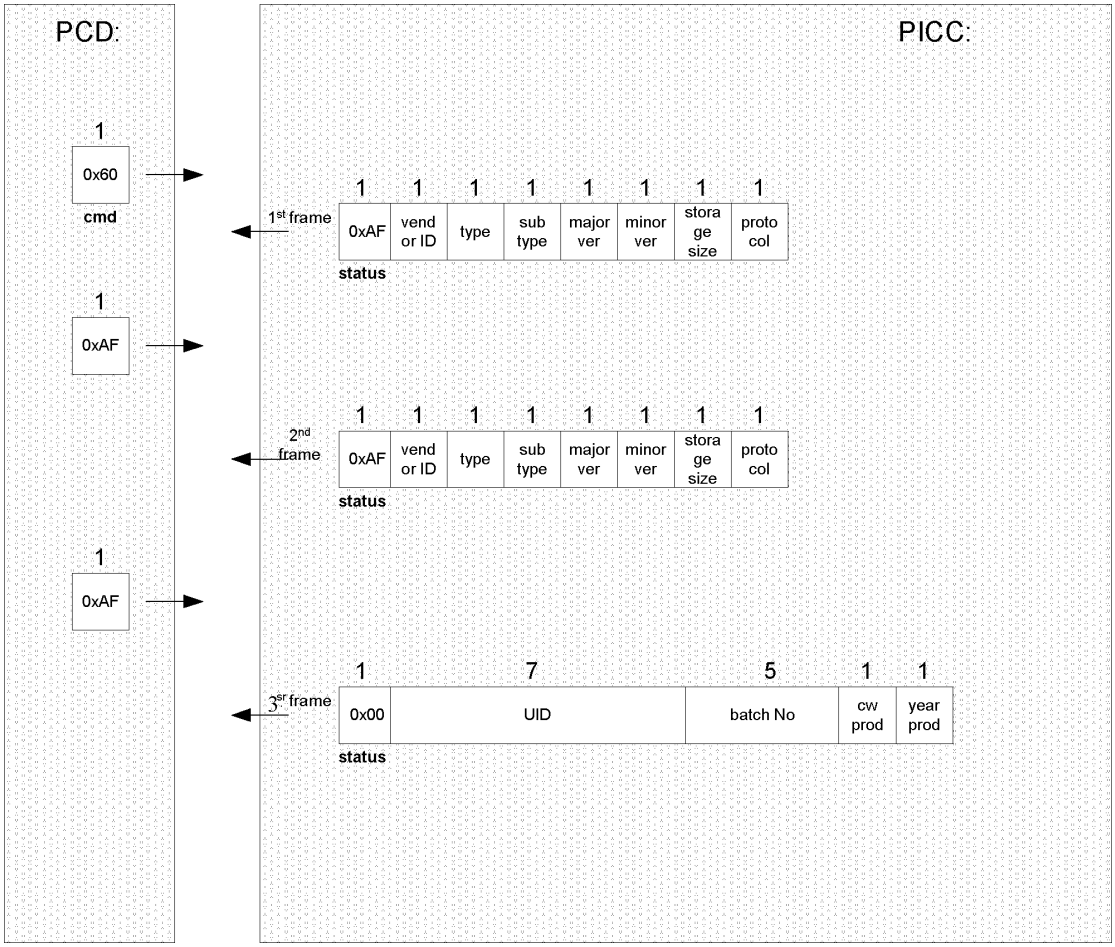
PICC 主密钥和 PICC 主密钥设定保持当前设定的值，不受该命令的影响。

执行该命令之前总是要求使用 PICC 主密钥进行验证。

4.4.6 GetVersion

GetVersion 命令返回 PICC 与制造有关的数据。

GetVersion[1 字节]



该命令不带参数。

PICC 返回 3 帧与制造相关的数据：

第 1 帧包含硬件相关的信息：

字节 1：厂商 ID（PHILIPS 为 0x04）

字节 2：类型（此处为 0x01）

字节 3：子类型（此处为 0x01）

字节 4：主版本号（此处为 0x00）

字节 5：子版本号（此处为 0x00）

字节 6：存储规格*（此处为 0x18＝4096 字节）

字节 7：通信协议类型（此处为 0x05，表示 ISO 14443-4）

第 2 帧包含软件相关的信息：

字节 1：厂商 ID（PHILIPS 为 0x04）

字节 2：类型（此处为 0x01）

字节 3：子类型（此处为 0x01）

字节 4：主版本号（此处为 0x00）

字节 5：子版本号（此处为 0x00）

字节 6：存储规格*（此处为 0x18＝4096 字节）

字节 7：通信协议类型（此处为 0x05，表示 ISO 14443-4）

第 3 帧返回唯一序列号、批号、生产的年份和周数：

字节 1~字节 7：唯一的序列号

字节 8~字节 12：生产批号

字节 13：生产的周数

字节 14：生产的年份

* 高 7 位（＝n）根据 2^n 编码存储器规格。如果规格恰好等于 2^n，最低位为 0，如果规格介于 2^n 和 2^{n+1} 之间，则为 1。对于该版本的 DESFire，高 7 位为 0x0C（2^{12}＝4096），最低位为 0。

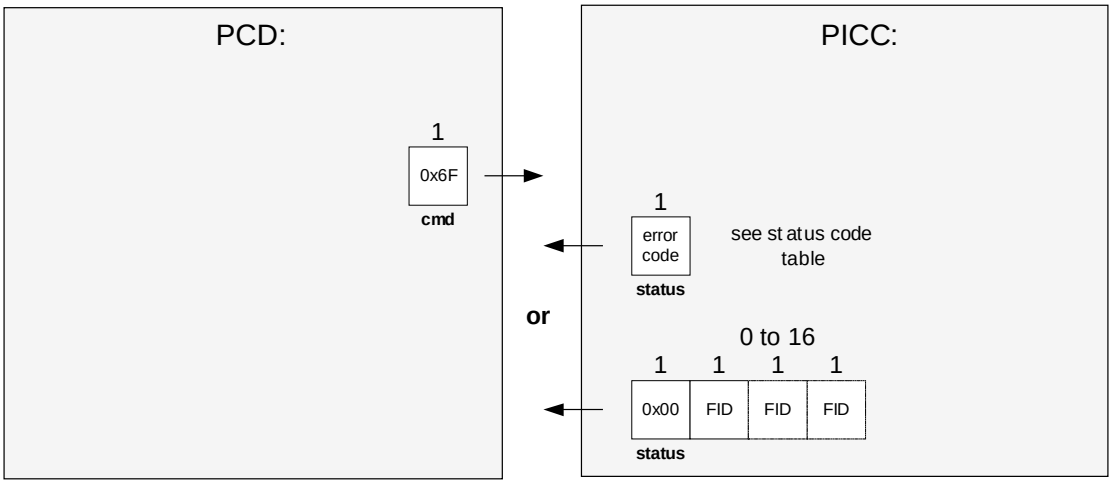
4.5 MF3 IC D40 命令集－应用级命令：

MF3 IC D40 为应用级功能提供下列命令集：

4.5.1 GetFileIDs

GetFileIDs 命令返回当前所选应用中所有有效文件的文件标识符。

GetFileIDs()[1 字节]



该命令不带参数。

根据应用主密钥的设定，见 4.3.2，该命令在执行之前可能需要进行应用主密钥的验证。

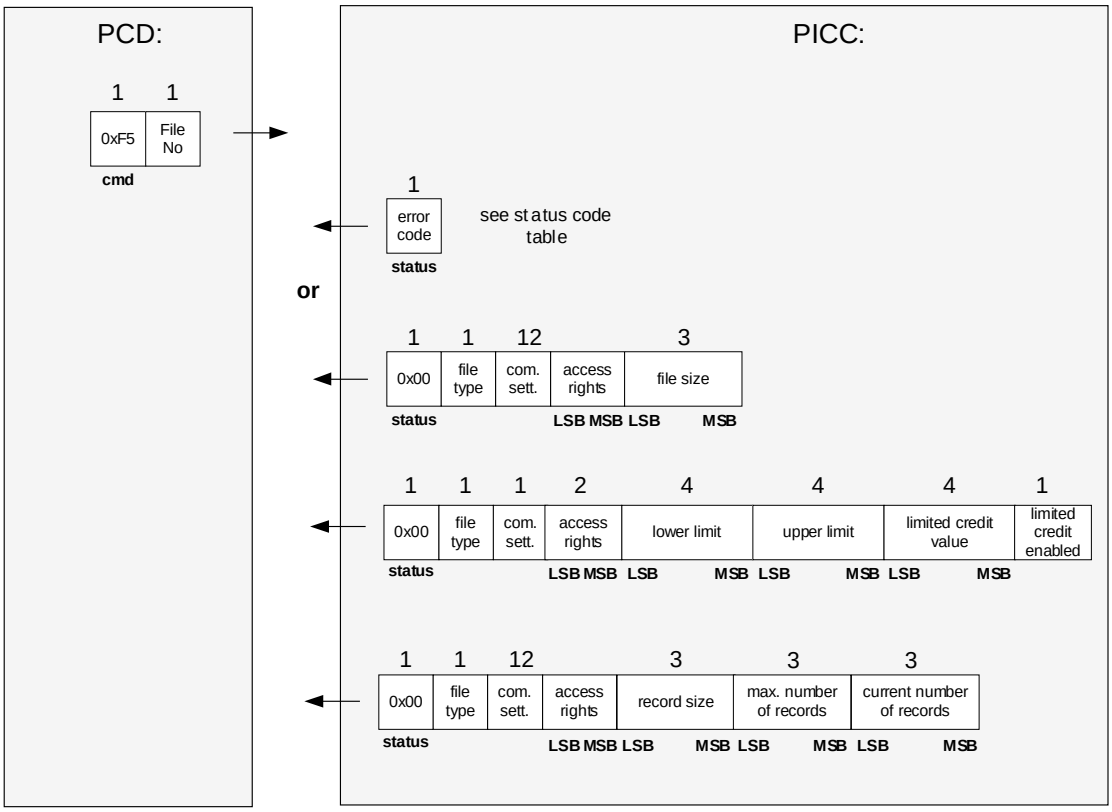
每个文件 ID 编码为一个字节，范围为 0x00~0x0F。由于每个文件都有一个唯一确定的标识符，因此不可能出现重复的值。

由于一个应用中的文件数目限制为 8 个，因此响应总是适合用单个数据帧发送。

4.5.2 GetFileSettings

GetFileSettings 命令可以从指定文件属性获得信息。该命令获取的信息取决于被查询文件的类型。

GetFileSettings()[2 字节]



GetFileSettings 命令带一个参数，用于编码被查询文件的文件编号。该文件编号必须在 0x00 和 0x0F 之间。

根据应用主密钥的设定，见 4.3.2，该命令在执行之前可能需要进行应用主密钥的验证。

在更新值文件的值之后，但在执行 CommitTransaction 命令之前，GetFileSettings 命令总是将有限信用值恢复为旧的、未改变的限制值。

所有文件类型的返回信息的第一部分都是相同的：

第 1 个字节表示文件类型，见 3.1。第 2 个字节提供文件通信设定的信息（明码/MAC/加密），见 3.2。

该信息后面是 2 字节文件访问权限值，见 3.3。

之后的字节根据文件类型的不同，有着不同的含义：

- 标准数据文件和备份文件：

3 字节长度的场返回用户文件的规格（字节为单位）。

- 值文件：

返回 3 个场，每个场长度为 4 字节。第一个场返回文件的“下限”（在文件创建时定义），第二个场返回“上限”（在文件创建时定义），第三个场返回当前最大的“有限信用”值，见 4.6.5。如果不使用有限信用值，第三个场全零。如果该文件允许使用 LimitedCredit 命令，则最后一个字节编码为 0x00（禁止），0x01 或 0x01（使能）。

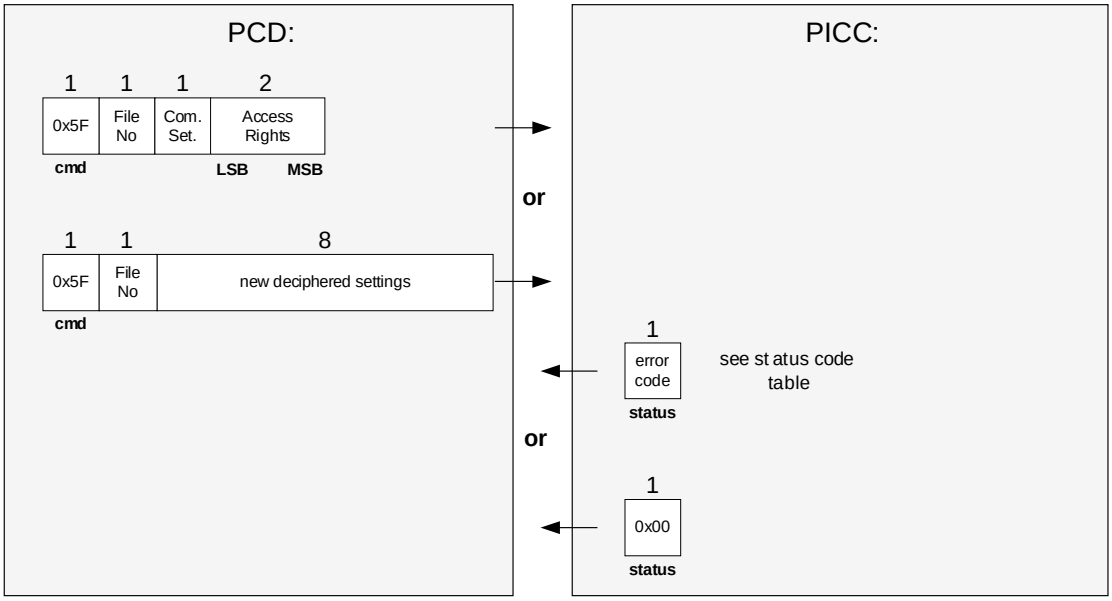
- 线性记录文件和循环记录文件：

返回 3 个场，每个场为 3 个字节。第一个场返回单个记录的规格（在文件创建时定义），第二个场为记录文件中的最大记录数（在文件创建时定义），第三个场返回记录文件中当前的记录数。该数字等于当前可读记录的最大数目，见 4.6.8。

4.5.3 ChangeFileSettings

该命令改变一个现有文件的访问参数。

ChangeFileSettings(FileNo, Com.Set., AccessRights)[5 字节]



参数的第一个字节表示当前所选应用中的文件编号。

下一个字节定义新的通信设定，见 3.2。

最后两个字节定义新的访问权限，见 3.3。

只有在当前访问权不是“永远禁止”时，设定的改变才会成功，见 3.3。

为了保证 ChangeFileSettings 命令是由之前通过验证的同一方发出的，必须使用与 ChangeKey 命令相同的加密机制，见 4.3.4。

对后 3 个字节进行 CRC 计算并将结果附在结尾。由于修改的数据流现在长度为 5 个字节，再加上 3 个全零字节构成一个符合 DES/3DES 操作的 8 字节长度的数据流。最好在 PCD 端对该数据执行 DES/3DES 译码。

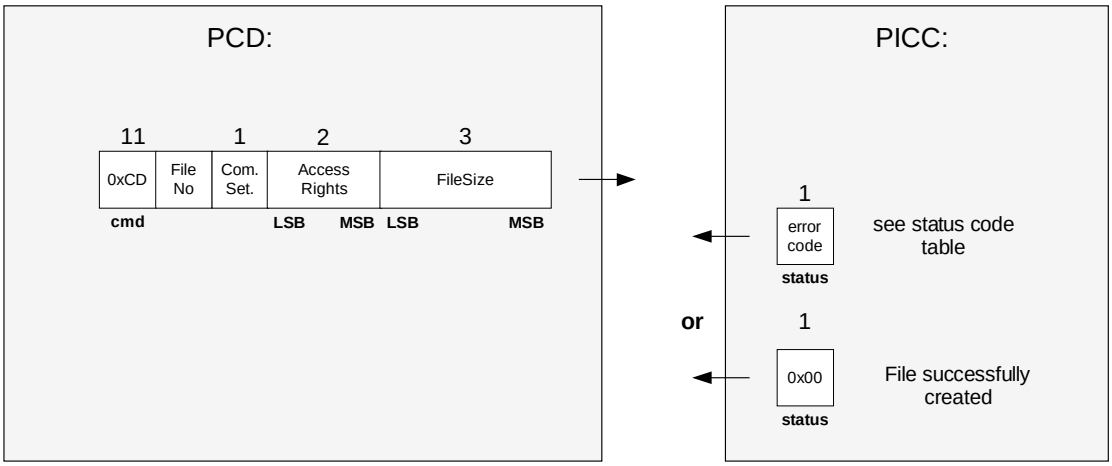
PICC 通过运行 DES/3DES 加密来胭脂接收数据并对数据进行 CRC 校验。

但是如果 ChangeAccessRights 访问权限设置为“自由访问”，则不需要任何安全机制，数据以明码形式发送（总共 5 个字节）。

4.5.4 CreateStdDataFile

CreateStdDataFile 命令在 PICC 现有的一个应用中创建用于保存无格式用户数据的文件。

CreateStdDataFile(FileNo, Com.Set., AccessRights, FileSize)[8 字节]



该命令的第一个参数要求新的文件编号在范围 0x00~0x0 之内。在当前所选的应用中创建文件。文件的创建不需要遵循特定的顺序。如果指定编号的文件已经存在于当前所选的应用中，则返回一个出错代码。

第二个字节定义通信设定，见 3.2。

接下来两个字节定义新的访问权限，见 3.3。

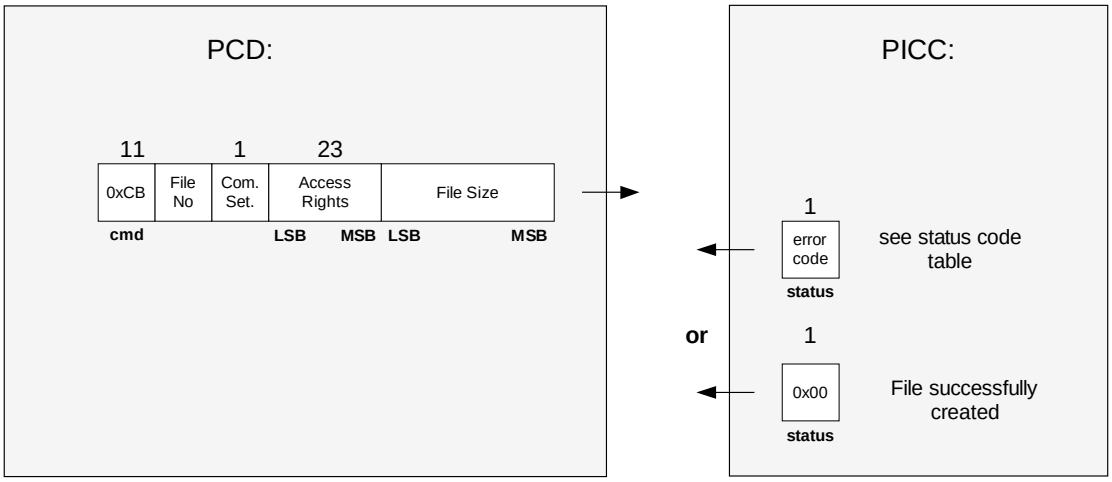
参数的最后 3 个字节指定文件的规格（以字节为单位）。

注：MF3 IC D40 在内部按照 32 字节的倍数来分配 NV 存储器，因此参数 FileSize 为 0x00 00 01 的文件创建命令将在内部消耗与 0x00 00 20（32 字节文件规格）相同数目的 NV 存储器，也就是 32 字节。

4.5.5 CreateBackupDataFile

CreateBackupDataFile 命令在 PICC 现有的一个应用中创建用于保存无格式用户数据的文件，另外还支持集成的备份机制特性。

CreateBackupDataFile(FileNo, Com.Set., AccessRights, FileSize) [8 字节]



正如名称“BackupDataFile”所指，这种类型的文件具有集成的备份机制特性。

每个写命令都在该文件的独立镜像中完成。为了使写操作对该文件类型有效，必须使用 CommitTransaction 命令进行确认，见 4.6.10。如果 PCD 没有发送 CommitTransaction 命令，那么只改变了镜像，而原始数据保持不变并继续有效。

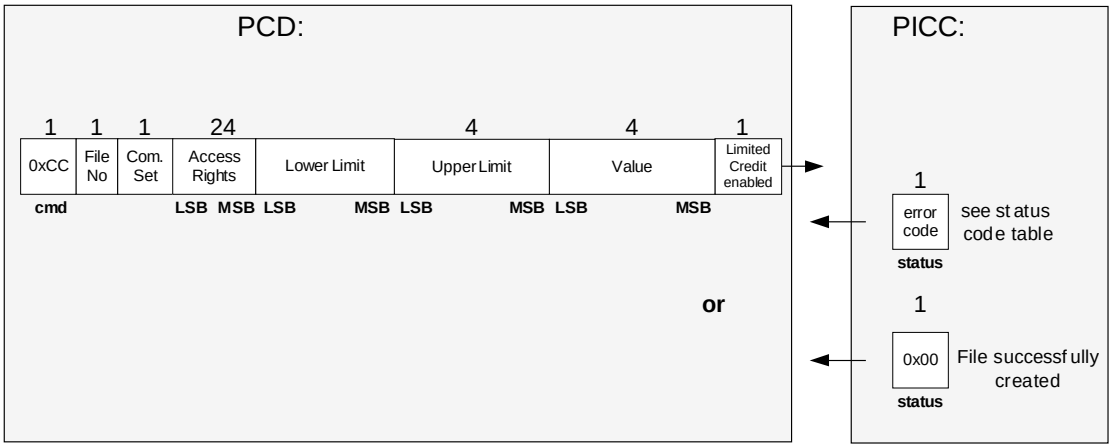
所有参数的格式都与 CreateStdDataFile 命令的参数格式相同，见 4.5.4，但参数 FileNo 例外。由于一个应用只有前 8 个字节支持集成的备份机制特性，因此 FileNo 的范围只能为 0x00~0x07。

由于产生镜像，因此 BackupDataFile 和指定相同规格的 StdDataFile 相比，总是消耗两倍的 NV 存储器。

4.5.6 CreateValueFile

CreateValueFile 命令在 PICC 的一个现有文件中创建文件，该文件用于保存和处理 32 位有符号整数值。

CreateValueFile(FileNo, Com.Set., AccessRights, LowerLimit, UpperLimit, Value, LimitedCreditEnabled)[8 字节]



第一个参数字节是文件编号，范围为 0x00~0x07，是当前所选应用中新创建的文件。

下一个字节定义通信设定，见 3.2。

接下来两个字节定义新的访问权限，见 3.3。

然后是 4 字节长度的参数，编码该文件的有效长度下限。下限表示对当前值进行 Debit 计算时不能超过的边界，见 4.6.5。下限参数是一个 4 字节有符号整数，可以是负值。

接着是 4 字节的上限参数，它以同样的方式设置边界，不过是对 Credit 操作。见 4.6.4。该参数也是 4 字节有符号整数。

上限必须大于等于下限，否则 PICC 会发送出错信息，并且文件创建也被取消。

下一个参数还是 4 字节有符号整数，它指定值文件的初始值。PICC 对上限和下限进行检查，如果出现不一致，则取消文件创建并发送出错信息。

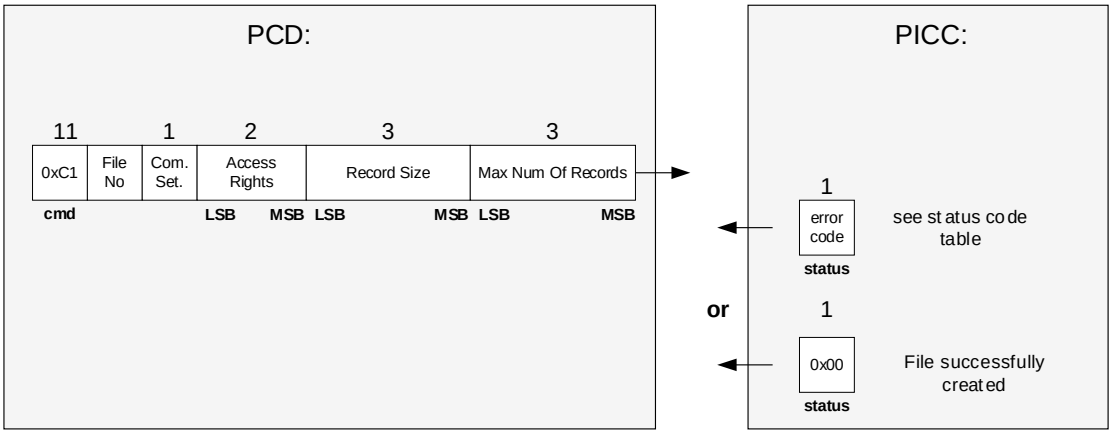
最后一个参数字节用于使能 LimitedCredit 特性，见 4.6.6。0x00 表示 LimitedCredit 被禁止，0x01 表示使能该特性。

值文件总是具有集成的备份机制特性，因此每次改变值的访问都需要通过 CommitTransaction 命令进行验证，见 4.6.10。

4.5.7 CreateLinearRecordFile

CreateLinearRecordFile 命令在 PICC 的一个现有文件中创建文件，该文件用于结构相似的数据（例如 loyalty 程序）的多次存储。一旦该文件被数据记录填满，那么将无法再对该文件执行写操作，除非该文件被清空，见命令 ClearRecordFile 一节 4.6.9。

CreateLinearRecordFile(FileNo, Com.Set., AccessRights, Rec.Size, MaxNumOfRecords) [11 字节]



第一个参数字节是文件编号，范围为 0x00~0x07，是当前所选应用中新创建的文件。

下一个字节定义通信设定，见 3.2。

接下来两个字节定义新的访问权限，见 3.3。

然后是 3 字节参数，定义单个记录的规格（以字节为单位）。该参数的范围必须是 0x000001~0xFFFFFF。

最后一个参数长度为 3 字节，定义记录的数目。该参数的范围也是 0x000001~0xFFFFFF。

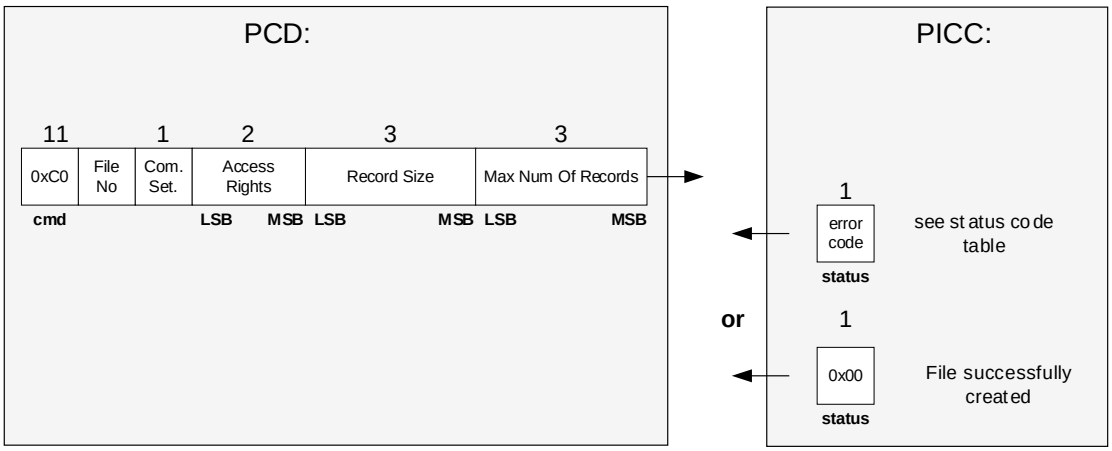
因此整个文件的规格为 RecordSize*MaxNumOfRecords。

线性记录文件总是具有集成的备份机制特性，因此每次改变值的访问都需要通过 CommitTransaction 命令才生效，见 4.6.10。

4.5.8 CreateCyclicRecordFile

CreateCyclicRecordFile 命令在 PICC 的一个现有文件中创建文件，该文件用于结构相似的数据（例如 loyalty 程序）的多次存储。一旦该文件被数据记录填满，PICC 自动将最新写入的记录覆盖最早的记录。该循环操作对 PCD 是完全透明的。

CreateCyclicRecordFile(FileNo, Com.Set., AccessRights, Rec.Size, MaxNumOfRecords) [11 字节]



第一个参数字节是文件编号，范围为 0x00~0x07，是当前所选应用中新创建的文件。

下一个字节定义通信设定，见 3.2。

接下来两个字节定义新的访问权限，见 3.3。

然后是 3 字节参数，定义单个记录的规格（以字节为单位）。该参数的范围必须是 0x000001~0xFFFFFF。

最后一个参数长度为 3 字节，定义记录的数目。该参数的范围是 0x000002~0xFFFFFF。

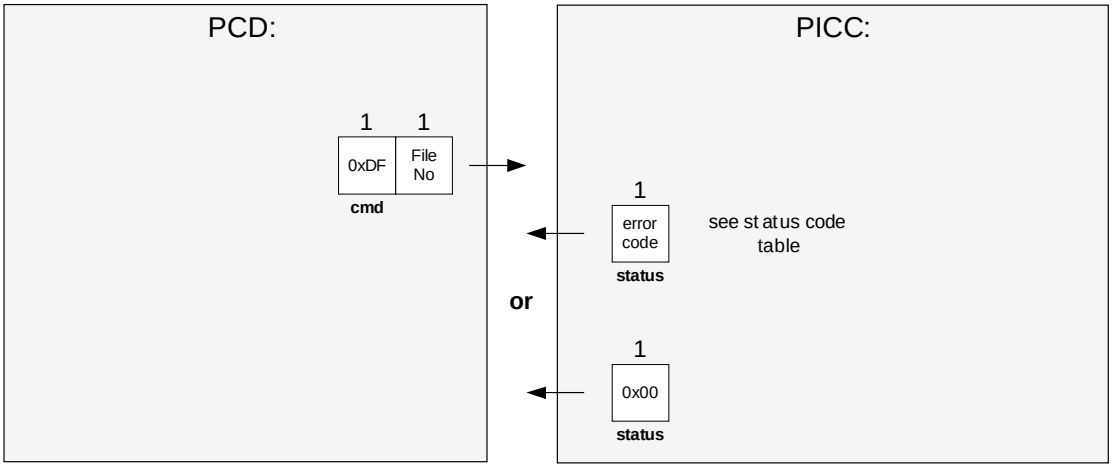
因此整个文件的规格为 RecordSize*MaxNumOfRecords。

循环记录文件总是具有集成的备份机制特性，因此每次添加记录的访问都需要通过 CommitTransaction 命令才生效，见 4.6.10。

4.5.9 DeleteFile

DeleteFile 命令使当前所选应用的文件目录中的一个文件永远失效。

DeleteFile(FileNo)[2 字节]



该命令带一个字节的参数，指定范围 0x00~0x0F 内的文件编号。

该命令的操作使指定文件的文件目录入口无效，这意味着文件从此将无法进行访问。

根据应用主密钥的设定，见 4.3.2，该命令在执行之前可能需要进行应用主密钥的验证。

与被删除文件相关的存储器块不会被释放。在该应用中，被删除文件的文件编号可以重新用来创建一个新的文件。

要释放存储器块实现重新利用，必须使用 FormatPICC 命令将整个 PICC 用户 NV 存储器擦除，见 4.4.5。

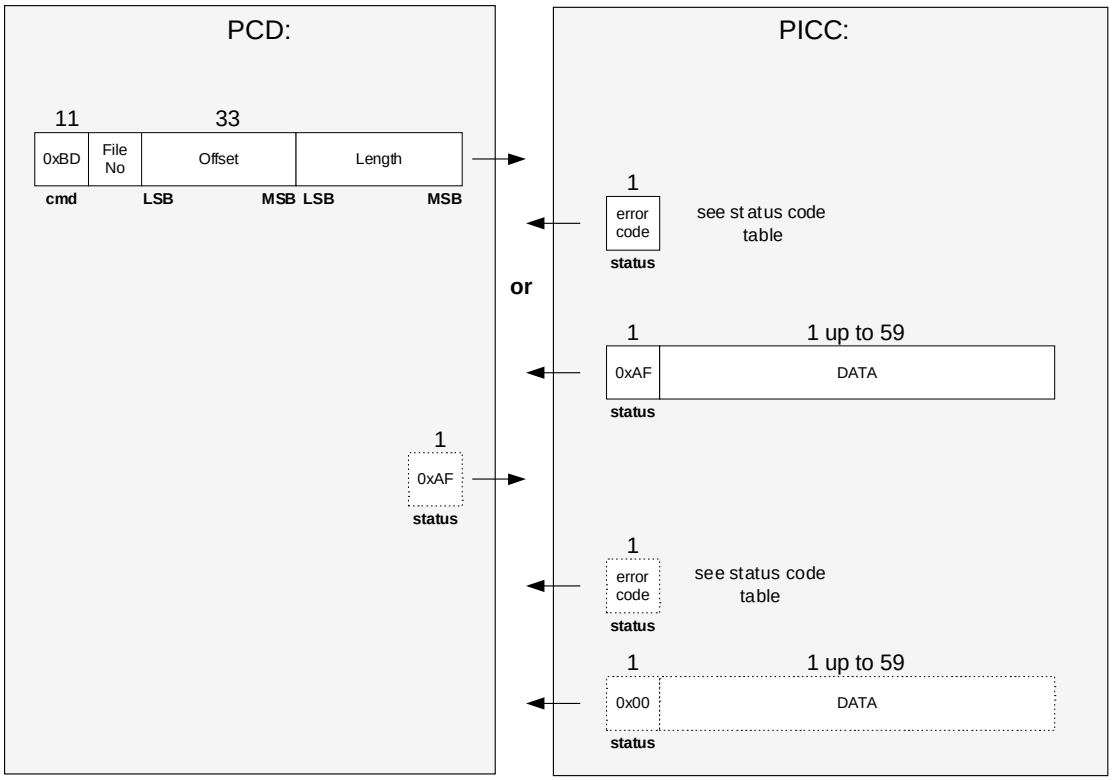
4.6 MF3 IC D40 命令集－数据处理命令

MF3 IC D40 为数据处理提供下列命令集。

4.6.1 ReadData

ReadData 命令可从标准数据文件或备份数据文件中读出数据。

ReadData(FileNo, Offset, Length)[8 字节]



第一个参数长度为 1 字节，定义了读取文件的编号。该参数范围必须为 0x00~0x0F（对于标准数据文件）或 0x00~0x0F（对于备份数据文件）。

下一个参数长度为 3 字节，定义文件中读操作的起始位置（偏移值）。该参数范围必须为 0x00000~ (文件规格-1)。

第三个参数长度也是 3 字节，指定了读取的数据字节个数。该参数范围为 0x000000~0xFFFFFF。

如果第三个参数编码为 0x000000，则从偏移值指定的位置开始读取整个数据文件。

如果发送到 PCD 的字节数不是恰好一帧，PICC 在发送下一帧之前等待一个状态字节为 0xAF 的状态帧。

如果在写备份数据文件之后，执行 CommitTransaction 命令之前读取备份文件，ReadData 命令总是获得 PICC 中保存的旧的、未变的数据。所有写入备份数据文件的数据只有在 CommitTransaction 命令之后才有效并可被外部 ReadData 命令读取。

读取命令在执行之前需要进行验证，该验证可以是“读”或“读&写”访问验证，见 3.3。

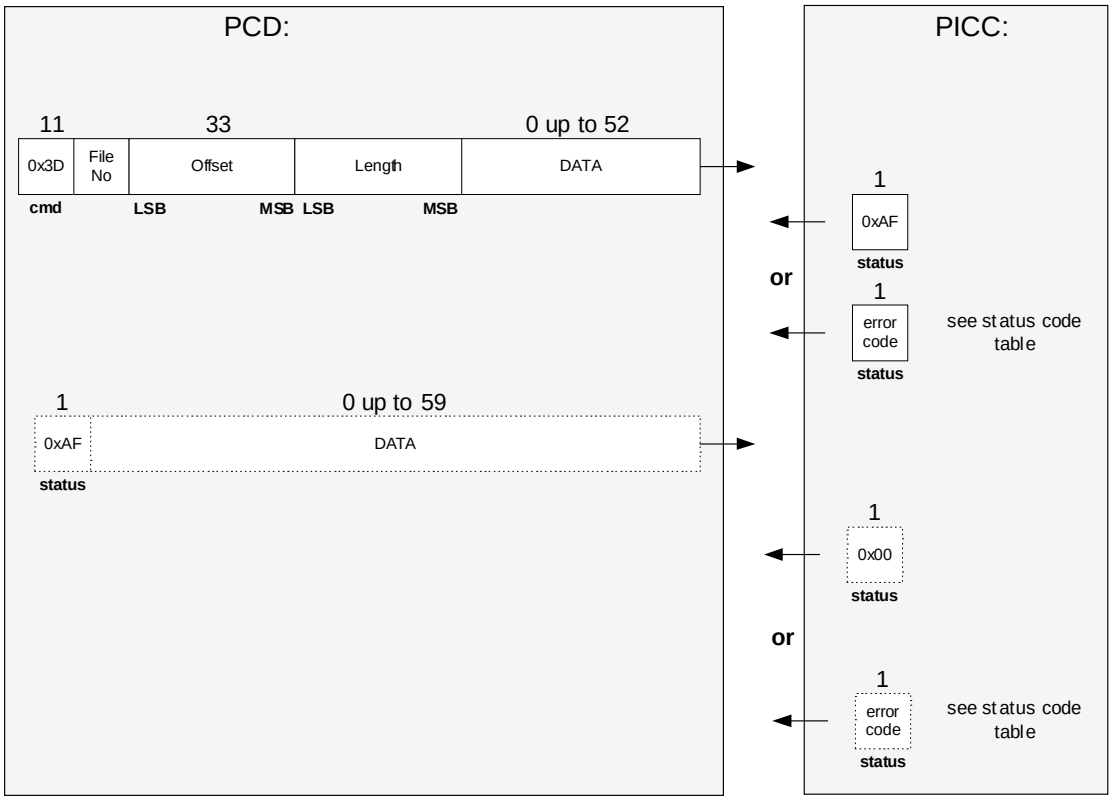
根据连接到文件的通信设定，PICC 使用明码/MAC/加密方式发送数据。所有加密操作都在 CBC 模式下完成。

对于 MAC 和加密的通信数据填充，必须使整个数据长度为 8 的倍数。在指定数据长度（命令参数长度≠0x000000）的情况下，填充字节都为 0x00。如果一直读取到文件边界为止（命令参数长度＝0x000000），附加的第一个填充字节为 0x80，附加的其它字节都为 0x00（符合 ISO 9797-1）。

4.6.2 WriteData

WriteData 命令可以将数据写入标准数据文件或备份数据文件。

WriteData(FileNo, Offset, Length)[8 字节]



第一个参数长度为 1 字节，定义了写入文件的编号。该参数范围必须为 0x00~0x0F（对于标准数据文件）或 0x00~0x0F（对于备份数据文件）。

下一个参数长度为 3 字节，定义文件中写操作的起始位置（偏移值）。该参数范围必须为 0x00000~ (文件规格-1)。

第三个参数长度也是 3 字节，指定了写入的数据字节个数。该参数范围为 0x000001~0xFFFFFF。

如果发送的字节数不是恰好一帧，PCD 必须在向 PICC 发送下一帧之前等待一个状态字节为 0xAF 的状态帧。

写命令在执行之前需要进行验证，该验证可以是“写”或“读&写”访问的密钥验证，见 3.3。

根据连接到文件的通信设定，PCD 使用明码/MAC/加密方式发送数据。所有加密操作都在 CBC 模式下完成。

对于 MAC 和加密的通信数据填充，必须使整个数据长度为 8 的倍数。填充字节都为 0x00。

如果数据写入备份数据文件，必须执行 CommitTransaction 命令使写入的数据生效，见 4.6.10。AbortTransaction 命令将使所有更改无效，见 4.6.11。

如果数据写入标准数据文件（不具有集成的备份机制），数据将直接编程到文件的可见 NV 存储器中。任何随后的 ReadData 命令都可立即读出新写入的数据。

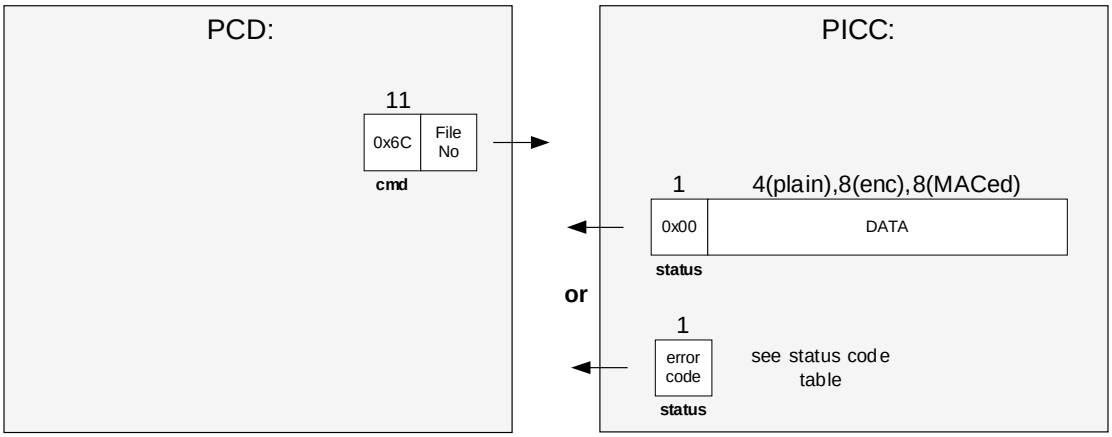
如果使用 MAC/加密通信，数据的有效性通过 PICC 对数据帧末尾的 MAC 或 CRC（包括必要的填充字节）进行检验来确认。如果验证失败（MAC/CRC 与数据不匹配，填充字节无效），PICC 停止 NV 编程并向 PCD 返回一个完整性错误，见 3.4。之后任何可能开始执行的处理都自动中止。

注：写一个标准数据文件时如果出现完整性错误，文件的内容会被破坏。

4.6.3 GetValue

GetValue 命令可以从值文件中读出当前保存的值。

GetValue(FileNo)[2 字节]



该命令唯一参数的长度是 1 个字节，表示文件编号。该参数的范围必须是 0x0~0x07。

PICC 响应值文件的当前值。根据通信模式的不同，数据可采用明码（4 字节）、加密（8 字节加密数据：4 字节值、2 字节 CRC、2 字节填充数据 0x00）或 MAC（4 字节值+4 字节 MAC）方式发送。

值总是 LSB 在前。

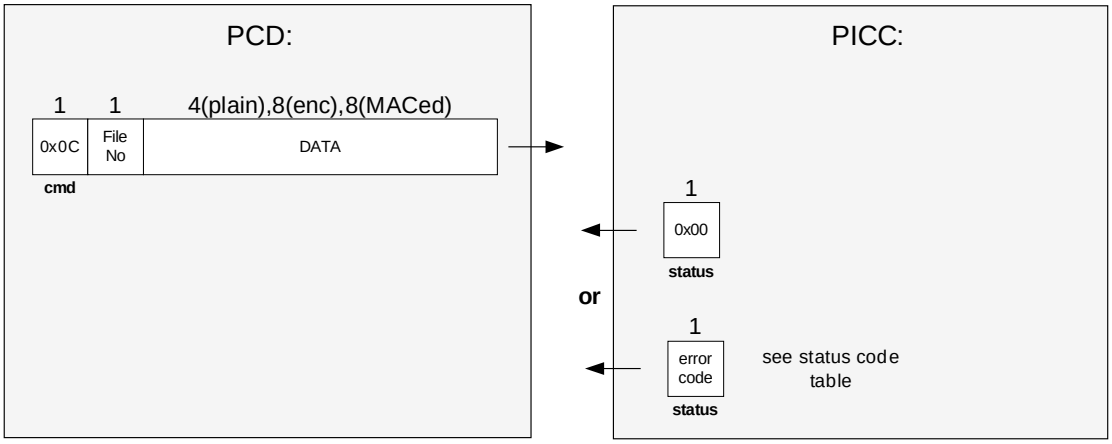
GetValue 命令在执行之前需要进行验证，该验证可以是“读”、“写”或“读&写”访问的密钥验证，见 3.3。

在更新值文件的值之后，但在执行 CommitTransaction 命令之前，GetVetVulue 命令总是得到旧的、未变的数据。

4.6.4 Credit

Credit 命令用于增加值文件中保存的值。

Credit(FileNo, Data)[6/10 字节]



第一个参数长度为 1 个字节，表示文件编号。该参数的范围必须是 0x0~0x07。

该命令使文件中的保存的当前值加上一个特定的数值（4 字节有符号整数），该数值通过数据场发送。Credit 命令只允许使用正整数。

根据通信模式的不同，数据可采用明码（4 字节）、加密（8 字节加密数据：4 字节值、2 字节 CRC、2 字节填充数据 0x00）或 MAC（4 字节值+4 字节 MAC）方式发送。

该值总是 LSB 在前。

必须执行 CommitTransaction 命令使更新的值生效，见 4.6.10。AbortTransaction 命令将使所有更改无效，见 4.6.11。

Credit、Debit 和 LimitedCredit 命令对值的修改是累加的，除非执行了 CommitTransaction 命令。

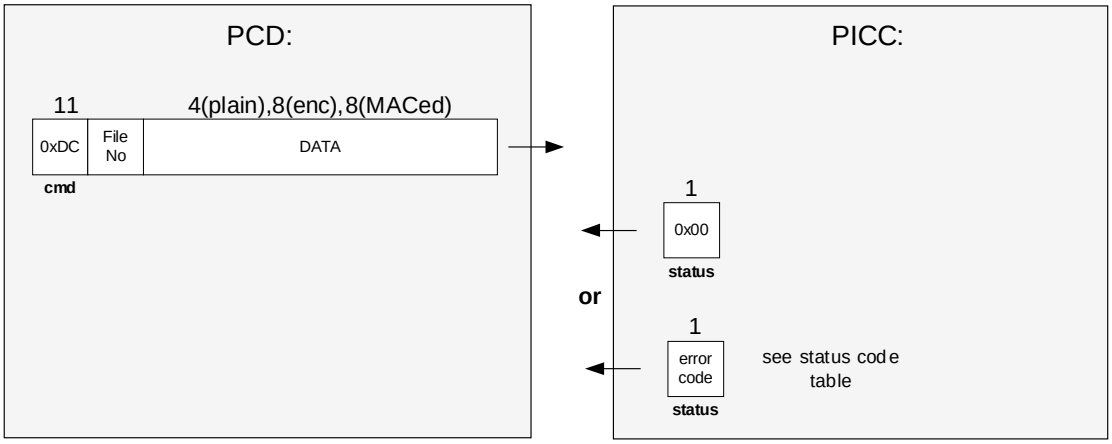
Credit 命令永远不会修改值文件的 LimitedCredit 值。但如果需要将 LimitedCredit 值设置为 0，则可以使用值为 0 的 LimitedCredit。

Credit 命令在执行之前需要进行“读&写”访问的密钥验证，见 3.3。

4.6.5 Debit

Dedit 命令用于减少值文件中保存的值。

Dedit(FileNo, Data)[6/10 字节]



第一个参数长度为 1 个字节，表示文件编号。该参数的范围必须是 0x0~0x07。

该命令使文件中的保存的当前值减去一个特定的数值（4 字节有符号整数），该数值通过数据场发送。Dedit 命令只允许使用正整数。

根据通信模式的不同，数据可采用明码（4 字节）、加密（8 字节加密数据：4 字节值、2 字节 CRC、2 字节填充数据 0x00）或 MAC（4 字节值+4 字节 MAC）方式发送。

该值总是 LSB 在前。

必须执行 CommitTransaction 命令使更新的值生效，见 4.6.10。AbortTransaction 命令将使所有更改无效，见 4.6.11。

Credit、Debit 和 LimitedCredit 命令对值的修改是累加的，除非执行了 CommitTransaction 命令。

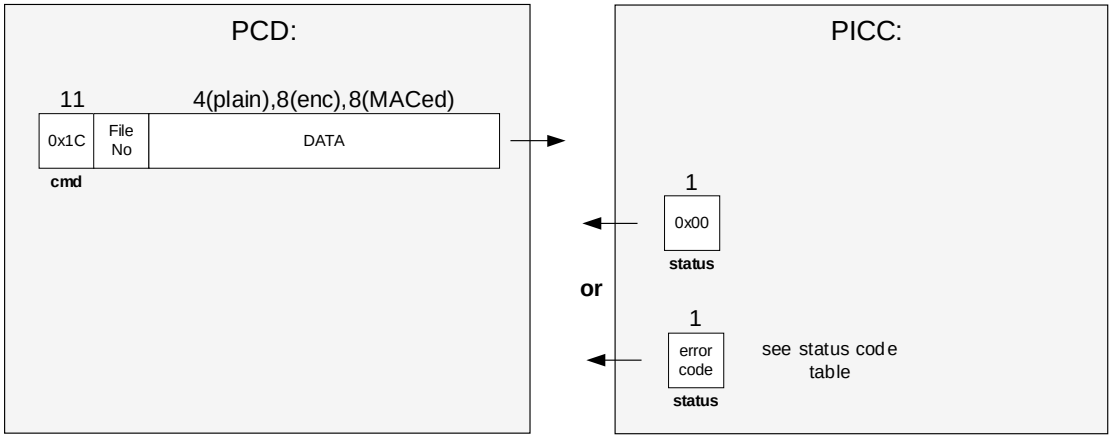
Dedit 命令在执行之前需要进行“读&写”访问的密钥验证，见 3.3。

如果使能 LimitedCredit 特性， LimitedCredit 命令将新的限制值设置为最近一次处理当中 Debit 命令的总和。这确保了 LimitedCredit 命令重新增加的值不可能多于之前的 Debit 处理减去的值。

4.6.6 LimitedCredit

LimitedCredit 命令在对文件没有完全的读&写访问权限的情况下，可以使值文件中保存的值实现有限的增加。这一特性可以在文件创建时使能或禁止。

LimitedCredit(FileNo, Data)[6/10 字节]



第一个参数长度为 1 个字节，表示文件编号。该参数的范围必须是 0x0~0x07。

该命令使文件中的保存的当前值减去一个特定的数值（4 字节有符号整数），该数值通过数据场发送。LimitedCredit 命令只允许使用正整数。

根据通信模式的不同，数据可采用明码（4 字节）、加密（8 字节加密数据：4 字节值、2 字节 CRC、2 字节填充数据 0x00）或 MAC（4 字节值+4 字节 MAC）方式发送。

该值总是 LSB 在前。

必须执行 CommitTransaction 命令使更新的值生效，见 4.6.10。AbortTransaction 命令将使所有更改无效，见 4.6.11。

Credit、Debit 和 LimitedCredit 命令对值的修改是累加的，除非执行了 CommitTransaction 命令。

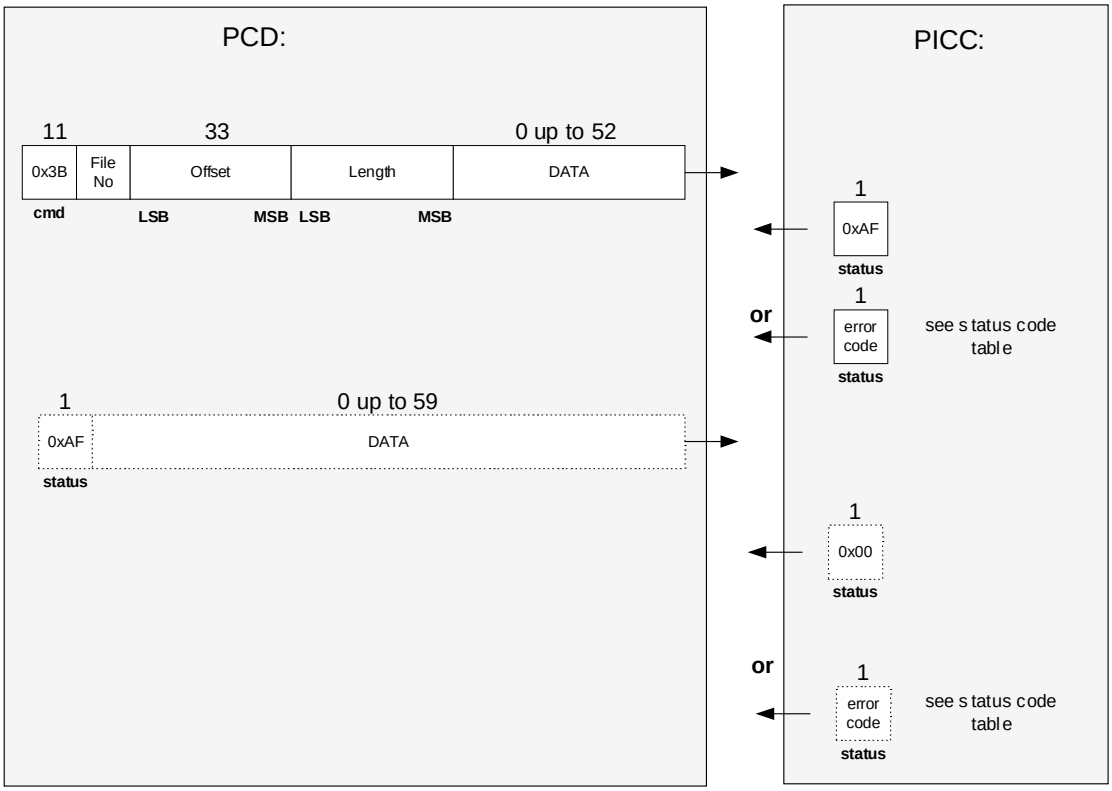
LimitedCredit 命令在执行之前需要进行“读&写”访问的密钥验证，见 3.3。

LimitedCredit 命令的值限制为最近一次处理当中 Debit 命令的总和，至少包含一个 Debit 命令。在执行 LimitedCredit 命令之后，不管曾经重新增加的值是多少，新的限制值总是设置为 0。因此在一次 Debit 处理之后只能使用一次 LimitedCredit 命令。

4.6.7 WriteRecord

WriteRecord 命令用于将数据写入一个循环或线性的记录文件。

WriteRecord(FileNo, Offset, Length, Data) [8+字节]



第一个参数长度为 1 个字节，表示文件编号。该参数的范围必须是 0x0~0x07。

接下来 3 个字节参数表示单个记录中的偏移量（以字节为单位）。该参数的范围必须为 0x00000~ (记录规格-1)。

第三个参数长度也是 3 字节，指定了写入记录文件的数据长度。该参数范围必须为 0x000001~(记录规格-偏移量)。

WriteRecord 命令将一个记录附加在线性记录文件的末尾，如果一个循环记录已满，它擦除并覆盖最早的记录。在数据写入之前，整个新数据被清除。

如果在 WriteRecord 命令之后没有发送 CommitTransaction 命令，同一个文件的下一次 WriteRecord 命令会写入已创建的记录。在发送 CommitTransaction 命令之后，新的 WriteRecord 命令将在记录文件中创建一个新的记录。AbortTransaction 命令将使所有更改无效，见 4.6.11。

在执行 ClearRecordFile 命令之后，但在 CommitTransaction 命令之前，对同一个记录执行的 WriteRecord 命令将失败。

根据连接到文件的通信设定，PCD 使用明码/MAC/加密方式发送数据。所有加密操作都在 CBC 模式下完成。

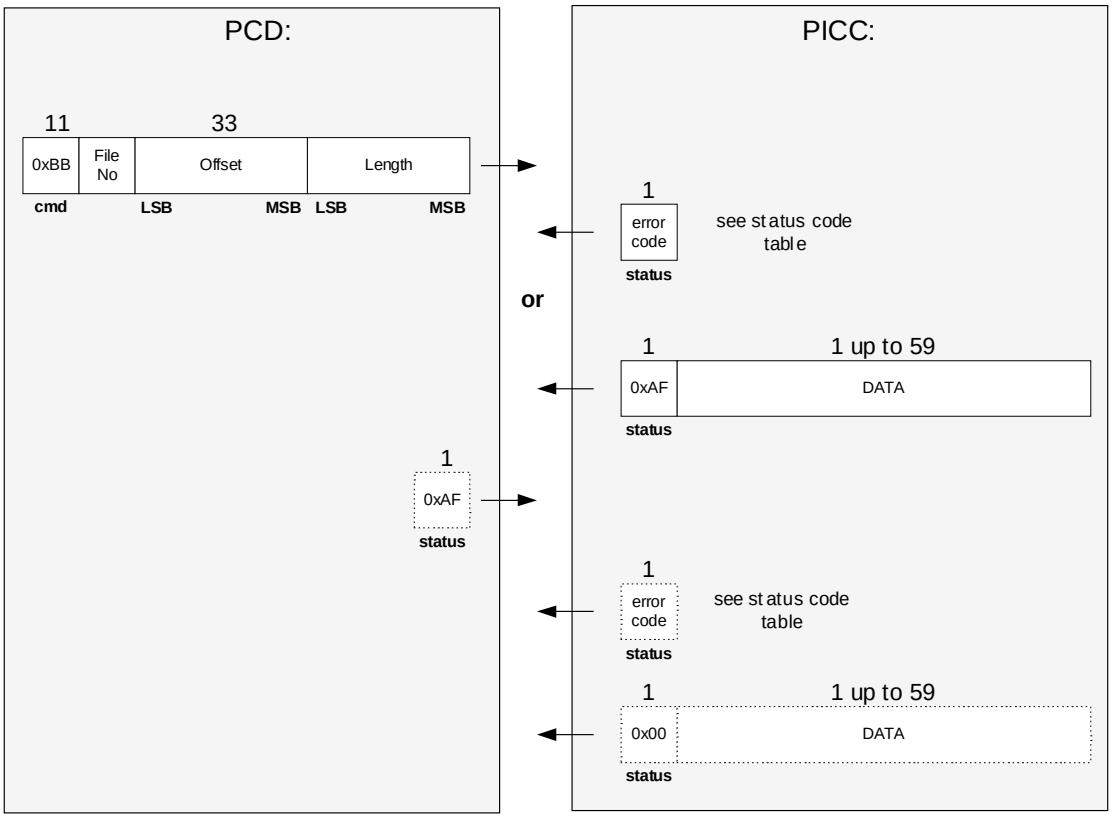
对于 MAC 和加密的通信数据填充，必须使整个数据长度为 8 的倍数。填充字节都为 0x00。

WriteRecord 命令在执行之前需要进行验证，该验证可以是“写”或“读&写”访问的密钥验证，见 3.3。

4.6.8 ReadRecords

ReadRecords 可从一个循环或线性记录文件中读出一套完整的记录。

ReadRecord(FileNo, Offset, Length, Data) [8 字节]



第一个参数长度为 1 个字节，表示文件编号。该参数的范围必须是 0x0~0x07。

接下来 3 个字节参数表示读取的最新记录的偏移量。如果为 0x000000，则读出最后写入的记录。偏移量的范围必须是 0x00000~ (现有记录数-1)。

第三个参数长度也是 3 字节，指定了读取记录的数据长度。PICC 总是以时间顺序发送记录（从最早的记录开始）。如果该参数为 0x000000，则读出包括最早和最新记录在内的所有记录。该参数的允许范围是 0x000000~(现有记录数-偏移量)。注意在循环记录文件中，保存的有效记录的最大数目等于 CreateCyclicRecordFile 命令所指定的记录数减去 1。

对一个空的记录文件执行 ReadRecords 命令会返回出错信息。

根据连接到文件的通信设定，PCD 使用明码/MAC/加密方式发送数据。所有加密操作都在 CBC 模式下完成。

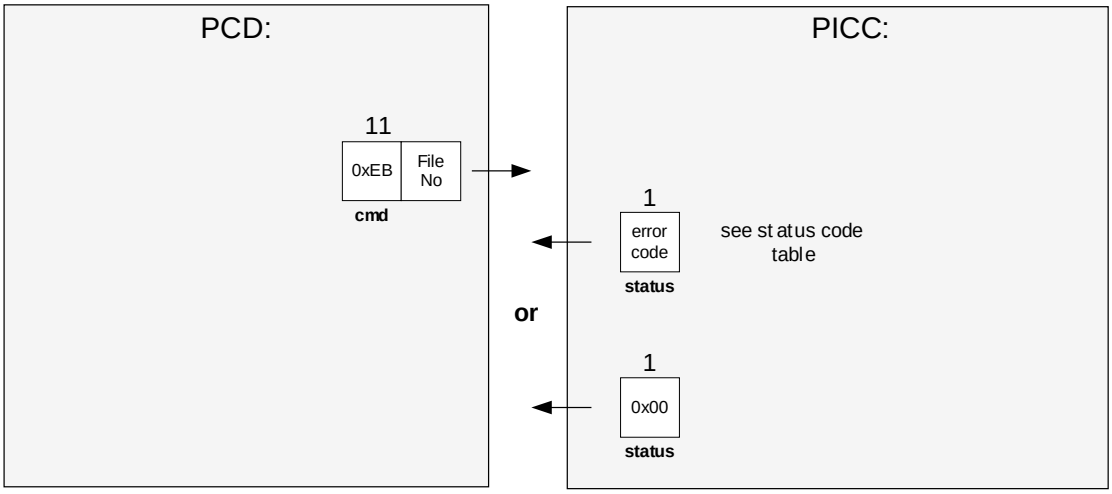
对于 MAC 和加密的通信数据填充，必须使整个数据长度为 8 的倍数。在指定数据长度（命令参数长度≠0x000000）的情况下，填充字节都为 0x00。如果一直读取到文件边界为止（命令参数长度＝0x000000），附加的第一个填充字节为 0x80，附加的其它字节都为 0x00（符合 ISO 9797-1）。

ReadRecords 命令在执行之前需要进行验证，该验证可以是“读”或“读&写”访问的密钥验证，见 3.3。

4.6.9 ClearRecordFile

ClearRecordFile 命令将一个循环或线性记录文件复位为清空的状态。

ClearRecordFile(FileNo)[2 字节]



该命令唯一参数的长度是 1 个字节，表示文件编号。该参数的范围必须是 0x0~0x07。

在执行 ClearRecordFile 命令之后，但在 CommitTransaction 命令之前，执行 WriteRecord 命令将会失败，见 4.6.7。执行 ReadRecords 命令将返回仍然有效的旧数据。

在执行 CommitTransaction 命令之后，ReadRecords 命令将失败，而 WriteRecord 命令将成功。

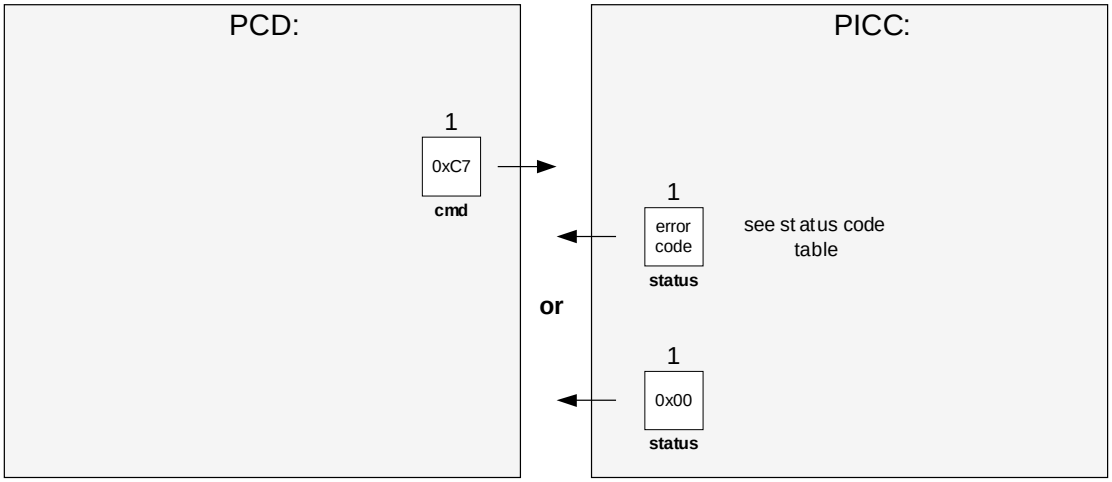
AbortTransaction 命令将使清空命令无效，见 4.6.11。

执行该命令需要对文件的完全“读&写”许可，见 3.3。

4.6.10 CommitTransaction

CommitTransaction 命令使应用中备份数据文件、值文件和记录文件的写操作生效。

CommitTransaction(AID)[1 字节]



该命令不带参数。

在对具有集成的备份机制特性的文件执行写操作后，CommitTransaction 命令使写操作生效。这些文件包括：

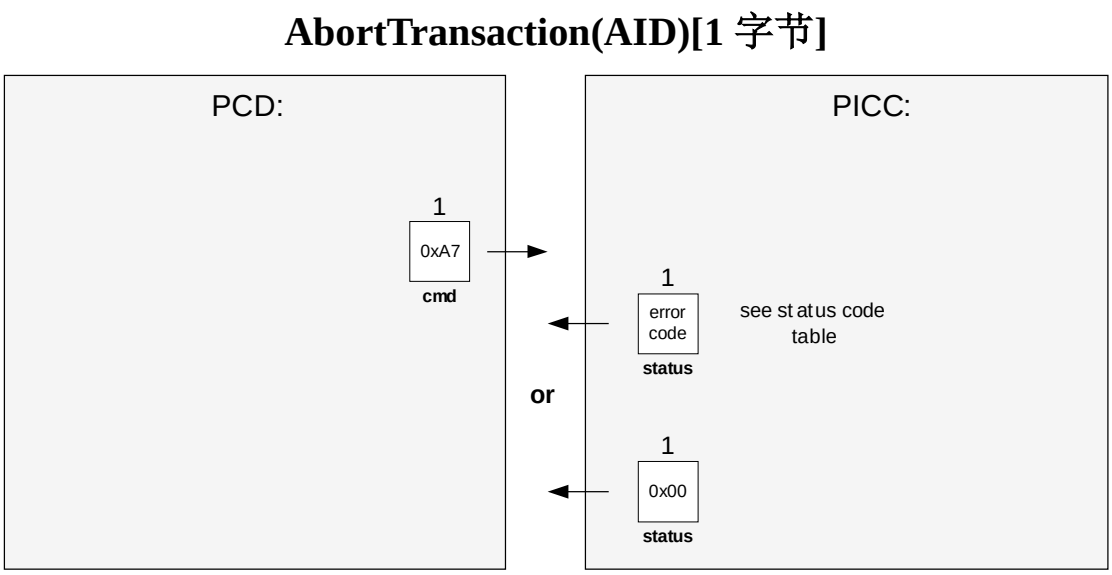
- 备份数据文件
- 值文件
- 线性记录文件
- 循环记录文件

在 ISO 14443-4 的 Deselect 命令之前，或处理另一应用（SelectApplication 命令）之前，CommitTransaction 通常是处理的最后一个命令。

作为与 CommitTransaction 命令在逻辑上相反的命令，AbortTransaction 命令使对具有集成备份机制特性的文件所做的更改无效，见 4.6.11。

4.6.11 AbortTransaction

AbortTransaction 命令使对应用中备份数据文件、值文件和记录文件的写操作无效。



该命令不带参数。

在对具有集成的备份机制特性的文件执行写操作后，AbortTransaction 命令使写操作无效，而不会改变验证状态。这些文件包括：

- 备份数据文件
- 值文件
- 线性记录文件
- 循环记录文件