Mifare DESFire® Specification

Version 1.0 29’th September 2009

Revision History

Version	Date	Author	Description of Changes
1.0	29/09/09	O McLaughlin	Ratified by LASSeO
0.2	28/07/09	O McLaughlin	Minor changes following dialogue with ITSO
0.1	01/06/09	O McLaughlin	Initial version for review.

Preface

Scope

This document is intended as specification for the provision of local authority services on the following DESFire® platforms. MF3.IC.D40, MF3.IC.D21, MF3.IC.41, MF3.IC.D81.

Intended Readership

Developers requiring to integrate with a DESFire® card encoded in accordance with this specification.

Related Documents

[1] UK Government Data Standards Catalogue
[2] ITSO Technical Specification 1000-10 – Interoperable public transport ticketing using contactless smart customer media – Part 10: Customer Media definitions (ITSO TS 1000-10)
[3] Smart Card Systems: Interoperable Citizen Services: Extended User Related Information: Part 1 – Definition of User Related Information and Implementation (CWA 13987-1)
[4] LASSeO Services and Data Definitions
[5] LASSeO Specification for Javacard Implementations
[6] LASSeO Mifare 4k specification
[7] ISO 7816-4 – Inter-industry Commands for Interchange Annex D
[8] Mifare DESFire Functional Specification Rev 3.4

Glossary

Term	*Definition*
AES	Advanced Encryption Standard
AID	Application Identifier
AMK	Application Master Key
ASCII	American Standard Characters for Information Interchange
BCD	Binary Coded Decimal
CCDA	Common Cardholder Data Application
CDO	Constructed Data Object
DES	Data Encryption Standard. Single key and single cryptographic process
3DES	A DES operation using 2 keys and a three stage cryptographic process
3k3DES	A DES operation using 3 keys and a three stage cryptographic process
EF	Elementary File
LASSeO	Local Authority Smartcard Standards e-Organisation
LSB	Least Significant Byte
LSb	Least Significant Bit
MAC	Message Authentication Code
MAD3	Mifare Application Directory for DESFire cards
MSB	Most Significant Byte
MSb	Most Significant Bit
NSCP	National Smartcard Project
PICC	Proximity Card
PIN	Personal Identification Number
RFU	Reserved For Future Use
TLV	Tag, Length, Value
USID	Unique Services Identifier

Contents

1 Introduction

This is a specification for storing and accessing the data items defined in the LASSeO Services and Data Specification on DESFire® platforms. The DESFire® platform is a memory technology like the Mifare Classic, but with enhanced file handling and security features. This specification is designed to be compatible with all DESFire® variants.

Table 1. DESFire® Platform Differences

	MF3IC40	MF3IC21–EV1	MF3IC41-EV1	MF3IC81-EV1
Memory Size	4k	2k	4k	8k
Free Space	4096 bytes	2272 bytes	4832 bytes	7936 bytes
Max. Applications	28	28	28	28
Max. Files per application	16	32	32	32
Crypto	DES,TDES	DES, TDES, 3KTDES, AES	DES, TDES, 3KTDES, AES	DES, TDES, 3KTDES, AES
Support for ISO 7816 Cmds	Very Limited	Yes	Yes	Yes

**** All values shown in the table above are decimal***.

To ensure that this specification is compatible with all the DESFire® platforms listed above, certain preconditions must be set.

Although the EV1 range of DESFire® cards supports a substantial number of ISO 7816-4 commands, this is not the case with the MF3ICD40 platform. This specification is therefore based on the proprietary command set which is common to all platforms.

Similarly, the 'compatible crypto mode' must be used. This supports TDES and the Authenticate command 0Ah. AuthenticateISO and AuthenticateAES are not supported.

The Directory application contains certain public data which can be read without any authorisation. Other citizen data stored within Free Access Files inside applications can be accessed without authentication and data transmission will be in plain mode without encryption or MACing. Citizen data stored within individual Elementary Files is subject to scheme specific conditions. In deciding on which items to store in elementary files (EFs), and the access conditions pertaining to them, inter-operability requirements should be considered as well as security needs.

All numeric values appearing in tables in this document are hexadecimal unless otherwise stated. Numeric values in normal text are decimal unless denoted as hexadecimal with an 'h' suffix.

Note that MAD3 is not supported and randomID should be disabled.

2 Applications and Files

Unlike Mifare Classic, DESFire® comes with a flexible filing system based on distinct applications and files contained within these applications. Every Service, as defined in the LASSeO Services and Data definitions document, will be an application on the card, and all the data items for a Service will be held in files within that application.

There will be a dedicated Directory application which will contain directory information on mapping Service USIDs on to applications, and other card level data such as cardholder number and expiry date.

2.1 DESFire® AIDs

DESFire® allows up to 32 applications on a card. Every application has a three byte Application Identifier (AID) by means of which it can be found and selected.

An already reserved Mifare® Classic AID can be used to create 16 unique DESFire® application AIDs. This means that there is no need to register any further AIDs for DESFire® use as the following Mifare® Classic AIDS are already registered; 4011h, 4012h and 4013h.

The mapping to 3 byte AIDs is as follows.

DESFire AID byte 1		DESFire AID byte 2		DESFire AID byte 3	
Nibble 0	Nibble 1	Nibble 2	Nibble 3	Nibble 4	Nibble 5
F	Mifare Classic AID				0 - F

Mifare® Classic AIDs 4011h, 4012h and 4013h generate the following 48 DESFire® AIDs.

F40110	F40111	F40112	F40113	F40114	F40115	F40116	F40117
F40118	F40119	F4011A	F4011B	F4011C	F4011D	F4011E	F4011F
F40120	F40121	F40122	F40123	F40124	F40125	F40126	F40127
F40128	F40129	F4012A	F4012B	F4012C	F4012D	F4012E	F4012F
F40130	F40131	F40132	F40133	F40134	F40135	F40136	F40137
F40138	F40139	F4013A	F4013B	F4013C	F4013D	F4013E	F4013F

These AIDs will be used as follows:

F40110	Reserved for the Service Directory Application
F40111	Reserved for the CCDA Application
F40112 – F4012F	Available for Service Applications
F40130 – F4013F	RFU

The CCDA application has been given a dedicated AID as it is a mandatory application. All other applications are optional.

DESFire® AIDs in the range F40112h- F4012Fh are available to identify citizen services on a DESFire® card. Which of these AIDs are used to identify which USIDs is a scheme specific issue.

Please note that when creating applications using the Create Application command the AID is specified in little endian format.

CMD	AID	Key Settings 1	Key Settings 2
CA	2F01F4	xx	yy

This example shows the creation of an application with AID F4012F. Also note that for compatibility across the range of DESFire® platforms bits 6 & 7 in Key Settings 2 should be set to '00' indicating DES and 3DES operation.

2.2 The Service Directory Application

The AID F40110h will always identify the Service Directory application. The application will contain four mandatory EFs.

File Description	File ID	File Type
Services Directory	00	Linear Record File
CardHolder Number	01	Standard File
Card Expiry Date	02	Standard File
Specification Versions	03	Backup File
RFU	04 -> 09	RFU
Scheme Specific	0A -> 0F	As required

File IDs 04 to 09 are Reserved for Future Use. However, schemes may create scheme specific files in this application with IDs in the range 0A to 0F. The nature and content of these files is beyond the scope of this specification.

2.2.1 Services Directory File:

This file maps the USIDs defined in document [4] on to application AIDs. The size of this file may change during the life of a card as services may be added or removed. The CCDA application need not be included as it will always have the same AID as described in section 2.1. The format of the file is as shown in the example below:

Example Directory File 0

Record No.	USID – 2 bytes	AID – 3 bytes
1	0002	F40114
2	0007	F4013C
3	0004	F4011A
4	FFFF (RFU)	FFFFFF (RFU)
5	FFFF (RFU)	FFFFFF (RFU)
6	FFFF (RFU)	FFFFFF (RFU)

The Services Directory file facilitates interoperability between different schemes in which the mapping between USIDs and AIDs may be different. When creating this file it is recommended that space be reserved for a sufficient number of entries to cater for any future expansion. Reserved records should be padded with FFh. Please see section 3 for guidance on how many records this file may need to support.

A scheme may use any of the 30 AIDs in the range F40112h – F4012Fh which are reserved for citizen applications. All Linear record files are treated as backup files. Following any change to the Services Directory File a Commit Transaction command must be issued to activate the backup process.

This file is created with the CreateLinearRecordFile command.

Cmd	File No	Comms	Access	Record Size	Max Records
C1	00	00	Fx Ex*	5	As required

This file is read using the ReadRecord command and written using the WriteRecord command. When writing data CommitTransaction must be used to complete the writing and backup process. Total NV space used will be double that needed for the actual data due to the creation of a backup file.

*Note these two bytes are sent LSB first. 'x' denotes scheme specific key number.

2.2.2 CardHolder Number:

This file is included in the Services Directory for convenience and ease of access. It is also part of the CCDA data set. The standard size and format of the Cardholder Number is as described in document [4] in Service ID 0001h, Tag DF23h. However, there are some schemes with non-standard numbering systems and these can also be accommodated. The size of this example file is 10 bytes, one byte for the length of the following data, one byte for the format identifier and 8 bytes of data.

Example File 1

Length	Format	Cardholder Number							
09	01 (BCD)	63	45	89	12	78	90	23	05

A standard file can be used for this data as the contents will not be changed during the lifecycle of the card. This file is created with the CreateStdDataFile command.

Cmd	File No	Comms	Access	File Size
CD	01	00	FF EF	0A 00 00*

This file is read using the ReadData command and written with the WriteData command.
* The file size information is presented LSB first. Although the file itself is only 10 bytes long, a minimum of 32 bytes of NV memory will be allocated.

2.2.3 Card Expiry Date:

This item is also included for convenience. The size and format of the Card Expiry Date is as described in document [4] in Service ID 0001h, Tag DF63h. The overall size of this file is 6 bytes, one byte giving the length of the following data, one byte for the format identifier and 4 bytes of data.

Example File 2

Length	Format	Date			
05	02 (DATE)	20	09	05	15

A standard file can be used for this data as the contents will not be changed during the lifecycle of the card. This file is created with the CreateStdDataFile command.

Cmd	File No	Comms	Access	File Size
CD	02	00	FF EF	06 00 00*

This file is read using the ReadData command and written with the WriteData command.
* The file size information is presented LSB first. 32 bytes of NV memory will be allocated.

2.2.4 Specification Versions:

This item can be used to notify the versions of the relevant specifications which were used for card encoding. There are two relevant specifications. These are the DESFire® specification (this specification) and the Services and Data definitions.

Example File 3

Length	Format	DESFire		Services and Data	
05	01 (BCD)	01	07	10	05

Each version identifier consists of two bytes BCD. The first byte indicates the major version number, and the second byte the minor version number. The above examples show versions as follows:

DESFire 1.7
Services & Data Definitions 10.5

A backup file type should be used as the data in this file may change if a card is re-encoded to a later specification. This file is created using the CreateBackupDataFile command.

Cmd	File No	Comms	Access	File Size
CB	03	00	Fx Ex	06 00 00*

This file is read using the ReadData command and written with the WriteData command. When writing data CommitTransaction must be used to complete the writing and backup process. Total NV space used will be doubled due to the creation of a backup file.

* The file size information is presented LSB first. 64 bytes of NV memory will be allocated.

2.2.5 File Access and Communication Conditions:

The table below summarises the communications and access settings for these files.

Description	File ID	Comms Settings	Access Conditions			
			R	W	RW	Ch
Services Directory	00	00	0xE	*K	0xF	*K
CardHolder Number	01	00	0xE	0xF	0xF	0xF
Card Expiry Date	02	00	0xE	0xF	0xF	0xF
Version Information	03	00	0xE	*K	0xF	*K

*K denotes the scheme specific DES or TDES key number in range 0 to 13.
The Communications Settings specify data transmission in clear with no encryption or MACing.

The access conditions allow unconditional Read Access to the files in the Services Directory. Only the Directory File can be written to. Access conditions are coded on to two bytes as follows;

MSb															LSb
0F	0E	0D	0C	0B	0A	09	08	07	06	05	04	03	02	01	00
Read				Write				Read & Write				Change			

Each 4 bit nibble references a permission in the range 0 to 15 with the following meaning.

Value	Meaning
0	Use Key 0
1	Use Key 1
2	Use Key 2
3	Use Key 3
4	Use Key 4

5	Use Key 5
6	Use Key 6
7	Use Key 7
8	Use Key 8
9	Use Key 9
A	Use Key 10
B	Use Key 11
C	Use Key 12
D	Use Key 13
E	Free Access
F	No Access

2.3 Service Application Structure

2.3.1 A DESFire Service Application

An application will contain two mandatory files, an Index File and a Free Access File and an optional number of Elementary Files containing individual data items.

Description	File ID	File Type
Index File	00	Linear Record File
Free Access File	01	Backup File
Individual Data File(s)	02 -> Max Files-1	Backup File

It must be remembered that the DESFire MF3IC40 only permits up to 16 files per application, so the maximum number of individual data files on this platform would be only 14, whereas it would be 30 on all other DESFire EV1 platforms.

2.3.2 The Index File

The Index file maps the tagged items within a service on to the files in which they are to be found. In most circumstances the majority (or all) of items will be located in the Free Access File, but others may reside in their own files with individual access rights.

Example Index File for CCDA Service

Tag (2 bytes)	File ID(1 byte)

5F2B	01
DF23	01
DF32	01
DF33	01
DF56	02
DF57	03
DF6D	01
FFFF (RFU)	FF (RFU)
FFFF (RFU)	FF (RFU)

In the above example the majority of items are located in the Free Access File with the exception of address information accessed via tags DF56h and DF57h. Space has also been reserved in this Linear Record File to allow new entries to be added. The index file must be updated whenever data items are added to the card or deleted.

The decision on where to store data items and the access permissions associated with them is largely a scheme specific issue, although document [4] makes recommendations about this. This file is created with the CreateLinearRecordFile command.

Cmd	File No	Comms	Access	Record Size	Max Records
C1	00	00	Fx Ex*	3	As required

This file is read using the ReadRecord command and written using the WriteRecord command. When writing data CommitTransaction must be used to complete the writing and backup process. Total NV space used will be doubled due to the creation of a backup file.
*Note these two bytes are sent LSB first. 'x' denotes a scheme specific key number.

2.3.3 The Free Access File

This file contains data in the form of a constructed BER-TLV Data Object as described in section 5 but without the final checksum. The same data structure would be represented as below on a DESFire platform.

Example of a Free Access File Contents

E0 **45** 50 05 00 43 43 44 41 DF 23 09 01 63 37 10 00 00 04 13 01 DF 32 0A 00 46 72 65 64 65 72 69
63 6B DF 33 08 00 59 65 75 6C 65 74 74 5F 2B 05 02 19 39 05 16 DF 56 03 00 32 34 DF 57 09 00 50
45 31 35 20 39 4C 58 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

This data block is parsed as follows.

CDO Tag	CDO Length	Data Tag	Length	Format	Contents	Meaning
E0	45					

		50	05	00	43 43 44 41	Service Label ‘CCDA’
		DF23	09	01	63 37 10 00 04 13 01	Card No. ‘63371000041301’
		DF32	0A	00	46 72 65 64 65 72 69 63 6B	Forename ‘Frederick’
		DF33	08	00	59 65 75 6C 74 74	Surname ‘Yeultt’
		5F2B	05	02	19 39 05 16	DOB ‘16/5/1939’
		DF56	03	00	32 34	House No. ‘24’
		DF57	09	00	50 45 31 35 20 39 4C 58	PostCode ‘PE15 9MX’

Although the Service Label (Tag 50h) must appear first in the order of data tags, the ordering of the remainder is not mandated.

Note that the CDO length value (45h underlined) is the same as the Mifare Classic example in section 5. This is because the service checksum required on the Mifare 4k platform resided in a separate primitive data object outside the CDO and did not therefore form part of its length.
When creating this file the requirement for future expansion should be considered. In the above example the file size is 128 bytes of which only 71 bytes are in use. Unused space will be padded with FFh. When the file contents are changed the entire CDO must be rewritten.

Even if the original size of the Free Access File is exceeded there are two options available. The file can be deleted and recreated with the same file ID and a larger size. However, this will result in the loss of the original NV memory as there is no memory recovery on DESFire platforms. The second option is to locate some data items in individual EFs with the same permissions as the Free Access File. As long as the Index File is kept up-to-date there will be no difficulty in locating these items.

This file is created using the CreateBackupDataFile command.

Cmd	File No	Comms	Access	File Size
CB	01	00	Fx Ex	80 00 00*

This file is read using the ReadData command and written with the WriteData command. When writing data CommitTransaction must be used to complete the writing and backup process. Total NV space used will be double whatever is specified when the file is created due to the existence of a backup file.

* The file size information is presented LSB first. The file size of 128 bytes is the recommended minimum. The total NV space used will be doubled due to the creation of a backup file.

2.3.4 Elementary Data Files

Elementary Data Files contain a single data item. In the example below file ‘02’ holds the data for CCDA item DF56h, House Name/Number.

Example EF file structure

Length	Format	Data
13	00 (ASCII)	44,45,20,53,41,49,4C,57,41,59,20,54,45,53,53,41,43,45

'Length' specifies the length of actual data within the file including the data format byte.
'Format Byte' declares the format type of the data
'Data' is the data itself.

In some circumstances it may be expedient to create files that are larger than the immediate data requirement if contents are likely to change during the life of the card. As 32 bytes of NV is always allocated at file creation there is no disadvantage in using this as a default size. Unused space should be packed with FFh.

This file is created using the CreateBackupDataFile command.

Cmd	File No	Comms	Access	File Size
CB	02	00	Fx yx*	As required

This file is read using the ReadData command and written with the WriteData command. When writing data CommitTransaction must be used to complete the writing and backup process. Total NV space used will be doubled due to the creation of a backup file.

* 'x' represents a scheme specific key number. 'y' represents a User PIN if required.

2.3.5 Access Conditions:

The table below summarises the communications and access settings for these files.

Description	File ID	Comms Settings	Access			
			R	W	RW	Ch
Index File	00	00 (Plain)	0xE	*K	0xF	*K
Free Access File	01	00 (Plain)	0xE	*K	0xF	*K
Data Files	0x	00 (Plain)	*P	*K	0xF	*K

*K denotes the diversified, scheme specific DES or TDES write key in the range 0 -13
*P denotes the read permission associated with individual EF files. This may be 'Free Read' or subject to authentication depending on scheme requirements.

3 Memory Considerations

The available space on a particular platform is important if other non-LASSeO applications such as ITSO are also required. The ITSO specification stipulates that the default ITSO configuration requires 1760 bytes of memory. Because citizen data is to be stored in backup files, which effectively require double the amount of memory as standard files, space may become a significant issue on some platforms where ITSO is also resident.

Table 2. Space and Application Requirements

	MF3IC40	**MF3IC21–EV1**	**MF3IC41-EV1**	**MF3IC81-EV1**
Free Space	4096 bytes	2272 bytes	4832 bytes	7936 bytes
Max.Citizen applications	7	4	8	15
With ITSO application	4	N/A	5	12

**** All values shown in the table above are decimal***.

The table above is a guide to how many citizen applications can be stored on a particular platform besides the Service Directory. The estimate is based upon a typical application with an Index File of 2 blocks and a Free Access File of 4 blocks. It does not include any provision for individual data EFs within applications. Each additional EF will require at least 64 bytes on NV memory.

Because NV memory is always allocated in 32 byte blocks, this could be used as a default size for many single item EFs even though all the space is not required. As some data items will change within the lifecycle of a card this extra space may be useful where data items increase in size. The only guidance available for allocating space for specific items is in the Government Data Catalogue where maximum sizes for some variable items are specified. However, these sizes are in many cases inappropriate to smartcard platforms where space may be at a premium. For example, Forename (DF32h) has a stated maximum size of 80 characters. On a DESFire platform using a Backup file, adherence to this limit would require the allocation of 192 bytes of NV memory for this one item alone. This is clearly excessive.

In most circumstances a minimum 32 byte allocation (64 bytes with backup) should be sufficient. For items that could exceed this minimum a judgement needs to be made on how much space might be needed. Even in a scenario where not enough space has been allocated there is always the option to delete the file and create a new one with sufficient room. The only caveat here is that the original NV memory will be lost when the file is deleted.

4 Security Strategy

This specification seeks to define a minimum set of requirements to support scheme interoperability. It does not seek to lay down specific rules for ensuring that schemes have adequate or appropriate security provision.

To promote interoperability at a basic level it is essential that all terminals are able to read the open data sets on the card without having to perform cryptographic operations. The only alternative to this approach is to mandate the use of SAMs for all terminals and impose a security regime that all schemes should follow. For open access it is essential that read access to open citizen applications and data is not protected by scheme specific keys.

4.1 PICC Master Key

Every DESFire card has a PICC Master Key. This is identified as key 0 at the card level which is identified as an application with an AID of 000000h. This key controls critical functions at the card level such as creating and deleting applications and changing PICC key settings. For scheme integrity it is important that this key is set and strictly controlled at the scheme level.

4.2 Application Master Keys

Every application can have its own Master Key which is identified as key 0 within that application. This key should be used to control the creation and deletion of files within an application. If this permission is not specified then the integrity of scheme applications may be compromised.
To avoid the need for controlling a number of different application keys it may be practical to use the same AMK for securing data in all non-value citizen applications.

4.3 Proprietary Application Master Keys

Some applications which store or control value, such as e-purses or parking applications should have their own proprietary keys for write access. This is in line with published recommendations for other platforms. Schemes wishing to co-operate in the use of such applications would need to ensure that they share the same cryptographic secrets although this is often provided by a third party.

4.4 User Keys

Some schemes may wish to protect individual data items within applications so that read access is at the discretion of the card holder. With javacards it is possible to protect data with the use of PINs associated with individual EFs. This mechanism is not available on the DESFire although it is possible to map

PINs on to single DES keys. This will lead to relatively weak DES keys being deployed but the level of security achieved is certainly not less than with a conventional PIN mechanism. How this mapping is achieved is a scheme issue.

4.5 Platform Compatibility

In order to accommodate a number of different DESFire platforms in a scheme where common access permission is required then only the basic Authenticate Command (0Ah) can be used. This is because the MF3IC40 platform does not support 3K3DES or AES cryptography. Security features must also be restricted to the use of TDES crypto in DESFire native mode.

The ITSO specification makes the same stipulation. See document [2].

Table 3. Example Scheme Key Strategy

Application		Read	Write
Service Directory			
	All files	Free	Scheme Key
CCDA			
	Index File	Free	Scheme Key
	Free Access	Free	Scheme Key
	EF1	User PIN	Scheme Key
	EF2	User PIN	Scheme Key
Special Needs			
	Index File	Free	Scheme Key
	Free Access	Free	Scheme Key
Library			
	Index File	Free	Scheme Key
	Free Access	Free	Scheme Key
Parking			
	Index File	Free	Proprietary Key
	Free Access	Free	Proprietary Key
ITSO			
	ITSO Files	Free	ITSO Key

In this example the same Scheme Key is used across a number of applications involving non-value transactions.
It is highly recommended that Scheme Keys be diversified on a per card basis.

NB: Although inter-operability across all DESFire platforms requires the use of Native Mode, this does not prevent proprietary applications employing more secure cryptography such as AES on EV1 platforms where this is considered to be necessary.

5 Implementation Approach

On a Mifare Classic Platform it is only practical to store application data in a single memory structure with the same access conditions for all items. Mifare Classic also has no integral filing system therefore data can only be stored and referenced by its actual location in memory.
Below is a block of memory containing a CCDA service within a Constructed Data Object (CDO). The absolute address of this service is known from the Services Directory on the Mifare card.

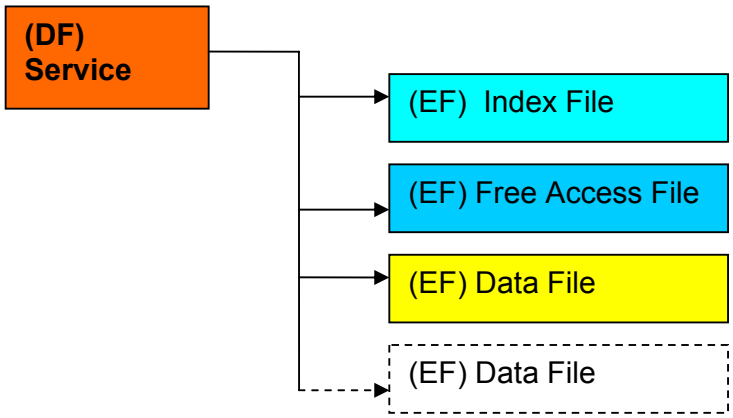
Block	Data
128	E0 45 50 05 00 43 43 44 41 DF 23 09 01 63 37 10
129	00 00 04 13 01 DF 32 0A 00 46 72 65 64 65 72 69
130	63 6B DF 33 08 00 59 65 75 6C 65 74 74 5F 2B 05
131	02 19 39 05 16 DF 56 03 00 32 34 DF 57 09 00 50
132	45 31 35 20 39 4C 58 C0 02 FB 15 00 00 00 00 00
133	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
134	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
135	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

This structure is described fully in document [6], the Mifare 4k specification. Because of these constraints only certain kinds of citizen data are deemed suitable for storage on a Mifare Classic platform.

At the other end of the scale document [5] describes how citizen data is stored on a processor card such as a JCOP31. The power and size of these platforms allows for sophisticated filing and security systems as well as other desirable features such as garbage collection whereby memory occupied by deleted items can be recovered. It is also possible to store individual data items within a service in individual files with their own access requirements.

Each service application contains two mandatory files. These files are the Index file, and the Free Access file. The Index file stores the location and display description of all data items held by the service application. The Free Access file contains all the data items which are freely readable. All other data items are held in EF files with their individual access conditions.

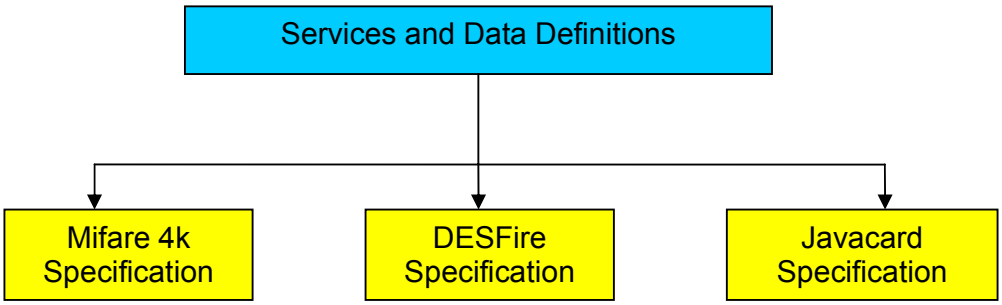
Figure 1. Service Application Contents on Javacards



In terms of capability, the DESFire® platforms lie somewhere in between the two. Although the DESFire® is still essentially a memory card, the filing system and enhanced security features allow for a more flexible file-based approach. However, the relatively limited memory sizes and lack of garbage collection facilities mean that a full implementation along javacard lines is impractical. What is therefore proposed is an optimised version better suited to the memory limitations of the DESFire®.

It is important to remember that whatever the means of access or storage the service USIDs, data tags and formats described in document [4] are consistent across all the LASSeO platform specifications.

Figure 2. The LASSeO Family of Specifications



** End of Document **