

---

# MULTOS KMA File Interface Formats (external)

---

Document Number	maos-gkc-spc-002/1
Version	6-0
Date	14 Nov 2003
Author	Andy Calvert

## Copyright

© Copyright 2003 MAOSCO Ltd. This document is confidential. No part of this document may be reproduced, published or disclosed in whole or part, by any means: mechanical, electronic, photocopying, recording or otherwise without the prior written permission of MAOSCO Ltd. This document is made available under the terms of the confidentiality agreement signed with MAOSCO Ltd. and must not be disclosed to any other person or organisation otherwise than as set out in the terms of that confidentiality agreement.

## History

Version 1-0		Following formal review
Version 1-1	12/01/98	Incorporating changes required by initial implementation design
Version 1-2	20/01/98	Added ALU formats & Appendix 3
Version 2-0	27/01/98	Following review
Version 2-1	23/03/98	Incorporating feedback from Bureau, and additional file formats ready for the CA systems
Version 2-2	31/03/98	Incorporating feedback from developers
Version 2-3	31/05/98	Additional information on ALC / ADC formats, removal of Appendix 2 (information now available in greater detail elsewhere). Addition of extra files and file types. Major restructure of document in attempt to improve readability. Added support for MULTOS 4 certificate structures. Released for internal MXI / MAOSCO review.
Version 3-0	12/06/98	Incorporating minor review comments received. Issued for external distribution
Version 3-1	28/08/98	Incorporating review comments from MAOSCO to improve readability, and from early implementers to provide clarifications. Minor corrections to support released MULTOS 4 specifications.
Version 3-2	26/08/99	To incorporate feedback from the MGKC, COO and M-SPI 2 developments. Common elements moved into section 7. Format of naming fields made more consistent. For internal review only.
Version 4-0	22/10/1999	To include feedback from AH, RC and CL. Issued for external distribution.
Version 4-1	15/06/2000	To incorporate updated file formats for Application Registration, ALC Request, Card Block Request and Response. Also minor editorial throughout.
Version 4-2	03/05/2001	Incorporating new MSM Controls request file formats. Introduction of version byte into all request files. Removal of obsolete file formats. For internal review only.
Version 4-3	19/09/2001	Addition of file formats for importing application details into M-SPI, importing application provider key file into M-SPI, and exporting the MSM and / or mkd_pk_c from M-SPI. Editorial changes from CA to KMA. Separated into maos-gkc-spc-002/1 for external parties and maos-gkc-spc-002 for all MXI / MAOSCO developers.
Version 4-4	20/11/2001	Issued for review
Version 5-0	04/01/2002	Incorporating review comments. For external distribution.

# Proprietary

---

Version 5-1	03/11/2003	Introducing new files for M-SPI v4-2. Separated document into three sections for different M-SPI products. Issued for internal review
Version 6-0	14/11/2003	Updated after formal review. Issued for external release.

## Preface

This document defines the data files used to communicate with a MULTOS KMA (Key Management Authority).

## Objectives

The objective is to ensure that Bureaux and Issuers can easily communicate with a KMA and a KMA can efficiently process orders.

## Scope

This document covers only the external interfaces to a KMA from Issuer and Bureau M-SPI (MULTOS-Service Provider Interface) tools. The equivalent document (maos-gkc-spc-002) describes the internal interfaces within a KMA.

## Audience

Anyone involved in the design and implementation of elements within the MULTOS scheme.

## Related Documents

MULTOS KMA File Interface Formats (internal)  
maos-gkc-spc-002

Integrated circuit card(s) with contacts - Part 3: Electronic Signals and transmission protocols  
ISO/IEC 7816-3

Integrated circuit card(s) with contacts - Part 5: Registration System for international applications in integrated circuit(s) cards  
ISO/IEC 7816-5

Identification cards -- Contactless integrated circuit(s) cards -- Proximity cards -- Part 4: Transmission protocol  
ISO-IEC 14443-4

Secure Hash Standard (SHS)  
FIPS-PUB 180-1

## Assumptions

It is assumed that the reader knows the basics of MULTOS and KMA Interfacing.

## Issues

None.

## Table Of Contents

<b>1</b>	<b>Introduction .....</b>	<b>1</b>
<b>2</b>	<b>File Types .....</b>	<b>2</b>
<b>3</b>	<b>Files used to communicate from Issuer / Bureau M-SPI 4-0 to a KMA outer-office system (COO).....</b>	<b>4</b>
3.1	MSM Request File (version 04) .....	5
3.2	ALC Request Files (version 05).....	8
3.3	ADC Request files (version 03) .....	11
3.4	Application Registration (version 04) .....	13
3.5	Issuer Key Addition (version 02) .....	18
3.6	Issuer Key Deletion (version 02).....	20
3.7	Bureau Association Registration.....	21
3.8	Card Block / Unblock Request File (version 02) .....	22
3.9	Bureau Key Addition .....	24
3.10	Bureau Key Deletion.....	26
3.11	MSM Controls Data List Files (version 02) .....	27
3.12	ALC Response (version 03).....	30
3.13	ADC Response (version 03).....	34
3.14	Card Block / Unblock Response File (version 02).....	38
<b>4</b>	<b>Files used to communicate from Issuer / Bureau M-SPI 4-2 to a KMA outer-office system (COO).....</b>	<b>41</b>
4.1	MSM Request File (version 04) .....	42
4.2	ALC Request Files (version 05).....	45
4.3	ADC Request files (version 03) .....	48
4.4	Application Registration (version 07) .....	50
4.5	Issuer Key Addition (version 03) .....	56
4.6	Issuer Key Deletion (version 03).....	58
4.7	Bureau Association Registration (version 02) .....	60
4.8	Card Block / Unblock Request File (version 02).....	61
4.9	Bureau Key Addition (version 02).....	63
4.10	Bureau Key Deletion (version 02) .....	65
4.11	MSM Controls Data List Files (version 02) .....	67
4.12	ALC Response (version 03).....	70
4.13	ADC Response (version 03).....	74
4.14	Card Block / Unblock Response File (version 02).....	78
<b>5</b>	<b>Additional MAOSCO Advisory File Formats.....</b>	<b>81</b>
5.1	ALU Request List File.....	82
5.2	ALU Data List Files (version 01) .....	84

5.3	ALU Data List Files (version 02) .....	86
<b>6</b>	<b>Key Disk Formats .....</b>	<b>90</b>
6.1	TKCK.....	90
6.2	Hash Modulus.....	91
<b>7</b>	<b>Request and Management Data Signatures .....</b>	<b>92</b>
7.1	Signature Generation .....	92
7.2	Signed Data .....	93
7.3	Signature Ciphertext.....	93
7.4	Pictorial Explanation .....	94
<b>8</b>	<b>DATA DICTIONARY.....</b>	<b>95</b>

## 1 Introduction

This document defines the external interface to a KMA for data transfer to / from Issuers, Bureaux and similar bodies.

It defines the format and content of the files used to transfer information. Most of the files in this document are generated by, or input into, the Issuer or Bureau variant of M-SPI.

In this document, numbers prefixed with 0x are hexadecimal representations of single or multiple byte numbers.

### **IMPORTANT**

Since the last release of this document in January 2002 the MAOSCO Consortium has decided that future MULTOS specifications and products will continue to enhance MULTOS 4 in a fully backwards compatible fashion. Therefore features that were previously introduced with MULTOS 5 such as support for default applications and contactless interfaces have been retrofitted to MULTOS 4. This specification is MULTOS 4-2. At this time it is not envisaged that MULTOS 5 products will be released.

M-SPI 4-2 introduces subtle amendments to the M-SPI 4-0 file formats in order to support MULTOS 4-2 products. These file formats are described later in this document.

All references to M-SPI 5-0 and its associated file formats that support MULTOS 5 products have, for the sake of clarity, been removed from this release of this document.

## 2 File Types

The following file type codes are used to define the different types of data file. This list may grow with time.

In the following table :-

I = Issuer M-SPI      B = Bureau M-SPI      M = MULTOS KMA

File Type Code	File Contents	File Name Extension	From	To
MCDL	MSM Controls request sent to KMA	.mcd	I,B	M
ALCQ	ALC request sent to KMA	.alq	I	M
ADCQ	ADC request sent to KMA	.adq	I	M
APLR	Application Registration request sent to KMA	.apr	I	M
ISKA	Issuer Secret Key Addition request sent to KMA	.ika	I	M
ISKD	Issuer Secret Key Deletion request sent to KMA	.ikd	I	M
BARF	Bureau Association request sent to KMA	.bar	I	M
CUBQ	Card Block / Unblock request sent to KMA	.cbq	I	M
BSKA	Bureau Secret Key Addition request sent to KMA	.bka	B	M
BSKD	Bureau Secret Key Deletion request sent to KMA	.bkd	B	M
MSML	MSM Controls response sent by KMA	.msm	M	I,B
ALCR	ALC response sent by KMA	.alr	M	I,B
ADCR	ADC response sent by KMA	.adr	M	I,B
CUBR	Card Block / Unblock response sent by KMA	.cbr	M	I
ALUQ	Application Load Unit request file	.auq	-	-
ALUR	Application Load Unit Data	.aur	-	-
TKCK	MULTOS Transport Key Certification Key	.key	-	-
HASH	MULTOS Hash Modulus	.mhm	-	-

# Proprietary

---

## Notes

Each file carries at the start a 4 byte File Type Code followed by two one byte fields. The first of these is the File Protection method ID - this defines the way the file is protected. The second byte is the File Structure Method ID. This defines the way the data in that type of file is assembled. The meanings of these control bytes are context dependent on the file type.

Files will be named <Consignment\_File\_ID>.<filename extension>, where <Consignment\_File\_ID> is a filename that will allow the Issuer / Bureau to track the file uniquely. It is otherwise free format. The Consignment\_File\_ID field within each file's header may differ from the filename, but it would make handling simpler if they were the same. Filenames follow a standard DOS "8.3" format. Note that the file extension is important and must be preserved as per this specification since it is used as an indication of file content.

Application Load Unit related files are not sent to or received from a MULTOS KMA. They are however included here for completeness. All participants in MULTOS are encouraged to use them for consistency.

All file data is stored as binary data with no record delimiters. All multi-byte binary values are stored in "Big-Endian" or Motorola format.

Please note that details of how to create digital signatures are given in section 7.

In the following sections, data items in *italics* will be found in the data dictionary. Items in **bold** are the subject of tables in their own right. Items in parentheses thus {} are repeated. Items in normal font are simple length or integer values that do not need further explanation.

All requests are sent to a KMA either by E-mail ( Binary format, MIME attachment ) or on a DOS format 1.44 Mbyte 3.5" diskette. Responses may be returned from a KMA via E-mail, DOS format diskette or CD-ROM.

## **3 Files used to communicate from Issuer / Bureau M-SPI 4-0 to a KMA outer-office system (COO)**

This section of the document defines the file formats to be used when communicating between a KMA outer-office system (COO – Combined Outer-Office system) and Issuers and Bureaux, when M-SPI 4-0 is being used. At the time of this document being released all external parties have migrated from earlier versions to M-SPI 4-0, and so this is the baseline state.

Note that a key hierarchy exists. Issuer / Bureau key version numbers in the range 0 to 0x3F FF FF FF are generated by a KMA and distributed on M-SPI cards. The Issuer / Bureau M-SPI will not allow these key versions (known as Distribution keys) to be used for signing production requests. The only use that can be made of these keys is to sign Key Addition requests. Similarly, when the Issuer / Bureau wishes to generate a new key (known as an Operational key), they will not be permitted to generate a key with a version number in that range, all the keys they produce themselves will have version numbers in the range 0x80 00 00 00 upwards. (Note that on M-SPI the Operational keys will appear to be in the numerical range 0 to 1 073 741 823 inclusive as it is felt that a low decimal number is easier to comprehend for the operator than a very large hexadecimal number).

## 3.1 MSM Request File (version 04)

### 3.1.1 MSM Request File

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "MCDL"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x04
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Header_Record</b>	
{ <b>MCD_ID_List_Record</b> }	Occurs Number_Of_MCD_ID_List_Records times (min = 1, max. = 200,000). Entries are not sorted in any particular order
<b>Signed_Management_Information_Record</b>	

## 3.1.2 Header Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>AMD_ID</i>	ASCII, 8 characters. Set to "0000v000" to indicate that a KMA should use the appropriate default AMD for this mask
<i>x_parameter</i>	Binary, 1 byte. Set to 0x00 to indicate that a KMA should use the appropriate default value for this mask
<i>y_parameter</i>	Binary, 1 byte. Set to 0x00 to indicate that a KMA should use the appropriate default value for this mask
<i>length_ATR</i>	Binary, 1 byte. The number of relevant bytes of the primary ATR
<i>ATR</i>	Binary, 18 bytes. The primary ATR characters, padded with 0xFF as required
<i>length_ATR</i>	Binary, 1 byte. The number of relevant bytes of the secondary ATR. Only relevant for MULTOS 4 onwards
<i>ATR</i>	Binary, 18 bytes. The secondary ATR characters, padded with 0xFF as required. Only relevant for MULTOS 4 onwards
<i>MSM_MCD_Permissions_Scheme_ID</i>	Binary, 1 byte. Set to 0x01
<i>MCD_Issuer_Product_ID</i>	Binary, 1 byte
<i>Bureau_ID</i>	Binary, 4 bytes. If the request is sent by an Issuer then this field should be set to be the same as the Issuer ID
<i>Number_Of_MCD_ID_List_Records</i>	Binary, 4 bytes. This is the number of entries in the following list of MCD IDs

## 3.1.3 MCD ID List Record

Data	Definition
<i>IC_Manufacturer_ID</i>	Binary, 1 byte
<i>IC_Type</i>	Binary, 1 byte
<i>MCD_ID</i>	Binary, 6 bytes

Note that all MCD ID List records within one file must be for the same *IC\_Manufacturer\_ID* and *IC\_Type* (i.e. for the same ROM mask).

### 3.1.4 Note for users of the M-SPI software application

When using M-SPI to generate the MCD ID List File, a facility is provided to import a list of MCD IDs from file. This list is expected to be in a single MS-DOS file, consisting simply of repeated MCD ID List Records as shown above, with no record delimiters, header information or checksums of any kind. It is not expected that the MCD IDs within the file are in any order. This facility is provided to allow the easy generation of MSM requests from a list provided by the card manufacturer. Note therefore that the Issuer or Bureau generating this request will have to obtain that file from the manufacturer first (or generate the list themselves by interrogating the cards).

It is essential that every MCD in a request file is of the same ROM mask, as indicated by the IC\_Manufacturer\_ID and IC\_Type fields.

### 3.1.5 Signed Management Information Record

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the MSM header record and the MCD_ID list
<i>Padding</i>	Binary, see section 7

## 3.2 ALC Request Files (version 05)

### 3.2.1 ALC Request File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ALCQ"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x05
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_ALC_Request_Record</b>	
<b>Signed_ALC_Management_Record</b>	

### 3.2.2 Signed ALC Request Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Rom_Identifier</i>	Binary, 2 bytes
<i>History_Required</i>	Binary, 8 bytes. Set each byte to 0x00 to indicate history support not required, or any other value for history required. Only relevant for MULTOS 4 onwards
<i>MSM_App_Permissions_Scheme_ID</i>	Binary, 1 byte. Set to 0x01
<i>MSM_App_Permissions</i>	Binary, 76 bytes
<i>Application_ID_Field</i>	Binary, 17 bytes
<i>Application_Variant</i>	Binary, 2 bytes
<b>Application Provider Modulus</b>	
<b>ALC Encryption public key</b>	
<i>Padding</i>	Binary, see section 7

### 3.2.3 Application Provider Modulus

The Application Provider Modulus is used to sign applications. The public key used to do this consists of a public exponent of 0x03 and a modulus as defined by this structure.

Data	Definition
Modulus_Length	Binary, 2 bytes. The length in bytes of the Application Provider modulus
<i>Modulus</i>	Binary, Modulus_Length bytes. Modulus of key used to sign the Application Unit

### 3.2.4 Note for users of the M-SPI software application

When using M-SPI to request ALCs a facility is provided to allow the importing of the application provider public key. This must be in a binary file with the format as specified in the Application\_Provider\_Modulus record above i.e. a 2 byte length field followed by the actual application provider public key.

### 3.2.5 ALC Encryption Public Key

The ALC Encryption Public Key, if present, is used to protect the data returned from a KMA. If not used, both length fields (Modulus\_Length and Exponent\_Length) are set to 0x00s.

Data	Definition
Modulus_Length	Binary, 2 bytes. Length of following modulus field
<i>Modulus</i>	Binary, Modulus_Length bytes. Modulus to be used
Exponent_Length	Binary, 2 bytes. Length of following exponent field
<i>Exponent</i>	Binary, Exponent_Length bytes. Exponent to be used

## 3.2.6 Signed ALC Management Record

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 3.3 ADC Request files (version 03)

### 3.3.1 ADC Request File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ADCQ"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x03
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_ADC_Request_Record</b>	
<b>Signed_ADC_Management_Record</b>	

### 3.3.2 Signed ADC Request Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>MSM_App_Permissions_Scheme_ID</i>	Binary, 1 byte. Set to 0x01
<i>MSM_App_Permissions</i>	Binary, 76 bytes
<i>ALC_ID</i>	Binary, 8 bytes
<b>ADC Encryption public key</b>	
<i>Padding</i>	Binary, see section 7

### 3.3.3 ADC Encryption Public Key

The ADC Encryption Public Key, if present, is used to protect the data returned from a KMA. If not used, both length fields (Modulus\_Length and Exponent\_Length) are set to 0x00s.

Data	Definition
Modulus_Length	Binary, 2 bytes. Length of following modulus field
<i>Modulus</i>	Binary, Modulus_Length bytes. Modulus to be used
Exponent_Length	Binary, 2 bytes. Length of following exponent field
<i>Exponent</i>	Binary, Exponent_Length bytes. Exponent to be used

### 3.3.4 Signed ADC Management Record

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 3.4 Application Registration (version 04)

Note that no two applications will be accepted with the same Application ID from the same Issuer unless they have distinct variant identifiers.

### 3.4.1 Application Registration File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "APLR"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x04
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
Application_Registration_Length	Binary, 4 bytes. The length of the following field
<b>Signed_Application_Registration_Record</b>	
Extended_Registration_Length	Binary, 4 bytes. The length of the following field
<b>Signed_Extended_Registration_Record</b>	
Management_Registration_Length	Binary, 4 bytes. The length of the following field
<b>Signed_Application_Registration_Management_Record</b>	

### 3.4.2 Signed Application Registration Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_Number</i>	Binary, 4 bytes
<b>APP_DATA</b>	
<i>Padding</i>	Binary, see section 7

### 3.4.3 APP\_DATA

Data	Definition
<i>Application_ID_Field</i>	Binary, 17 bytes
<i>Application_Variant</i>	Binary, 2 bytes
<i>File_Mode_Type</i>	Binary, 1 byte
<i>Code_Size</i>	Binary, 2 bytes
<i>Data_Size</i>	Binary, 2 bytes
<i>Session_Data_Size</i>	Binary, 2 bytes
<i>DIR_File_Record_Size</i>	Binary, 2 bytes
<i>FCI_Record_Size</i>	Binary, 2 bytes
<i>App_ATR_Type</i>	Binary, 1 byte
<i>Verify_Certificate_Flag</i>	Binary, 1 byte
<i>Verify_KTU_Flag</i>	Binary, 1 byte
<b>Access_List</b>	
<i>Application_Code_Hash_Length</i>	Binary, 1 byte
<i>Application_Code_Hash</i>	Binary, Application_Code_Hash_Length bytes

### 3.4.4 Note for users of the M-SPI software application

When using M-SPI to register applications the Issuer may import many of the details from a file rather than having to type them into M-SPI. This could save time and help prevent typing mistakes. The file would be supplied by the application provider. The format of such a file is:

<application id - hex>,  
<mnemonic - text>,  
<code size - decimal>,  
<data size - decimal>,  
<session size - decimal>,  
<DIR size - decimal>,  
<FCI size - decimal>,  
<ATR setting – 'P' (for primary), 'A' (for alternate i.e. secondary), or 'N' (for no ATR effect)>,  
<shell mode – 'Y' (for shell), or 'N' (for a MULTOS app)>,  
<signed app – 'Y' (for signed) or 'N' (for non signed)>,  
<encrypted app – 'Y' (for encrypted) or 'N' (for clear)>,  
<strong crypto – 'Y' (for strong crypto required), or 'N' (for not required)>,  
<code hash - hash>

# Proprietary

---

Note that there should not be any line-breaks between parameters; they should all be in one long line. They have been shown split for clarity.

If an application provider requires assistance in creating such a file then please contact Customer Support at a MULTOS KMA for guidance. Note that the variant number is not included in this file since that is not determined by the application provider, but by the Issuer.

## 3.4.5 Access\_List

The Access List will be used to indicate the services needed by an application. One bit is currently used, the rest are yet to be defined.

Data	Definition
Bit 0 (lsb)	If set, application may call strong crypto primitives without abend. If clear, MULTOS will abend the application if it attempts to use a controlled primitive.
Bits 1 - 15	Reserved for future use.

## 3.4.6 Signed Extended Registration Record

This record contains a signed declaration from the Issuer which assists a KMA to determine whether or not export controls permissions are required for this application.

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
Export_Flags	Binary, 4 bytes. A series of bit flags indicating the answers to various questions presented to the Issuer at M-SPI
Application_Name_Length	Binary, 1 byte. The length of the next field (with a maximum value of 50)
Application_Name	ASCII, Application_Name_Length bytes. Free text description of this application
<i>Padding</i>	Binary, see section 7

# Proprietary

---

The export flags are a series of bit flags indicating the answers to various questions presented to the Issuer regarding export controls. Note that these questions may change at any time to reflect current legislative requirements. The current mappings between questions and bit flags are as follows:

- bit 0 (lsb) The Application will not be used for any purpose connected with chemical, biological or nuclear weapons or missiles capable of delivering such weapons, nor will the Application be resold if the undersigned knows or suspects that it is intended or likely to be used for such purposes.
- bit 1 The Application will not be used for any military purpose.
- bit 2 The undersigned agrees to indemnify and hold harmless Mondex International Limited, its officers, directors, agents, representatives, affiliates and employees from and against all and any losses, costs, claims, liabilities, damages, and expenses as a result of or in connection with the use or export of any Application or any breach of any of the terms or conditions set forth herein.
- bit 3 The undersigned agrees that the information provided in this Export Declaration may be disclosed by Mondex International Limited to relevant government departments where necessary in accordance with applicable law or regulation.
- bit 4 Each Application intended to be downloaded on the Card is not capable of message traffic encryption or encryption of user-supplied data or related key management functions therefore.

The remaining bits contain the declaration as to whether the application fits into any of the following predefined categories.

- bit 5 a. Access control equipment, such as automatic teller machines, self-service statement printers or point of sale terminals, which protects password or personal identification numbers (PIN) or similar data to prevent unauthorized access to facilities but does not allow encryption of files or text, except as directly related to the password or PIN protection.
- bit 6 b. Data authentication equipment which calculates a Message Authentication Code (MAC) or similar result to ensure no alternation of text has taken place, or to authenticate users, but does not allow for encryption of data, text or other media other than that needed for the authentication.
- bit 7 c. Cryptographic equipment specially designed and limited for use in machines for banking or money transactions, such as automatic teller machines, self-service statement printers or point of sale terminals.
- bit 8 d. Decryption functions specially designed to allow the execution of copy-protected "software", provided the decryption functions are not user-accessible.

# Proprietary

---

- bit 9 e. Receiving equipment for radio broadcast, pay television or similar restricted audience television of consumer type, without digital encryption and where digital decryption is limited to video, audio or management functions.
- bit 10 f. Portable or mobile radiotelephones for civil use (e.g., for use with commercial civil cellular radio communications systems) that are not capable of end-to-end encryption.
- bit 11 g. Equipment containing 'fixed' data compression or coding techniques.
- bit 12 h. Equipment using 'fixed' band scrambling not exceeding 8 bands and in which the transpositions change not more frequently than once every second.
- bit 13 i. Equipment using 'fixed' band scrambling exceeding 8 bands and in which the transpositions change not more frequently than once every 10 seconds.
- bit 14 j. Equipment using 'fixed' frequency inversion and in which the transpositions change not more frequently than once every second.
- bit 15 k. Facsimile equipment.
- bit 16 l. Restricted audience broadcast equipment.
- bit 17 m. Civil television equipment.
- bits 18-31 Currently RFU

In all cases a binary 1 indicates a 'Yes' or true answer, and a binary 0 indicates a 'No' or false answer.

## 3.4.7 Signed Application Registration Management Record

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 3.5 Issuer Key Addition (version 02)

### 3.5.1 Issuer Key Addition File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ISKA"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_Issuer_Key_Addition_Record</b>	
<b>Signed_Addition_Management_Record</b>	

### 3.5.2 Signed Issuer Key Addition Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_Number</i>	Binary, 4 bytes. The serial number for the current key
<i>Key_Serial_Number</i>	Binary, 4 bytes. The serial number for this new key
<i>Issuer_Modulus_Length</i>	Binary, 2 bytes. The length of the associated modulus
<i>Modulus</i>	Binary, Issuer_modulus_length bytes. The modulus of the new Issuer key
<i>Issuer_Public_Exponent_Length</i>	Binary, 2 bytes. The length of the associated public exponent
<i>Exponent</i>	Binary, Issuer_public_exponent_length bytes. The public exponent of the new Issuer key
<i>Padding</i>	Binary, see section 7

### 3.5.3 Signed Addition Management Record

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 3.6 Issuer Key Deletion (version 02)

### 3.6.1 Issuer Key Deletion File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ISKD"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_Issuer_Key_Deletion_Record</b>	
<b>Signed_Deletion_Management_Record</b>	

### 3.6.2 Signed Issuer Key Deletion Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_Number</i>	Binary, 4 bytes. The serial number for the current key
<i>Key_Serial_Number</i>	Binary, 4 bytes. The serial number for the key to be deleted
<i>Padding</i>	Binary, see section 7

### 3.6.3 Signed Deletion Management Record

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 3.7 Bureau Association Registration

### 3.7.1 Bureau Association Registration File

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "BARF"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_Management_Record</b>	

### 3.7.2 Signed Management Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_No</i>	Binary, 4 bytes
<i>Bureau_ID</i>	Binary, 4 bytes
<i>Associate_Disassociate</i>	Binary, 1 byte. Set to 0x5a to allow the association, or 0xa5 to remove the association
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 3.8 Card Block / Unblock Request File (version 02)

Each file will contain a request for a card block or unblock MAC, or both, for one or more cards.

### 3.8.1 Card Block / Unblock File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "CUBQ"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
Card_Block_Request_length	Binary, 4 bytes. The length of the next field
<b>Signed_Card_Block_Unblock_Request</b>	
{ <b>CBUB_Card_Record</b> }	Occurs one or more times
<b>Signed_CBUB_Management_Record</b>	

### 3.8.2 Signed Card Block / Unblock Request

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_Number</i>	Binary, 4 bytes. The serial number for the current key
Block_Unblock_Flag	Binary, 1 byte. Set to 0x5a for Block, or 0xa5 for Unblock, or 0x55 for both
Number_Of_Card_Records	Binary, 4 bytes. The number of CBUB_Card_Records, in the range 1 to 50,000
Card_Hash	Binary, 20 bytes. A SHA-1 hash of the CBUB_Card_Records
<b>CBUB_Encryption_Public_Key</b>	
<i>Padding</i>	Binary, see section 7

### 3.8.3 CBUB\_Card\_Record

Data	Definition
<i>MCD_Number</i>	Binary, 8 bytes

### 3.8.4 CBUB Encryption Public Key

The CBUB Encryption Public Key, if present, is used to protect the data returned from a KMA. If not used, both length fields are set to zero and the other fields are null (i.e. not present)

Data	Definition
<i>Modulus_Length</i>	Binary, 2 bytes. Length of following modulus field
<i>Modulus</i>	Binary, <i>Modulus_Length</i> bytes. Modulus to be used
<i>Exponent_Length</i>	Binary, 2 bytes. Length of following exponent field. Set to 0x01 for this implementation
<i>Exponent</i>	Binary, <i>Exponent_Length</i> bytes. Exponent to be used. Set to 0x03 for this implementation

### 3.8.5 Signed CBUB Management Record

Data	Definition
<i>Key_Serial_No</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 3.9 Bureau Key Addition

### 3.9.1 Bureau Key Addition File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "BSKA"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_Bureau_Key_Addition_Record</b>	
<b>Signed_Addition_Management_Record</b>	

### 3.9.2 Signed Bureau Key Addition Record

Data	Definition
<i>Bureau_ID</i>	Binary, 4 bytes
<i>Key_Serial_No</i>	Binary, 4 bytes. The serial number for the current key
<i>Key_Serial_No</i>	Binary, 4 bytes. The serial number for this new key
<i>Bureau_Modulus_Length</i>	Binary, 2 bytes. The length of the associated modulus
<i>Modulus</i>	Binary, <i>Bureau_Modulus_Length</i> bytes. The modulus of the new Bureau key
<i>Bureau_Public_Exponent_Length</i>	Binary, 2 bytes. The length of the associated public exponent
<i>Exponent</i>	Binary, <i>Bureau_Public_Exponent_Length</i> bytes. The public exponent of the new Bureau key
<i>Padding</i>	Binary, see section 7

### 3.9.3 Signed Addition Management Record

Data	Definition
<i>Key_Serial_No</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 3.10 Bureau Key Deletion

### 3.10.1 Bureau Key Deletion File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "BSKD"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_Bureau_Key_Deletion_Record</b>	
<b>Signed_Deletion_Management_Record</b>	

### 3.10.2 Signed Bureau Key Deletion Record

Data	Definition
<i>Bureau_ID</i>	Binary, 4 bytes
<i>Key_Serial_No</i>	Binary, 4 bytes. The serial number for the current key
<i>Key_Serial_No</i>	Binary, 4 bytes. The serial number for the key to be deleted
<i>Padding</i>	Binary, see section 7

### 3.10.3 Signed Deletion Management Record

Data	Definition
<i>Key_Serial_No</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 3.11 MSM Controls Data List Files (version 02)

### 3.11.1 MSM Controls Data List File

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "MSML"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>KMA_Consignment_ID</i>	ASCII, 10 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<i>Consignment_File_ID</i>	ASCII, 8 characters
<b>Header_Record</b>	
{ <b>MSM_Permissions_List_Record</b> }	Occurs Number_Of_MSM_Permissions_List_Records times. Entries are not sorted into any particular order
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the complete header record and permissions list records

### 3.11.2 Header Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>MSM_Permissions_Scheme_ID</i>	Binary, 1 byte. Set to value 0x01
<i>MCD_Issuer_Product_ID</i>	Binary, 1 byte
<i>Bureau_ID</i>	Binary, 4 bytes. Echoed from the input file
<i>MSM_Controls_Data_Record_Size</i>	Binary, 2 bytes. The size in bytes of each MSM Controls Data entry
<i>MKD_PK_C_Size</i>	Binary, 2 bytes. The size in bytes of each card public key certificate
<i>Number_Of_MSM_Permissions_List_Records</i>	Binary, 4 bytes. This is the number of entries in the subsequent list and should match that of the original request file

### 3.11.3 MSM Permissions List Record

Data	Definition
<i>MCD_ID</i>	Binary, 6 bytes
<i>MSM_Control_Data_Record</i>	Binary, <i>MSM_Controls_Data_Record_Size</i> bytes. The MSM Permissions record for the corresponding <i>MCD_ID</i>
<b>MKD_PK_C_Record</b>	

### 3.11.4 MKD PK C Record

Data	Definition
Internal data	Binary, 11 bytes
<i>MKD_Cert_Method_ID</i>	Binary, 2 bytes
<i>MKD_Hash_Method_ID</i>	Binary, 2 bytes
Internal data	Binary, 11 bytes
<i>MCD_Issuer_Product_ID</i>	Binary, 1 byte
<i>Issuer_ID</i>	Binary, 4 bytes
<i>MSM_Controls_Data_Date</i>	Binary, 1 byte
<i>MCD_Number</i>	Binary, 8 bytes
Internal data	Binary, remaining bytes

### **3.11.5 Note for users of the M-SPI software application**

When using M-SPI to receive the MSM Controls Data List File, a facility is provided to export the contents of this file into an alternative file format.

If the user chooses to export the MSM Controls then they will be given a file containing repeated lines of:

```
<MCD_Number>,<MSM_Control_Data_Record><CR><LF>
```

Or, if they choose to export the mkd\_pk\_c then they will be given a file containing repeated lines of:

```
<MCD_Number>,<MKD_PK_C_Record><CR><LF>
```

In each case the records will be output in hex-ASCII format (i.e. a text file), and <CR><LF> indicates ASCII carriage-return and line-feed.

The use of this facility within M-SPI is entirely optional, and is provided in case it is of any use when interfacing with other devices or systems.

## 3.12 ALC Response (version 03)

The ALC response may be encrypted using a key supplied by the originator of the ALC request.

Note that the ALC response files contain fields which are conditional on the type of device being targeted. These are indicated as **MULTOS 4 ONWARDS** if the field is only present for that particular target device type.

### 3.12.1 ALC Response File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ALCR"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x03
<i>KMA_Consignment_ID</i>	ASCII, 10 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<i>Consignment_File_ID</i>	ASCII, 8 characters
<b>ALC_Response_Record</b>	
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the ALC_Response_Record

## 3.12.2 ALC Response Record

The structure of this data is as shown below. Note that the ALC is passed in clear and no KTU is present unless the ALC request included a key used to protect it.

Data	Definition
ALC_Record_Length	Binary, 2 bytes. The length of the encrypted ALC record
<b>Encrypted_ALC_Record</b>	ALC data from a KMA. If the ALC was not requested to be encrypted, this is the plaintext ALC
KTU_Length	Binary, 2 bytes. The length of the KTU. If the ALC is not encrypted ( i.e. no optional key was supplied in the ALC request ) this field takes value 0x0000 and no KTU is present
<b>ALC_Protection_KTU</b>	If present, this is the KTU used to protect the ALC. It contains the session key used to encrypt the ALC and is encrypted with the public key supplied in the ALC request

## 3.12.3 Encrypted ALC Record

Data	Definition
<i>ALC_ID_Length</i>	Binary, 2 bytes. The length of the following ID
<i>ALC_ID</i>	Binary, ALC_ID_Length bytes (always 8 bytes at present)
<i>Rom_Identifier</i>	Binary, 2 bytes. Copied from the request file
<i>Pad_Length</i>	Binary, 1 byte. Length of subsequent padding
<i>Padding</i>	Binary, Pad_Length bytes
<b>ALC_DATA</b>	Binary, ALC_Length bytes

# Proprietary

## 3.12.4 ALC\_DATA

Data	Definition
ALC_Length	Binary, 2 bytes. The length of this actual ALC_DATA record, including these length bytes
Internal data	Binary, 9 bytes
<i>Cert_Method_ID</i>	Binary, 2 bytes
<i>Hash_Method_ID</i>	Binary, 2 bytes
Public_Key_Length	Binary, 2 bytes. The length of the key being certified
Certifying_Key_Length	Binary, 2 bytes. The length of the certifying key
Internal data	Binary, 7 bytes
<i>MSM_App_Permissions</i>	Binary, 76 bytes. Copied from the input request
Internal data	Binary, 18 bytes
<i>Application_ID_Field</i>	Binary, 17 bytes. Copied from the input request
<i>Random_Seed</i>	Binary, 8 bytes. <b>MULTOS 4 ONWARDS</b>
<i>File_Mode_Type</i>	Binary, 1 byte
<i>Code_Size</i>	Binary, 2 bytes
<i>Data_Size</i>	Binary, 2 bytes
<i>Session_Data_Size</i>	Binary, 2 bytes
<i>DIR_File_Record_Size</i>	Binary, 2 bytes
<i>FCI_Record_Size</i>	Binary, 2 bytes
<i>App_ATR_Type</i>	Binary, 1 byte
<i>Verify_Certificate_Flag</i>	Binary, 1 byte
<i>Verify_KTU_Flag</i>	Binary, 1 byte
<i>Access_List</i>	Binary, 2 bytes
<i>Application_Code_Hash_Length</i>	Binary, 1 byte. The length of the following hash. <b>MULTOS 4 ONWARDS</b>
<i>Application_Code_Hash</i>	Binary, <i>Application_Code_Hash_Length</i> bytes. <b>MULTOS 4 ONWARDS</b>
<i>Key_Certificate</i>	Binary, variable bytes. The certified key

## 3.12.5 ALC Protection KTU

This is the key used to decrypt the ALC. It is encrypted using the public key supplied by the ALC requester.

The ALC is protected by DES Cipher Block Chain encryption of the ALC starting with the first byte of the ALC (Triple DES is used with Encryption by the first key, then Decryption by the second followed by Encryption by the first one again). The plaintext\_KTU is shown in the following table. This is obtained by decrypting the ALC\_Protection\_KTU with the secret key corresponding to the public key sent with the ALC request.

<b>Data</b>	<b>Definition</b>
<i>Header_Byte</i>	Binary, 1 byte
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the following 3 keys
<i>ALC_Encryption_Key_1</i>	Binary, 8 bytes. The first component of the key
<i>ALC_Encryption_Key_2</i>	Binary, 8 bytes. The second component of the key
<i>ALC_Encryption_IV</i>	Binary, 8 bytes. The initial IV for ALC encryption
<i>Padding</i>	Binary, N bytes of padding of value 0x55. The padding is used to extend the length of this structure to the size of the key used to encrypt the ALC record
<i>Trailer_Byte</i>	Binary, 1 byte

## 3.13 ADC Response (version 03)

The ADC response may be encrypted using a key supplied by the originator of the ADC request.

Note that the ADC response files contain fields which are conditional on the type of device being targeted. These are indicated as **MULTOS 4 ONWARDS** if the field is only present for that particular target device type.

### 3.13.1 ADC Response File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ADCR"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x03
<i>KMA_Consignment_ID</i>	ASCII, 10 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<i>Consignment_File_ID</i>	ASCII, 8 characters
<b>ADC_Response_Record</b>	
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the ADC_Response_Record

## 3.13.2 ADC Response Record

The structure of this data is as shown below. Note that the ADC is passed in clear and no KTU is present unless the ADC request included a key used to protect it.

Data	Definition
ADC_Record_Length	Binary, 2 bytes. The length of the encrypted ADC record
<b>Encrypted_ADC_Record</b>	ADC data from a KMA. If the ADC was not requested to be encrypted, this is the plaintext ADC
KTU_Length	Binary, 2 bytes. The length of the KTU. If the ADC is not encrypted ( i.e. no optional key was supplied in the ADC request ) this field takes value 0x0000 and no KTU is present
<b>ADC_Protection_KTU</b>	If present, this is the KTU used to protect the ADC. It contains the session key used to encrypt the ADC and is encrypted with the public key supplied in the ADC request

## 3.13.3 Encrypted ADC Record

Data	Definition
<i>ADC_ID_Length</i>	Binary, 2 bytes
<i>ADC_ID</i>	Binary, ADC_ID_Length bytes
<i>ROM_Identifier</i>	Binary, 2 bytes. Copied from the request file
<i>Pad_Length</i>	Binary, 1 byte. Length of subsequent padding
<i>Padding</i>	Binary, Pad_Length bytes
<b>ADC_DATA</b>	

## 3.13.4 ADC\_DATA

Data	Definition
ADC_Length	Binary, 2 bytes. The length of this actual ADC_DATA record, including these length bytes
Internal data	Binary, 9 bytes
<i>Cert_Method_ID</i>	Binary, 2 bytes
<i>Hash_Method_ID</i>	Binary, 2 bytes
Public_Key_Length	Binary, 2 bytes. The length of the key being certified
Certifying_Key_Length	Binary, 2 bytes. The length of the certifying key
Internal data	Binary, 7 bytes
<i>MSM_App_Permissions</i>	Binary, 76 bytes
Internal data	Binary, 18 bytes
<i>Application_ID_Field</i>	Binary, 17 bytes
<i>Random_Seed</i>	Binary, 8 bytes. <b>MULTOS 4 ONWARDS</b>
Internal data	Binary, remaining bytes. The meaning of this data is irrelevant to this document (but includes the certified key)

## 3.13.5 ADC Protection KTU

This is the key used to decrypt the ADC. It is encrypted using the public key supplied by the ADC requester.

The ADC is protected by DES Cipher Block Chain encryption of the ADC starting with the first byte of the ADC (Triple DES is used with Encryption by the first key, then Decryption by the second followed by Encryption by the first one again). The plaintext\_ktu is shown in the following table. This is obtained by decrypting the ADC\_Protection\_KTU with the secret key corresponding to the public key sent with the ADC request.

<b>Data</b>	<b>Definition</b>
<i>Header_Byte</i>	Binary, 1 byte
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the following 3 keys
<i>ADC_Encryption_Key_1</i>	Binary, 8 bytes. The first component of the key
<i>ADC_Encryption_Key_2</i>	Binary, 8 bytes. The second component of the key
<i>ADC_Encryption_IV</i>	Binary, 8 bytes. The initial IV for ADC encryption
<i>Padding</i>	Binary, N bytes of padding of value 0x55. The padding is used to extend the length of this structure to the size of the key used to encrypt the ADC record
<i>Trailer_Byte</i>	Binary, 1 byte

## 3.14 Card Block / Unblock Response File (version 02)

Each file will contain one or more card block or unblock MACs, or both.

### 3.14.1 Card Block / Unblock Response File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "CUBR"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>KMA_Consignment_ID</i>	ASCII, 10 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<i>Consignment_File_ID</i>	ASCII, 8 characters
<b>CBUB_Response_Record</b>	
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the CBUB_Response_Record

### 3.14.2 CBUB Response Record

The structure of this data is as shown below. Note that the Card Block / Unblock MACs are passed in clear and no KTU is present unless the request included a key used to protect it.

Data	Definition
CBUB_Record_Length	Binary, 4 bytes. The length of the CBUB record
<b>CBUB_Record</b>	CBUB data from the CA.
KTU_Length	Binary, 2 bytes. The length of the KTU. If CBUB is not encrypted ( i.e. no optional key was supplied in the CBUB request ) this field takes value 0x0000 and no KTU is present
<b>CBUB_Protection_KTU</b>	If present, this is the KTU used to protect the CBUB. It contains the session key used to encrypt the CBUB and is encrypted with the public key supplied in the CBUB request

### 3.14.3 CBUB Record

Data	Definition
Block_Unblock_Flag	Binary, 1 byte. Set to 0x5a for Block, or 0xa5 for Unblock, or 0x55 for both
Number_Of_Card_Records	Binary, 4 bytes. The number of CBUB_Card_Records, in the range 1 to 50,000
{ Encrypted_CBUB_Data }	Occurs Number_Of_Card_Records times. If the CBUB was not requested to be encrypted, this is the plaintext CBUB

### 3.14.4 Encrypted CBUB Data

Data	Definition
<i>MCD_Number</i>	Binary, 8 bytes
MAC	Binary, 8 bytes. The Block or Unblock MAC requested
MAC	Binary, 8 bytes. If both MACs have been requested then the Block MAC is first, then the Unblock MAC. If only one MAC has been requested then this field is not present.

## 3.14.5 CBUB Protection KTU

This is the key used to decrypt the CBUB. It is encrypted using the public key supplied by the CBUB requester.

The CBUB is protected by DES Cipher Block Chain encryption of the CBUB starting with the first byte of the CBUB (Triple DES is used with Encryption by the first key, then Decryption by the second followed by Encryption by the first one again). The plaintext\_ktu is shown in the following table. This is obtained by decrypting the CBUB\_Protection\_KTU with the secret key corresponding to the public key sent with the CBUB request.

<b>Data</b>	<b>Definition</b>
<i>Header_Byte</i>	Binary, 1 byte
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the following 3 keys
<i>CBUB_Encryption_Key_1</i>	Binary, 8 bytes. The first component of the key
<i>CBUB_Encryption_Key_2</i>	Binary, 8 bytes. The second component of the key
<i>CBUB_Encryption_IV</i>	Binary, 8 bytes. The initial IV for CBUB encryption
<i>Padding_Random</i>	Binary, N bytes of random padding. The padding is used to extend the length of this structure to the size of the key used to encrypt the CBUB record
<i>Trailer_Byte</i>	Binary, 1 byte

## **4 Files used to communicate from Issuer / Bureau M-SPI 4-2 to a KMA outer-office system (COO)**

This section of the document defines the file formats to be used when communicating between a KMA outer-office system (COO) and Issuers and Bureaux, when M-SPI 4-2 is being used.

Six of the files have changed. Five of these (Issuer Key Addition, Issuer Key Deletion, Bureau Key Addition, Bureau Key Deletion, and Bureau Association) have had an additional address field added. The Global KMA is no longer the only active KMA in the world; there are now additional IKMAs (Independent KMAs). It has been recognised that different IKMAs may wish to communicate differently with their customers. As such all request files now include a Sender address, a Billing address, and a Delivery address. It is left to each KMA to determine how they wish to use those addresses (for examples a 'Delivery' address may be used to send acknowledgements for orders that otherwise require no delivery of output).

The other file that has changed is the Application Registration. This has been enhanced to allow MULTOS 4-2 features to be specified. Specifically some of the features previously proposed for MULTOS 5 have been 'retro-fitted' into the existing MULTOS 4 application registration format.

## 4.1 MSM Request File (version 04)

### 4.1.1 MSM Request File

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "MCDL"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x04
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Header_Record</b>	
{ <b>MCD_ID_List_Record</b> }	Occurs Number_Of_MCD_ID_List_Records times (min = 1, max. = 200,000). Entries are not sorted in any particular order
<b>Signed_Management_Information_Record</b>	

## 4.1.2 Header Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>AMD_ID</i>	ASCII, 8 characters. Set to "0000v000" to indicate that a KMA should use the appropriate default AMD for this mask
<i>x_parameter</i>	Binary, 1 byte. Set to 0x00 to indicate that a KMA should use the appropriate default value for this mask
<i>y_parameter</i>	Binary, 1 byte. Set to 0x00 to indicate that a KMA should use the appropriate default value for this mask
<i>length_ATR</i>	Binary, 1 byte. The number of relevant bytes of the primary ATR
<i>ATR</i>	Binary, 18 bytes. The primary ATR characters, padded with 0xFF as required
<i>length_ATR</i>	Binary, 1 byte. The number of relevant bytes of the secondary ATR. Only relevant for MULTOS 4 onwards
<i>ATR</i>	Binary, 18 bytes. The secondary ATR characters, padded with 0xFF as required. Only relevant for MULTOS 4 onwards
<i>MSM_MCD_Permissions_Scheme_ID</i>	Binary, 1 byte. Set to 0x01
<i>MCD_Issuer_Product_ID</i>	Binary, 1 byte
<i>Bureau_ID</i>	Binary, 4 bytes. If the request is sent by an Issuer then this field should be set to be the same as the Issuer ID
<i>Number_Of_MCD_ID_List_Records</i>	Binary, 4 bytes. This is the number of entries in the following list of MCD IDs

## 4.1.3 MCD ID List Record

Data	Definition
<i>IC_Manufacturer_ID</i>	Binary, 1 byte
<i>IC_Type</i>	Binary, 1 byte
<i>MCD_ID</i>	Binary, 6 bytes

Note that all MCD ID List records within one file must be for the same *IC\_Manufacturer\_ID* and *IC\_Type* (i.e. for the same ROM mask).

## 4.1.4 Note for users of the M-SPI software application

When using M-SPI to generate the MCD ID List File, a facility is provided to import a list of MCD IDs from file. This list is expected to be in a single MS-DOS file, consisting simply of repeated MCD ID List Records as shown above, with no record delimiters, header information or checksums of any kind. It is not expected that the MCD IDs within the file are in any order. This facility is provided to allow the easy generation of MSM requests from a list provided by the card manufacturer. Note therefore that the Issuer or Bureau generating this request will have to obtain that file from the manufacturer first (or generate the list themselves by interrogating the cards).

It is essential that every MCD in a request file is of the same ROM mask, as indicated by the IC\_Manufacturer\_ID and IC\_Type fields.

## 4.1.5 Signed Management Information Record

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the MSM header record and the MCD_ID list
<i>Padding</i>	Binary, see section 7

## 4.2 ALC Request Files (version 05)

### 4.2.1 ALC Request File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ALCQ"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x05
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_ALC_Request_Record</b>	
<b>Signed_ALC_Management_Record</b>	

### 4.2.2 Signed ALC Request Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Rom_Identifier</i>	Binary, 2 bytes
<i>History_Required</i>	Binary, 8 bytes. Set each byte to 0x00 to indicate history support not required, or any other value for history required. Only relevant for MULTOS 4 onwards
<i>MSM_App_Permissions_Scheme_ID</i>	Binary, 1 byte. Set to 0x01
<i>MSM_App_Permissions</i>	Binary, 76 bytes
<i>Application_ID_Field</i>	Binary, 17 bytes
<i>Application_Variant</i>	Binary, 2 bytes
<b>Application Provider Modulus</b>	
<b>ALC Encryption public key</b>	
<i>Padding</i>	Binary, see section 7

## 4.2.3 Application Provider Modulus

The Application Provider Modulus is used to sign applications. The public key used to do this consists of a public exponent of 0x03 and a modulus as defined by this structure.

Data	Definition
Modulus_Length	Binary, 2 bytes. The length in bytes of the Application Provider modulus
<i>Modulus</i>	Binary, Modulus_Length bytes. Modulus of key used to sign the Application Unit

## 4.2.4 Note for users of the M-SPI software application

When using M-SPI to request ALCs a facility is provided to allow the importing of the application provider public key. This must be in a binary file with the format as specified in the Application\_Provider\_Modulus record above i.e. a 2 byte length field followed by the actual application provider public key.

## 4.2.5 ALC Encryption Public Key

The ALC Encryption Public Key, if present, is used to protect the data returned from a KMA. If not used, both length fields (Modulus\_Length and Exponent\_Length) are set to 0x00s.

Data	Definition
Modulus_Length	Binary, 2 bytes. Length of following modulus field
<i>Modulus</i>	Binary, Modulus_Length bytes. Modulus to be used
Exponent_Length	Binary, 2 bytes. Length of following exponent field
<i>Exponent</i>	Binary, Exponent_Length bytes. Exponent to be used

## 4.2.6 Signed ALC Management Record

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 4.3 ADC Request files (version 03)

### 4.3.1 ADC Request File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ADCQ"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x03
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_ADC_Request_Record</b>	
<b>Signed_ADC_Management_Record</b>	

### 4.3.2 Signed ADC Request Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>MSM_App_Permissions_Scheme_ID</i>	Binary, 1 byte. Set to 0x01
<i>MSM_App_Permissions</i>	Binary, 76 bytes
<i>ALC_ID</i>	Binary, 8 bytes
<b>ADC Encryption public key</b>	
<i>Padding</i>	Binary, see section 7

### 4.3.3 ADC Encryption Public Key

The ADC Encryption Public Key, if present, is used to protect the data returned from a KMA. If not used, both length fields (Modulus\_Length and Exponent\_Length) are set to 0x00s.

Data	Definition
Modulus_Length	Binary, 2 bytes. Length of following modulus field
<i>Modulus</i>	Binary, Modulus_Length bytes. Modulus to be used
Exponent_Length	Binary, 2 bytes. Length of following exponent field
<i>Exponent</i>	Binary, Exponent_Length bytes. Exponent to be used

### 4.3.4 Signed ADC Management Record

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 4.4 Application Registration (version 07)

Note that no two applications will be accepted with the same Application ID from the same Issuer unless they have distinct variant identifiers.

### 4.4.1 Application Registration File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "APLR"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x07
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
Application_Registration_Length	Binary, 4 bytes. The length of the following field
<b>Signed_Application_Registration_Record</b>	
Extended_Registration_Length	Binary, 4 bytes. The length of the following field
<b>Signed_Extended_Registration_Record</b>	
Management_Registration_Length	Binary, 4 bytes. The length of the following field
<b>Signed_Application_Registration_Management_Record</b>	

### 4.4.2 Signed Application Registration Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_Number</i>	Binary, 4 bytes
<b>APP_DATA</b>	
<i>Padding</i>	Binary, see section 7

# Proprietary

---

## 4.4.3 APP\_DATA

<b>Data</b>	<b>Definition</b>
<i>Application_ID_Field</i>	Binary, 17 bytes
<i>Application_Variant</i>	Binary, 2 bytes
<i>File_Mode_Type</i>	Binary, 1 byte
<i>Code_Size</i>	Binary, 2 bytes
<i>Data_Size</i>	Binary, 2 bytes
<i>Session_Data_Size</i>	Binary, 2 bytes
<i>DIR_File_Record_Size</i>	Binary, 2 bytes
<i>FCL_Record_Size</i>	Binary, 2 bytes
<i>App_ATR_Type</i>	Binary, 1 byte
<i>Verify_Certificate_Flag</i>	Binary, 1 byte
<i>Verify_KTU_Flag</i>	Binary, 1 byte
<b>Access_List</b>	
<i>Application_Code_Hash_Length</i>	Binary, 1 byte
<i>Application_Code_Hash</i>	Binary, Application_Code_Hash_Length bytes

## 4.4.4 Note for users of the M-SPI software application

When using M-SPI to register applications the Issuer may import many of the details from a file rather than having to type them into M-SPI. This could save time and help prevent typing mistakes. The file would be supplied by the application provider. The format of such a file is as follows. Note that this file format is a slightly extended version of that for M-SPI 4-0. If an existing file from M-SPI 4-0 is imported into M-SPI 4-2 then all fields will be completed on screen correctly apart from the selection of which communications interface is required. Conversely, if an M-SPI 4-2 file is imported into M-SPI 4-0 then the extra field will simply be ignored by M-SPI 4-0.

```
<application id - hex>,  
<mnemonic - text>,  
<code size - decimal>,  
<data size - decimal>,  
<session size - decimal>,  
<DIR size - decimal>,  
<FCI size - decimal>,  
<ATR setting – 'P' (for primary), 'A' (for alternate i.e. secondary), or 'N' (for no ATR effect)>,  
<shell mode – 'Y' (for shell) , 'N' (for a MULTOS app)>, or 'D' (for default)  
<signed app – 'Y' (for signed) or 'N' (for non signed)>,  
<encrypted app – 'Y' (for encrypted) or 'N' (for clear)>,  
<strong crypto – 'Y' (for strong crypto required), or 'N' (for not required)>,  
<code hash - hex>,  
<communications interface required - 'C' (for contact only), 'L' (for contactless only), or 'B' (for both)>
```

Note that there should not be any line-breaks between parameters; they should all be in one long line. They have been shown split for clarity.

If an application provider requires assistance in creating such a file then please contact Customer Support at a MULTOS KMA for guidance.

## 4.4.5 Access\_List

The Access List will be used to indicate the services needed by an application. Three bits are currently used, the rest are yet to be defined.

Data	Definition
Bit 0 (lsb)	If set, application may call strong crypto primitives without abend. If clear, MULTOS will abend the application if it attempts to use a controlled primitive.
Bit 1	If set, the application may use the contact communications interface. Only relevant from MULTOS 4-2 onwards
Bit 2	If set, the application may use the contactless communications interface. Only relevant from MULTOS 4-2 onwards
Bits 3 - 15	Reserved for future use.

## 4.4.6 Signed Extended Registration Record

This record contains a signed declaration from the Issuer which assists a KMA to determine whether or not export controls permissions are required for this application.

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
Export_Flags	Binary, 4 bytes. A series of bit flags indicating the answers to various questions presented to the Issuer at M-SPI
Application_Name_Length	Binary, 1 byte. The length of the next field (with a maximum value of 50)
Application_Name	ASCII, Application_Name_Length bytes. Free text description of this application
<i>Padding</i>	Binary, see section 7

# Proprietary

---

The export flags are a series of bit flags indicating the answers to various questions presented to the Issuer regarding export controls. Note that these questions may change at any time to reflect current legislative requirements. The current mappings between questions and bit flags are as follows:

- bit 0 (lsb) The Application will not be used for any purpose connected with chemical, biological or nuclear weapons or missiles capable of delivering such weapons, nor will the Application be resold if the undersigned knows or suspects that it is intended or likely to be used for such purposes.
- bit 1 The Application will not be used for any military purpose.
- bit 2 The undersigned agrees to indemnify and hold harmless Mondex International Limited, its officers, directors, agents, representatives, affiliates and employees from and against all and any losses, costs, claims, liabilities, damages, and expenses as a result of or in connection with the use or export of any Application or any breach of any of the terms or conditions set forth herein.
- bit 3 The undersigned agrees that the information provided in this Export Declaration may be disclosed by Mondex International Limited to relevant government departments where necessary in accordance with applicable law or regulation.
- bit 4 Each Application intended to be downloaded on the Card is not capable of message traffic encryption or encryption of user-supplied data or related key management functions therefore.

The remaining bits contain the declaration as to whether the application fits into any of the following predefined categories.

- bit 5 a. Access control equipment, such as automatic teller machines, self-service statement printers or point of sale terminals, which protects password or personal identification numbers (PIN) or similar data to prevent unauthorized access to facilities but does not allow encryption of files or text, except as directly related to the password or PIN protection.
- bit 6 b. Data authentication equipment which calculates a Message Authentication Code (MAC) or similar result to ensure no alternation of text has taken place, or to authenticate users, but does not allow for encryption of data, text or other media other than that needed for the authentication.
- bit 7 c. Cryptographic equipment specially designed and limited for use in machines for banking or money transactions, such as automatic teller machines, self-service statement printers or point of sale terminals.
- bit 8 d. Decryption functions specially designed to allow the execution of copy-protected "software", provided the decryption functions are not user-accessible.

# Proprietary

---

- bit 9 e. Receiving equipment for radio broadcast, pay television or similar restricted audience television of consumer type, without digital encryption and where digital decryption is limited to video, audio or management functions.
- bit 10 f. Portable or mobile radiotelephones for civil use (e.g., for use with commercial civil cellular radio communications systems) that are not capable of end-to-end encryption.
- bit 11 g. Equipment containing 'fixed' data compression or coding techniques.
- bit 12 h. Equipment using 'fixed' band scrambling not exceeding 8 bands and in which the transpositions change not more frequently than once every second.
- bit 13 i. Equipment using 'fixed' band scrambling exceeding 8 bands and in which the transpositions change not more frequently than once every 10 seconds.
- bit 14 j. Equipment using 'fixed' frequency inversion and in which the transpositions change not more frequently than once every second.
- bit 15 k. Facsimile equipment.
- bit 16 l. Restricted audience broadcast equipment.
- bit 17 m. Civil television equipment.
- bits 18-31 Currently RFU

In all cases a binary 1 indicates a 'Yes' or true answer, and a binary 0 indicates a 'No' or false answer.

## 4.4.7 Signed Application Registration Management Record

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 4.5 Issuer Key Addition (version 03)

### 4.5.1 Issuer Key Addition File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ISKA"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x03
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_Issuer_Key_Addition_Record</b>	
<b>Signed_Addition_Management_Record</b>	

### 4.5.2 Signed Issuer Key Addition Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_Number</i>	Binary, 4 bytes. The serial number for the current key
<i>Key_Serial_Number</i>	Binary, 4 bytes. The serial number for this new key
<i>Issuer_Modulus_Length</i>	Binary, 2 bytes. The length of the associated modulus
<i>Modulus</i>	Binary, Issuer_modulus_length bytes. The modulus of the new Issuer key
<i>Issuer_Public_Exponent_Length</i>	Binary, 2 bytes. The length of the associated public exponent
<i>Exponent</i>	Binary, Issuer_public_exponent_length bytes. The public exponent of the new Issuer key
<i>Padding</i>	Binary, see section 7

## 4.5.3 Signed Addition Management Record

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 4.6 Issuer Key Deletion (version 03)

### 4.6.1 Issuer Key Deletion File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ISKD"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x03
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_Issuer_Key_Deletion_Record</b>	
<b>Signed_Deletion_Management_Record</b>	

### 4.6.2 Signed Issuer Key Deletion Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_Number</i>	Binary, 4 bytes. The serial number for the current key
<i>Key_Serial_Number</i>	Binary, 4 bytes. The serial number for the key to be deleted
<i>Padding</i>	Binary, see section 7

## 4.6.3 Signed Deletion Management Record

Data	Definition
<i>Key_Serial_Number</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 4.7 Bureau Association Registration (version 02)

### 4.7.1 Bureau Association Registration File

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "BARF"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_Management_Record</b>	

### 4.7.2 Signed Management Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_No</i>	Binary, 4 bytes
<i>Bureau_ID</i>	Binary, 4 bytes
<i>Associate_Disassociate</i>	Binary, 1 byte. Set to 0x5a to allow the association, or 0xa5 to remove the association
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 4.8 Card Block / Unblock Request File (version 02)

Each file will contain a request for a card block or unblock MAC, or both, for one or more cards.

### 4.8.1 Card Block / Unblock File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "CUBQ"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
Card_Block_Request_length	Binary, 4 bytes. The length of the next field
<b>Signed_Card_Block_Unblock_Request</b>	
{ <b>CBUB_Card_Record</b> }	Occurs one or more times
<b>Signed_CBUB_Management_Record</b>	

### 4.8.2 Signed Card Block / Unblock Request

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>Key_Serial_Number</i>	Binary, 4 bytes. The serial number for the current key
Block_Unblock_Flag	Binary, 1 byte. Set to 0x5a for Block, or 0xa5 for Unblock, or 0x55 for both
Number_Of_Card_Records	Binary, 4 bytes. The number of CBUB_Card_Records, in the range 1 to 50,000
Card_Hash	Binary, 20 bytes. A SHA-1 hash of the CBUB_Card_Records
<b>CBUB_Encryption_Public_Key</b>	
<i>Padding</i>	Binary, see section 7

## 4.8.3 CBUB\_Card\_Record

Data	Definition
<i>MCD_Number</i>	Binary, 8 bytes

## 4.8.4 CBUB Encryption Public Key

The CBUB Encryption Public Key, if present, is used to protect the data returned from a KMA. If not used, both length fields are set to zero and the other fields are null (i.e. not present)

Data	Definition
<i>Modulus_Length</i>	Binary, 2 bytes. Length of following modulus field
<i>Modulus</i>	Binary, <i>Modulus_Length</i> bytes. Modulus to be used
<i>Exponent_Length</i>	Binary, 2 bytes. Length of following exponent field. Set to 0x01 for this implementation
<i>Exponent</i>	Binary, <i>Exponent_Length</i> bytes. Exponent to be used. Set to 0x03 for this implementation

## 4.8.5 Signed CBUB Management Record

Data	Definition
<i>Key_Serial_No</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 4.9 Bureau Key Addition (version 02)

### 4.9.1 Bureau Key Addition File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "BSKA"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_Bureau_Key_Addition_Record</b>	
<b>Signed_Addition_Management_Record</b>	

### 4.9.2 Signed Bureau Key Addition Record

Data	Definition
<i>Bureau_ID</i>	Binary, 4 bytes
<i>Key_Serial_No</i>	Binary, 4 bytes. The serial number for the current key
<i>Key_Serial_No</i>	Binary, 4 bytes. The serial number for this new key
<i>Bureau_Modulus_Length</i>	Binary, 2 bytes. The length of the associated modulus
<i>Modulus</i>	Binary, <i>Bureau_Modulus_Length</i> bytes. The modulus of the new Bureau key
<i>Bureau_Public_Exponent_Length</i>	Binary, 2 bytes. The length of the associated public exponent
<i>Exponent</i>	Binary, <i>Bureau_Public_Exponent_Length</i> bytes. The public exponent of the new Bureau key
<i>Padding</i>	Binary, see section 7

## 4.9.3 Signed Addition Management Record

Data	Definition
<i>Key_Serial_No</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 4.10 Bureau Key Deletion (version 02)

### 4.10.1 Bureau Key Deletion File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "BSKD"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Signed_Bureau_Key_Deletion_Record</b>	
<b>Signed_Deletion_Management_Record</b>	

### 4.10.2 Signed Bureau Key Deletion Record

Data	Definition
<i>Bureau_ID</i>	Binary, 4 bytes
<i>Key_Serial_No</i>	Binary, 4 bytes. The serial number for the current key
<i>Key_Serial_No</i>	Binary, 4 bytes. The serial number for the key to be deleted
<i>Padding</i>	Binary, see section 7

## 4.10.3 Signed Deletion Management Record

Data	Definition
<i>Key_Serial_No</i>	Binary, 4 bytes
<i>Sender_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Delivery_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Billing_Details</i>	An instance of a <i>Name_Address_Record</i>
<i>Padding</i>	Binary, see section 7

## 4.11 MSM Controls Data List Files (version 02)

### 4.11.1 MSM Controls Data List File

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "MSML"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>KMA_Consignment_ID</i>	ASCII, 10 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<i>Consignment_File_ID</i>	ASCII, 8 characters
<b>Header_Record</b>	
{ <b>MSM_Permissions_List_Record</b> }	Occurs Number_Of_MSM_Permissions_List_Records times. Entries are not sorted into any particular order
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the complete header record and permissions list records

### 4.11.2 Header Record

Data	Definition
<i>Issuer_ID</i>	Binary, 4 bytes
<i>MSM_Permissions_Scheme_ID</i>	Binary, 1 byte. Set to value 0x01
<i>MCD_Issuer_Product_ID</i>	Binary, 1 byte
<i>Bureau_ID</i>	Binary, 4 bytes. Echoed from the input file
<i>MSM_Controls_Data_Record_Size</i>	Binary, 2 bytes. The size in bytes of each MSM Controls Data entry
<i>MKD_PK_C_Size</i>	Binary, 2 bytes. The size in bytes of each card public key certificate
<i>Number_Of_MSM_Permissions_List_Records</i>	Binary, 4 bytes. This is the number of entries in the subsequent list and should match that of the original request file

## 4.11.3 MSM Permissions List Record

Data	Definition
<i>MCD_ID</i>	Binary, 6 bytes
<i>MSM_Control_Data_Record</i>	Binary, <i>MSM_Controls_Data_Record_Size</i> bytes. The MSM Permissions record for the corresponding <i>MCD_ID</i>
<b>MKD_PK_C_Record</b>	

## 4.11.4 MKD PK C Record

Data	Definition
Internal data	Binary, 11 bytes
<i>MKD_Cert_Method_ID</i>	Binary, 2 bytes
<i>MKD_Hash_Method_ID</i>	Binary, 2 bytes
Internal data	Binary, 11 bytes
<i>MCD_Issuer_Product_ID</i>	Binary, 1 byte
<i>Issuer_ID</i>	Binary, 4 bytes
<i>MSM_Controls_Data_Date</i>	Binary, 1 byte
<i>MCD_Number</i>	Binary, 8 bytes
Internal data	Binary, remaining bytes

## 4.11.5 Note for users of the M-SPI software application

When using M-SPI to receive the MSM Controls Data List File, a facility is provided to export the contents of this file into an alternative file format.

If the user chooses to export the MSM Controls then they will be given a file containing repeated lines of:

```
<MCD_Number>,<MSM_Control_Data_Record><CR><LF>
```

Or, if they choose to export the mkd\_pk\_c then they will be given a file containing repeated lines of :

```
<MCD_Number>,<MKD_PK_C_Record><CR><LF>
```

In each case the records will be output in hex-ASCII format (i.e. a text file), and <CR><LF> indicates ASCII carriage-return and line-feed.

The use of this facility within M-SPI is entirely optional, and is provided in case it is of any use when interfacing with other devices or systems.

## 4.12 ALC Response (version 03)

The ALC response may be encrypted using a key supplied by the originator of the ALC request.

Note that the ALC response files contain fields which are conditional on the type of device being targeted. These are indicated as **MULTOS 4 ONWARDS** if the field is only present for that particular target device type.

### 4.12.1 ALC Response File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ALCR"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x03
<i>KMA_Consignment_ID</i>	ASCII, 10 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<i>Consignment_File_ID</i>	ASCII, 8 characters
<b>ALC_Response_Record</b>	
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the ALC_Response_Record

## 4.12.2 ALC Response Record

The structure of this data is as shown below. Note that the ALC is passed in clear and no KTU is present unless the ALC request included a key used to protect it.

Data	Definition
ALC_Record_Length	Binary, 2 bytes. The length of the encrypted ALC record
<b>Encrypted_ALC_Record</b>	ALC data from a KMA. If the ALC was not requested to be encrypted, this is the plaintext ALC
KTU_Length	Binary, 2 bytes. The length of the KTU. If the ALC is not encrypted ( i.e. no optional key was supplied in the ALC request ) this field takes value 0x0000 and no KTU is present
<b>ALC_Protection_KTU</b>	If present, this is the KTU used to protect the ALC. It contains the session key used to encrypt the ALC and is encrypted with the public key supplied in the ALC request

## 4.12.3 Encrypted ALC Record

Data	Definition
<i>ALC_ID_Length</i>	Binary, 2 bytes. The length of the following ID
<i>ALC_ID</i>	Binary, ALC_ID_Length bytes (always 8 bytes at present)
<i>Rom_Identifier</i>	Binary, 2 bytes. Copied from the request file
<i>Pad_Length</i>	Binary, 1 byte. Length of subsequent padding
<i>Padding</i>	Binary, Pad_Length bytes
<b>ALC_DATA</b>	Binary, ALC_Length bytes

# Proprietary

## 4.12.4 ALC\_DATA

Data	Definition
ALC_Length	Binary, 2 bytes. The length of this actual ALC_DATA record, including these length bytes
Internal data	Binary, 9 bytes
<i>Cert_Method_ID</i>	Binary, 2 bytes
<i>Hash_Method_ID</i>	Binary, 2 bytes
Public_Key_Length	Binary, 2 bytes. The length of the key being certified
Certifying_Key_Length	Binary, 2 bytes. The length of the certifying key
Internal data	Binary, 7 bytes
<i>MSM_App_Permissions</i>	Binary, 76 bytes. Copied from the input request
Internal data	Binary, 18 bytes
<i>Application_ID_Field</i>	Binary, 17 bytes. Copied from the input request
<i>Random_Seed</i>	Binary, 8 bytes. <b>MULTOS 4 ONWARDS</b>
<i>File_Mode_Type</i>	Binary, 1 byte
<i>Code_Size</i>	Binary, 2 bytes
<i>Data_Size</i>	Binary, 2 bytes
<i>Session_Data_Size</i>	Binary, 2 bytes
<i>DIR_File_Record_Size</i>	Binary, 2 bytes
<i>FCI_Record_Size</i>	Binary, 2 bytes
<i>App_ATR_Type</i>	Binary, 1 byte
<i>Verify_Certificate_Flag</i>	Binary, 1 byte
<i>Verify_KTU_Flag</i>	Binary, 1 byte
<i>Access_List</i>	Binary, 2 bytes
<i>Application_Code_Hash_Length</i>	Binary, 1 byte. The length of the following hash. <b>MULTOS 4 ONWARDS</b>
<i>Application_Code_Hash</i>	Binary, <i>Application_Code_Hash_Length</i> bytes. <b>MULTOS 4 ONWARDS</b>
<i>Key_Certificate</i>	Binary, variable bytes. The certified key

## 4.12.5 ALC Protection KTU

This is the key used to decrypt the ALC. It is encrypted using the public key supplied by the ALC requester.

The ALC is protected by DES Cipher Block Chain encryption of the ALC starting with the first byte of the ALC (Triple DES is used with Encryption by the first key, then Decryption by the second followed by Encryption by the first one again). The plaintext\_KTU is shown in the following table. This is obtained by decrypting the ALC\_Protection\_KTU with the secret key corresponding to the public key sent with the ALC request.

<b>Data</b>	<b>Definition</b>
<i>Header_Byte</i>	Binary, 1 byte
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the following 3 keys
<i>ALC_Encryption_Key_1</i>	Binary, 8 bytes. The first component of the key
<i>ALC_Encryption_Key_2</i>	Binary, 8 bytes. The second component of the key
<i>ALC_Encryption_IV</i>	Binary, 8 bytes. The initial IV for ALC encryption
<i>Padding</i>	Binary, N bytes of padding of value 0x55. The padding is used to extend the length of this structure to the size of the key used to encrypt the ALC record
<i>Trailer_Byte</i>	Binary, 1 byte

## 4.13 ADC Response (version 03)

The ADC response may be encrypted using a key supplied by the originator of the ADC request.

Note that the ADC response files contain fields which are conditional on the type of device being targeted. These are indicated as **MULTOS 4 ONWARDS** if the field is only present for that particular target device type.

### 4.13.1 ADC Response File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ADCR"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x03
<i>KMA_Consignment_ID</i>	ASCII, 10 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<i>Consignment_File_ID</i>	ASCII, 8 characters
<b>ADC_Response_Record</b>	
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the ADC_Response_Record

## 4.13.2 ADC Response Record

The structure of this data is as shown below. Note that the ADC is passed in clear and no KTU is present unless the ADC request included a key used to protect it.

Data	Definition
ADC_Record_Length	Binary, 2 bytes. The length of the encrypted ADC record
<b>Encrypted_ADC_Record</b>	ADC data from a KMA. If the ADC was not requested to be encrypted, this is the plaintext ADC
KTU_Length	Binary, 2 bytes. The length of the KTU. If the ADC is not encrypted ( i.e. no optional key was supplied in the ADC request ) this field takes value 0x0000 and no KTU is present
<b>ADC_Protection_KTU</b>	If present, this is the KTU used to protect the ADC. It contains the session key used to encrypt the ADC and is encrypted with the public key supplied in the ADC request

## 4.13.3 Encrypted ADC Record

Data	Definition
<i>ADC_ID_Length</i>	Binary, 2 bytes
<i>ADC_ID</i>	Binary, ADC_ID_Length bytes
<i>ROM_Identifier</i>	Binary, 2 bytes. Copied from the request file
<i>Pad_Length</i>	Binary, 1 byte. Length of subsequent padding
<i>Padding</i>	Binary, Pad_Length bytes
<b>ADC_DATA</b>	

## 4.13.4 ADC\_DATA

Data	Definition
ADC_Length	Binary, 2 bytes. The length of this actual ADC_DATA record, including these length bytes
Internal data	Binary, 9 bytes
<i>Cert_Method_ID</i>	Binary, 2 bytes
<i>Hash_Method_ID</i>	Binary, 2 bytes
Public_Key_Length	Binary, 2 bytes. The length of the key being certified
Certifying_Key_Length	Binary, 2 bytes. The length of the certifying key
Internal data	Binary, 7 bytes
<i>MSM_App_Permissions</i>	Binary, 76 bytes
Internal data	Binary, 18 bytes
<i>Application_ID_Field</i>	Binary, 17 bytes
<i>Random_Seed</i>	Binary, 8 bytes. <b>MULTOS 4 ONWARDS</b>
Internal data	Binary, remaining bytes. The meaning of this data is irrelevant to this document (but includes the certified key)

## 4.13.5 ADC Protection KTU

This is the key used to decrypt the ADC. It is encrypted using the public key supplied by the ADC requester.

The ADC is protected by DES Cipher Block Chain encryption of the ADC starting with the first byte of the ADC (Triple DES is used with Encryption by the first key, then Decryption by the second followed by Encryption by the first one again). The plaintext\_ktu is shown in the following table. This is obtained by decrypting the ADC\_Protection\_KTU with the secret key corresponding to the public key sent with the ADC request.

<b>Data</b>	<b>Definition</b>
<i>Header_Byte</i>	Binary, 1 byte
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the following 3 keys
<i>ADC_Encryption_Key_1</i>	Binary, 8 bytes. The first component of the key
<i>ADC_Encryption_Key_2</i>	Binary, 8 bytes. The second component of the key
<i>ADC_Encryption_IV</i>	Binary, 8 bytes. The initial IV for ADC encryption
<i>Padding</i>	Binary, N bytes of padding of value 0x55. The padding is used to extend the length of this structure to the size of the key used to encrypt the ADC record
<i>Trailer_Byte</i>	Binary, 1 byte

## 4.14 Card Block / Unblock Response File (version 02)

Each file will contain one or more card block or unblock MACs, or both.

### 4.14.1 Card Block / Unblock Response File format

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "CUBR"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>KMA_Consignment_ID</i>	ASCII, 10 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<i>Consignment_File_ID</i>	ASCII, 8 characters
<b>CBUB_Response_Record</b>	
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the CBUB_Response_Record

### 4.14.2 CBUB Response Record

The structure of this data is as shown below. Note that the Card Block / Unblock MACs are passed in clear and no KTU is present unless the request included a key used to protect it.

Data	Definition
CBUB_Record_Length	Binary, 4 bytes. The length of the CBUB record
<b>CBUB_Record</b>	CBUB data from the CA.
KTU_Length	Binary, 2 bytes. The length of the KTU. If CBUB is not encrypted ( i.e. no optional key was supplied in the CBUB request ) this field takes value 0x0000 and no KTU is present
<b>CBUB_Protection_KTU</b>	If present, this is the KTU used to protect the CBUB. It contains the session key used to encrypt the CBUB and is encrypted with the public key supplied in the CBUB request

## 4.14.3 CBUB Record

Data	Definition
Block_Unblock_Flag	Binary, 1 byte. Set to 0x5a for Block, or 0xa5 for Unblock, or 0x55 for both
Number_Of_Card_Records	Binary, 4 bytes. The number of CBUB_Card_Records, in the range 1 to 50,000
{ Encrypted_CBUB_Data }	Occurs Number_Of_Card_Records times. If the CBUB was not requested to be encrypted, this is the plaintext CBUB

## 4.14.4 Encrypted CBUB Data

Data	Definition
<i>MCD_Number</i>	Binary, 8 bytes
MAC	Binary, 8 bytes. The Block or Unblock MAC requested
MAC	Binary, 8 bytes. If both MACs have been requested then the Block MAC is first, then the Unblock MAC. If only one MAC has been requested then this field is not present.

## 4.14.5 CBUB Protection KTU

This is the key used to decrypt the CBUB. It is encrypted using the public key supplied by the CBUB requester.

The CBUB is protected by DES Cipher Block Chain encryption of the CBUB starting with the first byte of the CBUB (Triple DES is used with Encryption by the first key, then Decryption by the second followed by Encryption by the first one again). The plaintext\_ktu is shown in the following table. This is obtained by decrypting the CBUB\_Protection\_KTU with the secret key corresponding to the public key sent with the CBUB request.

<b>Data</b>	<b>Definition</b>
<i>Header_Byte</i>	Binary, 1 byte
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the following 3 keys
<i>CBUB_Encryption_Key_1</i>	Binary, 8 bytes. The first component of the key
<i>CBUB_Encryption_Key_2</i>	Binary, 8 bytes. The second component of the key
<i>CBUB_Encryption_IV</i>	Binary, 8 bytes. The initial IV for CBUB encryption
<i>Padding_Random</i>	Binary, N bytes of random padding. The padding is used to extend the length of this structure to the size of the key used to encrypt the CBUB record
<i>Trailer_Byte</i>	Binary, 1 byte

## **5 Additional MAOSCO Advisory File Formats**

There are two files which are necessary for the MULTOS scheme which are not sent to, or received from, a MULTOS KMA. These files are the ALU (Application Load Unit) request and response files.

The ALU request file is sent from an Issuer to an Application Provider. The Application Provider responds with the ALU.

These files do not interface to a MULTOS KMA; however MAOSCO advises that all parties should adopt the following file formats.

## 5.1 ALU Request List File

This file is not sent to a MULTOS KMA. It is sent to the Application Provider in order to obtain a set of ALUs. It is included in this document for completeness. It is recognised that the following fields are likely to be a subset of those required by any one Application Provider.

### 5.1.1 ALU Request File

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ALUQ"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x00
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>Consignment_File_ID</i>	ASCII, 8 characters
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<b>Header_ALU_Request</b>	
<i>Application_ID_Field</i>	Binary, 17 bytes
{ <b>Card_Data_List_Record</b> }	Occurs Number_Of_MCD_Number_List_Records_Or_ALUs times.

### 5.1.2 Header ALU Request

Data	Definition
<i>Bureau_ID</i>	Binary, 4 bytes. If the request is sent by an Issuer then this field should be set to be the same as the Issuer ID
<i>Issuer_ID</i>	Binary, 4 bytes
MKD_pk_c_Length	Binary, 2 bytes. Indicates the length of the MKD_pk_c record entry associated with each MCD number.
Number_Of_MCD_Number_List_Records_Or_ALUs	Binary, 4 bytes. If the MKD_pk_c_length is non-zero, then this is the number of entries in the following list.  If the MKD_pk_c_length is zero, then this is the number of generic ALUs required.

Note that if the ALU is not MCD specific, then set the MKD\_pk\_c\_length to 0x0000 and the Number\_Of\_MCD\_Number\_List\_Records\_Or\_ALUs to the number of ALUs required. In this case do not include any Card\_Data\_List\_Records.

### 5.1.3 Card Data List Record

Data	Definition
<i>MCD_Number</i>	Binary, 8 bytes. This is the individual card's MCD_Number used to generate the KTU by the application provider
<i>MKD_pk_c</i>	Binary, MKD_pk_c_length bytes. This is the card public key certificate for the corresponding MCD.

## 5.2 ALU Data List Files (version 01)

This file is not sent by a MULTOS KMA. It is sent by the Application Provider and is included in this document for completeness. Whilst outside the scope of a MULTOS KMA, all MULTOS participants are strongly encouraged to use this file format.

It should not be assumed that Application Providers will always package ALUs as multiple entries in a single file. It is also possible for a set of ALUs to be returned as a set of files with one or more ALUs per file.

Consignment identifiers are a matter for agreement with Application providers.

### 5.2.1 ALU Data List File

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ALUR"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x00
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>AP_consignment_ID</i>	ASCII, 10 characters. This is a unique reference created by the Application Provider to allow them to track files
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<i>Consignment_File_ID</i>	ASCII, 8 characters
<b>Header_ALU_Data</b>	
{ <b>ALU_Data_List_Record</b> }	Occurs Number_Of_ALU_Data_List_Records times

## 5.2.2 Header ALU Data

Data	Definition
<i>Application_ID_Field</i>	Binary, 17 bytes
ALU_record_length	Binary, 4 bytes. The length of each ALU record. Note that it is a requirement that each ALU be of the same length. If not, then separate files will be required for each size.
Number_Of_ALU_Data_List_Records	Binary, 4 bytes. This is the number of entries in the subsequent list. Note that the original request file may require a number of separate response files in order to satisfy it (if the ALU sizes are different). Hence the sum of all the Number_Of_ALU_Data_List_Records in all the response files to one request should match that of the original request file

## 5.2.3 ALU Data List Record

Data	Definition
<i>MCD_Number</i>	Binary, 8 bytes. The MCD_Number that this ALU corresponds to. If it is an unencrypted application, this is set to 8 bytes of 0x00
Code_Record_Length	Binary, 2 bytes
Code_Record	Binary, Code_Record_Length bytes
Data_Record_Length	Binary, 2 bytes
Data_Record	Binary, Data_Record_Length bytes
DIR_Record_Length	Binary, 2 bytes
DIR_Record	Binary, DIR_Record_Length bytes
FCI_Record_Length	Binary, 2 bytes
FCI_Record	Binary, FCI_Record_Length bytes
App_Signature_Length	Binary, 2 bytes
App_Signature	Binary, App_Signature_Length bytes
KTU_Length	Binary, 2 bytes
KTU	Binary, KTU_Length bytes

## 5.3 ALU Data List Files (version 02)

This is an alternative to version 01, not a replacement.

This file is not sent by a MULTOS KMA. It is sent by the Application Provider and is included in this document for completeness. Whilst outside the scope of a MULTOS KMA, all MULTOS participants are strongly encouraged to use this file format.

It should not be assumed that Application Providers will always package ALUs as multiple entries in a single file. It is also possible for a set of ALUs to be returned as a set of files with one or more ALUs per file.

Consignment identifiers are a matter for agreement with Application providers.

### 5.3.1 ALU Data List File

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "ALUR"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x02
<i>AP_Consignment_ID</i>	ASCII, 10 characters. This is a unique reference created by the Application Provider to allow them to track files
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<i>Consignment_File_ID</i>	ASCII, 8 characters
<b>Header_Encrypted_ALU_Data</b>	
{ <b>Encrypted_ALU_Data_List_Record</b> }	Occurs Number_Of_ALU_Data_List_Records times
ALU_Padding	Binary N bytes of padding of value 0x55. The padding is used to extend the length of Number_Of_ALU_Data_List_Records * ALU_Record_Length to an integer multiple of 8.

## 5.3.2 Header Encrypted ALU Data

Data	Definition
<i>Application_ID_Field</i>	Binary, 17 bytes
ALU_Record_Length	Binary, 4 bytes. The length of each ALU record. Note that it is a requirement that each ALU be of the same length. If not, then separate files will be required for each size.
Number_Of_ALU_Data_List_Records	Binary, 4 bytes. This is the number of entries in the subsequent list. Note that the original request file may require a number of separate response files in order to satisfy it (if the ALU sizes are different). Hence the sum of all the Number_Of_ALU_Data_List_Records in all the response files to one request should match that of the original request file
ALU_Protection_KTU_Length	Binary, 2 bytes. This is the length of the KTU used to protect the ALU.
<b>ALU_Protection_KTU</b>	This is the KTU used to protect the ALU. It contains the session key used to encrypt the ALU and is encrypted with a public key supplied from the Bureau.

## 5.3.3 ALU Protection KTU

This is the key used to decrypt the ALU. It is encrypted using the public key supplied by the Bureau.

The ALU is protected by DES Cipher Block Chain encryption of the ALU starting with the first byte of the ALU (Triple DES is used with Encryption by the first key, then Decryption by the second followed by Encryption by the first one again). The plaintext\_ktu is shown in the following table. This is obtained by decrypting the ALU\_Protection\_KTU with the secret key corresponding to the public key.

Note that all the ALUs are encrypted as one large block of plaintext data, with one set of padding on the end (as required). This is for speed of encryption.

<b>Data</b>	<b>Definition</b>
Header_Byte_5b	Binary, 1 byte. Set to 0x5B
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the following 3 keys
<i>ALU_Encryption_Key_1</i>	Binary, 8 bytes. The first component of the key
<i>ALU_Encryption_Key_2</i>	Binary, 8 bytes. The second component of the key
<i>ALU_Encryption_IV</i>	Binary, 8 bytes. The initial IV for ALU encryption
<i>Padding</i>	Binary N bytes of padding of value 0x55. The padding is used to extend the length of this structure to the size of the key used to encrypt the ALU
Trailer_Byte_a4	Binary, 1 byte. Set to 0xA4

## 5.3.4 Encrypted ALU Data List Record

Data	Definition
<i>MCD_Number</i>	Binary, 8 bytes. The MCD number that this ALU corresponds to. If it is an unencrypted application, this is set to 8 bytes of 00
Code_Record_Length	Binary, 2 bytes
Code_Record	Binary, Code_Record_Length bytes
Data_Record_Length	Binary, 2 bytes
Data_Record	Binary, Data_Record_Length bytes
DIR_Record_Length	Binary, 2 bytes
DIR_Record	Binary, DIR_Record_Length bytes
FCI_Record_Length	Binary, 2 bytes
FCI_Record	Binary, FCI_Record_Length bytes
App_Signature_Length	Binary, 2 bytes
App_Signature	Binary, App_Signature_Length bytes
KTU_Length	Binary, 2 bytes
KTU	Binary, KTU_Length bytes

## 6 Key Disk Formats

### 6.1 TKCK

This is required by an Application Provider when generating encrypted ALUs, to allow the verification of the `mkd_pk_c`. It is provided by a MULTOS KMA.

These files will always be distributed with a filename of the form:

'tkckxxxx.key'

where 'xxxx' represents the `MKD_Cert_Method_ID`.

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "TKCK"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>Consignment_File_ID</i>	ASCII, 8 characters. Set to 'TKCK', followed by 4 characters presenting the identifier (in hex). This will be the same as the <i>MKD_Cert_Method_ID</i> .  For example 'TKCK0113', would be version 19 (decimal) of a MULTOS 4 96 byte TKCK
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<i>MKD_Cert_Method_ID</i>	Binary, 2 bytes. Comprised of : scheme ID, 1 byte + key version number, 1 byte
<i>Key_Length</i>	Binary, 2 bytes. This should match the value inferred from the scheme ID byte of the <i>MKD_Cert_Method_ID</i>
<i>Key_Data</i>	Binary, <i>Key_Length</i> bytes. The actual TKCK public key
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the <i>MKD_Cert_Method_ID</i> , <i>Key_Length</i> and <i>Key_Data</i>

## 6.2 Hash Modulus

This is required by an Application Provider when generating application signatures. It is provided by a MULTOS KMA.

These files will always be distributed with a filename of the form:

'hashxxxx.mhm'

where 'xxxx' represents the Hash\_Method\_ID.

Data	Definition
<i>File_Type_Code</i>	ASCII, 4 characters. Set to "HASH"
<i>File_Protection_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>File_Structure_Method_ID</i>	Binary, 1 byte. Set to 0x01
<i>Consignment_File_ID</i>	ASCII, 8 characters. Set to 'HASH', followed by 4 characters presenting the identifier (in hex). This will be the same as the <i>Hash_Method_ID</i> .  For example 'HASH0105'
<i>Date</i>	Date, 4 bytes
<i>Time</i>	Time, 3 bytes
<i>Hash_Method_ID</i>	Binary, 2 bytes
<i>Key_Length</i>	Binary, 2 bytes
<i>Key_Data</i>	Binary, <i>Key_Length</i> bytes. The actual Hash Modulus
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the <i>Hash_Method_ID</i> , <i>Key_Length</i> and <i>Key_Data</i>

## 7 Request and Management Data Signatures

### 7.1 Signature Generation

Data is split into two parts, a plaintext part and a ciphertext part. The plaintext part of all the files in this document contains the following, in clear:

- If the data record constitutes a request record, then the plaintext part contains the Issuer\_ID / Bureau\_ID, followed by the Key\_Serial\_No.
- If the data record constitutes a management record, then the plaintext part contains the Key\_Serial\_No.

This is summarised as follows:

Request source	Request record header	Management record header
Issuer	Issuer_ID Key_Serial_No	Key_Serial_No
Bureau	Bureau_ID Key_Serial_No	Key_Serial_No

Now, let the length of the data record (excluding padding) be LARD.

Let the length of the key be LIK.

Let the length of the plaintext header (0, 4 or 8 bytes) be LMID

If  $LARD - LMID + 22 < LIK$  then padding of length  $LIK - LARD + LMID - 22$  must be added.

Now, let  $LARD' = LIK + LMID - 22$ . This is the total length of data including any padding that may have been added.

The file header is made up of  $LARD' - LIK + 22$  bytes. The remaining  $LIK - 22$  bytes is the file trailer which is added to the 1 byte header, hash and 1 byte trailer and then encrypted using the key indicated to form the Signature\_Ciphertext.

## 7.2 Signed Data

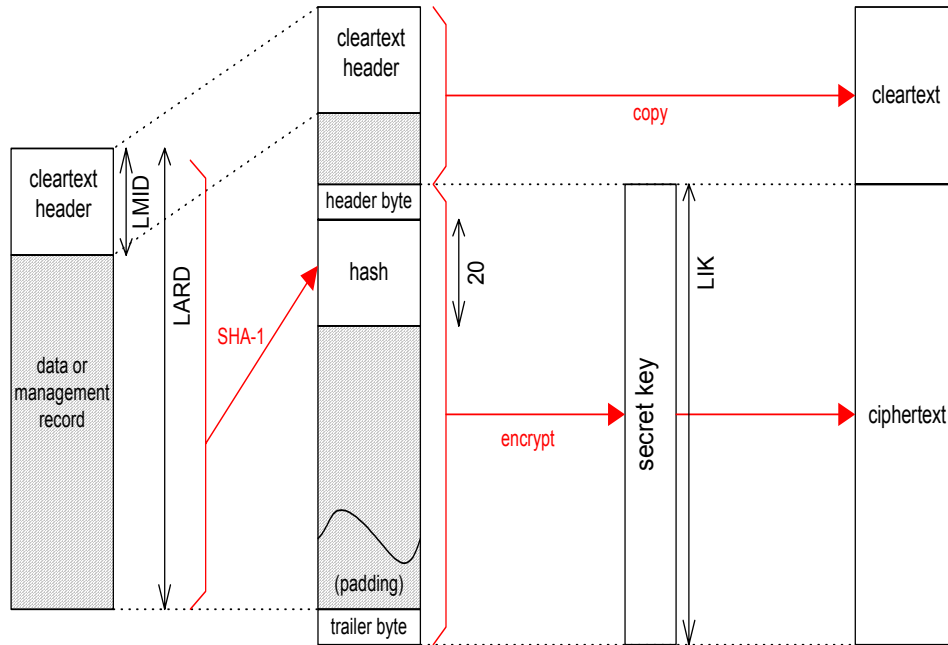
Data	Definition
Signature_PLAINTEXT	Plaintext part of the signed data. LARD' – LIK + 22 bytes long
Signature_CIPHERTEXT	LIK bytes ciphertext as defined below

## 7.3 Signature Ciphertext

Data	Definition
<i>Header_Byte</i>	Binary, 1 byte. Set to 0x5A (encrypted)
<i>Hash_Code</i>	Binary, 20 bytes. A SHA-1 hash of the request / management data
File_Trailer	Binary, LIK – 22 bytes. The ciphertext part of the request
<i>Trailer_Byte</i>	Binary, 1 byte. Set to 0xA3 (encrypted)

### 7.4 Pictorial Explanation

Here is the same information in pictorial form:



## 8 DATA DICTIONARY

Type	Definition	Description
<i>???</i> _Encryption_IV	Binary, 8 bytes	An Initialisation vector used to protect data. ??? can be one of ALC, ADC, CBUB, ALU
<i>???</i> _Encryption_Key_1, 2, 3	Binary, 8 Bytes	A cryptographic key used to protect data. ??? can be one of ALC, ADC, CBUB, ALU
<i>Access_List</i>	Binary, 2 bytes	List of bit fields used to define MULTOS features such as access to cryptographic primitives
<i>ALC_ID</i>	Binary, 8 bytes	An identifier for an ALC or ADC, assigned by a KMA
<i>ALC_ID_Length</i>	Binary, 1 byte	The length of an ALC_ID, currently always set to 8
<i>AMD_ID</i>	ASCII, 8 characters	Of the form "NNNNvMMM" where N and M indicate ASCII digits. For example "0009v002". Set to "0000v000" to indicate that a KMA should use a default AMD
<i>AP_Consignment_ID</i>	Binary, 10 bytes	A unique identifier created by an Application Provider to allow tracking of a file
<i>App_ATR_Type</i>	Binary, 1 byte	Set to 0x41 for App_ATR Set to 0x00 for no App_ATR (Set to 0x42 for Alt_App_ATR for <b>MULTOS 4 ONWARDS</b> )
<i>Application_Code_Hash</i>	Binary, variable	A hash over the application code space
<i>Application_Code_Hash_Length</i>	Binary, 1 byte	Either 0x14 for SHA-1, or 0x10 for proprietary, or 0x00 for no hash
<i>Application_ID</i>	Binary, 16 bytes	The Application ID for an Issuer's application as defined in ISO/IEC 7816-5 (padded with 0xff if required)
<i>Application_ID_Field</i>	Binary, 17 bytes	Comprised of : 1 byte indicating how many bytes of the Application_ID are significant + <i>Application_ID</i>
<i>Application_Variant</i>	Binary, 2 bytes	Differentiates between differing instances of a registered application with the same application ID. This allows an Issuer to register many slight variations of the same application simultaneously and select between them when ordering.

# Proprietary

Type	Definition	Description
<i>ATR</i>	Binary, 18 bytes	The Initial, Format and optional Interface characters of an Answer-To-Reset as defined in ISO/IEC 7816-3. Actual length required may be less than 18 bytes, but is padded with 0xFF to length
<i>Billing_Details</i>	<i>Name_Address_Record</i>	The party to be billed for KMA services
<i>Bureau_ID</i>	Binary, 4 bytes	Identifies a Bureau as the sender of a request for MSM Controls Data
<i>Card_ID</i>	Binary, 4 bytes	A unique identifier for an M-SPI card distributed to an Issuer or Bureau
<i>Cert_Method_ID</i>	Binary, 2 bytes	Used to match an ALC or ADC against a ROM mask in an MCD, to ensure correct algorithms and keys are applied
<i>Code_Size</i>	Binary, 2 bytes	The amount of code space an application needs on a MULTOS chip, as detailed in the Application Load Certificate
<i>Consignment_File_ID</i>	ASCII, 8 characters	This is an identifier created by the creator of the file by which it will be managed and tracked. A KMA will incorporate this in the returned file to allow the originator to track it
<i>Data_Size</i>	Binary, 2 bytes	The amount of data space an application needs on a MULTOS chip, as detailed in the Application Load Certificate
<i>Date</i>	Binary, 4 bytes year (2 bytes) + month (1 byte, 1..12) + day (1 byte, 1..31)	Note that data is stored big-endian, so that the year 1997 (0x07CD), would be stored as a byte of value 07 followed by a byte of value CD. Please note that these items are binary NOT BCD. 31 <sup>st</sup> December 1997 would be coded in Hexadecimal as <b>07 CD 0C 1F</b>
<i>Delivery_Details</i>	<i>Name_Address_Record</i>	The intended address for delivery of KMA services output
<i>DIR_File_Record_Size</i>	Binary, 2 bytes	The length of the entry in the EF <sub>DIR</sub> file for this application
<i>Exponent</i>	Binary, variable	An RSA exponent ( usually public ) used with a modulus to sign or encrypt data
<i>FCI_Record_Size</i>	Binary, 2 bytes	The number of bytes to be returned when an application is selected by a terminal
<i>File_Mode_Type</i>	Binary, 1 byte	An identifier for the type of application. A value of 0x5A defines a 'Shell' application. A value of 0xA5 defines a 'Default' application. A value of 0x00 defines a standard application
<i>File_Protection_Method_ID</i>	Binary, 1 byte	Identifier for method used to protect the file. Meaning is in context of file type

# Proprietary

Type	Definition	Description
<i>File_Structure_Method_ID</i>	Binary, 1 byte	Identifier for how data is structured within the file. Its meaning is in context of file type
<i>File_Type_Code</i>	ASCII, no terminator, 4 bytes	Defines file type as defined in section 2
<i>Hash_Code</i>	Binary, 20 bytes	A SHA-1 hash as defined in FIPS PUB 180-1
<i>Hash_Method_ID</i>	Binary, 2 bytes	Used in an ALC or ADC to compare against the value in the ROM mask of an MCD to verify that the correct Hash Modulus is in use
<i>Header_Byte</i>	Binary, 1 byte	One byte header value used in signed requests. Value of 0x5A
<i>IC_Manufacturer_ID</i>	Binary, 1 byte	Used to identify the manufacturer of the silicon chip. A current list of possible identifiers can be requested from MAOSCO or MULTOS Customer Support
<i>IC_Type</i>	Binary, 1 byte	Used to identify a particular mask of a silicon chip. A current list of possible identifiers can be requested from MAOSCO or MULTOS Customer Support
<i>Issuer_ID</i>	Binary, 4 bytes	The MCD Issuer Id, assigned by the RA upon Issuer registration, for a specific Issuer. A Unique handle by which the Issuer is known
<i>Key_Certificate</i>	Binary, variable	An actual MULTOS enablement, load or delete certificate
<i>Key_Serial_Number</i>	Binary, 4 bytes	Used to identify a key. Unique per Issuer, Bureau or RA. Split into 'Distribution' keys (top bit = 0) and 'Operational' keys (top bit = 1).
<i>Key_Type</i>	Binary, 1 byte	Indicates whether ALC or ADC is requested. Takes value of 0x43 for ALC or 0x44 for ADC
<i>KMA_Consignment_File_ID</i>	ASCII, 10 characters	A unique identifier created by a KMA for a file
<i>Length_ATR</i>	Binary, 1 byte	Indicates how many of the 18 bytes of ATR are actually relevant (in the range 0<=length<=18)
<i>MCD_ID</i>	Binary, 6 bytes	The identifier for an MCD, assigned during MCD manufacture
<i>MCD_Issuer_Product_ID</i>	Binary, 1 byte	The product ID to be assigned to this set of MCDs, denoting the subclass of the Issuer's card base
<i>MCD_Issuer_Product_IDs</i>	Binary, 32 bytes	A 256 bit field of flags corresponding to each of the possible 256 <i>MCD_Issuer_Product_ID</i> values

# Proprietary

Type	Definition	Description
<i>MCD_Number</i>	Binary, 8 bytes	Unique reference for a card assigned by KMA in MSM Controls Data
<i>MKD_Cert_Method_ID</i>	Binary, 2 bytes	Used to match a TKCK against an MCD when generating encrypted ALUs
<i>MKD_PK_C</i>	Binary, variable	Card public key certificate. Sent with MSM data
<i>Modulus</i>	Binary, variable	An RSA modulus used with an exponent ( either specified explicitly or implicitly assumed to be of value 3 depending on context ) used to sign or encrypt data
<i>MSM_App_Permissions</i>	Binary, 76 bytes	<i>MCD_Issuer_Product_IDs</i> + <i>Issuer_ID</i> + <i>MSM_Controls_Data_Dates</i> + <i>MCD_Number</i>
<i>MSM_App_Permissions_Scheme_ID</i>	Binary, 1 byte	Indicates the structure of the following MSM Permissions in an ALC or ADC request
<i>MSM_Controls_Data_Date</i>	Binary, 1 byte	The date on which MSM Controls were generated. This is an offset in months from January 1998 ( = 0). Hence controls generated in September 1999 for example would have a date of 20 (decimal).
<i>MSM_Controls_Data_Dates</i>	Binary, 32 bytes	A 256 bit field of flags corresponding to each of the possible 256 <i>MCD_Controls_Data_Date</i> values
<i>MSM_MCD_Permissions_Scheme_ID</i>	Binary, 1 byte	Indicates the structure of the following MSM Permissions request
<i>Name_Address_Record</i>	ASCII, 796 characters  First_Name (30 char) + Last_Name (30 char) + Company_Name (70 char) + Address_Line1 (60 char) + Address_Line2 (50 char) + Address_Line3 (50 char) + Address_Line4 (50 char) + City_Town (30 char) + County_State (30 char) + Country (50 char) + Postcode_Zipcode (30 char) + Telephone (30 char) + Fax (30 char) + Email(s) (254 char) + MSPI_Version (1 byte) + Delivery_Option (1 byte)	Used between M-SPI and COO to allow electronic ordering of services.  Note that there may be multiple email addresses, separated by commas.
<i>Padding</i>	Binary, variable	Optional padding of zero or more bytes of a fixed value of 0x55

# Proprietary

Type	Definition	Description
<i>Padding_ff</i>	Binary, variable	Optional padding of zero or more bytes of a fixed value of 0xFF
<i>Random_Seed</i>	Binary, 8 bytes	Used in MULTOS 4 onwards to implement a history list within the card
<i>ROM_Identifier</i>	Binary, 2 bytes	An identifier for a set of ROMs that share a common KCK
<i>Sender_Details</i>	<i>Name_Address_Record</i>	The originator of a service request
<i>Session_Data_Size</i>	Binary, 2 bytes	The amount of session data space an application needs on a MULTOS chip, as detailed in the Application Load Certificate
<i>Time</i>	Binary, 3 bytes hour (1 byte, 0..23) + minute (1 byte, 0..59) + second (1 byte, 0..59)	Note that values are stored in binary, so 42 seconds past half ten in the evening would be 22:30:42 on a 24 hour clock and would be coded as <b>16 1E 2A</b>
<i>Trailer_Byte</i>	Binary, 1 byte	One byte header value used in signed requests. Value of 0xA3
<i>Verify_Certificate_Flag</i>	Binary, 1 byte	Flag used to denote whether an ALU signature is present or not Set to 0x4E for not present Set to 0x50 for present
<i>Verify_KTU_Flag</i>	Binary, 1 byte	Flag used to denote whether an ALU KTU is present or not Set to 0x4E for not present Set to 0x50 for present
<i>x_parameter</i>	Binary, 1 byte	The inter character transmission time. Set to 0x00 to indicate that a KMA should use a default value
<i>y_parameter</i>	Binary, 1 byte	The message processing time. Set to 0x00 to indicate that a KMA should use a default value

--end of document--