FLAM training – compose your programme

Questionnaire and catalogue of units

Our training is free of charge. You provide room and time, we bring the speakers and the material. So that we can put together a programme that fits your people and the way you use FLAM, please answer the questions on this page and mark in the catalogue behind it what you are interested in. The form is fillable -- open it in a PDF reader, tick the boxes and send it back to education@flam.de.

1. Your training

Who is taking part?

☐ Systems programming, operations ☐ Application development ☐ Decision makers, architecture
☐ Information security, audit

How much FLAM do the participants already know?

☐ Beginners – FLAM is new ☐ Experienced – FLAM is in production ☐ Mixed

What do you use FLAM for?

☐ Data exchange ☐ Archiving ☐ Both

On which platforms?

☐ z/OS ☐ Linux, Unix ☐ Windows ☐ Cloud, object storage

Which way into FLAM do you use?

☐ Utility in batch or a script ☐ Subsystem, file system – application unchanged ☐ Programming interface
☐ Still open – please advise

How should the training run?

☐ On site at your place ☐ Remote ☐ With exercises on your own system
☐ With a discussion of your own use cases

Frame

Preferred duration in days

Preferred date or period

Number of participants

Language of the training

Contact

Company

Name

E-mail

Telephone

2. Catalogue of units

Three possibilities per unit: focus means in depth with exercises, short means an overview, leave out means do not cover it. To treat a whole group the same way, mark the group's own row. Without a mark we compose the unit from your answers on page 1. The detailed description of every unit is in the appendix.

	focus	short	leave out
Level 0 – fundamentals			
1. FLAM basics What FLAM is, what it consists of and how to reach it	☐	☐	☐
2. Tools: FLCL and FLCC The command line, the framework beneath it and the GUI above it	☐	☐	☐
3. Documentation and support Documentation, download area, ticket route and newsletter	☐	☐	☐
4. The command language CLE/P How a parameter string is built – and what the language itself can do	☐	☐	☐
5. The built-in functions Looking up, presetting, setting the environment, diagnosing	☐	☐	☐
6. Selecting files, forming names Wildcards, members, directory walk – and how the output name comes about	☐	☐	☐
7. Administration Configuration, environment, libraries, licence, authorisations	☐	☐	☐
8. Operation and maintenance Quality assurance, delivery, installation, update	☐	☐	☐
Level 1 – ways in			
9. The three access routes Batch, subsystem and API compared - and what distinguishes FLAM4 from FLAM5	☐	☐	☐
10. Access: batch utility flcl in bash, PowerShell and JCL - and how to evaluate the result	☐	☐	☐
11. Access: subsystem and file system One DD statement on z/OS, one mount on Linux and Unix	☐	☐	☐
12. Access: the APIs Byte, record, element, subprogram - plus C++ and Java	☐	☐	☐
13. Network: SSH and FLIES SSH instead of a path – and what became of FLIES	☐	☐	☐
Level 2 – functional subjects			
Archive	☐	☐	☐
14. FLAM4 - architecture and interfaces Four entrances into one access method – and how to pick the right one	☐	☐	☐

	focus	short	leave out
15. The FLAM5 archive Structure, keys, versioning and storage	☐	☐	☐
16. Indexing, filters and search How FLAM5 searches in encrypted data sets – and why that costs little	☐	☐	☐
17. Managing archived data Search, change, insert, delete – and verify	☐	☐	☐
18. Pot design Set compression, encryption and signature per column	☐	☐	☐
19. Segment store and cloud Where a FLAM5 archive lies – locally, remote, in the cloud or in your own storage	☐	☐	☐
Data and formats	☐	☐	☐
20. Conversion with FLUC Read, convert, write more than once	☐	☐	☐
21. Compression Seven methods, one measured comparison	☐	☐	☐
22. File formats and records Record formats, length fields, delimiters – and the automatic detection	☐	☐	☐
23. Tables and CSV Recognise formats, rebuild columns, write them out true to type	☐	☐	☐
24. The format compiler The structure description becomes the table description	☐	☐	☐
25. Detecting data without a description Undescribed records become a table	☐	☐	☐
26. character sets 186 code pages, four key questions – and what is different on the host	☐	☐	☐
27. XML processing Reading and writing XML – and turning it into rows and columns	☐	☐	☐
28. Foreign formats and exchange ZIP, Base64, FLAMFILE – and the transparent reading	☐	☐	☐
Security	☐	☐	☐
29. Crypto and keys Encrypt, sign, manage keys	☐	☐	☐
30. PGP and signatures OpenPGP to the standard, hashes and signatures	☐	☐	☐
31. Security in operation What the operator has to observe - and which backdoors are open on purpose	☐	☐	☐
32. SAF and RACF Control the functions of a program through RACF	☐	☐	☐

	focus	short	leave out
Tools	☐	☐	☐
33. Auxiliary functions Information, search, checksums, comparison, keys	☐	☐	☐
Operations	☐	☐	☐
34. Embedding into batch jobs Best practices for JCL, parameter files and return codes	☐	☐	☐
35. Performance and optimisation Why FLAM is built for mass data - and what you can adjust	☐	☐	☐
36. Accelerators on z/OS zEDC, CPACF and zIIP – what really helps and what does not	☐	☐	☐

3. The units in detail

The detailed description of every unit in the catalogue, in the same order and with the same number.

1. FLAM basics

What FLAM is, what it consists of and how to reach it

The unit every course begins with: FLAM is an access method and not a compression tool – it reads and writes data, and compression, encryption, signature and character conversion happen while it does. You learn to tell the three component groups FLC/ORG, FLM and FLS apart and thus to say which part sees clear data at all. Plus the sentence that carries the whole course: one parameter string, three ways in. At the end you can place where to look things up for your question, and you know the platform and release policy.

2. Tools: FLCL and FLCC

The command line, the framework beneath it and the GUI above it

The tools unit: 15 commands in one program, plus 25 built-in functions and a GUI that assembles the call string. You learn to read the syntax of the command language – command, object, overlay, array, number, string --, to use the built-in help and to generate parts of the manual yourself. FLCC is more than a surface: it is the interactive manual and the generator for the shell line, the parameter file and the JCL. Whoever follows here types no long string by hand afterwards.

3. Documentation and support

Documentation, download area, ticket route and newsletter

Where to find what, and how to get help. One manual holds for the utility, the subsystem and the interface together, because all three take the same parameter string – that explains its structure. You learn to find documentation, software, add-ons and libraries on flam.de without searching, and to raise a ticket that can be worked without a question back. Plus the rule that prevents a data protection incident: no dumps and no production data to the support.

4. The command language CLE/P

How a parameter string is built – and what the language itself can do

The language every FLAM call is written in – and it is the same on every way. You learn the five parameter forms, the lexemes with their comments and number formats, and the operators that make a string shorter: the value out of a file for passphrases, the parameter file per command, overlay and object. At the end you read a syntax line out of the manual without help and know why a call with one bracket more means something else.

5. The built-in functions

Looking up, presetting, setting the environment, diagnosing

The 25 built-in functions are not an accessory but the frame around the commands: syntax and help, generating manual pages and documentation, presetting and maintaining properties, setting and listing environment variables, printing version, licence and the list of errors. You learn which function helps in which situation, how a property file comes about and how to run several shapes side by side with the owner – test against production, or one tenant per customer.

6. Selecting files, forming names

Wildcards, members, directory walk – and how the output name comes about

One call, many files: this unit shows how files are selected and how the fitting output name arises from every input name. You learn the wildcards and their limits, the selection of members in libraries and archives, the directory walk over subdirectories and the name building with placeholders. Plus the naming rules of the platforms: when a name is a path, when a data set and when an allocation out of the JCL – and what happens when an environment variable is not defined.

7. Administration

Configuration, environment, libraries, licence, authorisations

The unit for those who operate FLAM. Five sources decide what a call does – command line, parameter file, property file, configuration and environment – and having their order of precedence in your head saves the most frequent support question: why does my parameter not take effect. You learn to have every environment variable listed with its meaning, which libraries and load modules your platform needs, how the licence is bound in and where the authorisations take hold.

8. Operation and maintenance

Quality assurance, delivery, installation, update

What is in a delivery and how to bring it into operation. You learn with which methods we check before delivering, what is signed and how to verify it, and how an update runs without a standstill. Plus the answer to the question that comes up in every audit: how do I answer a CVE question about FLAM and about the external libraries in it – with the two lists we publish for every version.

9. The three access routes

Batch, subsystem and API compared - and what distinguishes FLAM4 from FLAM5

The unit that stands before the decision: there are three ways into FLAM, and all three take the same parameter string. You learn to justify the fitting way for your case – by the question who has to change something: operations or development. Plus the distinction of the generations: what FLAM4 can do, what FLAM5 adds and why both are operated side by side. At the end you recognise the same call in every wrapping.

10. Access: batch utility

flcl in bash, PowerShell and JCL - and how to evaluate the result

The way most houses go first: FLAM as a command in a job chain. You learn to write the same call in bash, PowerShell and JCL, to move long parameters into files, and to read the completion code correctly – including the bracket that keeps a warning from turning a chain red. Plus the allocation when writing: why a dynamic allocation is the better choice and what a static assignment does to the shape of your records.

11. Access: subsystem and file system

One DD statement on z/OS, one mount on Linux and Unix

The transparent way: the application stays unchanged and FLAM hangs itself underneath. On z/OS through a DD statement with the subsystem, on Linux and Unix through a mounted directory. You learn how the parameter string is handed over in this world, what the subsystem can do in conversion, compression and encryption, and what expressly does not work this way. The case that convinces: an unchanged utility writes a compressed file to a remote machine.

12. Access: the APIs

Byte, record, element, subprogram - plus C++ and Java

The unit for application development: four interfaces that differ in the level of abstraction, not in the functionality. You learn which interface fits which task – bytes, records, elements or the call of a subprogram with the same parameter string the command line takes. Plus the language bindings for C++ and Java and the rules you have to know when building it into an existing application.

13. Network: SSH and FLIES

SSH instead of a path – and what became of FLIES

Reading and writing over the network without operating a server of your own: a URL stands where the path would otherwise stand. You learn the two spellings for the same connection, the five levels of the host key check and the batch-capable one among them, the use of the existing OpenSSH configuration and the pre- and post-processing on the other side. Plus the clear answer what became of the old FLIES methods and what stands in their place today.

14. FLAM4 - architecture and interfaces

Four entrances into one access method – and how to pick the right one

FLAM4 is not a compression tool with an interface attached but an access method with four entrances: utility, subsystem, record interface and subprogram. You learn to tell those four apart and to choose the fitting one for a given file, and you see how a single record is found in a large archive. Plus FKME: how the key management stays in the house. The unit also places the limits of FLAM4 and where FLAM5 begins.

15. The FLAM5 archive

Structure, keys, versioning and storage

The FLAM5 archive from the inside: matrices, segments and the three keys with which directory, data and index are protected separately. You learn how an archive grows, what a version is and why an operator can manage an archive without seeing the data. Plus the separation of archive management and storage, which explains why the same archive can lie locally, on a remote machine or in the cloud.

16. Indexing, filters and search

How FLAM5 searches in encrypted data sets – and why that costs little

The unit that explains how to search in compressed, encrypted and signed data without rendering it in the clear. You learn the mechanics of the Bloom filter from the hash through the partial signature to the segment signature, design an indexing per column – how many hashes, how long the signature – and understand the two-stage access in which a hit in the filter is not yet a hit in the record. Plus the limits and the question when a real data indexing would be the better choice.

17. Managing archived data

Search, change, insert, delete – and verify

What you do with an existing archive: search, change records, insert and delete, manage versions and clean up – all without rendering the whole file in the clear. You learn the commands for it, the result sets and how to process them further, the checking depths when verifying and the difference between deleting and cleaning up. Plus the capacity question: how an archive with many changes grows and when it is worth rewriting it.

18. Pot design

Set compression, encryption and signature per column

Pot design is design work, not configuration: a pot is a group of columns for which it holds jointly how it is compressed, encrypted and signed. You learn the two questions with which columns are grouped, the parameters of a pot with its three streams and the trade-off behind it: one pot disturbs the compression, too many pots have too little data per pot. Plus the defaults of the archive, among them the integrity level – the answer to the security question.

19. Segment store and cloud

Where a FLAM5 archive lies – locally, remote, in the cloud or in your own storage

The lower layer of the archive: where the segments lie physically. You get to know the storages – file, folder, S3, Azure, IBM CDA and the access over SSH --, the buffering without which an archive in the cloud would be unusably slow, and the two programming interfaces with which a house connects a storage of its own. Plus the honest separation between what works today and what is planned as cloudFLAM.

20. Conversion with FLUC

Read, convert, write more than once

The core of the converter: one input, any number of conversions, several outputs – and all of it in one step, without temporary files. You learn which read object fits which source and which combinations of reading and writing make sense, how character sets are stated on three levels and how a chain of nine job steps becomes a single one. The case from practice is worked through completely.

21. Compression

Seven methods, one measured comparison

Which compression method when, and what it costs. You see a comparison of the seven supported methods with measured figures, learn why no specification is needed when reading, and what compression does to the record boundaries. Plus the placing of the hardware acceleration on the mainframe and the case where a method is chosen not for its rate but because the other side expects it.

22. File formats and records

Record formats, length fields, delimiters – and the automatic detection

The unit about the shape of the data: fixed and variable records, length fields and delimiters, print control characters and the formats of the mainframe world. You learn to state a format, to have it read out of the file name or to have FLAM recognise it from the data – and where the recognition reaches its limit. Plus the net rule for the record length and the two edge cases that produce support calls: the empty and the very large record.

23. Tables and CSV

Recognise formats, rebuild columns, write them out true to type

Records become rows and columns: this unit shows how a structure is described, how FLAM recognises it by itself in many cases and how CSV, JSON or another table representation comes out of it. You get to know the column types and their formats, the rebuilding of columns and the three command products for binary, ASCII and foreign-format CSV. The thread is the question who has the description – you, a copybook or nobody.

24. The format compiler

The structure description becomes the table description

Whoever has described a structure does not have to describe it a second time: FLAM compiles a COBOL copybook, a PL/I structure definition or an HLASM DSECT into the table description itself. This unit shows what the compiler makes of the fields, what it brings along that is not in the source at all, how four annotations at the field steer it and how IMS unloads get their copybook segment by segment.

25. Detecting data without a description

Undescribed records become a table

When there is no copybook any more the file is still readable: FLAM derives the column structure from the data itself. This unit shows the three detectors for fixed-length records, character-separated data and XML, what a detector uses to place a column boundary, where it cannot find one, how hints narrow the search space and how an IMS unload without any segment description becomes a CSV file. And it shows the way back: look at a detected description, correct it and write it down.

26. character sets

186 code pages, four key questions – and what is different on the host

Character sets are the most frequent cause of data that looks wrong. You learn to separate the four questions that are always thrown together: in which character set is the data, in which should it be afterwards, what happens to characters that do not exist in the target, and in which character set was the call actually typed. Plus the peculiarities of the EBCDIC world, the Unicode forms with their identifiers and the tools with which a conversion is verified.

27. XML processing

Reading and writing XML – and turning it into rows and columns

For FLAM, XML is a data format like others: it is read, converted and written. You learn how a document is broken into elements, how a table comes out of XML and XML out of a table, what happens to namespaces, CDATA and line endings, and how to minimise or indent a document. Plus the limits you should know before giving a foreign format to the parser.

28. Foreign formats and exchange

ZIP, Base64, FLAMFILE – and the transparent reading

The exchange with systems that do not have FLAM: ZIP archives to the standard, Base64 and other encodings, GZIP and the old FLAMFILEs. You learn how FLAM recognises by itself what it has in front of it when reading, how to address an archive member directly and which encoding fits which target system. Plus the rule that saves time: the target of an encoding is a binary stream, not a text.

29. Crypto and keys

Encrypt, sign, manage keys

The unit in which methods become an operational decision. You learn the methods FLAM offers, the difference between a passphrase and key management, and how FKME keeps the keys in the house. Plus the questions that come before the technology: who may decrypt, who manages the keys, what happens when staff change – and how to translate the answers into parameters.

30. PGP and signatures

OpenPGP to the standard, hashes and signatures

PGP is the common denominator in the data exchange between houses. You learn how FLAM implements OpenPGP to the RFC, how keys are imported and managed, what ASCII armor is and when you need it. Plus the hash and signature methods with their use cases and the difference that counts in an audit: a checksum proves that nothing changed, a signature proves in addition who it was.

31. Security in operation

What the operator has to observe - and which backdoors are open on purpose

The unit for the security department: what FLAM presupposes in operations, where the clear data travels and which possibilities exist that an attacker could use. You learn the switches with which a house seals itself off, and the places that are deliberately open because operations need them. The tone is open: we name the limits ourselves instead of admitting them when asked.

32. SAF and RACF

Control the functions of a program through RACF

On z/OS not only the access to files can be regulated but the use of the functions of FLAM itself – through a SAF class of its own. You learn how a resource is built, what is controllable (policies, parameter strings, interfaces, URLs, connections, keys, logging and licence), how the class is defined and activated and in which order to introduce it without halting operations. Plus the honest answer about what is missing of that on other platforms.

33. Auxiliary functions

Information, search, checksums, comparison, keys

The small commands that save the most time in daily work: information about the system and about files, search with regular expressions on the host as well, checksums, the logical comparison of two files and the check against a format description. You learn which command answers which question – bytes, content or structure – and which of them belongs in a ticket before you raise one.

34. Embedding into batch jobs

Best practices for JCL, parameter files and return codes

The unit that turns a job chain into one step. You see how seven temporary files become one call, how parameter files and JCL symbols make a procedure reusable, how return codes are read and what should stand in the environment of the step. Plus the switches for cleaning up, renaming and empty results – the points where an automation otherwise comes to a halt.

35. Performance and optimisation

Why FLAM is built for mass data - and what you can adjust

The unit with the figures: what the throughput costs, where the time goes and which adjusting screws really take effect. You learn the effect of block size and record count, the difference between one run over many files and many runs, the cost of the character conversion and the logging as an often forgotten item. The calculation is done with measured values, not with estimates.

36. Accelerators on z/OS

zEDC, CPACF and zIIP – what really helps and what does not

The unit for mainframe operations: which hardware accelerators FLAM uses, how to verify it and what they actually bring. You learn to tell the methods apart – compression in the card, crypto instructions in the processor, work that can be offloaded – and get the honest answer to the question about offloading. Plus the operation of the started task and the places where an update asks for more than exchanging a library.

4. Remarks

Whatever was not asked for belongs here: your own use cases you would like to discuss, files or formats we should look at beforehand, organisational constraints.

Our ticket / quotation (filled in by us)

Send the completed form to education@flam.de. We compose a programme from it, propose a date and a duration, and send you the material for the units you chose as a PDF beforehand.