limes datentechnik

FLAMfi-sub (MVS)

FRANKENSTEIN-LIMES-ACCESS-METHOD

USER MANUAL

— Edition May 2014 Version 4.5 —

' Copyright 1992-2014 by limes datentechnikfi gmbh ■ Louisenstra e 21 ■ D-61348 Bad Homburg
Telephone ++49 6172 / 5919-0 ■ Telefax ++49 06172 / 5919-39
http://www.flam.de ■ http://www.limes-datentechnik.de

User Manual FLAMfi -sub V4.5 (MVS)

'   Copyright 2014 by limes datentechnikfi gmbh

All rights reserved. The reproduction, transmission or use of this document is not permitted without express written authority.

Offender will be liable for damages.

Delivery subject to availability, right of technical modifications reserved.

Preface

This manual describes how to handle the FLAM subsystem in the IBM z/OS (OS/390, MVS) operating system.

FLAMfi -sub is an interface to FLAMfi (MVS), the compression and encryption utility, and may only be used in conjunction with a license for FLAMfi -utility.

FLAMfi (MVS) is described in the manual FLAMfi (MVS) V4.5, the encryption method is found in the manual FLAMfi & AES.

FLAMfi, FLAMFILEfi und limes datentechnikfi are international trademarks.

FLAMfi-sub (MVS)

User Manual

Summary of changes

Summary of changes 7 FLAMfi-sub V4.5

This release is an adaption to the FLAMfi (MVS) utility V4.5.

It includes the following new features

- **KME=FKMEFILE**

This key management extension routine reads a key from a sequential file to encrypt/decrypt the written/read data via the FLAMfi subsystem.

The file may be of fixed or variable record length. Trailing blanks are ignored.

The syntax of the key is the same as of parameter CRYPTOKEY.

The file may be protected by RACF, so this is an easy way for encryption/decryption without any key-protocol.

- **License check**

The subsystem modules are now included in the FLAMfi (MVS) LOAD library. There is an own license check implemented avoiding using the FLAM-subsystem without a special license.

Summary of changes 6 FLAMfi-sub V4.4

The Key Management Interface of FLAM (MVS) (-> manual FLAM (MVS) V4.x, ch. 3.5.5) is implemented in this subsystem release:

A user written module is invoked when opening a data set to provide FLAM-sub with a key for encryption/decryption.

This method allows an automatic encryption/decryption of files without any manual user action. The key used is not shown in any protocol.

Summary of changes 5 FLAMfi-sub V4.3

Introducing crypto hardware CPACF in newer hardware systems z9 and z10 allows FLAMfi-sub V4.3 to use these new routines for AES encryption.

FLAMfi-sub automatically checks the availability, so no parameter other than CRYPTOMODE=AES is needed.

Encryption by hardware increases perfomance and decreases cputime. Up to 30 % may be. It depends on the compression ratio, less compression increases time savings. Particularly using MODE=NDC, packing files without compression.

Summary of changes 4 FLAMfi-sub V4.1

FLAMfi-sub V4.1 benefits from the new and faster implementation of the AES algorithm in FLAM (MVS) V4.1.

Up to 50 % CPU time is saved using the new AES routines in the subsystem.

Summary of changes 3 FLAMfi-sub V4.0

FLAMfi-sub V4.0 incorporates the following changes as compared with Version 3:

Support of AES-encryption (Advanced Encryption Standard)

Using parameter CRYPTOMODE=AES all data will be encrypted in Advanced Encryption Standard mode, introduced in FLAM (MVS) V4. Additional information (hash-MACs) is stored in the FLAMFILE.

FLAMFILE split is supported

During output mode (writing records into an empty file) a FLAMFILE can be split serially or in parallel into several parts, subject to the settings of the parameters SPLITMODE, SPLITNUMBER, and SPLITSIZE.

MODE=NDC (No Data Compression) is supported

Data compression can be suppressed using MODE=NDC. Data are only formatted and, if requested, encrypted. This saves CPU time with data that do not compress efficiently (e.g. FLAMFILEs or compressed image files). The same security features are available as for compressed data.

MODE=NDC is downwards compatible with FLAM V3.x.

Summary of changes 2 FLAMfi-sub V3.0

FLAMfi-sub V3.0 incorporates the following changes as compared with Version 2:

MODE=ADC (Advanced Data Compression) is supported

Using MODE=ADC all data are compressed by the new ADC-algorithm of FLAM V3.x, the highest efficient compression method for all data (even non-structured data).

Loading VSAM-KSDS with a Utility (i.e. IEBGENER, SORT)

Although utilities use a (sequential) DCB to open a subsystem data set, it is now possible to load a VSAM-KSDS file with FLAMfi-sub.

Activate FLAMfi-sub as a 'Started Task'

Start FLAMfi-sub as a Started Task (S FLAM). So it is not longer necessary to wait for an IPL to test FLAMfi-sub.

Summary of changes 1 FLAMfi-sub V2.0

FLAMfi-sub V2.0 incorporates the following changes as compared with Version 1:

VSAM accesses supported

The calling programs can be written either in Assembler or in COBOL.

All logical VSAM accesses are supported by FLAMfi-sub V2.0, in both MOVE and LOCATE mode and for both synchronous and asynchronous calls.

In particular, VSAM-KSDS commands are also converted logically if an UPDATE is performed.

The index entries have been reduced to a minimum and compressed, so that less space is taken up in the memory and the overall performance is enhanced when VSAM files are accessed.

Physical accesses via memory addresses (RBA) are not supported.

Reading of uncompressed files supported

If the input is an uncompressed file, it can also optionally be read via the subsystem.

In this case, the keyword parameter IG10 must be entered for the subsystem:

```
//... DD ...,SUBSYS=(FLAM,IG10,'flam-parameter')
```

This parameter has no effect on compressed files.

TRACE function

A TRACE function can be activated for test purposes.

In this case, the keyword parameter TRACE must be entered for the subsystem:

```
//... DD ...,SUBSYS=(FLAM,TRACE,'flam-parameter')
```

All function calls to FLAM, as well as ACB and RPL of the calling program, are then traced in a file.

The trace file must be specified in the JCL. The FLAM parameter ’MSGDDN=ddname’ indicates the DD name. If no name is specified the parameter for the DEFAULT specification is used (generally FLPRINT; cf. INST02 job of the FLAM utility).

The trace file can also be specified as a subsystem file:

```
//FLPRINT  DD DSN=trace-file,SUBSYS=FLAM
```

New interface to FLAM utility

As of this version, all the required FLAM utility modules are loaded as resident programs, in other words the subsystem modules no longer need to be linked to the utility. Consequently, no ’relinks’ are necessary if the license number for the utility changes.

On the other hand, the load module library of FLAM-utility now also needs APF authorization.

We recommend keeping the subsystem and utility load modules in one library.

FLAMfi-sub (MVS)

User Manual

Contents

Chapter 1	**1.**	**FLAMfi as a subsystem**	**1**
Chapter 2	**2.**	**Subsystem call**	**1**
Chapter 3	**3.**	**Preconditions**	**1**
Chapter 4	**4.**	**Principle of operation**	**1**
Chapter 5	**5.**	**Restrictions**	**1**
	5.1	In general	1
	5.2	DCB calls (PS files)	2
	5.3	ACB/RPL calls (VSAM files)	2
Chapter 6	**6.**	**Parameters for FLAMfi**	**1**
Chapter 7	**7.**	**Parameters for controlling the subsystem**	**1**
Chapter 8	**8.**	**Subsystem messages**	**1**
Chapter 9	**9.**	**Installing FLAMfi-sub**	**1**
	9.1	LINKLST and authorization	2
	9.2	Subsystem start	3
Chapter 10	**10.**	**Examples**	**1**
	10.1	Input/output with catalogued files	1
	10.2	FLAMfi parameters and subsystem	2
	10.3	Creating a new file	2
	10.4	Temporary files	3
	10.5	Other subsystems	3
	10.6	Loading a VSAM-KSDS file	3
	10.6.1	Load a real VSAM-KSDS file	4
	10.6.2	Load a VSAM-KSDS file via an utility	4
	10.7	TRACE function	5
	10.8	FLAMFILEfi split	7
	10.8.1	Serial split	7

10.8.2 Parallel split 8

10.9 Encryption 9
10.9.1 Encryption via CRYPTOKEY parameter 9
10.9.2 Encryption via KME module 9
10.9.3 Encryption with FKMEFILE 10

' 2010 by limes datentechnik gmbh

FLAMfi-sub (MVS)

User Manual

Chapter 1:

FLAMfi as a subsystem

1. FLAMfi as a subsystem

FLAMfi -sub supports the subsystem interface of the command language (JCL) in z/OS. Files can thus be processed in compressed an encrypted form without having to modify the associated programs. The additional compression and decompression steps which were necessary in the past can now be dispensed with.

The calling program is not aware of any differences as compared with the conventional mode of file processing. It receives a record for a read call in the same way as before, while for a write call the data management system (DMS) receives the record as usual. The subsystem decompresses before reading and compresses before writing.

The compression results are similar to those of the FLAM utility (approx. 70-90%).

The FLAM subsystem even enables programs to be supported with file formats for which they were not originally written, in other words the program and the data formats can be separated from one another.

The FLAM subsystem is loaded almost entirely in the high address space (above 16 MB); only a small tuning module for 24-bit addressing works in the low address space. Since 24-bit addressing is used for the DMS accesses to PS files, a memory area corresponding to the length of one data record is created in the low address space. FLAM creates all work areas above the 16 MB limit, so that the usual memory area is still made available to the applications.

No changes are normally necessary in the calling programs.

The compressed files created by the subsystem (FLAMFILEs) can be decompressed by the FLAM utility at any time, and all FLAMFILEs created by the FLAM utility are accepted and processed logically by the FLAM subsystem. The same applies likewise to FLAMFILEs that are created/read via the record interface of FLAM (see also manual for FLAM V4.x).

' 2014 by limes datentechnik gmbh

FLAMfi-sub (MVS)

User Manual

Chapter 2:

Subsystem call

2. Subsystem call

```
//ddname    DD  DSN=filename,
//                DISP=OLD,
//                SUBSYS=(FLAM,'flam-parameter')
```

This call designates the cataloged file filename for processing with the FLAM subsystem.

FLAM parameters can be specified in addition, in the same way as with the FLAM utility.

Invalid parameters are rejected by FLAM as JCL errors and the job they refer to is not even started. The DD command is still verified by JES.

The catalog entry for the file is not modified by FLAM. All entries such as the file name, the record and block lengths, the volume, etc. remain unchanged.

The files that must be processed with the FLAM subsystem are thus still verified by the check mechanisms, such as SMS and RACF.

There is no FLAM restriction on the number of files that can be processed "simultaneously" by the subsystem. The only restrictions are those imposed by the system itself, such as the maximum amount of available memory or the maximum possible number of DD statements.

If files for which the SUBSYS specification in the DD statement is not supported must be processed (for example, JES files), the problem can be overcome by specifying a FLAM parameter and a second DD statement:

```
//ddname    DD SUBSYS=(FLAM,'FLAMDDN=ddname1'),
//             DCB=(LRECL=....,BLKSIZE=....)
//ddname1   DD SYSOUT=G,DEST=(.....),
//             DCB=(LRECL=80,.....)
```

Specifying a DD name ddname1 as a FLAM parameter causes the subsystem to process the file assigned by means of this name (in this case by writing in it). This file must be specified in the JCL. Similarly, ddname1 could be used as a decompression input. The calling program, on the other hand, uses the file assigned with ddname!

This method permits reading or writing in any file (JES, RJE, other subsystems, magnetic tapes, temporary files, etc.) which cannot otherwise be processed with the usual method (see examples).

If DCB attributes are specified for ddname, FLAM interprets them as values for the original (uncompressed) file. DCB specifications for ddname1 apply to the compressed file (the FLAMFILE).

Record and block formats which are completely different from one another can thus be set for the original and compressed files depending on the particular problem, in other words the subsystem allows an application to use files that have absolutely nothing in common with the entries in the program (see examples) and that without the subsystem might have to be converted.

If the FLAMFILE is accessed without a SUBSYS specification, it behaves like a "normal" file, in other words it can be read (copied, transferred) by means of utilities or file transfer programs without being decompressed!

Similarly, a FLAMFILE which is transferred by a file transfer program can be read and processed via the subsystem.

' 2014 by limes datentechnik gmbh

FLAMfi-sub (MVS)

User Manual

Chapter 3:

Preconditions

3. Preconditions

FLAM must be installed on the computer (see installation instructions) both as a utility (V4.0 or higher) and as a subsystem.

Data records can be written, read, modified, inserted or deleted. The data set organization may be of PS or VSAM.

The accesses must be logical, in other words either in sequential order or according to a key. Accesses according to the RBA (relative byte address) of a data record or an alternate index are not possible!

A DCB for PS files or an ACB for VSAM files can be coded in the calling program. The programs (or more precisely the file control block DCB) must be stored in the low address space.

A FLAMFILE in KSDS format must have the same structure as in the FLAM utility (see manual for FLAM (MVS)), i.e.

Relative key position	0
Key length	one byte longer than the original
Record length	between 80 and 32760 bytes
Control interval size	any, depending on record length

Although FLAM supports a wide range of record length, please take care on performance views.

FLAM compresses a number of data records in a row (depending on MAXR and MAXB parameter). MAXB=64 means 64 KB of Data will be compressed. Assume a compression rate of 90 %, this will lead to a compression result of 6.4 KB compressed data. If you had defined the VSAM-KSDS-FLAMFILE like your original file (i.e. with record size of 400 byte) FLAM has to write 17 VSAM records for the compressed data, each with a new built key. So has FLAM to read 17 records to decompress one original record. So it is better to define a larger RECSIZE for the FLAMFILE, in this example you should use RECSIZE(7168 7168).

Note:

The runtime system of COBOL (PL/I, C) checks all assigned VSAM files.

Frankenstein-Limes-Access-Method ' 2014 by limes datentechnik gmbh

The FLAMFILE must therefore be tuned to the VSAM file type if COBOL programs are used: for accesses to VSAM-ESDS it must also have been created as ESDS, while for accesses to VSAM-KSDS a KSDS-FLAMFILE must exist as well.

With Assembler programs, on the other hand, the FLAMFILE and the original file need not necessarily be of the same type (providing this is also meaningful).

FLAMfi-sub (MVS)

User Manual

Chapter 4:

Principle of operation

4. Principle of operation

All the JCL specifications are checked before a batch job is started by JES. The parameters for the subsystem are transferred and verified by FLAM-sub. If an error is detected, the DD statement is rejected as a JCL error. In addition, a message is output in the JCL list (see messages in examples section). The job is thus only started if there are no errors; otherwise it is aborted in the usual way.

An OPEN command call in the program causes a connection to be set up to the FLAM subsystem. FLAM-sub decides at this stage whether the file must be compressed or decompressed. An open input means decompression (read only), while an open output means compression (write only). An open I/O allows full I/Os to the FLAMFILE via the key of the original records. At the same time, the record and block lengths are taken from the calling program (or from the DCB specifications in the JCL) and - in the case of an open output - transferred to the FLAM file header of the FLAMFILE (unless the HEADER=NO parameter has been specified).

The FLAM parameters specified in the JCL are also activated at the time of the open and the compression/decompression routines set accordingly.

All read and write accesses take place in accordance with the specifications in the program. Asynchronous VSAM calls are executed synchronously by the subsystem, while return codes are not returned (or the error routines activated) until the CHECK call.

It makes no difference whether the file is accessed by the program as BSAM, QSAM or VSAM. FLAM-sub maps all calls to the currently available FLAMFILE.

The record format can always be either variable or fixed.

If an error occurs, the error routine remains active in the calling program and is not overlaid by the subsystem. The error return codes that are returned by the subsystem correspond to those of the DMS (see also 'MVS/DFP Macro Instructions for VSAM Data Sets' for VSAM error codes), in other words the error routines need not be modified; the IEC ... messages of the DMS (MVS Message Library: System Messages) are still traced if necessary. Additional messages are output by FLAM-sub with WTO,Routcde=11 for FLAM errors and subsystem errors (see chapter 8, 'Subsystem messages'). They are documented both in the system log and in the JCL list.

No other messages are generated (the FLAM SHOW=ALL parameter of the utility has no effect).

A CLOSE for the file causes FLAM to close the FLAMFILE and free all the work areas again that were requested with the OPEN. New OPEN calls are now allowed.

FLAMfi-sub (MVS)

User Manual

Chapter 5:

Restrictions

5. Restrictions

It is not normally necessary to modify the JCL as a result of using the FLAM subsystem.

If specifications such as the record and block lengths are omitted from the programs however (utilities such as IEBGENER and SORT function in this way), DCB specifications must be included in the DD statement!

The usual principle whereby the values are automatically taken from a catalog entry in such cases does not work if a subsystem is used. An OPEN routine only recognizes the actual presence of a subsystem file and no longer accesses a catalog entry (even if one exists).

5.1 In general

ADC MODE=ADC is not allowed in update mode.

AES AES-encryption of a VSAM-KSDS file is not allowed

DGD Data Generation Groups must be specified with the absolute name (FILE.G0004V00), not with the relative name (FILE(+1)).

DISP The DISP parameter in the DD statement (e.g. DISP=NEW or DISP=(..., DELETE) cannot be interpreted and thus has no effect.

All files that are assigned directly via FLAM-sub (DSN=filename, SUBSYS=FLAM) must be cataloged.

If new files must be created, a second DD statement can be used:

```
//ddname   DD  SUBSYS=(FLAM,'FLAMDDN=newfile')
//newfile  DD  DSN=...,
               DISP=(NEW,CATLG),
               UNIT=...,SPACE=...
```

Alternatively, the file must have been created in a previous STEP.

All types of access (GET, WRITE, PUT, POINT, ERASE, CHECK) with appropriate RPL modifications for VSAM are possible via the subsystem.

The following restrictions apply (see also manual 'DFSMS/MVS Macro Instructions for Data Sets, SC26-4913'):

5.2 DCB calls (PS files)

CLOSE

A temporary close call in conjunction with BSAM accesses (CLOSE ...,TYPE=T) is not passed on to the subsystem and has no effect.

LOCATE

All attempts to write in LOCATE mode are rejected.

POINT

A POINT in conjunction with BSAM accesses is rejected with the error message IEC141I 013-BC .. (it is not even passed on to the subsystem).

VBS

A file with RECFM=VS or VBS (spanned records) can be written in but not read (IEC141I 013-A8 ...).

5.3 ACB/RPL calls (VSAM files)

CHECK

Asynchronous calls are executed synchronously by FLAM-sub (though, as usual, no response to errors until CHECK).

CNVTAD

CNVTAD returns invalid specifications or zero.

ENDREQ

ENDREQ (terminate a request) has no effect (FLAM does not support locks and only holds the last key).

RPL

Accesses with more than one RPL (RPL list) are not supported (only the first RPL is interpreted).

SHOWCB

SHOWCB ACB=...
The CINV, KEYLEN, LRECL and RKP fields contain the values for the original file, while all other specifications (such as ENDRBA, HALCRBA, etc.) refer to the FLAMFILE.

VERIFY

VERIFY (synchronize end of data) has no effect (VSAM control blocks are not updated).

VSAM-KSDS

If a VSAM-KSDS file must be loaded via FLAM-sub (OPEN OUTPUT in the program), the position and length of the key must be notified to the subsystem (important: the key position is 1 byte higher for FLAM than the RKP for IDCAMS!):

```
//... DD
...,SUBSYS=(FLAM,'OKEYP=value1,OKEYL=value2')
```

The reason for this is as follows:
These specifications are normally transferred by means of IDCAMS when the file is cataloged. With VSAM accesses these specifications are contained in the catalog entry and not in the ACB. The FLAMFILE has been cataloged there with completely different values however, so that the parameters of the original file are no longer available. This information is contained in the compressed file for all subsequent accesses (INPUT or I/O) and does not need to be specified again.

' 2014 by limes datentechnik gmbh

FLAMfi-sub (MVS)

User Manual

Chapter 6:

Parameters for FLAMfi

6. Parameters for FLAMfi

Parameters for FLAM are transferred in the DD statement of the SUBSYS specification:

SUBSYS=(FLAM,**'parameter1=value1,parameter2=value2,...'**)

or

SUBSYS=(FLAM,**'parameter1=value1','parameter2=value2','...'**)

The parameters correspond to those of the FLAM utility (see also manual for FLAM V4.x).

Parentheses () can also be used instead of the equals sign '=', e.g. MO(ADC).

Any apostrophes contained within the parameter string literal must be duplicated
(e.g. SUBSYS=(FLAM,'CRYPTOKEY=C''PASS WORD WITH BLANKS''').

CRYPTOKEY

CRYPTOK

Key to encrypt or decrypt a FLAMFILE

This parameter activates the cryptographic method, entered with parameter CRYPTOMODE.

Possible values:

1 - 64 characters starting with A'...', C'...', X'...' or a string

Using A'..' all characters are translated to ASCII with the internal translation table A/E.

Default:	no key
Valid for:	compression, decompression

Note:
Please take care of the different code tables or national character sets used on the different platforms.

E.g. using the key 'FLAM' both on Windows systems (ASCII) and on MVS (EBCDIC) leads to a cryptokey error. You have to pass X'464C414D20' (this is 'FLAM ' in ASCII) or A'FLAM' on MVS instead.

We recommend to use the hex input for a heterogeneous environment.

CRYPTOMODE

CRYPTOM

Choose the algorithm for encryption.

Possible values:

AES	Advanced Encryption Standard
FLAM	the internal FLAM algorithm
Default:	FLAM
Valid for:	Compression.

Note:
AES was introduced 2003 in FLAM V4.0 and is not compatible to older versions 3.x.

The encryption will be activated by the parameter CRYPTOKEY. The encryption mode is stored in the FLAMFILE, only the key is necessary on decompression and decryption.

Encryption implies MODE=ADC or NDC. Without entering a MODE-parameter ADC is used.

FILEINFO

FI

Transfer file name of original into file header.

Possible values:

YES	Transfer file name
NO	Do not transfer file name
Default:	YES
Valid for:	Compression

FLAMDDN

FLAMD

DD name of a FLAMFILE assigned by JCL.

Compresses to this file or decompresses from this file.

Possible values:

DD name with up to 8 characters

Default:	No name
Valid for:	Compression, decompression

Note:
The file name (DSN=name) with the SUBSYS-parameter is stored in the FLAMFILE, not the name of the file FLAMDD is pointing to.

HEADER

HE

Creates a file header.

Possible values:

YES	Create file header
NO	Do not create file header
Default:	YES
Valid for:	Compression

Note:
Only for special usages, the header should always be created, so that the subsystem can access it in future in order to process the file.

HEADER=YES is set using AES-encryption.

KMEXIT

KME

Use the key management exit module

Possible values:

name	name of the module (max. 8 characters)
Default:	none
Valid for:	en-/decryption

The module is loaded dynamically.

Note: This Parameter overrules CRYPTOKEY
This module is required to benefit from a fully automatic encryption/decryption (-> manual FLAM (MVS) V4).

Because of the specific conditions in a subsystem environment this module has to be written in Assembler! Using a high level language will lead to system errors.

KMPARM

KMP

Parameter used for the KMEXIT.

Possible values:

Any input up to 256 characters in the form
A'...', C'...', X'...', or as a string.

Default:	no parameter
Valid for:	encryption/decryption

MAXBUFFER

MAXB

Maximum size of the compressed-file matrix.

Possible values:

	0 - 7	(Compatibility with FLAM V2.0)
	8 - 2047	Size in kilobytes
	Default:	64 Kbytes
	Valid for:	Compression (CX8,VR8)
MAXRECORDS	Maximum number of records in a compressed-file matrix.	
MAXR	Possible values:	
	1 - 255	for MODE=CX7/CX8/VR8
	1 - 4095	for MODE=ADC
	Default:	255 or 4095 (depends on MODE-parameter)
	Valid for:	Compression

Note for FLAMFILEs in KSDS format:
If accesses are normally direct, a lower value should be specified for MAXR (<= 64).

MAXSIZE	Maximum record length of the compressed file.	
MAXS	Possible values:	
	80 - 32760	
	Default:	512 bytes
	Valid for:	Compression

Note:
The LRECL entry in the catalog takes priority for fixed files (RECFM=F,FB,FBS), while for variable files and VSAM, MAXS is valid as the maximum length, even if a higher value is specified in the catalog for LRECL or RECSIZE.

MODE	Compression method	
MO	Possible values:	
	ADC	8-bit compressed file with highest compression (Advanced Data Compression)
	NDC	no compression of data
	CX7	Transformable, 7-bit compressed file
	CX8	8-bit compressed file (runtime-optimized)

VR8	8-bit compressed file (memory-optimized)
Default:	VR8
Valid for:	Compression

Note:
MODE=ADC is set using encryption.

MSGDDN

MSGD

DD name of a trace file assigned by JCL.

Causes a TRACE to be written in this file (see TRACE parameter).

Possible values:

DD name with up to 8 characters

Default:	FLPRINT
Valid for:	Compression, decompression

ODSORG

ODSO

Specifies the original files data organization of the output.

Possible values:

PS, KSDS

Default:	PS (output for a sequential data set)
Valid for:	Compression

Note:
Use this parameter to load a VSAM-KSDS file although a DCB for a PS file is used (i.e. by a utility).

OKEYLEN

OKEYL

Key length of the original file at the time of the output.

Possible values:

0, 1 – 255

Default:	8 if key sequenced data, 0 else
Valid for:	Compression

OKEYPOS

OKEYP

Key position of the original file at the time of the output.

Possible values:

0, 1 up to record length minus key length

Default:	1 if key sequenced data

Frankenstein-Limes-Access-Method ' 2014 by limes datentechnik gmbh

Valid for:	Compression

SECUREINFO

SEC

Additional information stored in the FLAMFILE increasing the security of data. Changing the FLAMFILE (in any way) leads to an decompression error.

Possible values:

YES	create these information (default on encryption) on compression
NO	do not store any additional data
IGNORE	ignore any security violations on decompression
Default:	NO (without encryption)
	YES (with AES encryption)
Valid for:	compression, decompression

Note:
Concatenation of "secure" FLAMFILEs leads to security violations!

SECUREINFO=YES requires MODE=ADC or NDC and is set automatically using AES encryption.

SPLITMODE

SPLITM

Mode to split a FLAMFILE

Possible values:

NONE	no split
SERIAL	serial split
PARALLEL	parallel split
Default:	NONE
Valid for:	compression

Note:
Split of FLAMFILEs has been introduced in FLAM V4.0 and is not compatible to older versions.

The split information is stored in the FLAMFILE. So it is not necessary to use any parameter on decompression.

File- or DD-names must have numeric characters (ch. 3.1.5, or example in ch. 5.1.3 in manual FLAM (MVS) V4, or example 10.8 in this manual).

SPLITNUMBER

SPLITN

Number of fragments on parallel split

Possible values:

2 - 4	number of simultaneously written files
Default:	4
Valid for:	compression

Note:
The information is stored in the FLAMFILE. So it is not necessary to use any parameter on decompression.

All fragments have to be catalogued and ready to read. It is not possible, to decompress one fragment alone.

Using this parameter requires SPLITMODE=PARALLEL.

SPLITSIZE

SPLITS

Amount in MB of a fragment on serial split

Possible values:

1 - 4095

Default:	100
Valid for:	compression

Note:
The number of created files depends on the amount of compressed data. The information is stored in the FLAMFILE.

Using this parameter needs SPLITMODE=SERIAL.

TRANSLATE

TRA

Data conversion.

Possible values:

E/A	Converts from EBCDIC to ASCII
A/E	Converts from ASCII to EBCDIC
name	Name (up to 8 characters) of a data module containing a 256 byte long translation table.
Default:	No conversion
Valid for:	Compression or decompression (but not simultaneously, i.e. I/O mode)

FLAMfi-sub (MVS)

User Manual

Chapter 7:

Parameters for controlling the subsystem

7. Parameters for controlling the subsystem

These parameters are keyword parameters that must be specified separately from the FLAM parameters. They serve to control the subsystem and strictly speaking have nothing to do with FLAM.

...SUBSYS=(FLAM,**parm**,'flam-parameter')

IG10	Causes return code 10 (not a FLAMFILE) to be ignored when opening for reading. The records are transferred to the calling program without being decompressed. This parameter is meaningful in systems which can contain both compressed and uncompressed files (e.g. for tests).
TRACE	Activates the TRACE function. All function calls to FLAM and the control blocks (ACB, RPL) of the calling program are traced. The trace file must be specified in the JCL. The FLAM parameter 'MSGDDN=ddname' specifies the DD name (default: FLPRINT).

FLAMfi-sub (MVS)

User Manual

Chapter 8:

Subsystem messages

8. Subsystem messages

FLAM-sub only outputs a message on the console in the event of an error (by means of WTO ,ROUTCDE=11). This message is also documented in the JCL log.

The letter after the message number specifies the time at which the error was detected:

I	When the subsystem was initialized
C	When the JCL was analyzed
A	When the file was allocated
O	When the file was opened
D	During the I/O operation
E	When the file was closed

Except when the subsystem is initialized during the IPL of the operating system, errors only affect the specified file for a particular job, in other words the subsystem remains active for the other files.

FLM0500I **INITIALIZATION OF SUBSYSTEM FLAM COMPLETED**

The FLAM subsystem has been initialized correctly and is now available.

Response: None

FLM0501I **SUBSYSTEM FLAM NOT INSTALLED ON THIS SYSTEM**

The subsystem cannot be initialized. It has not been installed on the computer.

Response: Insert the entry FLAM,FLSSIPL in SYS1.PARMLIB(IEFSSN..).

FLM0502I **SUBSYSTEM FLAM ALREADY INITIALIZED**

The program for initializing the subsystem was restarted after the IPL and then rejected. The subsystem that was already active is still active.

Response: FLAM-sub need not be restarted.

FLM0503I **NO MEMORY RECEIVED FOR INITIALIZATION**

The operating system cannot make any memory available for initialization.
The subsystem is not active.

Response: An analysis is necessary (system dump).

FLM0504I **SUBSYSTEM COULD NOT FREE SYSTEM MEMORY**

The memory made available by the operating system cannot be freed. The subsystem has nevertheless been initialized and is now available.

Response: An analysis is necessary (system dump).

FLM0505I **SUBSYSTEM FLAM FUNCTION MODULE NOT FOUND**

A FLAM-sub module required to initialize the subsystem is missing. The module was named by a previous operating system message. The subsystem has not been initialized.

Response: Store the subsystem modules in a library belonging to the LINKLIST concatenation.

FLM0506I **MODULE IEFJSVEC NOT FOUND**

The IEFJSVEC operating system module required to initialize the subsystem is missing.
The subsystem has not been initialized.

Response: Set the missing module in a library that has been linked with the LINKLIB.

FLM0507I **IEFJSVEC: NO MEMORY FOR SSVT**

The subsystem initialization procedure was aborted by the IEFJSVEC operating system module because there is insufficient memory.

Response: An analysis is necessary (system dump).

FLM0508I **LOGIC ERROR IN IEFJSVEC**

The subsystem initialization procedure was aborted by the IEFJSVEC operating system module due to an internal error.

Response: An analysis is necessary (system dump).

FLM0510I	**INTERNAL ERROR. RC = no** An internal error has been detected. The subsystem has not been initialized. Response: Please inform your sales partner.
FLM0515I	**INITIALIZATION OF SUBSYSTEM FLAM FAILED** The subsystem cannot be initialized due to an error. The error situation was documented in a previous message.
FLM0519I	**FLAM SUBSYSTEM IS NOT LICENSED** Your license does not allow using the subsystem.
FLM0520C **FLM0502C**	**PARAMETER ERROR: ...** **SYNTAX ERROR: ...** An invalid parameter was entered in the DD statement of the JCL for FLAM-sub. Response: Correct the parameter (see 'Parameters') and restart the job.
FLM0523C	**NO MEMORY RECEIVED FOR JCL-CONVERTION** A subsystem module was not allocated any memory by the operating system. Response: An analysis is necessary (system dump).
FLM0530A **FLM0530A**	**PARAMETER ERROR: ...** **SYNTAX ERROR: ...** An invalid parameter was entered in the DD statement of the JCL for FLAM-sub. The invalid parameter is displayed. Response: Correct the parameter (see 'Parameters') and restart the job.
FLM0531A	**NO MEMORY RECEIVED FOR ALLOCATION** A subsystem module was not allocated any memory by the operating system. Response: An analysis is necessary (system dump).

Frankenstein-Limes-Access-Method ' 2014 by limes datentechnik gmbh

FLM0540O **OPEN ERROR. DDNAME=*ddname*. RC= *no* (FLAM)/(VSAM)**

An error was detected at the time of the open for the file assigned in the JCL with ddname. FLAM return codes are specified in decimal format and VSAM errors in hexadecimal format (see also manual for FLAM (MVS) V4.x; and DFSMS Macro Instructions).

This message is followed by the IEC141I 013-xx message of the data management system, whereby the DD name used by the program is output. The message IEC161 may be output in connection with VSAM accesses.

The reported FLAM message numbers are equal to the numbers in FLAM utility (ch. 8 in the manual FLAM(MVS)).

Here some examples:

no:

-1	Not enough memory (license rights may have been violated)
10	The file is not a FLAMFILE (input or I/O)
11	FLAMFILE format error
12	Record length error
13	File length error
14	Checksum error
21	Illegal matrix buffer
22	Illegal compression method
23	Illegal CODE in FLAMFILE
24	Illegal BLOCKMODE
25	Illegal record length
31	File not assigned (DD statement missing)
33	Invalid file type
34	Invalid record format
35	Invalid record length
36	Invalid block length
40	Cannot load module or table (TRANS parameter)
60 - 78	FLAM syntax error
120	Name generation in error. Generating a new file name is impossible due to the lack of digit numbers in the name (i.e. only 1 digit number is in the name, cannot generate more than 9 file names).
121	One fragment of the splitted FLAMFILE is missing.
122	Sequence check error of a serially splitted FLAMFILE. The fragments are not in ascending order.
123	Fragments of the splitted FLAMFILE do not belong together.
124	The FLAMFILE was split in more fragments than the actual version can handle.

130	Security violation: FLAMFILE is not in original state during read (i.e concatenation of FLAMFILEs, updates, inserts). Allowed to be ignored (SEC=IGNORE).
131	Missing records in a FLAMFILE. Allowed to be ignored.
132	A member was inserted into a group FLAMFILE. Allowed to be ignored.
133	Sequence error: FLAMFILE records are not in ascending order. Allowed to be ignored.
134	Security error: FLAMFILE starts without security information but then one were found. Cannot be ignored. Perhaps concatenation of FLAMFILEs without and with security information.
531	Some fragments missing during decompression and parallel split.

Response: If RC = -1 and correct license, specify a higher value for REGION in the EXEC statement; if RC = 11 - 14, the file is no longer available in its original state, the data contents have been modified (possibly by a file transfer program) or another type of error has occurred and must be rectified by following the instructions provided (see also manual for FLAM V4.x, Chapter 8).
Some security violations are allowed to be ignored, but take care of the situation.

FLM0541O

ALLOCATION FAILED FOR DSN *filename*
SVC99 ERROR CODE = *no1*. INFO CODE = *no2*.

An error was detected when an attempt was made to allocate the file *filename* dynamically. The codes are specified in hexadecimal format.

Response: Analyse the error and info codes of the SVC99 as described in the manual ('MVS Authorized Assembler Services Guide, GC28-1763').

e.g. **error code = 1708**, info code = 2 or 5 means: **File is not cataloged** (remember: only cataloged files are allowed using FLAM-sub).

FLM0543O

TRANSLATE-PARAMETER INVALID WITH OPEN I/O – IGNORED

The TRANSLATE parameter is ignored in connection with an OPEN I/O.

Frankenstein-Limes-Access-Method ' 2014 by limes datentechnik gmbh

Input and output records can only be translated according to the specified table with an OPEN INPUT or an OPEN OUTPUT.

Processing continues without translating the data.

FLM0544O

FORCED BY PARAMETER TO ACCESS AS UNCOMPRESSED DATA

No FLAMFILE was identified when a subsystem file was opened (message: FLM0440O with RC=10). The set subsystem parameter IG10 forces processing to continue on the basis of an uncompressed (original) file.

Important: If there is no original (uncompressed) file, unpredictable errors may occur.

FLM0552D

I/O ERROR. DDNAME = *ddname*. RC = *no* (FLAM)/(VSAM)
file-name

An error was detected by the FLAM subsystem while reading or writing in the specified file file-name. FLAM return codes are specified in decimal or hexadecimal format and VSAM errors in hexadecimal format.

This message may be followed by the IEC020I 001-.. message of the data management system.

You can find a list of VSAM return codes in the manual entitled 'DFSMS Macro Instructions'.

All FLAM return codes are listed in the FLAM-utility manual (ch. 8), but here some examples:

no:

11	FLAMFILE format error
12	Record length error
13	File length error
14	Checksum error

If one of the above codes is returned, the data in the FLAMFILE has been mutilated (e.g. as a result of a file transfer).

7	Password not declared
15	Record length greater than 32 KB
16	Record length greater than MAXB - 4
25	Illegal record length
29	Password in error

123-134 see FLM0540O

Detecting a security violation the FLAM error code is reported as a hexadecimal value (nnmmmm). nn describes, where the error was detected:

01	Header
02	Segment
03	Membertrailer
04	Filetrailer

Mmmm is the error

0001	MAC1, the data MAC
0002	MAC2, the chaining MAC
0003	MAC3, the MAC over MACs
0010	data missing
0020	data inserted
0040	data updated
0080	record counter compressed data
0100	byte counter compressed data
0200	record counter original data
0400	byte counter original data
0800	chaining on FLAM encryption

The error codes are or'd, e.g. 030180 means: record and byte counter of the compressed data do not fit with the stored information, detected in the member trailer.

It is allowed to ignore these violations (SEC=IGNORE), but take care of the situation.

FLM0553D **NO MEMORY RECEIVED FOR I/O OPERATIONS**

A subsystem module was not allocated any memory by the operating system. Consequently, no other information is available.

Response: Specify a higher value for REGION in the EXEC statement.

FLM0570E **CLOSE ERROR. DDNAME = *ddname*. RC = *no* *file-name***

An error was detected when an attempt was made to close the specified file.

Since compressed-file records may have to be written as well when a file is closed, see also FLM0552D message.

FLAMfi-sub (MVS)

User Manual

Chapter 9:

Installing FLAMfi-sub

9. Installing FLAMfi-sub

The software is shipped with a CD-ROM or may be downloaded from the FLAM homepage via internet.

The FLAM subsystem itself is supplied as a FLAMFILE to upload on to the z/OS system. It is compressed and encrypted. So it is prepared to send it via an unsecure method (e.g. public internet).

Decryption of this FLAMFILE with the FLAM utility expands all data to disk.

README files are stored to guide the installation.

Manuals are provided as PDF-files to read the documents on an appropriate system (e.g. Unix, Windows).

It is allowed to copy and print the documents as often it is needed, but for internal use only.

9.1 LINKLST and authorization

The programs of the subsystem must be stored in an APF-authorised load library. This library must be linked to the system library SYS1.LINKLIB. The subsystem also reloads modules belonging to the FLAM utility at the time of the OPEN. The library of the FLAM utility must therefore be APF-authorised as well.

We recommend keeping all the load modules of FLAM-sub and FLAM-utility in a common library and using only this library.

The following entries are necessary in the SYS1.PARMLIB library for this purpose (see also z/OS Initialisation and Tuning Reference). xx is the serial number of your active system generation.

There is more than one method, to link a LOAD library to LINKLST.

a) To generate a link to the system library, insert the following line in the member LNKLSTxx:

```
SYS1.FLAMLIB,
```

Libraries which are concatenated in this way are normally APF-authorised as default. If not, they can be authorised by inserting the following line:

```
SYS1.FLAMLIB  volume,
```

in the member IEAAPFxx,

where volume is the name of the disk on which the SYS1.FLAMLIB library is stored (if this entry is the last in the member, the comma must be omitted).

b) Insert some lines into the PROGxx member:

```
LINKLST    ADD
           NAME(LINKLSTxx)
           DSNAME(flamlib)
           VOLUME(volume)
```
And
```
APF        ADD
           DSNAME(flamlib)
           VOLUME(volume)
```

Many variants exist for this member, this is only one example.

9.2 Subsystem Start

There a two ways do start the subsystem. Defining a static subsystem that cannot be changed during the IPL session, and defining the more flexible dynamic subsystem via a started task.

a) The FLAM subsystem is initialized during a system IPL.

The following line must be inserted in the member IEFSSNxx of the SYS.PARMLIB library for this purpose:

```
FLAM,FLSSIPL
```

This entry must not precede the line containing 'JES'.

The order of the entries depends on how often the programs are called. The more frequently they are called, the higher up the entries should be inserted in the first third of the member.

The FLAM subsystem becomes available after the next system IPL, or you can use the console command:

```
SETSSI ADD,S=FLAM,I=FLSSIPL
```

to start FLAM subsystem manually without an IPL.

Note: you can't deactivate the subsystem and change any modules during the session.

b) A started task to activate FLAM subsystem is supplied.

The operator starts the subsystem entering

```
S FLAM
```

from the console.

If you insert the line

```
COM='S FLAM'
```

into the member COMMNDxx, the subsystem starts automatically without an operator interaction during the next IPL.

Frankenstein-Limes-Access-Method ' 2014 by limes datentechnik gmbh

You can stop the subsystem with

```
P FLAM
```

And

```
F FLAM,VER
```

shows the version of all subsystem modules.

Stopping the subsystem, changing the FLAM-modules in the library (don’t forget the LLA REFRESH!) and starting the subsystem again enables an easy way to update the subsystem.

FLAMfi-sub (MVS)

User Manual

Chapter 10:

Examples

10. Examples

10.1 Input/output with cataloged files

The USER.EXAMPLE.INPUT file has been created as a compressed file (FLAMFILE), for example with the FLAM utility. It must now be read by another program. The USER.EXAMPLE.OUTPUT file was created by this program previously and must now likewise be compressed. Both files are cataloged.

Two different procedures are possible.

a) When the file was opened, the program specified the record and block lengths in the ACB or the DCB (or in the FD section if COBOL programs are used).

Solution: The SUBSYS specification in the DD statement is sufficient.

```
//stepname EXEC PGM=program
//INPUT    DD   DSN=USER.EXAMPLE.INPUT,
//              DISP=SHR,
//              SUBSYS=FLAM
//OUTPUT   DD   DSN=USER.EXAMPLE.OUTPUT,
//              DISP=OLD,
//              SUBSYS=FLAM
```

Since the "correct" values are specified in the calling program, the data records are transferred to or from the subsystem in accordance with them.

b) The calling program assumes that an "original" catalog entry exists, i.e. neither a record length nor a block length is specified in the program (IEBGENER, SORT and numerous utilities function in this way).

Solution: Specify additional DCB attributes in the DD statement.

```
//stepname EXEC PGM=program
//INPUT    DD   DSN=USER.EXAMPLE.INPUT,
//              DISP=SHR,
//              DCB=(RECFM=FB,LRECL=121,BLKSIZE=3025),
//              SUBSYS=FLAM
//OUTPUT   DD   DSN=USER.EXAMPLE.OUTPUT,
//              DISP=OLD,
//              DCB=(RECFM=VB,LRECL=438,BLKSIZE=4096),
//              SUBSYS=FLAM
```

The program is presented with the specified DCB attributes, irrespective of the actual catalog entry. The subsystem transfers records in accordance with these specifications. FLAM, on the other hand, reads or writes compressed-file records on the disk in cataloged format.

The subsystem activity is confirmed by the system messages in the JCL listing.

```
IEF237I FLAM ALLOCATED TO INPUT
IEF237I FLAM ALLOCATED TO OUTPUT
   .
   .
IEF285I   USER.EXAMPLE.INPUT         SUBSYSTEM
IEF285I   USER.EXAMPLE.OUTPUT        SUBSYSTEM
```

10.2 FLAM parameters and subsystem

Cf. example 1, except that a specific compression method (ADC) must be set for the output file.

Solution:

Enter the necessary parameters in the SUBSYS specification of the DD statement.

```
//stepname EXEC PGM=program
//INPUT    DD   DSN=USER.EXAMPLE.INPUT,
//              DISP=SHR,
//              SUBSYS=FLAM
//OUTPUT   DD   DSN=USER.EXAMPLE.OUTPUT,
//              DISP=OLD,
//              SUBSYS=(FLAM,'MO=ADC')
```

The FLAM utility and the 'D,SHOW(ATT)' parameters or the I OPTION in the FLAM panels allow you to check whether or not the compression procedure is successful.

10.3 Creating a new file

A file called USER.EXAMPLE.NEWDAT must be created. It is not cataloged:

```
//OUTPUT   DD  DSN=USER.EXAMPLE.NEWDAT,
//             DISP=(NEW,CATLG,),
//             UNIT=SYSDA,SPACE=(CYL,(24,24)),
//             DCB=(RECFM=FB,LRECL=121,BLKSIZE=3025)
```

Solution: Specify a second DD statement for FLAM-sub.

The original file description is given a new DD name, to which a FLAM parameter in the SUBSYS specification refers.

```
//OUTPUT   DD  SUBSYS=(FLAM,'FLAMDD=NEWDAT'),
//             DCB=(RECFM=FB,LRECL=121,BLKSIZE=3025)
//NEWDAT   DD  DSN=USER.EXAMPLE.NEWDAT,
//             DISP=(NEW,CATLG,),
//             UNIT=SYSDA,SPACE=(CYL,(6,6)),
//             DCB=(RECFM=FB,LRECL=121,BLKSIZE=3025)
```

The compressed file USER.EXAMPLE.NEWDAT is thus created exactly the same as the original. The advantage of this method is that any subsequent copy or transfer programs which function without the SUBSYS specification will find the same catalog entry as for the original and therefore do not need to be specially tuned.

The new DD statement requires the DCB specification, since it was also mandatory in the original.

Frankenstein-Limes-Access-Method ' 2014 by limes datentechnik gmbh

Otherwise, the compressed file can be set in any way, e.g.:

```
//OUTPUT   DD  SUBSYS=(FLAM,'FLAMDD=NEWDAT'),
//             DCB=(RECFM=FB,LRECL=121,BLKSIZE=3025)
//NEWDAT   DD  DSN=USER.EXAMPLE.NEWDAT,
//             DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(CYL,(6,6)),
//             DCB=(RECFM=FB,LRECL=1024,BLKSIZE=26624)
```

As far as the calling program (and the subsystem) is concerned, the original records are 121 bytes long.

The FLAMFILE, on the other hand, has longer records and blocks for performance reasons (this is the normal, recommended case).

The file descriptions can thus be separated from one another.

10.4 Temporary files

FLAM-sub cannot process any temporary files directly. A second DD statement is necessary for this purpose.

```
//DDNAME      DD   SUBSYS=(FLAM,'FLAMDD=tempfile')
//tempfile    DD   DSN=&&file,DISP=...
```

It is not necessary to specify a file name on account of the FLAMDD specification for DDNAME. A temporary file name is generated automatically by JES. FLAM-sub branches directly to the temporary file.

10.5 Other subsystems

Other subsystems, such as JES for SYSOUT/SYSIN, must be activated by means of a second DD statement.

```
//DDNAME      DD   SUBSYS=(FLAM,'FLAMDD=subfile')
//subfile     DD   SYSOUT=A,DEST=(....),...
```

Depending on the program which is used, it may be necessary to specify DCB parameters in order to process the original file (DDNAME statement, cf. example 1).

10.6 Loading a VSAM-KSDS file

If a KSDS file is loaded, the file is opened for writing (OUTPUT). The keys must be transferred to VSAM in ascending order (this is also checked by FLAM-sub).

It is possible to load the file in I/O mode, in other words it must contain at least one record. At the same time, this leads to a deterioration in performance and an increase in the number of accesses.

The file itself is normally created using IDCAMS and is already cataloged when the job is executed.

The simplest method is to create a compressed KSDS file with the FLAM utility.

10.6.1 Load a real VSAM-KSDS file

A ’real VSAM-KSDS file’ means, an ACB is provided to open the data set.

When the file is loaded via the subsystem, the key description of the original data must be specified in the form of parameters:

```
//OUTPUT   DD  DSN=USER.EXAMPLE.OUTPUT,
//             SUBSYS=(FLAM,'OKEYPOS=21,OKEYLEN=64,MAXS=7168'),
//             DCB=(LRECL=512,BLKSIZE=20480)
```

The original records are up to 512 bytes long in this example, while the control interval takes up 20480 bytes and the key begins at position 21 and is 64 bytes long.

The subsystem should use the full record length of the VSAM file (7168 bytes).

Note: FLAM counts the key position starting at 1 (corresponds to RKP=0).

The VSAM file itself requires the entries (see also ’Preconditions’) for the example:

KEYS(65 0)
RECSZ(7168 7168)

Important: Even if a higher value is specified for RECSZ, the FLAMFILE record length is restricted to 7168 bytes by the MAXS parameter.

Note:

A VSAM file must be empty when it is loaded, in other words either it must not contain any records or the REUSE parameter must have been specified for IDCAMS, to allow the file to be overwritten.

10.6.2 Load a VSAM-KSDS file via an utility

Most utilities (like IEBGENER or SORT) use a DCB to access a subsystem data set. FLAM subsystem then normally declares the data as physical sequential (PS) and has no knowledge about a key description.

Subsystem parameter overwrite this situation and force the subsystem to recognize the output data as key sequenced records:

```
//OUTPUT   DD  DSN=USER.EXAMPLE.OUTPUT,
// SUBSYS=(FLAM,'ODSORG=KSDS,OKEYPOS=21,OKEYLEN=64,MAXS=7168'),
// DCB=(LRECL=512,BLKSIZE=20480)
```

The original records are up to 512 bytes long in this example, while the control interval takes up 20480 bytes and the key begins at position 21 and is 64 bytes long.

The subsystem should use the full record length of the VSAM file (7168 bytes).

Note: FLAM counts the key position starting at 1 (corresponds to RKP=0).

The VSAM file USER.EXAMPLE.OUTPUT itself requires the entries (see also 'Preconditions') for the example:

KEYS(65 0)
RECSZ(7168 7168)

Important: Even if a higher value is specified for RECSZ, the FLAMFILE record length is restricted to 7168 bytes by the MAXS parameter.

Note:
FLAM subsystem checks the ascending order of the key sequence of the output records and returns the VSAM return code, if in error. The utility normally is unaware of this situation and returns a 'WRITE ERROR' without any further information.

10.7 TRACE function

A trace function is activated for both the input file and the output file for test purposes:

```
//INPUT      DD   DSN=filename1,SUBSYS=(FLAM,TRACE)
//OUTPUT     DD   DSN=filename2,
//                SUBSYS=(FLAM,TRACE,'MSGDDN=FLTRACE')
//*
//FLPRINT    DD   SYSOUT=*
//FLTRACE    DD   DSN=filename3,DISP=(NEW,CATLG),
                  SPACE=(TRK,(12,12),RLSE),UNIT=SYSDA
```

Different files must be specified for the trace. FLTRACE is the assignment for the OUTPUT file trace. FLPRINT is taken as the default name, since no parameter has been specified in the DD statement.

The trace for INPUT is as follows (the records have been shortened here):

```
FLAM - TRACE FUNCTION -   COPYRIGHT 2012 BY LIMES DATENTECHNIK GMBH

ACB DURING OPEN: 009E13A0
 (A000004C 00000000 00000000 54000000 00000000 00000000 48900008 0000000
  00000000 00000000 002C0041 009E12A8 03000000 0000521C 00000000 0C30005
  00000000 00000000 00000000)

 FLMOPN, ON ENTRY:
 1:03700C04(037008A0) 2:03700C08(8349A192) 3:0349AA90(00000001) 4:0349AA
 FLMOPN, ON RETURN:
 1:03700C04(0372CD88) 2:03700C08(00000000) 3:0349AA90(00000001) 4:0349AA
 FLMOPD, ON ENTRY:
 1:03700C04(0372CD88) 2:03700C08(00000000) 3:0349AA90(00000001) 4:03700C
 5:03700CC4()
 6:03700C20(00000000) 7:03700C24(00000009) 8:03700C28(00000200) 9:03700C
 10:03700C5C(00000000 00000000
 00000001 00000008 00000000 00000000 00000000 00000000 00000000 000000000
 00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000
 11:03700C34(00001800) 12:03700C38(00000000) 13:03700C3C(00000000)
 FLMOPD, ON RETURN:
 1:03700C04(0372CD88) 2:03700C08(00000000) 3:0349AA90(00000001) 4:03700CC
 5:03700CC4()
 6:03700C20(00000000) 7:03700C24(00000008) 8:03700C28(00000100) 9:03700CC
 10:03700C5C(00000000 00000000
00000000 00000000 00000001 00000000 00000000 00000000 00000000 000000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000
11:03700C34(00001800) 12:03700C38(00000000) 13:03700C3C(00000000)
FLMOPF, ON ENTRY:
1:03700C04(0372CD88) 2:03700C08(00000000) 3:03700C40(00000000) 4:03700C4
7:03700C50(00000001) 8:03700C54(000000FF) 9:03700C5C(00000000 00000000
00000001 00000008 00000001 00000000 00000000 00000000 00000000 000000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000
10:03700C58(00000001) 11:03700CC4(        ) 12:03700CC(        )
FLMOPF, ON RETURN:

1:03700C04(0372CD88) 2:03700C08(00000000) 3:03700C40(000000C8) 4:03700C4
7:03700C50(00000001) 8:03700C54(000000FF) 9:03700C5C(00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000
10:03700C58(00000001) 11:03700CC4(        ) 12:03700CC(        )
FLMGHD, ON ENTRY:
1:03700C04(0372CD88) 2:03700C08(00000000) 3:03700E54(00000036)
4:03700E58(FLAM.DAT.CMP                                    )
5:03700E4C(00000000) 6:03700E48(00000009) 7:03700E3C(00000050) 8:03700E0
9:03700C5C(00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000
10:03700E40(00000C30) 11:03700E38(00000000) 12:03700E28(01010000)
FLMGHD, ON RETURN:
1:03700C04(0372CD88) 2:03700C08(00000000) 3:03700E54(0000001C)
4:03700E58(FLAM.FLAMV27C.CLIST(ORGFLAM))
5:03700E4C(00000000) 6:03700E48(00000009) 7:03700E3C(00000050) 8:03700E0
9:03700C5C(00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000
10:03700E40(00000C30) 11:03700E38(00000000) 12:03700E28(01010000)
I/O-REQUEST: 00000000 GET
ACB: 000052F0
(A000004C 000147C0 8003E170 54000000 00000000 00000000 48900008 0000000
 00000000 00000000 002C0041 009E12A8 12000000 0000521C 00000000 0C30005
00000000 00000000 00000000)
RPL: 00005390
(0000004C 00000000 00000000 00000000 00000000 00000000 000052F0 0000000
 00045E5C 00000000 20000000 00000000 00000050 0000050 00000000 0000000
 00000000 00000000 000053EC)
FLMGET, ON ENTRY:
1:03700C04(0372CD88) 2:03700C08(00000000) 3:000053C0(00000050)
4:00045E5C(
5:000053C4(00000050)
FLMGET, ON RETURN:
1:03700C04(0372CD88) 2:03700C08(00000000) 3:000053C0(00000050)
4:00045E5C(PROC 0
5:000053C4(00000050)
I/O-REQUEST: 00000000 GET
ACB: 000052F0
(A000004C 000147C0 8003E170 54000000 00000000 00000000 48900008 000000
 00000000 00000000 002C0041 009E12A8 12000000 0000521C 00000000 0C3000
 00000000 00000000 00000000)
RPL: 00005390
(0000004C 00000000 00000000 00000000 00000000 00000000 000052F0 000000
 00045EAC 00000000 20000000 00000000 00000050 0000050 00000000 000000
```

```
  00000000 00000000 000053EC)
 FLMGET, ON ENTRY:
 1:03700C04(0372CD88) 2:03700C08(00000000) 3:000053C0(00000050)
 4:00045EAC(
 5:000053C4(00000050)
 FLMGET, ON RETURN:

1:03700C04(0372CD88) 2:03700C08(00000000) 3:000053C0(00000050)
 4:00045EAC(CONTROL NOLIST NOSYMLIST NOCONLIST NOFLUSH NOMSG NOCAPS
 5:000053C4(00000050)
          .
          .
 FLMCLS, ON ENTRY:
 1:03700C04(0372CD88) 2:03700C08(00000002)
 FLMCLS, ON RETURN:
 1:03700C04(FFFFFFFF) 2:03700C08(00000000)
```

The names of the FLAM functions (FLMOPN, FLMGET, etc.) and their parameters are equal to those listed in the manual for FLAM V4 (Chapter 3, Interfaces). The parameters are numbered in the same way as in the manual. The next specification is the parameter address. The parameter contents are enclosed in parentheses ’()’.

The control blocks ACB and RPL are specified by the subsystem interface, even if a PS file is accessed.

Please refer to the IBM manuals for further details of these control blocks, e.g. ’Macro Instructions for VSAM Data Sets’, ’Data Areas’, or to the ’SYS1.AMODGEN(IFGACB)’ and ’SYS1.AMODGEN(IFGRPL)’ macros.

10.8 FLAMFILE split

10.8.1 Serial split

To split a file in several fragments, depending on the amount of data, the serial split is used.

```
//output   DD  DSN=FLAM.SUBDAT.A01.ADC,
//             SUBSYS=(FLAM,'MO=ADC,SPLITM=SER,SPLITS=200')
```

The file FLAM.SUBDAT.A01.ADC is cataloged, the filename has two numeric characters (at least 1 numeric character is necessary). Compression method ADC is used for higher compression. One file must not exceed 200 MB data.

When more than 200 MB data are to be written, file FLAM.SUBDAT.A01.ADC is closed and file FLAM.SUBDAT.A02.ADC is searched and allocated, or if not found in the catalog, created. With two numeric characters it is possible to generate up to 99 files.

If your organisation requires special filenames or a higher flexibility for data set allocation is needed, assigning with DD-names is recommended. Only the first DD-name is passed to the subsystem.

```
//output  DD SUBSYS=(FLAM,'SPLITM=SER,SPLITS=200,FLAMDD=CMP01')
//*
//CMP01      DD  DSN=filename1,DISP=OLD
//CMP02      DD  DSN=filename2,DISP=OLD
```

```
//CMP03       DD  DSN=filename3,DISP=(NEW,CATLG),
//                UNIT=TAPE,...
```

All files must have the same record length, but the data set organization or format may differ.

When more than 200 MB are to be written, the file with DD-name CMP01 is closed and CMP02 is opened.

To read a splitted FLAMFILE it is not necessary to pass any parameter to the subsystem. The filename of the first fragment is named in the DD-statement:

```
//input     DD  DSN=FLAM.SUBDAT.A01.ADC,
//              SUBSYS=FLAM
```

all following files will be allocated dynamically by the subsystem.

Assigning files via DD-statement is possible as well. The DD-name must have at least one numeric character, the first name is passed to the subsystem:

```
//input       DD  SUBSYS=(FLAM,'FLAMDD=CMP01')
//*
//CMP01    DD  DSN=filename1,DISP=SHR
//CMP02    DD  DSN=filename2,DISP=SHR
//CMP03    DD  DSN=filename3,DISP=SHR
```

Note:

The files must be assigned in the same order than they were written. The subsystem checks the sequence of the files. In case of errors, message FLM0540O and RC=122 is reported and the data management system detects an open error.

10.8.2 Parallel split

Parallel split means: up to 4 files are written or read "simultaneously".

All files must be cataloged and assigned within JCL! Therefore it is necessary to assign the subsystem files via the DD-statement. The DD-name must have at least one numeric character, the first name is passed to the subsystem.

```
//output   DD  DSN=any,
//      SUBSYS=(FLAM,'MO=ADC,SPLITM=PAR,SPLITN=3,FLAMDD=CMP01')
//*
//CMP01     DD  DSN=filename1,DISP=OLD
//CMP02     DD  DSN=filename2,DISP=OLD
//CMP03     DD  DSN=filename3,DISP=OLD
```

All files must have the same record length, but the data set organization or format may differ.

JCL required for reading a split FLAMFILE in parallel:

```
//input    DD  DSN=any,
//             SUBSYS=(FLAM,'FLAMDD=CMP01')
//*
//CMP01    DD  DSN=filename1,DISP=SHR
//CMP02    DD  DSN=filename2,DISP=SHR
```

```
//CMP03   DD  DSN=filename3,DISP=SHR
```

The files may be in different order than they were written, the subsystem uses the original order automatically.

Note:

If one file is missing, message FLM0540O with RC=531 is reported by the subsystem and the data management system detects an OPEN-error.

Using DISP=NEW in the assigned DD-statements will lead to a system error 50D, so it is really necessary to catalog all files.

10.9 Encryption

10.9.1 Encryption via CRYPTOKEY parameter

To encrypt/decrypt the file a key is passed (CRYPTOKEY=...) to the subsystem. The cryptographic method is choosed by parameter CRYPTOMODE. Without this parameter, method FLAM is used (compatible to FLAM-sub V3)

```
//DDNAME   DD  DSN=filename,
//             SUBSYS=(FLAM,'CRYPTOM=AES,CRYPTOK=OTTO')
```

Using parameter PASSWORD instead of CRYPTOKEY is allowed and compatible to FLAM-sub V3.

Please remember: any apostrophes contained within the parameter string literal must be duplicated:

```
//DDNAME   DD  DSN=filename,
//             SUBSYS=(FLAM,'CRYPTOM=AES,CRYPTOK=X''AE01EA''')
```

10.9.2 Encryption via KME-module

A program *name* that supports the KME-interface (-> manual FLAM (MVS) V4.x, ch. 3.5.5), will be used as followed

```
//DDNAME   DD  DSN=filename,
//             SUBSYS=(FLAM,'CRYPTOM=AES',
//             'KME=name,KMP=C''parameter for KME''')
```

This method benefits from a fully automatic encryption/decryption without any manual user action. The key used for encryption is never seen in any protocol.

10.9.3 Encryption using FKMEFILE

Program FKMEFILE is an example for using the KME interface. It reads a record containing a key from a sequential file and returns it to the subsystem. A parameter *ddname* is needed for the file to read.

The key must be the first record of the file. The syntax is as of parameter CRYPTOKEY: the key may be a string, a C'string with blanks', a A'key in ascii', or a X'hexvalue'. Trailing blanks are ignored.

```
 //DDNAME   DD  DSN=filename,
//              SUBSYS=(FLAM,'CRYPTOM=AES',
//              'KME=FKMEFILE,KMP=C''ddname1''')
//* THIS is the key-file
//ddname1   DD  DSN=keyfilename,DISP=SHR
```

Just to test it on the fly

//*ddname* DD *
C'this is my key'
/*