FLAM ®

FRANKENSTEIN-LIMES-ACCESS-METHOD

for WINDOWS

User Manual

— Issue July 2014 Version 4.3 —

Phone no. +49 (0)6172 5919-0 ■ Fax +49 (0)6172 5919-39
http://www.flam.de

NEUES IN FLAM® VERSION 4 FÜR WINDOWS

01. NEW FEATURES IN FLAM® VERSION 4 FOR WINDOWS

- **AES Encryption by recent standards**
- **“No-Data-Compression” for processing already compressed data**
- **High protection against manipulation by additional checksums and control records**
- **The command line version consists of only four files: flam4.exe, flamdef.dat, flamliz.dat and flammsg.dat**
- **Log file without message module**
- **In a non-German environment, messages are automatically issued in English**
- **Simplified product distribution for corporate licenses**
- **No entries in the registry necessary**

New features in FLAM® Version 4.0.0.1

- **FLAMCMD: a program to launch FLAM by command line**
- **Registration of FLAM environment variables**
- **Setting the priority of FLAM**

New features in FLAM® Version 4.0.0.2

- **Improved IO: 70-80% faster especially via network**
- **Full support of STDIN and STDOUT also with PIPES**
- **APPEND and ADD also more than 4GB**
- **Extended statistics about FLAMCMD**

New feature in FLAM® Version 4.0.0.3

- **Option “suppress” extended by “trimchar”**

New features in FLAM® Version 4.0.0.4

- **Parameter file processing corrected**
- **Reading of “var”records with record length 0 corrected**

New features in FLAM® Version 4.1.0.2

- **FLAM subprogram interface FLAMUPEX.DLL**
- **FKMS: connection to FLAM KEY MANAGEMENT SYSTEM**
- **FCTE FLAM code table editor**
- **New parameters kmdll, kmparam, kmexit, label, pafile**
- **Counter in log file of large files corrected**
- **File namelength up to 255 bytes.**

New feature in FLAM® Version 4.1.0.5

- **Output specifikation flamf=[txt=adc] flamin=*.txt generates a FLAMFILE xxx.adc for each xxx.txt**

03. AES ENCRYPTION

The *National Institute of Standards* (NIST) has defined the **Advanced Encryption Standard** (AES) concerning the encryption of data. This method was described with the *Federal Information Processing Standard* (FIPS 197) in November 2001 and it was released on May 26, 2002.

FLAM uses this algorithm to encrypt the compressed data. As a key, up to 64 characters can be specified. Internally a key length of 128 bit is used (AES 128). Control fields, which are generated with AES as well, are added to protect the data.

This encryption method is activated by the parameters **cryptmode=aes** and **cryptkey=key**. It is implemented for the compression methods ADC and NDC (**mode=adc** resp. **mode=ndc)** only.

04. SECURED FLAMFILES

In ADC/NDC mode **secureinfo=yes** has the effect that additional information is stored within the compressed file in order to guarantee the completeness and intactness of the compressed file without the need to decompress the **FLAMFILE®**. It is already recognized during formal checking if a thus secured **FLAMFILE®** has been altered (e.g. by updating, supplementing, deleting “members” in a **FLAMFILE®** archive).

During encryption with AES this additional information is always written.
In ADC and NDC mode this information is irrelevant for **FLAM®** Version 3 and it is ignored.

With **secureinfo=no** this data can be ignored during decompression. For example this is useful for concatenated secured **FLAMFILE**s or when data is to be decompressed despite a security incident.
If just a single “member” is extracted from a secured **FLAMFILE®**, only the security information of this specific member is interpreted.

05. REGISTRATION OF FLAM ENVIRONMENT VARIABLES

To use the FLAM applications flam4 and flamcmd without a path statement, entries for these programs must be added to the WINDOWS registry. This is done with the parameter /**reg**.
The environment variable “PATH” is available in the console window only after the **reboot of the system**.

The call **flam4** /**reg** generates the entries for **flam4.exe** in the registry.
The call **flamcmd** /**reg** generates the entries for **flamcmd.exe** in the registry.

Registry entries for starting from **“START – Run...”:**

HKEY_LOCAL_MACHINE
SOFTWARE
Microsoft\Windows\CurrentVersion\App Paths\flam4.exe
Microsoft\Windows\CurrentVersion\App Paths\flamcmd.exe

Registry entry for starting in the **console window** (CMD.EXE: DOS_Box):

HKEY_LOCAL_MACHINE
SYSTEM\CurrentControlSet\Control\Session Manager\Environment\Path

The environment variable %PATH% is supplemented by the FLAM path and is available after the reboot of the system.

06. FLAM LAUNCHER FLAMCMD

flamcmd.exe launches flam(4).exe with low priority, leaving the window available. In this way FLAM can be launched comfortably from the Start box (START – Run...), without the result disappearing. But starting from a console window (DOS box) has also been improved. FLAMCMD also works with flam.exe Version 3.

The call
flamcmd comp flamf=*FlamfileName* flamin=*inputfile* mode=adc inrecf=text

is equivalent to
flam comp flamf=*FlamfileName* flamin=*inputfile* mode=adc inrecf=text

with the difference that FLAM is executed in low priority and that the console window is waiting for the Return key.

FLAMCMD can be launched with the following commands:

flamcmd /newcon <flamparameter> :	FLAM is launched in a new window.
flamcmd /nowait <flamparameter>:	FLAM is launched in the background. This means that the starting program (console) is not blocked. FLAM is left to itself as a detached process. Very useful in combination with /newcon.
flamcmd /nocon <flamparameter>:	FLAM is launched without a console window. Useful for pure batch processing.
flamcmd /nopause <flamparameter>:	After finishing FLAM closes the window. Therefore FLAMCMD behaves like FLAM itself. However this is done with low priority so that parallel processing is less affected.
/newcon /nowait	FLAM starts in a new console in a detached manner.
/nocon /nowait /nopause	The unattended FLAM starts without a console window.
/stat	provides advanced process statistics

Note for advanced users and system administrators:

With flamcmd (especially with /newcon /nocon or /nowait) the message output cannot be redirected with 2>logfile or 2>>logfile any longer, it must be passed with the parameter *logfile* instead. If log=*logfile* is specified as the first parameter, FLAM4 uses this file immediately as

soon as the parameter has been interpreted. The redirection 2>*logfile* has the effect that the log file is deleted, so log=*logfile* writes into an empty file.

07. WORKING WITH STDIN, STDOUT AND PIPES

Since version 4.0.0.2 the input and output can also be done via **STDIN** and **STDOUT**:

For **input** '<' and for **output** '>' is added to the file name as prefix.

Depending on the function, the FLAMFILE is either input (decompress) or output (compress).

These file names **must not contain wildcards** (*,?). A selection with wildcards must still be carried out with the key words flamin, flamout or flamfile.

FLAM4 comp <*originalfile* >*FLAMFILE*

Or with PIPE:

TYPE *originalfile* | FLAM4 comp >*FLAMFILE*

Former syntax:

FLAM4 comp flamf=*FLAMFILE* flamin=*originalfile*

Looking at the decompressed FLAMFILE content on the console:

FLAM4 deco flamf=*FLAMFILE* | more
or
FLAM4 deco <*FLAMFILE* | more

Other parameters can be specified as before:

FLAM4 deco translate=e/a outrecf=text <*FLAMFILE* | more

Please note: The FLAM log is output to STDERR and can be written into a file with log=*FILE NAME* or 2>*FILE NAME*.

08. THE COMMAND LINE PARAMETERS

FLAM4.EXE is a **console application**, this means it is started

- via START – Run
- in a DOS box
- from a batch application

Simple example for compression:

flam4 comp flamfile=<name of new FLAMFILE> flamin=<name of file that is to be compressed>

Simple example for decompression:

flam4 deco flamfile=<name of FLAMFILE> flamout=<name of file that is to be created>

All parameters are separated by a space (blank) on the command line. If a file name or another parameter value contains a blank, the file name or the parameter value must be enclosed by "double quotes".

Abbreviation of the key words:

The key words may be abbreviated as long as the abbreviation is unambiguous. E.g. flamf for flamfile, outrecd for outrecdelim etc. Capital letters may be used as well. In the description the possible short forms are indicated by bold type, e.g. **inrecf**ormat may be abbreviated with **inrecf**.

Priority of the command line:

Parameters may also be passed in a parameter file. Specifying a parameter file (parfile=<name>) may be supplemented by command line parameters. Concerning this matter the parameters in the command line have a higher priority than the parameters in the file.

For the command line the following parameters are available:

add	further compressed data is added to an existing FLAMFILE®
append	the decompressed data is appended to an existing file
attributes=	controls the writing of the compression information into the FLAMFILE
codetable=	if translation is activated, the specified code table is used
compress	initiates compression
cryptkey=	FLAM 4 password for cryptmode=aes
cryptmode=	FLAM 4 states the encryption method
cut	controls behaviour for too long records
decompress	initiates decompression
flamcode=	instructs FLAM to use ASCII or EBCDIC control characters. The data is not changed.
flamfile=	names the FLAMFILE® regarding compression and decompression. Regarding decompression wildcards are allowed.
flamin=	refers to the file that is to be compressed (original file). Wildcards are allowed.
flamout=	refers to the file that is to be decompressed. Special characters are allowed.
inrecdelim=	The original file is to be read with a record delimiter.
inrecformat=	The original file is to be read with a certain record format.
inrecsize=	The original file has a certain record length.
label=	Insert comment in log file and FLAMFILE®. (V 4.1)
list	All default values are listed.
logfile=	refers to the file to which messages are to be redirected
maxbuffer=	determines the size of the buffer internally used by FLAM
maxrecords=	determines the number of original records that are to be collected in a buffer
mode=	determines the processing mode (ADC, AES, CX7,CX8, NDC,VR8).
nocut	controls the behaviour regarding too long records
nopath	The file names are inserted in the FLAMFILE without path specification.
outpath=	refers to the path (directory), where the decompressed files are stored
nosuppress	no suppression of spaces (blanks) at the end of record (writing)
outrecdelim=	refers to the record delimiter of the decompressed file
outrecformat=	refers to the record format of the decompressed file
outrecsize=	refers to the record length of the decompressed file
parfile=	refers to the parameter file
paascii=	password ASCII encoded
paebcdic=	password EBCDIC encoded
pafile=	parameter file for encryption (V 4.1)
password=	password
paxword=	hexadecimal password
priority=	sets a lower priority for FLAM than normal
recdelim=	refers to the record delimiter for a CX7-FLAMFILE
recformat=	refers to the record format of the compressed file
recsize=	refers to the record length of the FLAMFILE
secureinfo=	turn additional security structures on and off
show=	controls the output of the messages
suppress	suppresses spaces (blanks) at the end of record (writing)
translate=	turns translation of original data on
trimchar=	suppresses (blank) character at the end of record

add

Syntax **add**

Values none

Description Compression only**:** Further compressed data (=“member”) is added to an existing **FLAMFILE®**. If the file does not exist, it will be created.

This feature is not allowed for secure FLAMFILES.

append

Syntax **app**end

Value none

Description Decompression only: The decompressed data is appended to an existing file**.** If the file does not exist, it will be created.

attributes

Syntax **at**tributes=a*ttribute option*

Values *attribute option*

none	no compression information
common	common compression information
all	general compression information and system specific information about the original file

Description

This parameter has the effect that compression information is written into the FLAMFILE.

The insertion of compression information into the FLAMFILE allows a subsequent extraction of this information, without the necessity to totally decompress the FLAMFILE. In doing so, the file organization, the record format and the record length can be logged during compression. This is also true for the attribute under which operating system the compression has taken place. (attributes=common).

As result the decompressed file can be generated with the same characteristics during decompression if necessary.

The file specification of the original file may optionally be entered in addition (attributes=all). This option makes it possible to specify the output specification [ ] or [*] for decompression, with the result that FLAM will use the entered file specification and the related characteristics automatically.

With attributes=none no information about the original file is written into the FLAMFILE at all.

codetable

Syntax **cod**etable=<name of code table>

Description If translation has been activated, the specified code table is used.

The first 256 characters refer to the function translate=e/a .

The characters 257 – 512 refer to the function translate=a/e.

If the file has only 256 characters, these characters will be used for e/a and a/e as well.

In this way a table can serve as conversion tool for lower-case to capital letters or for the adjustment of umlauts, e.g. OEM<->ANSI.

See also: **translate**

compress

Syntax **com**press

Values none

Description With this parameter **FLAM®** compresses the original file(s).

cryptkey

Syntax **cryptk**ey=*SECRET*

Values *SECRET*
String of 1 to 64 characters

Description only **FLAM® Version 4**

The **FLAMFILE®** is encrypted with AES in combination with mode=adc or mode=ndc.
Alternative specification: cryptmode=*aes* password=*secret*

See also: **cryptmode**, **password**, **paeword**, **paxword**

Important note Regardless of the encryption algorithm, a key should consist of at least 6 – 8 characters and it should at least contain 1 character from the following sets:

Lower case letters
Capital letters
Numbers 0 - 9
Punctuation marks

cryptmode

Syntax **cr**yptmode=c*rypt option*

Values *crypt option*

none	no encryption
flam	FLAM encryption (**FLAM®** Version 3)
aes	encryption with AES (**FLAM®** Version 4)

Description This parameter has the effect that the FLAMFILE is encrypted.
Only for compression with mode=adc or mode=ndc

Password specification required

See also: **cryptkey, password**

cut

Syntax **cu**t

Values none

Description Decompression only: Records with a record length longer than the maximum record length may be shortened.

See also: **nocut**

decompress

Syntax **dec**ompress

Description This parameter has the effect that **FLAM®** decompresses the compressed file(s).

flamcode

Syntax **flamc**ode=*character set*

Values

ascii	use ASCII code for the FLAMFILE
ebcdic	use EBCDIC code for the FLAMFILE

Description In the context of compression this parameter specifies which character set is to be used to display the character coded information within the **FLAMFILE®**.

In CX7 mode all FLAM control characters of the compressed data are also encoded using this character set.

For the other modes this only concerns the information in the FLAM file header such as the name of the original file and some of the control characters since the compressed data is written in a binary manner.
Generally the characters of the original files remain unchanged though. Their translation can be achieved with the parameter **translate**.

flamfile

Syntax **flamf**ile=*file specification*

Values *file specification*

Description During compression (parameter **compress**) the compressed data is written into the specified file.

During decompression (parameter **decompress**) the compressed data is read from the specified file.

flamin

Syntax **flami**n=*"name of original files"*

Values =*"name of original files"*

Description With this parameter the files that are to be compressed are specified. The usual wildcards (*?) are allowed.

flamout

Syntax **flamo**ut=*output specification*

Values *output specification*

With the output specification one or more files are specified into which the decompressed data is to be written.

The following special characters are possible, especially in combination with **outpath**:

flamout=[]	the original file name stored in the FLAMFILE is used
flamout=[*]	just as [], but including path, if **outpath** not specified
flamout=[#]	File names are generated to FILE#nnn, e.g. File#002 = 2nd file in FLAMFILE
flamout=[#3]	Only the 3rd file from a FLAMFILE archive is decompressed, using the original name.
flamout=[#3=<name>]	Only the 3rd file is decompressed (with <name>).

Description With this parameter the names of the decompressed files are specified. During decompression the decompressed data of the FLAMFILE is written into the specified file**.**

inrecdelim

Syntax **inrecd**elim=*end-of-record characters (delimiter)*

Values Hexadecimal characters between 01 and FF, length: 1 or 2 bytes (2 or 4 hex characters).

Description With this parameter the end-of-record character of an original file is specified with **record format text**. When using **record format text** without specification of the end-of-record character, 0d0a (CRLF) is interpreted as the end of record.

If only “0a” is intended as the end of record, then inrecdelim=0a has to be specified in order to interpret a “0d” before a “0a” not as part of the end-of-record characters but as part of the data. If a file is meant to contain the end-of-record characters 0d0a only, inrecdelim=0d0a has to be specified in order to recognize isolated characters “0a” as part of the data.

Not only 0a and 0d0a are allowed as end-of-record characters, but also any other character combinations.

inrecformat

Syntax **inrecf**ormat=*format option*

Values *format option*

fix	fixed record length **inrecsize**
dtext	variable record length with end-of-record characters**,** last record without delimiter
text	variable record length with end-of-record characters
variable	variable record length with a binary record length of 2 bytes
var_2b	variable record length with a binary record length of 2 bytes
var_4b	variable record length with a binary record length of 4 bytes (host: length of 2 bytes, padding is ignored).
var_ascii	variable record length with a length field of 4 bytes in ASCII code
var_ebcdic	variable record length with a length field of 4 bytes in EBCDIC code
undefined	records without structure; records with the length **inrecsize** are read, last record may be shorter.

Description This parameter specifies the record format of the original file**.** With the specified record format the data are read as logical records from the file.

inrecsize

Syntax **inrecs**ize=*nnn*

Values Integer decimal number between 1 and 32760.
The default value is 512.

Description This parameter specifies the record length of original files with record format “fix” or “undefined” resp. the maximum record length.

kmdll

Syntax **kmd**ll=*nameOfKmDll*

Default **kmd**ll=*flamkme.dll*

Description With the parameter **kmdll** the name of a dynamically loadable library is specified. This library establishes the communication to a key management system, e.g. FKMS FLAM KEYMANAGEMENT SYSTEM.

The extension .dll may be omitted.

If the DLL is not located in the search path of the operating system, the file name must be specified in a fully qualified manner.

Availability **FLAM® Version 4.1**

kmexit

Syntax **kme**xit=*nameOfKmProcedure*

Default **kmd**ll=*flamkme*

Description With the parameter **kmexit** the name of the procedure that is to be used from the library is specified.

kmparam

Syntax **kmp**aram=*parameter string*

Default *none*

Description The string that is specified with kmparam is passed to the KM procedure.

See description of FLAM KME

label

Syntax **la**bel=comment

Description With the parameter **label** a comment can be made. This comment appears in the log and is written to the USER header of the FLAMFILE®. This header is logged during decompression.

FKMS The FKMS uses the USER header itself. The comment only appears in the log. During decompression the comment is passed to the FKMS in case of a missing USER header.

Availability **Since FLAM® Version 4.1.0.1.**

list

Syntax list

Values none

Description It is possible to display the installation specific standard settings on the screen with the parameter “list”.

flam4 list 2>*FILE*

Output of the default values to a file.

logfile

Syntax **lo**gfile

Values Name of the file to which the FLAM messages are to be written.

Description Concerning interactive mode FLAM messages are usually displayed on the user screen.
With msgfile= messagefile the output to a file can be initiated. The file can serve as a permanent copy in order to document compression. This is for example important for batch processing.

The logging in the message file does not begin until the syntax and the compatibility of the FLAM settings that had been used during the call were checked. Error messages due to syntax errors, missing files or incorrect settings are not logged. In these cases FLAM stops processing without creating a message file.

FLAM® Version 4: Generally the log file is opened with "append".
It is possible to have the log file created on the command line by the operating system:

flam4 *param1 param2 param3* 2>LOGFILE

Creates the file with the name "LOGFILE". Existing content is deleted.

flam4 *param1 param2 param3* 2>>LOGFILE

Uses the file named "LOGFILE". Existing content remains intact.

The redirection of STDERR instead of the specification msgfile=<name> has the advantage that all messages are written to the redirected file immediately. Otherwise the messages are not written to the specified file until the parameter msgfile is interpreted.

maxbuffer

Syntax **maxb**uffer=*nnn*

Values *nnn*

Integral decimal number between 1 and 2,621,440.

Values n with 0≤n≤2,560 are interpreted as number of KBytes (1 KByte = 1,024 bytes), greater values as number of bytes.

FLAM® chooses one of the buffer sizes from a table (in KBytes):
If the specified value is not in the table, then – if existent – the next higher buffer size is chosen from the table, apart from that always the maximum of 2,560 KB = 2,621,440 bytes.

Description During compression **FLAM®** reserves the matrix buffer for the temporary storage of the records from the original file. Please note that a matrix buffer of the same size has to be available during decompression, too. This is also the case when the decompression is done on other systems.

CX7, CX8 and VR8 only.
Regarding ADC, NDC and AES this parameter is not used. The selection is always a buffer of 64KB .

maxrecords

Syntax **maxr**ecords=*nnn*

Values Integral number between 1 and 255 (CX7, CX8, VR8)
Integral number between 1 and 4095 (ADC, NDC, AES).

Description FLAM buffers records up to the specified number within the matrix buffer and then compresses them. On a case-by-case basis this specified number may not be reached if the size of the matrix buffer is not big enough for the full number of records. Usually the compression efficiency inreases with the number. With maxrecords=1 a sequential, record-by-record compression is achieved.

mode

Syntax **mod**e=*FLAM mode*

Values *FLAM mode*

cx7 With mode=cx7 the input file is compressed in CX7 mode and a character coded **FLAMFILE®** is generated. However, in this mode only those files should be compressed that solely contain printable characters. This mode is a little less efficient, but the generated **FLAMFILE®** can be converted to other character codes without information loss, for example from ASCII to EBCDIC using remote data transmission.

cx8 With mode=cx8 the input file is compressed in cx8 mode and a binary **FLAMFILE®** is generated. This mode is more efficient and includes a stricter integrity check during decompression. But the generated **FLAMFILE®** cannot be transmitted via 7 bit lines or via line sections where automatic code conversion takes place.

vr8 With mode=vr8 the input file is compressed in vr8 mode and a binary **FLAMFILE®** is generated. In contrast to CX8 and CX7 mode the VR8-compressed data is a continuous bit stream and has no byte structure.

adc **FLAM®** Version 3: With mode=adc the input file is compressed in ADC mode and a binary **FLAMFILE®** is generated. In general this is the most efficient method.

ndc **FLAM®** Version 3 compatible: With mode=ndc the input file is **not** compressed. An ADC-compatible binary **FLAMFILE®** is generated. This mode is used for the packaging and the encryption of already compressed data.

aes **FLAM®** Version 4: With mode=aes the input file is compressed with the AES method. A **password** must be specified.

Description The input file is compressed with the specified FLAM mode**.**

The key word **mode=** may be omitted.

nocut

Syntax **noc**ut

Values none

Description Decompression only**:** Records with a record length larger than the maximum record length are not cut.

See also: **cut**

nopath

Syntax **nop**ath

Values none

Description Compression only: This specification has the effect that the file name of the original file is saved without path to the FLAMFILE. This is particularly helpful for the communication with external systems.

nosuppress

Syntax **nos**uppress

Values none

Description Decompression only: Blanks at the end of the record are not suppressed.

See also: **suppress**

notruncate

Syntax **notr**uncate

Values none

Description Synonym for **nocut**.

See also: **truncate**

outpath

Syntax **outpat**h=*path*

Values *path*

Description Decompression only: The specified path becomes the target directory.
The directory structure is generated if necessary.

outrecdelim

Syntax **outrecd**elim=*end-of-record characters (delimiter)*

Values Hexadecimal characters between 01 and FF. Length: 1 or 2 bytes (2 or 4 hex characters).

Description With this parameter the end-of-record character of the decompressed file is specified with **record format "text"**.

When using **record format "text"** without specifying the end-of-record character, 0D0A (CRLF) is written as the end-of-record character. With the specification outrecdelim=0a it is possible to generate the decompressed file in UNIX format.

Beyond that every other combination of characters is allowed as end-of-record characters.

outrecformat

Syntax **outrecf**ormat=*format option*

Values *format option*

fix	fixed record length
dtext	variable record length with end-of-record characters, last record without delimiter
text	variable record length with end-of-record characters
undefined	records without structure. The records are written with length **outrecsize**.
variable	variable record length with a binary record length of 2 bytes
var_2b	variable record length with a binary record length of 2 bytes
var_4b	variable record length with a binary record length of 4 bytes (host: length of 2 bytes, padding with **padchar**)
var_ascii	variable record length with a length field of 4 bytes in ASCII code
var_ebcdic	variable record length with a length fiel of 4 bytes in EBCDIC code

Description This parameter specifies the record format of the decompressed file**.** With the specified record format the decompressed data are written as logical records into the file.

outrecsize

Syntax **outrecs**ize=*nnn*

Values *nnn*

Integer decimal number between 1 and 32760
Default value: 512

Description This parameter specifies the record length of the decompressed files with record format “fix” or “undefined” resp. the maximum record length.

padchar

Syntax **pad**char=*pad character*

Values Hexadecimal characters between 00 and FF

Description With “padchar=xx” the character is specified with which – if necessary – the original record is padded. For example, if the output record has a fixed length, but the original record is shorter.

Please note: “padchar” is also used for padding the variable length field of 4 bytes when using “outrecformat=var_4b”

parfile

Syntax **par**file=*parameter file*

Values *parameter file*

Specification of a file that contains FLAM parameters. This file can be generated with a text editor and can be used interactively.

Description By specifiying a parameter file, extensive entries on the command line can be replaced**.**
All parameters can be kept in a simple text file.

IMPORTANT NOTE: One parameter per line.

The entries on the command line have a higher priority.

Since Version 4.1 At the end of line comments may be attached with ‘!’. Lines that begin with ; or ! are ignored.

password

Syntax **pas**sword=*SECRET*

Values *SECRET*
String with a length of 1 to 64 characters

Description
FLAM® Version 4 with mode=ADC
AES or NDC

During compression the FLAMFILE® is encrypted with the specified password and can only be decompressed with the same password.

See also: **cryptmode, cryptkey, paeword, paxword**

IMPORTANT NOTE: On the command line not all characters are allowed, e.g. commas and subsequent blanks. If required a hex-encoded password can be entered with **paxword**.

Regardless of the encryption algorithm a key should consist of at least 6 – 8 characters and it should contain at least 1 character from each of the following sets:

Lower case letters
Capital letters
Numbers 0 - 9
Punctuation marks

paascii

Syntax **paa**scii=*SECRET*

Values *SECRET*

String with a length of 1 to 64 characters
The password is converted to ASCII before it is used.

In WINDOWS equal to **password**

Availability **FLAM® Version 4**

paebcdic

Syntax **pae**bcdic=*SECRET*

Values *SECRET*

String with a length of 1 to 64 characters

Before usage the password is converted to EBCDIC. This is helpful when communicating with a main frame.
Only those characters that can be clearly classified as either ASCII or EBCDIC should be used. This applies to all letters and numbers plus many punctuation marks (usually !“#$%&*+,./:;=?@). German umlauts must be avoided. In critical cases hex input with “paxword” is helpful.

Description
FLAM® Version 4

with mode=adc
aes or ndc
During compression the **FLAMFILE®** is encrypted with the specified password and can only be decompressed with the same password.

See also: **cryptmode, cryptkey, password, paxword**

pafile

Syntax **paf**ile=*file*

Values *File*
Name of file that contains the parameters for en- and decryption.

The following parameters are allowed:

label=
password=
paebcdic=
paascii=
paxword=
cryptmode=
cryptkey=

Availability **FLAM® Version 4.1**

Description The character string for password= is not subject to the restrictions of the command line. This means it can contain all characters (especially , „ ()) except CR,LF and NUL. As password everything between = and the end of record is taken, also subsequent blanks.

IMPORTANT NOTE: A password with subsequent blanks and commas can only be specified in a hex-encoded manner on the command line using paxword.
Lines that begin with ; or ! are ignored.

paxword

Syntax **pax**word=*SECRET*

Values *SECRET*

Hexadecimal character string with a length of 2 to 128 characters

The word *SECRET* is coded in ASCII as 534543524554 and in EBCDIC as E2C5C3E1C5E3.

For EBCDIC the entry for *SECRET* as password is as follows:
pax= E2C5C3E1C5E3 or pae= *SECRET*.

Description
FLAM® Version 4 with mode=adc
aes

During compression the **FLAMFILE®** is encrypted with the specified password and can only be decompressed with the same password.

See also: **cryptmode, cryptkey, paeword**, **password**

IMPORTANT NOTE: Regardless of the encryption algorithm a key should consist of at least 6 – 8 characters and it should contain at least 1 character from each of the following sets:

Lower case letters
Capital letters
Numbers 0 - 9
Punctuation marks

priority

Syntax **pri**ority=*VALUE*

Values **b**elow
idle

Description priority=below sets the priority of FLAM to lower than normally.
priority=idle sets the priority of FLAM to low.
This has the effect that FLAM only gets CPU time in the background if other processes with a higher priority, especially the dialogue, are not suppressed.

recdelim

Syntax **recd**elim= *end-of-record characters (delimiter)*

Values Hexadecimal characters between 01 and FF. Length: 1 or 2 bytes (2 or 4 hex characters)

Description Regarding **Record format text:** With this parameter the end-of-record character of the **FLAMFILE®** can be specified in connection with recform=text in mode=cx7.

If no end-of-record character is specified for record format text, 0d0a (CRLF) is used.

For example, with recdelim=0a the output of data is possible in UNIX format.

This parameter can only be specified for compression.

recformat

Syntax **recf**ormat=*format option*

Values *format option*

fix records with a fixed record length (default for all FLAMFILES)

var records with a variable record length. The additional (!) record length field is 2 bytes long.

text records with a variable record length and end-of-record characters (CX7 only)

Description With this parameter the record format of the **FLAMFILE®** is specified.
The FLAMFILE always contains records of the same length.

Regarding CX8 and VR8 the records can be written with (variable) or without (fix) a record length field.

Record format text can only be used for CX7. If no specification is done with **recdelim**, 0d0a is used as end-of-record character.

The specification of recformat is only required for compression.

recsize

Syntax **recs**ize=*nnn*

Values *nnn*

Integer decimal number between 80 and 32760 for modes CX8, VR8, ADC, NDC, AES

Integer number between 80 and 4095 in CX7 mode

The default value is 512.

Description This parameter specifies the record length of the **FLAMFILE®**.

The compressed data is written as records of the same length regardless of the record format.

There is no relation between compressed data blocks and the records in the compressed file. A record can contain data from one or more compressed data blocks. A compressed data block can be included in one or more records.

There is no relation between the record length of the FLAMFILE and the record length of the original files.

This parameter is only used for compression.

secureinfo

Syntax **sec**ureinfo**=*character set***

Values

yes	generate resp. check secure information (default)
no	do not generate secure information resp. ignore secure information

Description only in modes ADC, NDC, AES

AES-encrypted data is always generated with secure information.

show

Syntax **sh**ow=*display option*

Values *display option*

none	no display
all	Shows all of the information available. The information depends on the invoked operation, compression or decompression, but not on the circumstance if **FLAM®** was called directly or with a parameter file.
attributes	decompression only suppresses the creation of the decompressed file only shows the saved compression information: • name of the original file (only if compressed with attributes=all) • format specifications of the original file • compression mode • character set ASCII or EBCDIC • system which generated the FLAMFILE
error	only shows error and warning messages
statistic	shows error and warning messages plus the number of compressed records

suppress

Syntax **su**ppress

Values none

Description **Decompression and text record formats only:** Blanks at the end of record are suppressed.

The character that is to be suppressed can be specified with trimchar=*character.*

If trimchar is not specified in addition to suppress, then the setting is trimchar=20 or trimchar=40, depending on the character set of the original data.

The specification of **trimchar** implicitly sets **suppress**.

See also: **nosuppress**

trimchar

Syntax **tri**mchar=*characters*

Values hex value 00 to FF

Description **Decompression and text record formats only:** *characters* are suppressed at the end of record.

trimchar refers to the character in the original character set.
With the specification trimchar=*characters*, "suppress" is set automatically.

Default values of **trimchar**:

if the original data was in ASCII, then the setting is trimchar=20 (ASCII blank)

if the original data was in EBCDIC, then the setting is trimchar=40 (EBCDIC blank)

If translate=e/a is specified for decompression and the translation table for blank is 40:20, then the setting is trimchar=40.

See also: **suppress, nosuppress, translate**

truncate

Syntax **tru**ncate

Values none

Description Synonym for **cut**

See also: **notruncate**

translate

Syntax **tra**nslate=*translation option*

Values *translation option*

none	no conversion
a/**e**	conversion from ASCII to EBCDIC
e/**a**	conversion from EBCDIC to ASCII

Description This parameter induces a code conversion of the data of the original file before compression or of the data of the decompressed file after its decompression.

If a specific table is to be used for translation, it must be specified with **codetable=table** or else the default table (see **flam4 list**) is used.

09. ERROR MESSAGES

000	FLAM_OK	FLAM completed successfully
001	1	*ABEND*Program aborted
002	FLAM_EOF	End of file
003	FLAM_GAP	Gap found in relative file
004	FLAM_LONG_REC	Decompressed record(s) extended (source was shorter)
005	FLAM_NO_RECORD	No (valid) record found
006	FLAM_NEW_FILE	Beginning new file
007	FLAM_NO_PW	No password found
009	FLAM_NO_FH	No file header preserved
010	FLAM_UNKOMP	File is not a FLAMFILE
011	FLAM_KOMP_ERR	Bad FLAMFILE format
012	FLAM_ERR_RS	Bad FLAMFILE record length
013	FLAM_FILE_LENGTH	Unexpected end of file encountered
014	FLAM_CHS	Bad checksum
015	FLAM_REC_GR_32KB	Source record longer than 32,764 bytes
016	FLAM_REC_GR_BUFFER	Source record too large for matrix buffer
017	FLAM_FLAM_V1	FLAMFILE created with FLAM Version 1
018	FLAM_FLAM_SPLIT_PAR	FLAMFILE is parallel splitted
019	FLAM_FLAM_SPLIT_SER	FLAMFILE is serial splitted
020	FLAM_ILLEGAL_FCT	Bad Open Mode
021	FLAM_ILLEGAL_BUFFER	Invalid size of matrix buffer
022	FLAM_ILLEGAL_MODE	Invalid compression mode
023	FLAM_ILLEGAL_CODE	FLAMFILE contains invalid code
024	FLAM_ILLEGAL_BLKSIZE	Illegal blocking size
025	FLAM_ILLEGAL_RECSIZE	Illegal record length
026	FLAM_ILLEGAL_CH_SET	Illegal character code
027	FLAM_ILLEGAL_RECFORM	Illegal record format
028	FLAM_DEVICE	Source file device check unsuccessful
029	FLAM_ERR_PW	Password wrong or missing
030	FLAM_ERR_LEER	Input file empty
031	FLAM_ERR_FILE	Input file not found
032	FLAM_ERR_OPEN	Bad open mode
033	FLAM_ERR_ORG	Bad file type
034	FLAM_ERR_REC_FORM	Invalid record format
035	FLAM_ERR_REC_SIZE	Invalid record length
036	FLAM_ERR_BLK_SIZE	Invalid block length
037	FLAM_ERR_KEY_POS	Invalid key position
038	FLAM_ERR_KEY_SIZE	Invalid key length
039	FLAM_ERR_FILENAME	Invalid file specification / archive contains files without file header
040	FLAM_ERR_LOAD	Module or table cannot be loaded
041	FLAM_ERR_CALL	Module cannot be called
042	FLAM_ERR_UNLOAD	Module cannot be unloaded
043	FLAM_ERR_EXIT	Abnormal end caused by exit routine

045	**FLAM_ERR_MSGLOAD**	**Message table cannot be loaded**
046	**FLAM_ERR_MSGFILE**	**Message file cannot be opened**
047	**FLAM_ERR_DEFAULTS**	**Default data not recognized**
050	**FLAM_ERR_VFC**	**Error with dtaus / asn1 / ... / vfc format**
051	**FLAM_GR_MAXSIGDBL**	**Length error with EAF / stream / asn1 format**
052	**FLAM_TOO_MUCH_DUP_KEY**	**Too many duplicate keys**
053	**FLAM_PATH_INVALID**	**Path name of FLAM not found**
054	**FLAM_TEMP_INVALID**	**FLAM_TEMP invalid**
055	**FLAM_ERR_KEY_SEQ**	**Bad sequence of records with duplicate keys**
056	**FLAM_ERR_FHLEN**	**FLAMFILE file header too long (> 32 kB)**
057	**FLAM_KOMP_LENGTH**	**FLAMFILE segment length corrupted**
058	**FLAM_SPEC_RECFORM**	**Stream format for file not allowed**
060	**FLAM_SYNTAX**	**FLAMFILE corrupted**
061	**61**	**Too many counters (vr8)**
062	**62**	**Length error**
063	**FLAM_ERR_CHS_CRC3**	**Bad checksum in compressed data (CRC3)**
064	**FLAM_ERR_CHS_CRC1**	**Bad checksum in compressed data (CRC1)**
065	**FLAM_ERR_KONS**	**Bad check point**
066	**66**	**Bad check point**
068	**FLAM_ERR_MATRIX**	**Bad length of decompressed record**
070	**FLAM_ERR_VERSION**	**Invalid version of FLAMFILE**
071	**FLAM_FL_CUT**	**Decompressed record truncated**
072	**FLAM_RECORD_CUT**	**Decompressed record(s) truncated**
073	**FLAM_ERR_KOMP_LENGTH**	**Compressed data length bad**
074	**FLAM_ERR_CHECK_CHAR**	**Bad check character**
081	**FLAM_PARAM_ERR**	**Unknown argument or qualifier**
082	**FLAM_ERR_PAR_VAL**	**Invalid argument value**
085	**FLAM_SAME_NAME**	**Input and output files names must be distinct**
086	**FLAM_ERR_KEY_SEG**	**Specified sequence of key segments not contiguous**
087	**FLAM_NO_INPUT**	**Input file name missing**
088	**FLAM_NO_OUTPUT**	**Output file name missing**
089	**FLAM_ERR_LONG_VALUE**	**Qualifier too long or too big: %s**
090	**FLAM_ERR_VALUE**	**Invalid qualifier value: %s**
091	**FLAM_ERR_KEY**	**Invalid key for FLAMFILE:**
092	**FLAM_NO_REPLACE**	**No substitution characters in input file name**
093	**FLAM_NOT_IDX**	**No substitution characters in input file name**
094	**FLAM_NO_KEY**	**Key definition missing for index file**
095	**FLAM_ERR_KEY_TYPE**	**Segmented key is not of type character string**
096	**FLAM_ERR_NEXT_FILE**	**Error during search of next file name**
097	**FLAM_FILE_NOT_PROC**	**File cannot be processed (XABITM/LOCK)**
098	**FLAM_READ_ERR**	**Read error**
099	**FLAM_WRITE_ERR**	**Write error**
100	**100**	**Flam code not ASCII/EBCDIC**
101	**FLAM_NOT_ALL_FILES**	**Only part of the input files was processed**
102	**FLAM_WAR_NO_FILES**	**Specified number of output files > number of input files**

103	**FLAM_NOT_DEL**	**File was not deleted**
104	**FLAM_NO_FILE_OF_LIST**	**No specified file can be found in the compressed archive**
105	**FLAM_NO_DELIM_AT_EOF**	**No delimiter at end of file**
106	**FLAM_FILES_WITHOUT_FH**	**Compressed file contains files without file header (file name)**
111	**FLAM_SHORT_REC**	**Record(s) truncated**
112	**FLAM_INVALID_CODETAB**	**Invalid code table, size not 256 or 512**
113	**FLAM_NO_CODETAB**	**Code table not found**
114	**FLAM_SEC_NOT_ALLOWED**	**Secure with given mode not allowed**
115	**FLAM_ADD_NOT_ALLOWED**	**Add not allowed with secure**
340	**340**	**Min.number of not decompressed records:**
341	**341**	**Bad checksum in last block**
342	**342**	**SKIP_FLAM-position over corrupted block**
343	**343**	**mode=ADC**
344	**344**	**encryption=FLAMenc**
345	**345**	**encryption=AES**
346	**346**	**no encryption**
348	**FLM9_PARAM**	**Invalid qualifier / value of qualifier:**
350	**FLM9_INVALID_FUNCTION**	**Error at calloc / invalid ID**
351	**FLAM_SEC_ERR_351**	**SECURITY: Member number not contiguous**
352	**FLAM_SEC_ERR_352**	**SECURITY: Member trailer: counters wrong**
353	**FLAM_SEC_ERR_353**	**SECURITY: File trailer: counters wrong**
354	**AES_MEM_MAC**	**AES:MemberMACs wrong**
355	**AES_FIL_MAC**	**AES:FileMACs wrong**
356	**AES_CRC_SUFF**	**Bad checksum in compressed data (CRC-Offs)**
357	**AES_CRC_5**	**Bad checksum in compressed data (CRC-5)**
369	**FLAM_MEM_NO**	**Member number: %04d**
370	**UNDEF_MSG**	**AES:Undefined message**
900	**FLAM_PARAM**	**Invalid qualifier / value of qualifier:**
908	**FLAM_LIC_SUCCESS**	**FLAM licensed sucessfully**
996	**FLAM_INVALID_DEFDAT**	**File(s) corrupt:**
997	**FLAM_LICENSE_EXPIRED**	**Licence expired on**
998	**FLAM_INVALID_LICENSE**	**Invalid license:**
999	**FLAM_INVALID_FUNCTION**	**Error at calloc / invalid ID**
-1		**None specified error. Additional text in message file.**

10. COMPATIBILITY OF PARAMETERS

FLAM® Version 4 for WINDOWS is downwards compatible. All familiar key words are still supported (in some cases across the systems).

List of synonyms:

Alias	Synonym for
aes	mode=adc cryptmode=aes secureinfo=yes
buffer_size	buffersize
character_set	flamcode
characterset	flamcode
code_table	codetable
comment	label
crykey	cryptkey
crymode	cryptmode
cryptmode=aes	cryptmode=aes securinfoe=yes
cryptokey	cryptkey
cryptkey=___	password=__ cryptmode=aes secureinfo=yes
cryptomode	cryptmode
eight_bit	cx8
fileinfo=no **h**eader=no	attributes=none
fileinfo=no **h**eader=yes	attributes=common
fileinfo=yes **h**eader=yes	attributes=all
iblksize	inblocksize
iblocksize	inblocksize
inblcksize	inblocksize
indelim	inrecdelim
input_file	flamin
insize	inrecsize
irdelim	inrecdelim
irecdelim	inrecdelim
irecformat	inrecformat
irformat	inrecformat
irsize	inrecsize
ja	yes
log	logfile
message_file	logfile
messages	logfile
msgfile	logfile
nein	no
nofinfo	fileinfo=no
noheader	header=no
notest	secureinfo=no
notruncate	nocut

oblksize	outblocksize
oblocksize	outblocksize
opath	outpath
ordelim	outrecdelim
orecdelim	outrecdelim
orecformat	outrecformat
orecsize	outrecsize
orformat	outrecformat
orsize	outrecsize
pad-char	padchar
pad_char	padchar
parameter_file	parfile
seven_bit	cx7
stream	text
truncate	cut
uncompress	decompress

11. INDEX