FLUCUPLB-API

1

Generated by Doxygen 1.8.14

Contents

Chapter 1

FLUC Load Module Interface

This interface provides all FLCL commands (CONV, XCNV, ...) based on the Frankenstein Limes Universal Converter (FLUC) as simple to use and programming language independent load modules. It is mainly meant to be used for Assembler, COBOL and PL1 on mainframe systems. It can also be used in C through the fetch() function, but the C like FLUCUP native interface is easier to use in this case.

For each command, there is a function that accepts a command line and property file string that follows the FLCL syntax and produces the same condition code as FLCL. The property file string is parsed before the command line string. The property string may contain command properties in the property file syntax of FLCL based on the default owner "limes". You can read an FLCL property file programmatically and provide its content as this string. To generate a property file please use:

```
FLCL GENPROP command=filename
```

It is currently not possible to change the owner id or program name. In case this is required, please use the native FLUCUP interface.

To manage a property file per command the function FCUPRO() is provided.

The interface accepts the content of such a file as string, not the filename itself! This allows to store properties in other formats such as JSON, XML or binary and use it with the UP interface.

The programming interface can be used to integrate the source (file) to target (file) data conversion capabilities provided by FLUC into a program.

The interface also provides the interactive help, documentation and syntax information to assist in understanding all commands.

1.1 Interface standards

Each function is provided as a separate load module. All parameters are call-by-reference and have no return value. There are 3 types of parameters:

- INTEGER: pointer to a 32 bit number in two's complement
- STRING[x]: pointer to a byte (8 bit) array of length x
- STRING: pointer to a variable length byte (8 bit) array

Input strings may be 0-terminated in which case the corresponding length pointer may be NULL. All output strings are 0-terminated.

For variable length byte arrays, the length will be determined based on an additional INTEGER parameter. On input, this parameter must contain the length of the string. On output, this parameter must be loaded with the available size for the byte array and will be set to the actually used size. If the size is too small, the string will be cut and the length will be set to the respective value. This case will be marked with the special condition code 10.

For output files you can define ":STDERR" or ":STDOUT" to assign the corresponding streams.

1.2 Environment variables

For all default character conversions, it is useful to set the environment variable LANG. Other used environment variables of FLAM can be found in the FLCL manual. With version 5.1.19 a new function (FCUENV()) to load the FLAM environment was introduced. This function can be used in front of the first API call to establish the same environment used by FLAM utilities, subsystems and so on. This give the application developer the possibility to adjust the environment before the first real call is done. Till 5.1.18 each function has read the system variables on z/OS. This is now part of the FCUENV() function to give complete control about the environment to the user of the API.

1.3 Condition codes

The return codes of the subprogram are also called condition codes and returned at subprogram termination.

- 0 - command line, command syntax, mapping, execution and finish of the command was successful
- 1 - command line, command syntax, mapping, execution and finish of the command was successful but a warning can be found in the log
- 2 - command line, command syntax, mapping, execution was successful but cleanup of the command failed (may not happened)
- 4 - command line, command syntax and mapping was successful but execution of the command returns with a warning
- 8 - command line, command syntax and mapping was successful but execution of the command returns with an error
- 10 - Provided buffer for about, version or license string is too small
- 12 - command line and command syntax was OK but mapping failed
- 16 - command line was OK but command syntax was wrong
- 20 - command line was wrong (user error)
- 24 - initialization of the parameter structure of command failed (may not happened)
- 28 - configuration is wrong (user error)
- 32 - table error (something within the predefined tables is wrong)
- 36 - system error (e.g. load of environment or open of file failed)
- 40 - access control or license error
- 44 - interface error (e.g. parameter pointer equals NULL)
- 48 - memory allocation failed (e.g. dynamic string handling)
- 64 - fatal error (basic things are damaged)
- >64 - for few special conditions codes (see FLCLBOOK)

The return code parameter can be a null pointer, then no return value is set. Special condition codes can be found in FLCL manual (FLCLBOOK).

1.4 Reason codes

The reason code (internal FLAM return code) can be determined together with a corresponding message using the function FCURSN. All FLAM reason codes are listed in FLCL manual (FLCLBOOK).

1.5 Help and Syntax

For the help, syntax and documentation functions, the command must be specified as a constant number called 'what':

- 1 - to get information about the INFO command string
- 2 - to get information about the CONV command string
- 3 - to get information about the XCNV command string
- 4 - to get information about the ICNV command string
- 5 - to get information about the FLAM command string

The help, syntax and documentation can be used to determine the complete context sensitive help, which is identical to the content of the FLCL user manual section for the respective command.

1.6 Compile and Link

1.6.1 On mainframes (z/OS, BS2000, ...):

The interface is provided as DLL and each function of the DLL is also available as separate load module.

For dynamic linking of the load module (fetch) the hlq.FLAM.LOAD library must be part of the STEPLIB concatenation.

If you link one of the load modules statically to your application (each load module contains all required entries of the interface), you must include the DLL import files below to resolve the missing external references:

```
hlq.FLAM.IMPORT(FLUCUP)
hlq.FLAM.IMPORT(FL5CORE)
```

To link dynamically against the DLL you must include the import DLL file form the IMPORT library below:

```
hlq.FLAM.IMPORT(FLUCUPLB)
```

1.6.2 On other platforms (Unix, Windows, ...):

The interface is only available as dynamic link library (DLL) or shared object (SO). You can link it dynamically, statically or load the library at runtime (dlopen(), LoadLibrary()) with the common DLL/SO mechanism of your operating system.

1.7 Sample programs

Sample programs in C with name SCFCUCNV/GZP can be found as part of the installation package for mainframe systems in the library SRCLIBC(SCFCUxxx), with the corresponding compile and link step in JOBLIB(SBUILD). For other platforms (Windows, UNIX) the sample program source of SCFCUCNV/GZP is located in the 'sample' directory and the compile and link procedures can be found in the Makefile of the same directory.

Additional COBOL samples with name SOFCUCNV/KME can be found in SRCLIB(SOFCUxxx). The corresponding compile and link procedures is also located as separate step in JOBLIB(SBUILD).

Chapter 2

Module Index

2.1 API definitions

Here is a list of the API parts:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Module Documentation

4.1 FLUC Function Codes

Macros

- #define FLUC_INFO 1
  INFO command.
- #define FLUC_CONV 2
  CONV command.
- #define FLUC_XCNV 3
  XCNV command.
- #define FLUC_ICNV 4
  ICNV command.
- #define FLUC_FLAM 5
  FLAM command.
- #define FLUC_XCHK 6
  XCHK command.
- #define FLUC_HASH 7
  HASH command.
- #define FLUC_UTIL 8
  UTIL command.
- #define FLUC_DIFF 9
  DIFF command.
- #define FLUC_KEY 10
  KEY command.

4.1.1 Detailed Description

Pass one of this to FCUHLP(), FCUVSN(), FCUDOC() or FCUPRO() to get information about the corresponding command.

4.1.2 Macro Definition Documentation

4.1.2.1 FLUC_INFO

```
#define FLUC_INFO 1
```

INFO command.

4.1.2.2 FLUC_CONV

```
#define FLUC_CONV 2
```

CONV command.

4.1.2.3 FLUC_XCNV

```
#define FLUC_XCNV 3
```

XCNV command.

4.1.2.4 FLUC_ICNV

```
#define FLUC_ICNV 4
```

ICNV command.

4.1.2.5 FLUC_FLAM

```
#define FLUC_FLAM 5
```

FLAM command.

4.1.2.6 FLUC_XCHK

```
#define FLUC_XCHK 6
```

XCHK command.

4.1.2.7 FLUC_HASH

```
#define FLUC_HASH 7
```

HASH command.

4.1.2.8 FLUC_UTIL

```
#define FLUC_UTIL 8
```

UTIL command.

4.1.2.9 FLUC_DIFF

```
#define FLUC_DIFF 9
```

DIFF command.

4.1.2.10 FLUC_KEY

```
#define FLUC_KEY 10
```

KEY command.

4.2 Functions

Functions

- void FCUVSN (int *rtc, int *vsnLen, char *vsn)

  Version.
- void FCUABO (int *rtc, int *aboLen, char *abo)

  About.
- void FCULIC (int *rtc, int *licLen, char *lic)

  Get license text.
- void FCUENV (int *rtc, const int *sys, const int *std, const int *envLen, const char *envStr, int *cnt)

  Load the FLAM environment.
- void FCUINFO (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC INFO - Provides various information.
- void FCUUTIL (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC UTIL - Runs various simply functions on wildcard file lists.
- void FCUCONV (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC CONV - Simplified data conversion.
- void FCUXCNV (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC XCNV - Extended data conversion.
- void FCUICNV (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC ICNV - Character conversion like ICONV.
- void FCUFLAM (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC FLAM - Support of FLAM4 command in FLUC.
- void FCUXCHK (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC XCHK - Support of XCHK command in FLUC.
- void FCUHASH (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC HASH - Support of HASH command in FLUC.
- void FCUDIFF (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC DIFF - Compares two data sources.
- void FCUKEY (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC KEY - Key management functions.
- void FCURSN (int *rsn, int *msgLen, char *msg)

  Returns reason code and message.
- void FCUHLP (int *rtc, const int *what, const int *depth, const int *pathLen, const char *path, const int *outLen, const char *out)

  Returns help information about the command strings.
- void FCUSYN (int *rtc, const int *what, const int *depth, const int *pathLen, const char *path, const int *outLen, const char *out)

  Return syntax information about the command string.
- void FCUDOC (int *rtc, const int *what, const int *pathLen, const char *path, const int *outLen, const char *out)

Return documentation.

- void FCUPRO (int ∗rtc, const int ∗what, const int ∗iproLen, const char ∗ipro, const int ∗propLen, const char ∗prop, const int ∗oproLen, const char ∗opro, const int ∗outLen, const char ∗out)

  Manage properties.

4.2.1 Detailed Description

4.2.2 Function Documentation

4.2.2.1 FCUVSN()

```
void FCUVSN (
             int * rtc,
             int * vsnLen,
             char * vsn )
```

Version.

This function writes a string containing version information for every used component into the 'vsn' parameter. This should be used in a support case.

Parameters

out	*rtc*	INTEGER Condition code
in,out	*vsnLen*	INTEGER Size/length of version string
out	*vsn*	STRING Version string (is null-terminated)

4.2.2.2 FCUABO()

```
void FCUABO (
             int * rtc,
             int * aboLen,
             char * abo )
```

About.

This function writes a string with about information for this library on multiple lines and license information for used external libraries into the 'abo' parameter .

Parameters

out	*rtc*	INTEGER Condition code
in,out	*aboLen*	INTEGER Size/length of about string
out	*abo*	STRING About string (is null-terminated)

4.2.2.3 FCULIC()

```
void FCULIC (
            int * rtc,
            int * licLen,
            char * lic )
```

Get license text.

This function can be used to get the current license text on multiple lines. The license text defines the permissible use of this library.

Parameters

out	*rtc*	INTEGER Condition code
in,out	*licLen*	INTEGER Size/length of license string
out	*lic*	STRING Current license text (is null-terminated)

4.2.2.4 FCUENV()

```
void FCUENV (
            int * rtc,
            const int * sys,
            const int * std,
            const int * envLen,
            const char * envStr,
            int * cnt )
```

Load the FLAM environment.

This function load the FLAM system variables (only on z/OS or in EDZ environment), the STDENV definitions ('DD:STDENV' or '<SYSUID>.STDENV' on z/OS or '.stdenv' file in working or home directory on other system) and a optional list if variables to the environment. You can adjust the environment before the first call to a FLAM API function is done. This function gives the possibility to work with the same environment used by FLAM utilities, subsystems aso.

Sample call in COBOL:

```
WORKING-STORAGE SECTION.
01  F-RETCO         PIC S9(8) COMP VALUE 0.
01  F-SYSENV        PIC S9(8) COMP VALUE 1.
01  F-STDENV        PIC S9(8) COMP VALUE 1.
01  F-ENVLEN        PIC S9(8) COMP VALUE 200.
01  F-ENVAR         PIC X(200)
    VALUE 'KEYWORD1=VALUE1;KEYWORD2=VALUE2'
01  F-ENVCNT        PIC S9(8) COMP VALUE 0.
PROCEDURE DIVISION.
CALL  'FCUENV' USING F-RETCO,F-SYSENV,F-STDENV,
                     F-ENVLEN,F-ENVAR,F-ENVCNT.
```

Parameters

out	*rtc*	INTEGER Return code
in	*sys*	INTEGER True to load system variables, NULL or false no system variables active
in	*std*	INTEGER True to load standard environment, NULL or false no standard environment variables active
in	*envLen*	INTEGER Length of the optional environment string (NULL if optional or if zero termination)
in	*envStr*	STRING Optional list (could be NULL or empty) of environment variables (KEYWORD=VALUE) separated by new line ('\n') or semicolon (';')
out	*cnt*	INTEGER Amount of environment variables set (optional, could be NULL)

4.2.2.5 FCUINFO()

```
void FCUINFO (
             int * rtc,
             const int * cmdLen,
             const char * cmd,
             const int * proLen,
             const char * pro,
             const int * outLen,
             const char * out,
             const int * trcLen,
             const char * trc )
```

FLUC INFO - Provides various information.

This function calls the INFO command of FLUC

Example (Cobol):

```
CALL "FCUINFO" USING rtc 19 "get.file='test.gz'" 0 " " 7 ":STDERR" 0 " "
```

Parameters

out	*rtc*	INTEGER Condition code
in	*cmdLen*	INTEGER Length of command string (must be set)
in	*cmd*	STRING Command string with GET and LOG instructions
in	*proLen*	INTEGER Length of property string (0 if no properties)
in	*pro*	STRING Property string with GET and LOG instructions
in	*outLen*	INTEGER Length of output filename (if 0 no output is produced)
in	*out*	STRING Filename for printouts
in	*trcLen*	INTEGER Length of trace filename (if 0 no trace is written)
in	*trc*	STRING Filename for tracing

4.2.2.6 FCUUTIL()

```
void FCUUTIL (
             int * rtc,
             const int * cmdLen,
             const char * cmd,
             const int * proLen,
             const char * pro,
             const int * outLen,
             const char * out,
             const int * trcLen,
             const char * trc )
```

FLUC UTIL - Runs various simply functions on wildcard file lists.

This function calls the UTIL command of FLUC

Example (Cobol):

```
CALL "FCUUTIL" USING rtc 20 "run.list='~.test.**'" 0 " " 0 " " 0 " "
```

Parameters

out	*rtc*	INTEGER Condition code
in	*cmdLen*	INTEGER Length of command string (must be set)
in	*cmd*	STRING Command string with RUN and LOG instructions
in	*proLen*	INTEGER Length of property string (0 if no properties)
in	*pro*	STRING Property string with RUN and LOG instructions
in	*outLen*	INTEGER Length of output filename (if 0 no output is produced)
in	*out*	STRING Filename for printouts
in	*trcLen*	INTEGER Length of trace filename (if 0 no trace is written)
in	*trc*	STRING Filename for tracing

4.2.2.7 FCUCONV()

```
void FCUCONV (
             int * rtc,
             const int * cmdLen,
             const char * cmd,
             const int * proLen,
             const char * pro,
             const int * outLen,
             const char * out,
             const int * trcLen,
             const char * trc )
```

FLUC CONV - Simplified data conversion.

This function calls the CONV command of FLUC

Example (Cobol):

```
CALL FCUCONV USING rtc 42 "read.text(file='test.txt') write.record()" 0 " " 7 ":STDERR" 0 " "
```

Parameters

out	*rtc*	INTEGER Condition code
in	*cmdLen*	INTEGER Length of command string (must be set)
in	*cmd*	STRING Command string with READ, WRITE and LOG instructions
in	*proLen*	INTEGER Length of property string (0 if no properties)
in	*pro*	STRING Property string with READ, WRITE and LOG instructions
in	*outLen*	INTEGER Length of output filename (if 0 no output is produced)
in	*out*	STRING Filename for printouts
in	*trcLen*	INTEGER Length trace filename (if 0 no trace is written)
in	*trc*	STRING Filename for tracing

4.2.2.8 FCUXCNV()

```
void FCUXCNV (
             int * rtc,
             const int * cmdLen,
             const char * cmd,
             const int * proLen,
             const char * pro,
             const int * outLen,
             const char * out,
             const int * trcLen,
             const char * trc )
```

FLUC XCNV - Extended data conversion.

This function calls the XCNV command of FLUC

Example (Cobol):

```
CALL FCUXCNV USING Rtc
                   123
                   "inp(sav.fil(fio.blk(name='test.txt')
                                        cnv.chr(from='IBM-1141'
                                        to='UTF-8') fmt.txt()))
                    out(sav.fil(fio.blk(name='test.out')))"
                    0 " " 7 ":STDERR" 0 " "
```

Parameters

out	*rtc*	INTEGER Condition code
in	*cmdLen*	INTEGER Length of command string ()
in	*cmd*	STRING Command string with INP, OUT and LOG instructions
in	*proLen*	INTEGER Length of property string (0 if no properties)
in	*pro*	STRING Property string with INP, OUT and LOG instructions
in	*outLen*	INTEGER Length of output filename (if 0 no output is produced)
in	*out*	STRING Filename for printouts
in	*trcLen*	INTEGER Length of trace filename (if 0 no trace is written)
in	*trc*	STRING Filename for tracing

4.2.2.9 FCUICNV()

```
void FCUICNV (
            int * rtc,
            const int * cmdLen,
            const char * cmd,
            const int * proLen,
            const char * pro,
            const int * outLen,
            const char * out,
            const int * trcLen,
            const char * trc )
```

FLUC ICNV - Character conversion like ICONV.

This function calls the ICNV command of FLUC

Example (Cobol):

```
CALL FCUICNV USING rtc
                   62
                   "in='test.txt' from='IBM-1141' to='UTF-8' ENL2LF
                    out='test.out'"
                   0 " " 7 ":STDERR" 0 " "
```

Parameters

out	*rtc*	INTEGER Condition code
in	*cmdLen*	INTEGER Length of command string (must be set)
in	*cmd*	STRING Command string with ICONV parameter
in	*proLen*	INTEGER Length of property string (0 if no properties)
in	*pro*	STRING Property string with ICONV parameter
in	*outLen*	INTEGER Length of output filename (if 0 no output is produced)
in	*out*	STRING Filename for printouts
in	*trcLen*	INTEGER Length of trace filename (if 0 no trace is written)
in	*trc*	STRING Filename for tracing

4.2.2.10 FCUFLAM()

```
void FCUFLAM (
            int * rtc,
            const int * cmdLen,
            const char * cmd,
            const int * proLen,
            const char * pro,
            const int * outLen,
            const char * out,
```

```
            const int * trcLen,
            const char * trc )
```

FLUC FLAM - Support of FLAM4 command in FLUC.

This function calls the FLAM command of FLUC

Example (Cobol):

```
CALL FCUFLAM USING rtc
                   50
                   "COMP MODE=ADC FLAMIN='test.txt' FLAMFILE='test.adc'"
                   0 " " 7 ":STDERR" 0 " "
```

Parameters

out	*rtc*	INTEGER Condition code
in	*cmdLen*	INTEGER Length of command string (must be set)
in	*cmd*	STRING Command string with FLAM parameter
in	*proLen*	INTEGER Length of property string (0 if no properties)
in	*pro*	STRING Property string with FLAM parameter
in	*outLen*	INTEGER Length of output filename (if 0 no output is produced)
in	*out*	STRING Filename for printouts
in	*trcLen*	INTEGER Length of trace filename (if 0 no trace is written)
in	*trc*	STRING Filename for tracing

4.2.2.11 FCUXCHK()

```
void FCUXCHK (
            int * rtc,
            const int * cmdLen,
            const char * cmd,
            const int * proLen,
            const char * pro,
            const int * outLen,
            const char * out,
            const int * trcLen,
            const char * trc )
```

FLUC XCHK - Support of XCHK command in FLUC.

This function calls the XCHK command of FLUC

Example (Cobol):

```
CALL FCUXCHK USING rtc
                   49
                   "inp(sav.fil(fio.blk(name='test.txt') cnv.hsh()))"
                    0 " " 7 ":STDERR" 0 " "
```

Parameters

out	*rtc*	INTEGER Condition code
in	*cmdLen*	INTEGER Length of command string (must be set)
in	*cmd*	STRING Command string with FLAM parameter
in	*proLen*	INTEGER Length of property string (0 if no properties)
in	*pro*	STRING Property string with FLAM parameter
in	*outLen*	INTEGER Length of output filename (if 0 no output is produced)
in	*out*	STRING Filename for printouts
in	*trcLen*	INTEGER Length of trace filename (if 0 no trace is written)
in	*trc*	STRING Filename for tracing

4.2.2.12 FCUHASH()

```
void FCUHASH (
             int * rtc,
             const int * cmdLen,
             const char * cmd,
             const int * proLen,
             const char * pro,
             const int * outLen,
             const char * out,
             const int * trcLen,
             const char * trc )
```

FLUC HASH - Support of HASH command in FLUC.

This function calls the HASH command of FLUC

Example (Cobol):

```
CALL FCUHASH USING rtc
                   28
                   "file='test.txt' algo=sha512"
                    0 " " 7 ":STDERR" 0 " "
```

Parameters

out	*rtc*	INTEGER Condition code
in	*cmdLen*	INTEGER Length of command string (must be set)
in	*cmd*	STRING Command string with FLAM parameter
in	*proLen*	INTEGER Length of property string (0 if no properties)
in	*pro*	STRING Property string with FLAM parameter
in	*outLen*	INTEGER Length of output filename (if 0 no output is produced)
in	*out*	STRING Filename for printouts
in	*trcLen*	INTEGER Length of trace filename (if 0 no trace is written)
in	*trc*	STRING Filename for tracing

4.2.2.13 FCUDIFF()

```
void FCUDIFF (
            int * rtc,
            const int * cmdLen,
            const char * cmd,
            const int * proLen,
            const char * pro,
            const int * outLen,
            const char * out,
            const int * trcLen,
            const char * trc )
```

FLUC DIFF - Compares two data sources.

This function calls the DIFF command of FLUC

Example (Cobol):

```
CALL FCUDIFF USING rtc 56 "read.file='./test.txt.gz' compare.file='~.test.fba' DATA" 0 " " 7 ":STD
```

Parameters

out	*rtc*	INTEGER Condition code
in	*cmdLen*	INTEGER Length of command string (must be set)
in	*cmd*	STRING Command string with READ, COMPARE and LOG instructions
in	*proLen*	INTEGER Length of property string (0 if no properties)
in	*pro*	STRING Property string with READ, COMPARE and LOG instructions
in	*outLen*	INTEGER Length of output filename (if 0 no output is produced)
in	*out*	STRING Filename for printouts
in	*trcLen*	INTEGER Length trace filename (if 0 no trace is written)
in	*trc*	STRING Filename for tracing

4.2.2.14 FCUKEY()

```
void FCUKEY (
            int * rtc,
            const int * cmdLen,
            const char * cmd,
            const int * proLen,
            const char * pro,
            const int * outLen,
            const char * out,
            const int * trcLen,
            const char * trc )
```

FLUC KEY - Key management functions.

This function calls the KEY command of FLUC

Example (Cobol):

```
CALL FCUKEY USING rtc 32 "import.pgp(file='./keyFile.pgp')" 0 " " 7 ":STDERR"
```

Parameters

out	*rtc*	INTEGER Condition code
in	*cmdLen*	INTEGER Length of command string (must be set)
in	*cmd*	STRING Command string with IMPORT/EXPORT/... and LOG instructions
in	*proLen*	INTEGER Length of property string (0 if no properties)
in	*pro*	STRING Property string with IMPORT/EXPORT/... instructions
in	*outLen*	INTEGER Length of output filename (if 0 no output is produced)
in	*out*	STRING Filename for printouts
in	*trcLen*	INTEGER Length trace filename (if 0 no trace is written)
in	*trc*	STRING Filename for tracing

4.2.2.15 FCURSN()

```
void FCURSN (
            int * rsn,
            int * msgLen,
            char * msg )
```

Returns reason code and message.

This function returns the reason code and optional a message after a command fails. If the buffer too small the message is truncated. 128 byte is a good size for the message buffer.

Parameters

out	*rsn*	INTEGER Reason code (internal FLAM return code)
in,out	*msgLen*	INTEGER Size/length of message string (if 0 no message is returned)
out	*msg*	STRING Message string (is null-terminated and optional)

4.2.2.16 FCUHLP()

```
void FCUHLP (
            int * rtc,
            const int * what,
            const int * depth,
            const int * pathLen,
            const char * path,
```

```
            const int * outLen,
            const char * out )
```

Returns help information about the command strings.

This function can be used to get all help messages or man pages for each parameter in the command string.

Parameters

out	*rtc*	INTEGER Condition code
in	*what*	INTEGER Constant number for the corresponding command string
in	*depth*	INTEGER Number of levels to display (0-Man page, 1-One Level, 2-Two Level, ..., <9-All)
in	*pathLen*	INTEGER Length of path string (0 to start at root)
in	*path*	STRING Path (e.g. conv.read.text.ccsid) to limit help to the (sub)tree of a certain object
in	*outLen*	INTEGER Length of output filename (if 0 stderr is used)
in	*out*	STRING Filename for printouts

4.2.2.17 FCUSYN()

```
void FCUSYN (
            int * rtc,
            const int * what,
            const int * depth,
            const int * pathLen,
            const char * path,
            const int * outLen,
            const char * out )
```

Return syntax information about the command string.

This function can be used to determine the complete syntax of the command strings.

Parameters

out	*rtc*	INTEGER Condition code
in	*what*	INTEGER Constant number for the corresponding command string
in	*depth*	INTEGER Number of levels to display (0-Man page, 1-One Level, 2-Two Level, ..., <9-All)
in	*pathLen*	INTEGER Length of path string (0 to start at root)
in	*path*	STRING Path (e.g. conv.read.text.ccsid) to limit syntax to the (sub)tree of a certain object
in	*outLen*	INTEGER Length of output filename (if 0 stderr is used)
in	*out*	STRING Filename for printouts

4.2.2.18 FCUDOC()

```
void FCUDOC (
            int * rtc,
```

```
            const int * what,
            const int * pathLen,
            const char * path,
            const int * outLen,
            const char * out )
```

Return documentation.

This function can be used to determine the complete documentation of the corresponding command.

Parameters

out	*rtc*	INTEGER Condition code
in	*what*	INTEGER Constant for the corresponding command string
in	*pathLen*	INTEGER Length of path string (0 to start at root)
in	*path*	STRING Path (e.g. conv.read.text.ccsid) to limit docu to the (sub)tree of a certain object
in	*outLen*	INTEGER Length of output filename (if 0 stderr is used)
in	*out*	STRING Filename for printouts

4.2.2.19 FCUPRO()

```
void FCUPRO (
            int * rtc,
            const int * what,
            const int * iproLen,
            const char * ipro,
            const int * propLen,
            const char * prop,
            const int * oproLen,
            const char * opro,
            const int * outLen,
            const char * out )
```

Manage properties.

This function can be used to manage a property file for one of the supported commands. You can generate a property file if 'ipro' not set and 'opro' defined. You can printout the properties if 'ipro' defined and 'opro' is not set. You can validate and copy property files if 'ipro' and 'opro' defined. You can adjust a property file if 'ipro' defined, 'prop' are used and 'opro' set.

Parameters

out	*rtc*	INTEGER Condition code
in	*what*	INTEGER Constant for the corresponding command string
in	*iproLen*	INTEGER Length of input property file string (0 to generate a property file)
in	*ipro*	STRING Input property file string
in	*propLen*	INTEGER Length of property string (if 0 then no manipulation of properties done)
in	*prop*	STRING Property string to update the property file
in	*oproLen*	INTEGER Length of output property file string (if 0 stdout is used)
in	*opro*	STRING Output property file string
in	*outLen*	INTEGER Length of output filename (if 0 no output is produced)
in	*out*	STRING Filename for printouts

Chapter 5

File Documentation

5.1 FLUCUPLB.h File Reference

FLUCUPLB - Description - Load module interface for FLUC subprogram.

Macros

- #define FLUC_INFO 1

  INFO command.
- #define FLUC_CONV 2

  CONV command.
- #define FLUC_XCNV 3

  XCNV command.
- #define FLUC_ICNV 4

  ICNV command.
- #define FLUC_FLAM 5

  FLAM command.
- #define FLUC_XCHK 6

  XCHK command.
- #define FLUC_HASH 7

  HASH command.
- #define FLUC_UTIL 8

  UTIL command.
- #define FLUC_DIFF 9

  DIFF command.
- #define FLUC_KEY 10

  KEY command.

Functions

- void FCUVSN (int *rtc, int *vsnLen, char *vsn)

  Version.
- void FCUABO (int *rtc, int *aboLen, char *abo)

  About.
- void FCULIC (int *rtc, int *licLen, char *lic)

  Get license text.
- void FCUENV (int *rtc, const int *sys, const int *std, const int *envLen, const char *envStr, int *cnt)

  Load the FLAM environment.
- void FCUINFO (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC INFO - Provides various information.
- void FCUUTIL (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC UTIL - Runs various simply functions on wildcard file lists.
- void FCUCONV (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC CONV - Simplified data conversion.
- void FCUXCNV (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC XCNV - Extended data conversion.
- void FCUICNV (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC ICNV - Character conversion like ICONV.
- void FCUFLAM (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC FLAM - Support of FLAM4 command in FLUC.
- void FCUXCHK (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC XCHK - Support of XCHK command in FLUC.
- void FCUHASH (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC HASH - Support of HASH command in FLUC.
- void FCUDIFF (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC DIFF - Compares two data sources.
- void FCUKEY (int *rtc, const int *cmdLen, const char *cmd, const int *proLen, const char *pro, const int *outLen, const char *out, const int *trcLen, const char *trc)

  FLUC KEY - Key management functions.
- void FCURSN (int *rsn, int *msgLen, char *msg)

  Returns reason code and message.
- void FCUHLP (int *rtc, const int *what, const int *depth, const int *pathLen, const char *path, const int *outLen, const char *out)

  Returns help information about the command strings.
- void FCUSYN (int *rtc, const int *what, const int *depth, const int *pathLen, const char *path, const int *outLen, const char *out)

  Return syntax information about the command string.
- void FCUDOC (int *rtc, const int *what, const int *pathLen, const char *path, const int *outLen, const char *out)

  Return documentation.
- void FCUPRO (int *rtc, const int *what, const int *iproLen, const char *ipro, const int *propLen, const char *prop, const int *oproLen, const char *opro, const int *outLen, const char *out)

  Manage properties.

5.1.1 Detailed Description

FLUCUPLB - Description - Load module interface for FLUC subprogram.

Author

limes datentechnik gmbh

Date

18.02.2015

Copyright

Index