C/C++ Interface Library
Quick Start Guide
AN00192

Application Note

Revision 1.1 – Jul 2026

SMH Technologies®

UNIVERSAL PRODUCTION IN-SYSTEM PROGRAMMING

Copyright © 2019 SMH Technologies
DC11843

SMH Technologies S.r.l.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

Introduction

This guide explains how to install and integrate the FR_COMM C/C++ DLL in a customer application on Windows.

The library exports a flat C API callable from C, C++, or any language that supports Windows DLL imports. It provides:

- command execution
- file transfer
- project execution
- public key retrieval and encryption
- real-time logger access
- log capture (streaming) and log file download

This document only covers the supported public API exposed to customer applications.

Package Delivery

The deliverable consists of three files:

FR_COMM.dll	runtime library
FR_COMM.lib	import library for the linker
FR_COMM.h	public header with all declarations

Supported Environments

- Compilers: MSVC (Visual Studio 2017 or later), MinGW-w64
- Operating system: Windows (x86 and x64)
- Language standards: C99 or later, C++11 or later
- Character encoding:
    - FR_COMM.h provides unsuffixed macros — FR_OpenCommunication, FR_SendCommand, etc. — that resolve automatically based on whether UNICODE is defined in the project settings
        - Unicode build (UNICODE defined, Visual Studio default since 2005): string parameters are wchar_t*; use L"..." wide string literals in code
        - ANSI build (UNICODE not defined): string parameters are char*; use plain string literals
        - Use the unsuffixed macro name (e.g. FR_OpenCommunication) in all customer code

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

Installation

Copy FR_COMM.dll, FR_COMM.lib and FR_COMM.h to your project tree. Then configure your build system as follows.

MSVC / Visual Studio:

```
# Project Properties > Linker > Input > Additional Dependencies
FR_COMM.lib

# Project Properties > C/C++ > General > Additional Include Directories
<path containing FR_COMM.h>
```

CMake:

```
target_include_directories(my_app PRIVATE path/to/header)
target_link_libraries(my_app PRIVATE path/to/FR_COMM.lib)
```

At runtime, FR_COMM.dll must be in the same directory as the application executable or in a directory listed in the system PATH.

Include the header at the top of each source file that uses the library:

```
#include "FR_COMM.h"
```

Installation Verification

Verify that the DLL loads and the version is readable with the following minimal program:

```
#include "FR_COMM.h"
#include <stdio.h>

int main(void)
{
    unsigned int major, minor, build, rev;
    GetDllVersion(&major, &minor, &build, &rev);
    printf("%u.%u.%u.%u\n", major, minor, build, rev);
    return 0;
}
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

Supported Public API

Customer applications should use only the following types and functions declared in FR_COMM.h:

- FR_COMM_HANDLE — opaque communication handle
- FR_LOGGER_HANDLE — opaque logger handle
- FR_COMM_ERR — error code type (unsigned long)
- FR_FileTransferProgressProc — file transfer callback type
- FR_ProjectExecutionProc — project execution callback type
- RET_OK — success constant (value 0)
- FR_LOGSTREAM_HANDLE – opaque log stream handle
- FR_LogStreamOptions, FR_LogStreamStats – real-time log stream configuration and counters
- FR_LogFetchOptions, FR_LogFetchResult – log file download configuration and result
- All FR_* and GetDllVersion functions listed in the API Summary

API Summary

API / function	Purpose	Typical use
FR_OpenCommunication	Open a LAN or USB communication session	Call first; returns NULL on failure
FR_CloseCommunication	Close and release a communication session	Always call when the handle is no longer needed
FR_SendCommand	Send one fully-prefixed command to FlashRunner	Request status or trigger a device action
FR_SendCommandEx	Send one command to an explicit set of channels	Multi-channel targeted commands without manual prefix
FR_GetAnswer	Read the answer for the last sent command	Provide a caller-allocated output buffer
FR_RunProject	Execute a project on one or more channels	Blocks until all channels complete or timeout
FR_SendFile	Upload one file to FlashRunner via YMODEM	Transfer .frb, .prj, .lic, .log, .lib files
FR_GetFile	Download one file from FlashRunner via YMODEM	Retrieve files from device storage
FR_GetPublicKey	Retrieve the FlashRunner RSA-2048 public key	Required before any encryption call
FR_EncryptData	Encrypt one payload using the device key	Legacy single-shot dynamic memory encryption

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

FR_DynMemBeginEncryption	Open a dynamic memory encryption session	Required before FR_DynMemEncryptData
FR_DynMemEncryptData	Encrypt one block within the active session	Sequential encryption with CBC chaining
FR_DynMemResetEncryption	Close the active dynamic memory session	Explicit session teardown
FR_HexToAsciiStream	Convert binary data to a hexadecimal ASCII string	Format encrypted payloads before sending
GetDllVersion	Read the DLL version as four integer fields	Installation and runtime validation
FR_GetLastErrorMessage	Read the last error description into a buffer	Call immediately after any failure
FR_SendCommandToAMPlus	Send a command to an AM+ module	Call FR_GetAnswer to read the response
FR_SendFileToAMPlus	Send a file to an AM+ module	AM+ port 1–8 maps to channel 55 + port
FR_GetFileFromAMPlus	Retrieve a file from an AM+ module	Requires YMODEM protocol
FR_GetLogger	Open a real-time LAN logger connection	Read logger output over LAN
FR_DisposeLogger	Close and dispose the logger connection	Always call when the logger is no longer needed
FR_LoggerRead	Read available logger bytes into a buffer	Non-blocking; poll in a loop
FR_StartLogStream	Start a real-time, filtered log capture (LAN only)	Capture a run's log into memory
FR_StopLogStream	Stop a log capture and return the accumulated data	Returns an allocated buffer; free it with FR_FreeLogData
FR_FreeLogData	Free a buffer returned by FR_StopLogStream	Always call after FR_StopLogStream
FR_FetchLogFile	Download the stored log file (LAN or USB)	Log retrieval with filtering and per-channel split

Connection Configuration

LAN Connection

Pass the following arguments to FR_OpenCommunication():

- com_port = L"LAN"
- com_settings = L"<ip>:<tcp_port>"

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

Example:

```
h = FR_OpenCommunication(L"LAN", L"192.168.1.100:1234");
```

USB or Serial Connection

Pass the following arguments to FR_OpenCommunication():

- com_port = L"COM3" (or L"COM10", etc.)
- com_settings = baud rate as a wide string

Examples:

```
h = FR_OpenCommunication(L"COM3",  L"115200");
h = FR_OpenCommunication(L"COM3",  L"3000000");
```

Supported baud rates are:

- 115200
- 3000000



Note: When using USB, FlashRunner powers up at 115200.
If the session must continue at 3000000, the host must first send #55*SETSERIALBAUDRATE HIGH while communicating at 115200.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

Public API Reference

FR_COMM_HANDLE

Opaque handle returned by FR_OpenCommunication. Pass it as the first argument to every subsequent API call. NULL indicates failure.

FR_LOGGER_HANDLE

Opaque handle returned via FR_GetLogger. Pass it to FR_LoggerRead and FR_DisposeLogger.

FR_COMM_ERR

Error code type (unsigned long). RET_OK (0) indicates success.

All functions except FR_OpenCommunication, GetDllVersion and FR_GetLastErrorMessage return FR_COMM_ERR. After any non-zero return, call FR_GetLastErrorMessage to retrieve the human-readable description.

FR_FileTransferProgressProc

Callback type for file transfer progress notifications.

```
typedef FR_COMM_ERR (WINAPI *FR_FileTransferProgressProc)(
   unsigned long file_size,   /* total file size in bytes */
   unsigned long file_pos);   /* current position in bytes */
```

Pass NULL to disable progress notifications. The return value of the callback is ignored by the library.

FR_ProjectExecutionProc

Callback type for per-channel project execution results.

```
typedef void (WINAPI *FR_ProjectExecutionProc)(
   int channel,   /* 1-based channel number */
   int result);   /* 1 = success, 0 = failure */
```

Pass NULL to disable per-channel notifications.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

FR_OpenCommunication

```
FR_COMM_HANDLE WINAPI FR_OpenCommunication(
   const wchar_t *com_port,
   const wchar_t *com_settings);
```

Parameters:

- com_port: L"LAN" for TCP, or L"COMx" for USB/serial
- com_settings: for LAN, L"<ip>:<port>"; for USB, the baud rate as a wide string

Return value:

- a valid FR_COMM_HANDLE on success
- NULL on failure — call FR_GetLastErrorMessage to get the error description

Behavior:

- opens the communication channel and queries FlashRunner for its active channel configuration
- the handle must be closed with FR_CloseCommunication when no longer needed

FR_CloseCommunication

```
FR_COMM_ERR WINAPI FR_CloseCommunication(FR_COMM_HANDLE handle);
```

Closes the communication channel and releases all resources associated with the handle. Always call this when the handle is no longer needed.

FR_SendCommand

```
FR_COMM_ERR WINAPI FR_SendCommand(
   FR_COMM_HANDLE handle,
   const wchar_t *command);
```

Parameter:

- command: full command string including the required FlashRunner prefix (e.g. L"#GETENGSTATUS")

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

Postcondition:

- the command is sent synchronously
- the next FR_GetAnswer call reads the response for this command

FR_SendCommandEx

```
FR_COMM_ERR WINAPI FR_SendCommandEx(
   FR_COMM_HANDLE  handle,
   const int      *target_channels,
   int            num_channels,
   const wchar_t  *command);
```

Parameters:

- target_channels: array of 1-based channel numbers, e.g. int ch[] = {1, 2, 3}
- num_channels: number of entries in target_channels
- command: command body without the channel prefix

Behavior:

- constructs the appropriate channel prefix automatically (#N*, bitmask, or broadcast)
- the next FR_GetAnswer call reads the response

FR_GetAnswer

```
FR_COMM_ERR WINAPI FR_GetAnswer(
   FR_COMM_HANDLE  handle,
   wchar_t        *answer,
   unsigned long   maxlen,
   unsigned long   timeout_ms);
```

Parameters:

- answer: caller-allocated output buffer for the decoded response string
- maxlen: size of the answer buffer in wide characters
- timeout_ms: maximum wait time in milliseconds

Precondition:

- FR_SendCommand or FR_SendCommandEx must have been called on the same handle

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

FR_RunProject

```
FR_COMM_ERR WINAPI FR_RunProject(
   FR_COMM_HANDLE        handle,
   const wchar_t         *project_name,
   const int             *channels,
   int                   num_channels,
   unsigned long         timeout_ms,
   FR_ProjectExecutionProc callback);
```

Parameters:

- project_name: project file name including the .prj extension
- channels: array of 1-based channel numbers
- num_channels: number of entries in channels
- timeout_ms: maximum time to wait for all channels to complete
- callback: FR_ProjectExecutionProc called once per channel on completion, or NULL to disable

Preconditions:

- the project must already be available on FlashRunner
- channels must identify valid target channels for the connected device

FR_SendFile

```
FR_COMM_ERR WINAPI FR_SendFile(
   FR_COMM_HANDLE            handle,
   const wchar_t             *protocol,
   const wchar_t             *src_filename,
   const wchar_t             *dst_path,
   FR_FileTransferProgressProc progress);
```

Parameters:

- protocol: must be L"YMODEM"
- src_filename: full local path of the file to send
- dst_path: file type string on the FlashRunner side — one of L"FRB", L"PRJ", L"LIC", L"LOG", L"LIB"
- progress: FR_FileTransferProgressProc, or NULL to disable callbacks

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

Preconditions:

- src_filename must exist and must not be empty
- the handle must be open

FR_GetFile

```
FR_COMM_ERR WINAPI FR_GetFile(
   FR_COMM_HANDLE          handle,
   const wchar_t           *protocol,
   const wchar_t           *src_filename,
   const wchar_t           *dst_path,
   const wchar_t           *filetype,
   FR_FileTransferProgressProc progress);
```

Parameters:

- protocol: must be L"YMODEM"
- src_filename: name of the file as stored on FlashRunner
- dst_path: local destination directory
- filetype: file type string — one of L"FRB", L"PRJ", L"LIC", L"LOG", L"LIB"
- progress: FR_FileTransferProgressProc, or NULL to disable callbacks

Preconditions:

- dst_path must already exist on the local file system
- the requested file must already exist on FlashRunner

FR_GetPublicKey

```
FR_COMM_ERR WINAPI FR_GetPublicKey(FR_COMM_HANDLE handle);
```

Retrieves the FlashRunner RSA-2048 public key and stores it internally per handle.

Postcondition:

- the handle keeps the imported public key for subsequent encryption calls

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

FR_EncryptData

```
FR_COMM_ERR WINAPI FR_EncryptData(
   FR_COMM_HANDLE       handle,
   const unsigned char  *data_to_encrypt,
   unsigned int         len_data,
   unsigned char        *header,
   unsigned char        *data_encrypted);
```

Parameters:

- data_to_encrypt: input payload
- len_data: length of the input in bytes
- header: caller-allocated 384-byte output buffer for the RSA-encrypted AES key and IV material
- data_encrypted: caller-allocated output buffer; size must be 16-byte aligned

Important:

- FR_GetPublicKey must be called first in the same session
- each call generates a new AES key and IV — the header changes every time
- calling this function automatically resets any active DynMem session

Notes:

- data_encrypted is the AES-256-CBC encrypted payload
- header contains the RSA-encrypted key material required by the device-side decryption flow

FR_DynMemBeginEncryption

```
FR_COMM_ERR WINAPI FR_DynMemBeginEncryption(
   FR_COMM_HANDLE  handle,
   unsigned char  *header);
```

Parameter:

- header: caller-allocated 384-byte output buffer for the RSA-encrypted session key and IV; send this to FlashRunner via #DYNMEMSETHEADER

Precondition:

- FR_GetPublicKey must have been called in the same session

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

Postcondition:

- subsequent FR_DynMemEncryptData calls use the session AES key
- any previously active session is reset before the new one begins

FR_DynMemEncryptData

```
FR_COMM_ERR WINAPI FR_DynMemEncryptData(
   FR_COMM_HANDLE       handle,
   const unsigned char  *data,
   unsigned int         len_data,
   unsigned char        *data_encrypted);
```

Parameters:

- data: input data block to encrypt
- len_data: length of the input in bytes
- data_encrypted: caller-allocated output buffer; size must be 16-byte aligned

Precondition:

- FR_DynMemBeginEncryption must have been called first

Behavior:

- the IV is updated to the last 16 bytes of the ciphertext after each call (CBC chaining)
- this allows sequential blocks to be encrypted without re-establishing the session

FR_DynMemResetEncryption

```
FR_COMM_ERR WINAPI FR_DynMemResetEncryption(FR_COMM_HANDLE handle);
```

Closes the active dynamic memory encryption session and securely wipes all session state from memory.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

FR_HexToAsciiStream

```
FR_COMM_ERR WINAPI FR_HexToAsciiStream(
   const unsigned char *data_hex,
   unsigned int        len_data,
   char                *data_ascii);
```

Parameters:

- data_hex: binary input buffer
- len_data: number of bytes to convert
- data_ascii: caller-allocated output buffer; must be at least len_data * 2 + 1 bytes

Converts binary data to a lowercase hexadecimal ASCII string. Useful for formatting encrypted headers before sending them as command arguments.

GetDllVersion

```
void WINAPI GetDllVersion(
   unsigned int *major,
   unsigned int *minor,
   unsigned int *build,
   unsigned int *revision);
```

Reads the DLL version from the file resources. All four fields are set to 0 if the version cannot be read.

FR_GetLastErrorMessage

```
void WINAPI FR_GetLastErrorMessage(
   wchar_t      *error_msg,
   unsigned long  string_len);
```

Parameters:

- error_msg: caller-allocated output buffer
- string_len: size of the buffer in wide characters

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

Returns a human-readable description of the last error. The message is cleared after retrieval. Call immediately after any failure to capture the error before making another API call.

FR_SendCommandToAMPlus

```
FR_COMM_ERR WINAPI FR_SendCommandToAMPlus(
   FR_COMM_HANDLE  handle,
   int          amplusPort,
   const wchar_t  *command);
```

Parameters:

- amplusPort: AM+ port number (1–8)
- command: command body without the channel prefix; the library constructs the target channel address automatically (channel = 55 + amplusPort)

Behavior:

- sends the command to the AM+ module connected to the specified port
- call FR_GetAnswer on the same handle to read the response

FR_SendFileToAMPlus

```
FR_COMM_ERR WINAPI FR_SendFileToAMPlus(
   FR_COMM_HANDLE            handle,
   int                  amplusPort,
   const wchar_t           *protocol,
   const wchar_t           *src_filename,
   const wchar_t           *filetype,
   FR_FileTransferProgressProc  progress);
```

Parameters:

- amplusPort: AM+ port number (1–8)
- protocol: must be L"YMODEM"
- src_filename: full local path of the file to send
- filetype: file type string on the AM+ side — one of L"FRB", L"PRJ", L"LIC", L"LOG", L"LIB"
- progress: FR_FileTransferProgressProc, or NULL to disable callbacks

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

Preconditions:

- src_filename must exist and must not be empty
- the handle must be open

FR_GetFileFromAMPlus

```
FR_COMM_ERR WINAPI FR_GetFileFromAMPlus(
   FR_COMM_HANDLE            handle,
   int                  amplusPort,
   const wchar_t           *protocol,
   const wchar_t           *src_filename,
   const wchar_t           *dst_path,
   const wchar_t           *filetype,
   FR_FileTransferProgressProc  progress);
```

Parameters:

- amplusPort: AM+ port number (1–8)
- protocol: must be L"YMODEM"
- src_filename: name of the file as stored on the AM+ module
- dst_path: local destination directory
- filetype: file type string — one of L"FRB", L"PRJ", L"LIC", L"LOG", L"LIB"
- progress: FR_FileTransferProgressProc, or NULL to disable callbacks

Preconditions:

- dst_path must already exist on the local file system
- the requested file must already exist on the AM+ module

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

Log Capture (Stream and Fetch)

The library offers three ways to obtain the FlashRunner log; pick the one that matches how you want to consume it:

- Raw logger (FR_GetLogger / FR_LoggerRead): a live, unfiltered byte stream over TCP port 1235 (LAN only). You poll it yourself and receive the log as it arrives, keeping full control of buffering and parsing. Use it when you want to display or forward the log continuously while a run is in progress, or when you need every byte unmodified.
- Log Stream (FR_StartLogStream / FR_StopLogStream): the same real-time stream, but the library runs the reader, applies channel and level filters, and buffers the result for you; you get the accumulated, filtered text in one block when you stop the stream, not incrementally. Use it to capture a run and read its filtered log afterwards, without managing the socket yourself. Several streams can run at the same time, even on the same handle.
- Log Fetch (FR_FetchLogFile): not a live stream but a download of the persistent log file already stored on the FlashRunner (LAN or USB). It applies the same channel and level filters, can split the log per channel, and preserves the raw download. Use it after a run, or on USB connections where real-time streaming is not available.

FR_GetLogger

```
FR_COMM_ERR WINAPI FR_GetLogger(
   const wchar_t    *ip_address,
   FR_LOGGER_HANDLE *logger);
```

Parameters:

- ip_address: logger endpoint in the form L"<ip>:<port>" (e.g. L"192.168.1.100:1235")
- logger: output pointer that receives the logger handle

Behavior:

- opens a separate TCP connection to the FlashRunner logger endpoint
- the logger connection is independent from any command handle

FR_DisposeLogger

```
FR_COMM_ERR WINAPI FR_DisposeLogger(FR_LOGGER_HANDLE logger);
```

Closes and destroys the logger connection. Always call this when the logger is no longer needed.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

FR_LoggerRead

```
FR_COMM_ERR WINAPI FR_LoggerRead(
   FR_LOGGER_HANDLE  logger,
   char          *buffer,
   int           size,
   int          *bytes_read);
```

Parameters:

- buffer: caller-allocated output buffer
- size: size of the buffer in bytes
- bytes_read: number of bytes actually read; 0 when no data is available

Behavior:

- non-blocking: returns RET_OK with bytes_read == 0 when no data is currently available
- call in a polling loop with a short sleep between iterations

FR_StartLogStream

```
FR_COMM_ERR WINAPI FR_StartLogStream(
   FR_COMM_HANDLE          handle,
   const FR_LogStreamOptions *options,
   FR_LOGSTREAM_HANDLE     *streamHandle);
```

Opens a dedicated TCP connection to the FlashRunner logger port (1235) on the same IP as the command handle, starts a background reader thread, and returns an opaque stream handle.
Parameters:

- handle: an open LAN communication handle
- options: capture filters and buffer settings (see FR_LogStreamOptions); must not be NULL
- streamHandle: output opaque handle to pass to FR_StopLogStream

Behavior:

- real-time streaming is supported on LAN only; a USB handle returns an error
- no file is written; the filtered log is kept in memory until the stream is stopped
- several streams may run concurrently, even on the same handle; each opens its own connection and buffer
- streams started from a handle are stopped automatically and discarded when FR_CloseCommunication is called on that handle

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

FR_StopLogStream

```
FR_COMM_ERR WINAPI FR_StopLogStream(
   FR_LOGSTREAM_HANDLE  streamHandle,
   char            **data,
   FR_LogStreamStats   *stats);
```

Stops the reader thread, closes the connection, and returns the accumulated, filtered log.
Parameters:

- streamHandle: the handle returned by FR_StartLogStream
- data: receives a newly-allocated, NUL-terminated UTF-8 buffer with the captured log; free it with FR_FreeLogData. Pass NULL to discard the text and read only the counters.
- stats: receives the capture counters (see FR_LogStreamStats), or NULL to ignore them

Behavior:

- pass the stream handle returned by FR_StartLogStream, not the command handle
- stopping an unknown or already-stopped handle returns an error
- the exact byte length of *data is stats->bytes_returned (the data may, in theory, contain embedded NUL bytes)

FR_FreeLogData

```
FR_COMM_ERR WINAPI FR_FreeLogData(char *data);
```

Frees a buffer previously returned in the data parameter of FR_StopLogStream. Passing NULL is a no-op. Always call this for every non-NULL buffer returned by FR_StopLogStream.

FR_FetchLogFile

```
FR_COMM_ERR WINAPI FR_FetchLogFile(
   FR_COMM_HANDLE          handle,
   const FR_LogFetchOptions *options,
   FR_LogFetchResult       *result);
```

Downloads the persistent log file (log.txt, and optionally the rotated log.txt.1) from the FlashRunner internal filesystem, applies channel and level filters, optionally splits the log per channel, and preserves the raw download. Works over both LAN and USB.
Parameters:

- handle: an open LAN or USB communication handle
- options: download location and filters (see FR_LogFetchOptions); log_dir is required

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

- result: output structure describing the files produced (see FR_LogFetchResult)

Behavior:

- a session folder is created under log_dir, named "SN<serial> - Date YYYY-MM-DD - Time HH-mm-ss" unless session_name is set
- the raw download is kept under the "raw" subfolder; the filtered log is written as "log.txt"
- when split_by_channel is non-zero, one file per worker channel (1-32) plus "log master.txt" (channel 55) is produced

FR_LogStreamOptions

Filters and buffer settings passed to FR_StartLogStream.

```
typedef struct {
    const int *channels;
    int       num_channels;
    int       min_level;
    int       max_buffer_bytes;
} FR_LogStreamOptions;
```

Fields:

- channels: channel whitelist; only listed channels are kept. NULL or num_channels == 0 keeps all. Valid ids: 1-32 worker, 55 master, 56-63 AM+.
- num_channels: number of entries in the channels array (required in C, since the pointer carries no length); 0 means keep all channels
- min_level: minimum log level to keep (1 = verbose, 2 = info, 3 = warning, 4 = error)
- max_buffer_bytes: maximum size of the in-memory buffer. Default 2 MB; minimum 4096. On overflow the oldest lines are dropped (the most recent log is always kept).

FR_LogStreamStats

Counters returned in the stats parameter of FR_StopLogStream.

```
typedef struct {
    unsigned long long lines_returned;
    unsigned long long bytes_returned;
    unsigned long long lines_dropped;
    unsigned long long bytes_dropped;
    unsigned long long malformed_lines;
} FR_LogStreamStats;
```

Fields:

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

- lines_returned: number of log lines available in the data buffer
- bytes_returned: size of the data buffer in bytes (the exact length of *data)
- lines_dropped: lines discarded (oldest first) because the buffer reached max_buffer_bytes
- bytes_dropped: bytes discarded for the same reason
- malformed_lines: lines that could not be parsed; still included raw in the data

FR_LogFetchOptions

Options passed to FR_FetchLogFile.

```
typedef struct {
    const wchar_t *log_dir;
    const int     *channels;
    int           num_channels;
    int           min_level;
    int           split_by_channel;
    const wchar_t *session_name;
    const wchar_t *folder_prefix;
    const wchar_t *file_prefix;
    int           include_backup;
} FR_LogFetchOptions;
```

Fields:

- log_dir: parent directory where the session folder is created (required; created if missing)
- channels: channel whitelist applied to the downloaded log; NULL keeps all
- num_channels: number of entries in the channels array; 0 means keep all channels
- min_level: minimum log level to keep (1-4)
- split_by_channel: when non-zero, also produces one file per channel
- session_name: full custom session folder name, used verbatim; NULL auto-generates "SN<serial> - Date ... - Time ...". When set, folder_prefix is ignored.
- folder_prefix: text prepended to the auto-generated folder name; ignored when session_name is set
- file_prefix: text prepended to every generated file name
- include_backup: when non-zero, also downloads the rotated log.txt.1 (if present) and prepends it with a separator
- session_name, folder_prefix and file_prefix must be valid name components: no path separators and none of the characters : * ? " < > |

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

FR_LogFetchResult

Result filled by FR_FetchLogFile. Each path field is a fixed-size buffer of FR_LOG_PATH_MAX characters.

```
typedef struct {
    wchar_t          session_dir[FR_LOG_PATH_MAX];
    wchar_t          raw_file[FR_LOG_PATH_MAX];
    wchar_t          raw_backup_file[FR_LOG_PATH_MAX];
    wchar_t          total_file[FR_LOG_PATH_MAX];
    unsigned long long lines_written;
    unsigned long long malformed_lines;
} FR_LogFetchResult;
```

Fields:

- session_dir: full path of the session folder created
- raw_file: full path of the raw, unmodified log.txt (under the raw subfolder)
- raw_backup_file: full path of the raw log.txt.1, or empty if not requested or not present
- total_file: full path of the combined, filtered log.txt
- lines_written: number of valid, filter-passing lines written
- malformed_lines: lines that could not be parsed; still written raw to the total file

When split_by_channel is set, the per-channel files are written next to total_file in session_dir, named "log channel N.txt" and "log master.txt". Their names are deterministic and are not returned individually.

Integration Workflow

Use this sequence when building a customer application around the DLL:

1. Add FR_COMM.dll, FR_COMM.lib and FR_COMM.h to the project and verify that the build succeeds.
2. Call GetDllVersion() and verify that the returned version matches the expected release.
3. Configure the correct LAN or USB connection parameters.
4. Start from a minimal program that calls FR_OpenCommunication(), sends "#GETENGSTATUS" and reads the answer with FR_GetAnswer(). Validate the answer before adding more operations.
5. Add file transfer only after basic communication is stable.
6. Add project execution after file transfer is confirmed. Use FR_ProjectExecutionProc to receive per-channel results if needed.
7. Add FR_GetLogger only for LAN-based logger endpoints. The logger handle is independent from the command handle.
8. Check every FR_COMM_ERR return value. Call FR_GetLastErrorMessage immediately on any failure. Always close handles before the process exits.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

Integration Examples

All examples below assume a Unicode build (UNICODE defined, the Visual Studio default). String parameters are wchar_t* and string literals use the L"..." prefix. For an ANSI build, remove L prefixes and change wchar_t to char.

Minimal Integration Example

The following example is the smallest recommended integration flow for a customer application:

```
#include "FR_COMM.h"
#include <stdio.h>

int main(void)
{
   FR_COMM_HANDLE h;
   FR_COMM_ERR    rc;
   wchar_t       answer[2048];
   wchar_t       err[512];

   h = FR_OpenCommunication(L"LAN", L"192.168.1.100:1234");
   if (h == NULL) {
      FR_GetLastErrorMessage(err, 512);
      wprintf(L"Open failed: %s\n", err);
      return 1;
   }

   rc = FR_SendCommand(h, L"#GETENGSTATUS");
   if (rc == RET_OK) {
      rc = FR_GetAnswer(h, answer, 2048, 1500);
      if (rc == RET_OK)
         wprintf(L"%s\n", answer);
   }

   FR_CloseCommunication(h);
   return 0;
}
```

Example 1: Command Execution

```
#include "FR_COMM.h"
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

```
#include <stdio.h>

FR_COMM_HANDLE h;
FR_COMM_ERR    rc;
wchar_t        answer[2048];
wchar_t        err[512];

h = FR_OpenCommunication(L"LAN", L"192.168.1.100:1234");
if (h == NULL) {
   FR_GetLastErrorMessage(err, 512);
   wprintf(L"Error: %s\n", err);
   return 1;
}

rc = FR_SendCommand(h, L"#GETENGSTATUS");
if (rc == RET_OK) {
   rc = FR_GetAnswer(h, answer, 2048, 1500);
   if (rc == RET_OK)
      wprintf(L"%s\n", answer);
   else
      wprintf(L"GetAnswer failed: %lu\n", rc);
}

FR_CloseCommunication(h);
```

Example 2: File Transfer and Project Execution

```
#include "FR_COMM.h"
#include <stdio.h>

static FR_COMM_ERR WINAPI on_progress(
   unsigned long file_size, unsigned long file_pos)
{
   if (file_size > 0)
      wprintf(L"transfer: %lu%%\n", file_pos * 100UL / file_size);
   return RET_OK;
}

static void WINAPI on_channel(int channel, int result)
{
   wprintf(L"ch%d: %s\n", channel, result ? L"PASS" : L"FAIL");
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

```
}

int channels[] = { 1, 2, 3 };

FR_SendFile(h, L"YMODEM", L"C:\\FR\\app.frb", L"FRB", on_progress);
FR_GetFile(h, L"YMODEM", L"app.frb", L"C:\\FR", L"FRB", on_progress);
FR_RunProject(h, L"demo.prj", channels, 3, 120000, on_channel);
```

Example 3: Dynamic Memory – "Single-shot" Encryption

This approach uses FR_EncryptData. Each call generates a new AES key and IV, so the 384-byte header must be sent to FlashRunner again for every encrypted payload.

```
#include "FR_COMM.h"
#include <stdlib.h>

unsigned char  input[]     = { 0xAA, 0xBB, 0xCC, 0xDD };
unsigned int   len         = sizeof(input);
unsigned int   aligned_len  = ((len + 15) / 16) * 16;
unsigned char  header[384];
unsigned char *encrypted    = (unsigned char *)malloc(aligned_len);

FR_GetPublicKey(h);

FR_COMM_ERR rc = FR_EncryptData(h, input, len, header, encrypted);
if (rc == RET_OK) {
   /* send header to device: #1*DYNMEMSETHEADER <hex>   */
   /* send encrypted block:  #1*DYNMEMSET2 addr len <hex> */
}

free(encrypted);
```

Dynamic memory encryption:

- The library provides two approaches for encrypting data destined for FlashRunner dynamic memory.
- **FR_EncryptData**: each call generates a new AES key, IV and header. The caller must send the header to FlashRunner via #DYNMEMSETHEADER before each encrypted block.
- **FR_DynMem***: FR_DynMemBeginEncryption generates the header once. FR_DynMemEncryptData can then be called multiple times with CBC chaining. The header is sent only once.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

- Use the session-based approach when encrypting multiple consecutive blocks to avoid regenerating and re-sending the header.

Example 4: Dynamic Memory – "Session-based" Encryption

The header is generated and sent once; multiple data blocks are encrypted sequentially with CBC chaining.

```
#include "FR_COMM.h"
#include <stdlib.h>
#include <string.h>

unsigned char  header[384];
char         header_hex[769];   /* 384 * 2 + 1 */
wchar_t      cmd[800];

unsigned char  blk1[] = { 0xAA, 0xBB, 0xCC, 0xDD };
unsigned char  blk2[] = { 0x11, 0x22, 0x33, 0x44 };
unsigned char  out1[16], out2[16];

FR_GetPublicKey(h);
FR_DynMemBeginEncryption(h, header);

/* convert header to hex and send to device */
FR_HexToAsciiStream(header, 384, header_hex);
swprintf(cmd, 800, L"#1*DYNMEMSETHEADER %hs", header_hex);
FR_SendCommand(h, cmd);

FR_DynMemEncryptData(h, blk1, sizeof(blk1), out1);
FR_DynMemEncryptData(h, blk2, sizeof(blk2), out2);

FR_DynMemResetEncryption(h);
```

Example 5: Real-Time Logger

```
#include "FR_COMM.h"
#include <stdio.h>
#include <windows.h>
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

```
FR_LOGGER_HANDLE logger = NULL;
char           buf[1024];
int            bytes_read;
FR_COMM_ERR     rc;

rc = FR_GetLogger(L"192.168.1.100:1235", &logger);
if (rc != RET_OK) {
   wchar_t err[512];
   FR_GetLastErrorMessage(err, 512);
   wprintf(L"Logger open failed: %s\n", err);
   return 1;
}

for (int i = 0; i < 20; i++) {
   rc = FR_LoggerRead(logger, buf, sizeof(buf), &bytes_read);
   if (rc == RET_OK && bytes_read > 0)
      fwrite(buf, 1, bytes_read, stdout);
   Sleep(100);
}

FR_DisposeLogger(logger);
```

Example 6: Real-Time Log Stream (LAN)

This example captures the filtered log of a run live (LAN only), then reads the accumulated data when the stream is stopped.

```
#include "FR_COMM.h"
#include <stdio.h>
#include <string.h>

FR_COMM_HANDLE     h;
FR_LOGSTREAM_HANDLE stream;
FR_LogStreamOptions opt;
FR_LogStreamStats   stats;
int              channels[] = { 1 };
char            *data = NULL;

h = FR_OpenCommunication(L"LAN", L"192.168.1.100:1234");
if (h == NULL) return 1;
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

```
memset(&opt, 0, sizeof(opt));
opt.channels       = channels;   /* NULL = all channels */
opt.num_channels   = 1;
opt.min_level      = 1;          /* keep everything */
opt.max_buffer_bytes = 2 * 1024 * 1024; /* 2 MB; minimum 4096 */

if (FR_StartLogStream(h, &opt, &stream) == RET_OK) {
   FR_RunProject(h, L"demo.prj", channels, 1, 120000, NULL);

   if (FR_StopLogStream(stream, &data, &stats) == RET_OK) {
      if (data) fputs(data, stdout);
      wprintf(L"%llu lines, %llu dropped\n",
            stats.lines_returned, stats.lines_dropped);
      FR_FreeLogData(data);
   }
}

FR_CloseCommunication(h);
```

Example 7: Download the Log File (LAN or USB)

This example downloads the stored log file, filters it by channel and level, and splits it per channel. It works on both LAN and USB handles.

```
#include "FR_COMM.h"
#include <stdio.h>
#include <string.h>

FR_COMM_HANDLE    h;
FR_LogFetchOptions opt;
FR_LogFetchResult  result;
int               channels[] = { 1, 55 };  /* worker 1 + master */

h = FR_OpenCommunication(L"LAN", L"192.168.1.100:1234");
if (h == NULL) return 1;

memset(&opt, 0, sizeof(opt));
opt.log_dir        = L"C:\\FlashRunner\\Logs";  /* required */
opt.channels       = channels;                  /* NULL = all */
opt.num_channels   = 2;
opt.min_level      = 1;
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

```
opt.split_by_channel = 1;
opt.include_backup   = 1;                    /* also fetch log.txt.1 */

if (FR_FetchLogFile(h, &opt, &result) == RET_OK) {
   wprintf(L"Session: %s\n", result.session_dir);
   wprintf(L"Total:   %s\n", result.total_file);
   wprintf(L"%llu lines written\n", result.lines_written);
}

FR_CloseCommunication(h);
```

Example 8: AM+ Module Integration

```
#include "FR_COMM.h"
#include <stdio.h>

static FR_COMM_ERR WINAPI on_progress(
   unsigned long file_size, unsigned long file_pos) {
   if (file_size > 0)
      wprintf(L"transfer: %lu%%\n", file_pos * 100UL / file_size);
   return RET_OK;
}

FR_COMM_ERR rc;
wchar_t     answer[2048];

/* Send a command to AM+ port 1 and read the response */
rc = FR_SendCommandToAMPlus(h, 1, L"GETENGSTATUS");
if (rc == RET_OK) {
   rc = FR_GetAnswer(h, answer, 2048, 1500);
   if (rc == RET_OK)
      wprintf(L"%s\n", answer);
}

/* Upload a file to AM+ port 2 */
FR_SendFileToAMPlus(h, 2, L"YMODEM", L"C:\\FR\\app.frb", L"FRB", on_progress);

/* Download a file from AM+ port 2 */
FR_GetFileFromAMPlus(h, 2, L"YMODEM", L"app.frb", L"C:\\FR\\dest", L"FRB", on_progress);
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

Error Handling Recommendations

All functions except FR_OpenCommunication, GetDllVersion and FR_GetLastErrorMessage return FR_COMM_ERR. Check every return value.

Recommended approach:

- validate connection parameters before calling FR_OpenCommunication
- check the handle for NULL immediately after FR_OpenCommunication; call FR_GetLastErrorMessage before doing anything else on failure
- call FR_GetLastErrorMessage immediately after any non-zero return — the message is cleared after each retrieval
- always call FR_CloseCommunication before the process exits, even after an error
- use reasonable timeout values for FR_GetAnswer and FR_RunProject

Pattern:

```
h = FR_OpenCommunication(port, settings);
if (h == NULL) {
   FR_GetLastErrorMessage(err, sizeof(err) / sizeof(wchar_t));
   wprintf(L"Error: %s\n", err);
   return 1;
}
/* ... operations ... */
FR_CloseCommunication(h);
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

Troubleshooting

DLL not found at runtime

Possible causes:

- FR_COMM.dll is not in the application directory or the system PATH
- the application was built for a different architecture than the DLL

Recommended checks:

- verify that FR_COMM.dll is in the same folder as the application executable
- confirm that both the DLL and the application are built for the same target architecture (x86 or x64)

Cannot open communication port

Possible causes:

- wrong COM port name or IP address
- invalid baud rate
- FlashRunner not powered or not connected correctly

Recommended checks:

- call FR_GetLastErrorMessage immediately after the NULL return to get the specific error description
- verify the exact COM port name or LAN endpoint
- confirm that the baud rate is one of the supported values (115200 or 3000000)

No answer from FR_GetAnswer

Possible causes:

- command prefix is incorrect
- timeout is too short
- device is connected but not in the expected state

Recommended checks:

- start with L"#GETENGSTATUS"
- increase the timeout value
- verify that the LAN endpoint or COM port is correct

FR_EncryptData or FR_DynMemBeginEncryption fails

Possible causes:

- FR_GetPublicKey was not called first

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

- the device did not return a valid public key

Recommended checks:

- call FR_GetPublicKey in the same session before any encryption call
- confirm that FR_GetPublicKey returned RET_OK
- verify that the output buffers are large enough: header = 384 bytes, data_encrypted = 16-byte-aligned input size

Logger does not produce output

Possible causes:

- wrong logger TCP port
- no logger traffic available at that moment

Recommended checks:

- try a longer sleep interval between FR_LoggerRead calls
- confirm the logger port configured on the FlashRunner side

AM+ function returns an error

Possible causes:

- amplusPort is out of range (valid range: 1–8)
- the AM+ module is not connected or not responding

Recommended checks:

- verify that amplusPort is in the range 1–8
- call FR_GetLastErrorMessage immediately after the error to get the specific error description
- confirm that the AM+ module is powered and connected to the FlashRunner

Log stream or fetch fails

Possible causes:

- FR_StartLogStream was called on a USB handle (real-time streaming is LAN only)
- log_dir is missing or cannot be created (FR_FetchLogFile)
- invalid options: min_level outside 1-4, an invalid channel id, max_buffer_bytes below 4096, or a session/file name with invalid characters
- FR_StopLogStream was passed the command handle instead of the stream handle

Recommended checks:

- for real-time capture use a LAN handle; for USB use FR_FetchLogFile instead
- set a valid, writable log_dir before calling FR_FetchLogFile
- keep min_level in 1-4, max_buffer_bytes >= 4096, and channel ids in 1-32, 55, or 56-63
- pass the stream handle returned by FR_StartLogStream to FR_StopLogStream, and call FR_GetLastErrorMessage for the specific error

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com