SMH Technologies®

Python Library
Quick Start Guide
AN00189

Application Note

Revision 1.1 — Jul 2026

SMH Technologies®

UNIVERSAL PRODUCTION IN-SYSTEM PROGRAMMING

Copyright © 2019 SMH Technologies
DC11841

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` Python wheel in a customer environment.

The library provides a high-level API to communicate with FlashRunner devices for:

- command execution
- file transfer
- project execution
- public key retrieval and encryption
- AM+ module support
- 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 is a Python wheel file:

```
fr_comm_python-1.0.0.2-py3-none-any.whl
```

Supported Environments

- Python: 3.10 or newer
- Operating systems:
  - Windows
  - Linux
  - ARM Linux systems

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

Install the wheel with `pip`:

```
python -m pip install fr_comm_python-1.0.0.2-py3-none-any.whl
```

If the target environment does not already contain the required runtime dependencies, `pip` will install them automatically.

Installation Verification

Verify that the package can be imported and that the logical DLL version is available:

```
python -c "import fr_comm; print('import ok')"
python -c "from fr_comm import FR_GetDllVersion; print(FR_GetDllVersion())"
```


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 public API:

- `FlashRunner`
- `FR_Logger`
- `FR_Comm_Exception`
- `FR_FileType`
- `ProgressHandler`
- `FR_GetDllVersion()`
- `LogStreamOptions, LogStreamResult (used by FlashRunner.FR_StartLogStream() / FR_StopLogStream())`
- `LogFetchOptions, LogFetchResult (used by FlashRunner.FR_FetchLogFile())`

API Summary

API	Purpose	Typical use
`FlashRunner(port, settings)`	Open a high-level communication session	Commands, transfers, project execution, encryption
`FlashRunner.FR_SendCommand()`	Send one command to the device	Request status or trigger one device-side action
`FlashRunner.FR_SendCommand_Ex()`	Send one command to explicit channels	Build channel header automatically from a channel list
`FlashRunner.FR_GetAnswer()`	Read the answer for the previously sent command	Retrieve command output
`FlashRunner.FR_SendFile()`	Upload one file to FlashRunner	Transfer `.frb`, `.prj`, `.lic`, `.log`, `.lib` files
`FlashRunner.FR_GetFile()`	Download one file from FlashRunner	Retrieve files from device storage
`FlashRunner.FR_RunProject()`	Execute one project on one or more channels	Multi-channel project execution
`FlashRunner.FR_GetPublicKey()`	Retrieve the FlashRunner public key	Required before encryption
`FlashRunner.FR_EncryptData()`	Encrypt payload using the device key	Prepare protected binary data for device-side usage

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

`FlashRunner.FR_DynMemBeginEncryption()`	Start a dynamic memory encryption session	Required before FR_DynMemEncryptData()
`FlashRunner.FR_DynMemEncryptData()`	Encrypt one block within the active session	Sequential encryption with CBC chaining
`FlashRunner.FR_DynMemResetEncryption()`	Close the active dynamic memory session	Explicit session teardown
`FlashRunner.FR_SendCommandToAMPlus()`	Send one command to an AMPlus module	Communicate with an AMPlus-connected device
`FlashRunner.FR_SendFileToAMPlus()`	Upload one file to an AMPlus module	Transfer files to an AMPlus-connected device
`FlashRunner.FR_GetFileFromAMPlus()`	Download one file from an AMPlus module	Retrieve files from an AMPlus-connected device
`FR_Logger("LAN", settings)`	Open the real-time logger stream	Read logger output over LAN
`FlashRunner.FR_StartLogStream()`	Start a real-time log stream (LAN only, port 1235)	Begin accumulating the filtered log in memory
`FlashRunner.FR_StopLogStream()`	Stop a stream and return the accumulated log	Retrieve captured log text and counters
`FlashRunner.FR_FetchLogFile()`	Download the persistent log file (LAN or USB)	Save filtered log.txt, optionally split per channel
`FR_GetDllVersion()`	Read the library version	Installation and runtime validation
`FR_Comm_Exception`	Report communication failures	Wrap all operational calls in `try/except`
`ProgressHandler`	Receive transfer callbacks	Optional UI or console progress updates

Connection Configuration

LAN Connection

Use:

• `port="LAN"`

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

• `settings="<ip>:<tcp_port>"`

Example:

```
port = "LAN"
settings = "192.168.1.100:1234"
```

USB or Serial Connection

Use:
• `port` as the serial device name
• `settings` as the baud rate string

Examples:

```
port = "COM3"
settings = "115200"
```

```
port = "/dev/ttyUSB0"
settings = "115200"
```

Supported baud rates are:
• 115200
• 3000000

Supported serial naming formats include:
• Windows: `COM3`, `COM10`, `\\.\COM12`
• Linux: `/dev/ttyUSB0`, `/dev/ttyACM0`, `/dev/ttyS0`
• macOS-style serial names: `/dev/cu.usbserial-*`



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_Exception

Library-specific exception raised when a communication or protocol error occurs.

Attributes:

- `code`: library error code, when available

Typical usage:

```
from fr_comm import FR_Comm_Exception

try:
   pass
except FR_Comm_Exception as exc:
   print(exc)
   print(exc.code)
```

ProgressHandler

Abstract callback interface used for file transfer progress.

Methods to implement:

- `showTransferProgress(self, progress: int)`

Example:

```
from fr_comm import ProgressHandler

class MyProgress(ProgressHandler):
   def showTransferProgress(self, progress: int):
      print(f"transfer: {progress}%")
```

FR_FileType

Enumeration used to identify the file type for transfer operations.

Available values:

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_FileType.FRB`
- `FR_FileType.PRJ`
- `FR_FileType.LIC`
- `FR_FileType.LOG`
- `FR_FileType.LIB`

FlashRunner

Main high-level class used for device communication.

Constructor

```
FlashRunner(port: str, settings: str)
```

Opens the communication channel immediately.

Parameters:

- `port`: connection target, for example `"LAN"` or `"COM3"`
- settings: connection settings, for example "192.168.1.100:1234" or "115200"

Behavior:

- if the connection cannot be opened, the constructor raises `FR_Comm_Exception`
- the object should be reused for related operations inside the same session

FR_CloseCommunication()

```
FR_CloseCommunication() -> None
```

Closes the active communication channel.

FR_SendCommand()

```
FR_SendCommand(command: str) -> None
```

Sends one command to FlashRunner.

Parameter:

- `command`: command string including the required FlashRunner prefix

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

Precondition:

- the `FlashRunner` instance must be open

Postcondition:

- the command is queued on the active session
- the next `FR_GetAnswer()` call reads the answer for this command

FR_SendCommand_Ex()

```
FR_SendCommand_Ex(channels: list[int], command: str) -> None
```

Sends one command to an explicit channel set.

Parameters:

- channels: target channels, for example [1, 2]
- command: command body without channel prefix

Precondition:

- the FlashRunner instance must be open

Postcondition:

- the command is queued on the active session
- the next FR_GetAnswer() call reads the answer for this command

FR_GetAnswer()

```
FR_GetAnswer(timeout_ts: int) -> str
```

Reads the answer for a previously issued command.

Parameter:

- `timeout_ts`: timeout in milliseconds

Return value:

- decoded answer string

Precondition:

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_SendCommand()` must already have been called on the same session

FR_SendFile()

```
FR_SendFile(src_name: str, file_type: FR_FileType, progress: ProgressHandler | None = None) -> None
```

Transfers one file from the host system to FlashRunner.

Parameters:
- `src_name`: full local file path
- `file_type`: `FR_FileType` value
- `progress`: optional callback handler

Preconditions:
- `src_name` must exist and must not be empty
- the session must be open

FR_GetFile()

```
FR_GetFile(src_filename: str, dest_path: str, file_type: FR_FileType, progress: ProgressHandler | None = None) -> None
```

Transfers one file from FlashRunner to the host system.

Parameters:
- `src_filename`: remote file name stored on FlashRunner
- `dest_path`: local destination directory
- `file_type`: `FR_FileType` value
- `progress`: optional callback handler

Preconditions:
- `dest_path` must already exist
- the requested file must already exist on FlashRunner

FR_RunProject()

```
FR_RunProject(project_name: str, channels: list[int], timeout_ms: int, callback: Callable[[int, bool], None] | None = None) -> None
```


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

Executes a project on one or more channels and waits for completion.

Parameters:

- `project_name`: project file name
- `channels`: list of channels, for example `[1, 2, 3]`
- `timeout_ms`: timeout in milliseconds
- callback: optional callable(channel, result) called per completed channel; pass None to disable

Preconditions:

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

FR_GetPublicKey()

```
FR_GetPublicKey() -> None
```

Retrieves the FlashRunner public key and stores it internally for later encryption.

Postcondition:

- the session keeps the imported public key for subsequent `FR_EncryptData()` calls

FR_EncryptData()

```
FR_EncryptData(input: bytearray) -> tuple[bytearray, bytearray]
```

Encrypts the input payload using the public key previously retrieved from FlashRunner.

Parameter:

- `input`: payload to encrypt

Return value:

- tuple `(output, header)`

Important:

- `FR_GetPublicKey()` must be called before `FR_EncryptData()`

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

Notes:

- `output` is the encrypted payload
- `header` contains the encrypted key material required by the device-side decryption flow
- calling FR_EncryptData() while a dynamic memory session is active automatically resets that session

FR_DynMemBeginEncryption()

```
FR_DynMemBeginEncryption() -> bytearray
```

Opens a dynamic memory encryption session.

Return value:

- header: RSA-encrypted session key material — send to FlashRunner via #DYNMEMSETHEADER

Precondition:

- FR_GetPublicKey() must have been called in the same session

Postcondition:

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

FR_DynMemEncryptData()

```
FR_DynMemEncryptData(input: bytearray) -> bytearray
```

Encrypts one data block within the active dynamic memory session.

Parameter:

- input: payload to encrypt

Return value:

- encrypted payload

Precondition:

- FR_DynMemBeginEncryption() must have been 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

Behavior:

- the IV is updated after each call
- this allows sequential blocks to be encrypted without re-establishing the session

FR_DynMemResetEncryption()

```
FR_DynMemResetEncryption() -> None
```

Closes the active dynamic memory encryption session and clears all session state.

FR_SendCommandToAMPlus()

```
FR_SendCommandToAMPlus(amplus_port: int, command: str) -> None
```

Sends one command to an AMPlus module connected to FlashRunner.

Parameters:

- amplus_port: target AMPlus port number (1 to 8)
- command: command body to send to the AMPlus module

Preconditions:

- the FlashRunner instance must be open
- amplus_port must be in the range 1–8

Postcondition:

- the next FR_GetAnswer() call reads the answer returned by the AMPlus module

FR_SendFileToAMPlus()

```
FR_SendFileToAMPlus(src_name: str, file_type: FR_FileType, amplus_port: int, progress:
ProgressHandler | None = None) -> None
```

Transfers one file from the host system to an AMPlus module connected to FlashRunner.

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

Parameters:

- src_name: full local file path
- file_type: FR_FileType value
- amplus_port: target AMPlus port number (1 to 8)
- progress: optional ProgressHandler instance

Preconditions:

- src_name must exist and must not be empty
- amplus_port must be in the range 1–8
- the session must be open

FR_GetFileFromAMPlus()

```
FR_GetFileFromAMPlus(src_filename: str, dest_path: str, file_type: FR_FileType, amplus_port: int, progress: ProgressHandler | None = None) -> None
```

Transfers one file from an AMPlus module to the host system.

Parameters:

- src_filename: remote file name stored on the AMPlus module
- dest_path: local destination directory
- file_type: FR_FileType value
- amplus_port: source AMPlus port number (1 to 8)
- progress: optional ProgressHandler instance

Preconditions:

- dest_path must already exist
- the requested file must already exist on the AMPlus module
- amplus_port must be in the range 1–8

FR_GetDllVersion()

```
FR_GetDllVersion() -> str
```

Returns the library version string.

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_Logger): a live, unfiltered stream over TCP port 1235 (LAN only). You poll it yourself with read() and receive the log as it arrives, keeping full control of buffering and parsing. Use it to display or forward the log continuously while a run is in progress, or when you need every line 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.
- 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_Logger

High-level real-time logger client.

Constructor

```
FR_Logger(port: str, settings: str)
```

Parameters:

- `port`: must be `"LAN"`
- `settings`: logger endpoint in the form `"<ip>:<port>"`

Behavior:

- invalid logger parameters raise `FR_Comm_Exception`

read()

```
read() -> str
```

Reads one logger chunk and returns it as text.

FR_StartLogStream()

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_StartLogStream(options: LogStreamOptions | None = None) -> object
```

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

- `options`: capture filters and buffer settings (see `LogStreamOptions`); pass None to keep the defaults

Return value:

- opaque stream handle to pass to FR_StopLogStream()

Behavior:

- real-time streaming is supported on LAN only; a USB session raises `FR_Comm_Exception`
- no file is written; the filtered log is kept in memory until the stream is stopped
- several streams may run concurrently on the same session; each opens its own connection and buffer
- streams are stopped automatically when `FR_CloseCommunication`() is called on the session

FR_StopLogStream()

```
FR_StopLogStream(stream) -> LogStreamResult
```

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

- `stream`: the handle returned by `FR_StartLogStream`()

Return value:

- `LogStreamResult` containing the captured data and counters

Behavior:

- stopping an unknown or already-stopped stream raises `FR_Comm_Exception`

FR_FetchLogFile()

```
FR_FetchLogFile(options: LogFetchOptions) -> LogFetchResult
```

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:

- `options`: download location and filters (see `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

Return value:

- `LogFetchResult` describing the files produced

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 True, one file per worker channel (1-32) plus “log master.txt” (channel 55) is produced

LogStreamOptions

Filters and buffer settings passed to `FR_StartLogStream`(). Attributes:

- `channels`: list[int] | None — channel whitelist; only listed channels are kept. None (default) keeps all. Valid ids: 1-32 worker, 55 master, 56-63 AM+.
- `min_level`: int — minimum log level to keep (1 = verbose, 2 = info, 3 = warning, 4 = error). Default 1 (keep everything).
- `max_buffer_bytes`: int — 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).

LogStreamResult

Returned by `FR_StopLogStream`(). Attributes:

- `data`: str — the accumulated filtered log as UTF-8 text, with lines separated by newline characters.
- `lines_returned`: int — number of log lines contained in data.
- `bytes_returned`: int — size of data in bytes.
- `lines_dropped`: int — lines discarded because the buffer reached `max_buffer_bytes` (oldest first).
- `bytes_dropped`: int — bytes discarded for the same reason.
- `malformed_lines`: int — lines that could not be parsed; still included raw in data.

LogFetchOptions

Options passed to `FR_FetchLogFile`(). Attributes:

- `log_dir`: str — parent directory where the session folder is created (required; created if missing).
- `channels`: list[int] | None — channel whitelist applied to the downloaded log; None (default) keeps all.
- `min_level`: int — minimum log level to keep (1-4). Default 1.
- `split_by_channel`: bool — when True, also produces one file per channel. Default False.
- `session_name`: str | None — full custom session folder name, used verbatim; None (default) auto-generates “SN<serial> - Date ... - Time ...”. When set, folder_prefix is ignored.
- `folder_prefix`: str | None — text prepended to the auto-generated folder name (e.g. “BatchA - ”). Ignored when session_name is set.

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

- `file_prefix`: str | None — text prepended to every generated file name (e.g. “BatchA_”).
- `include_backup_file`: bool — when True, also downloads the rotated log.txt.1 (if present) and prepends it with a separator. Default False.

`session_name`, `folder_prefix` and `file_prefix` must be valid name components: no path separators and none of the characters : * ? " < > |.

LogFetchResult

Returned by `FR_FetchLogFile()`. Attributes:

- `session_dir`: str — full path of the session folder created.
- `raw_file`: str — full path of the raw, unmodified log.txt (under the raw subfolder).
- `raw_backup_file`: str | None — full path of the raw log.txt.1, or None if not requested or not present.
- `total_file`: str — full path of the combined, filtered log.txt.
- `channel_files`: list[str] — per-channel files (log channel N.txt and log master.txt); empty when split_by_channel was False.
- `lines_written`: int — number of valid, filter-passing lines written.
- `malformed_lines`: int — lines that could not be parsed; still written raw to the total file.

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 Workflow

Use this sequence when building a customer script around the wheel:

1. Install the wheel with `pip`.
2. Verify that `import fr_comm` succeeds and that `FR_GetDllVersion()` returns the expected version.
3. Configure the correct LAN or USB connection parameters.
4. Start from a minimal script that opens `FlashRunner(...)`, sends a simple command such as #GETENGSTATUS, reads the answer, and catches `FR_Comm_Exception`.
5. Add file transfer only after basic communication is stable.
6. Add project execution after basic communication is stable. Use the optional callback parameter only when per-channel result tracking is needed.
7. Add `FR_Logger` only for LAN-based logger endpoints.
8. For log capture, use FR_StartLogStream / FR_StopLogStream over LAN to record a run live, or FR_FetchLogFile (LAN or USB) to download the stored log file afterwards.
9. Keep device operations inside `try/except` blocks and log both the exception message and `exc.code` when available.

Integration Examples

Minimal Integration Example

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

```
from fr_comm import FlashRunner, FR_Comm_Exception

def read_engine_status():
    try:
        with FlashRunner("LAN", "192.168.1.100:1234") as flashrunner:
            flashrunner.FR_SendCommand("#GETENGSTATUS")
            return flashrunner.FR_GetAnswer(1500)
    except FR_Comm_Exception as exc:
        print(f"FR error: {exc}")
        if exc.code is not None:
            print(f"FR code: {exc.code.name}")
        return None

status = read_engine_status()
if status is not None:
    print(status)
```


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 1: Command Execution

```
from fr_comm import FlashRunner, FR_Comm_Exception

try:
   with FlashRunner("LAN", "192.168.1.100:1234") as flashrunner:
      flashrunner.FR_SendCommand("#GETENGSTATUS")
      answer = flashrunner.FR_GetAnswer(1500)
      print(answer)
except FR_Comm_Exception as exc:
   print(f"FR error: {exc}")
```

Example 2: File Transfer and Project Execution

```
from fr_comm import FlashRunner, FR_FileType, ProgressHandler

class MyProgress(ProgressHandler):
   def showTransferProgress(self, progress: int):
      print(f"transfer: {progress}%")

progress = MyProgress()

with FlashRunner("LAN", "192.168.1.100:1234") as flashrunner:
   flashrunner.FR_SendFile(r"C:\FlashRunner\app.frb", FR_FileType.FRB, progress)
   flashrunner.FR_GetFile("app.frb", r"C:\FlashRunner", FR_FileType.FRB, progress)
   flashrunner.FR_RunProject("demo.prj", [1, 2, 3], 120000)
```

Example 3: Public Key Retrieval and Encryption

```
from fr_comm import FlashRunner

with FlashRunner("LAN", "192.168.1.100:1234") as flashrunner:
   flashrunner.FR_GetPublicKey()
   output, header = flashrunner.FR_EncryptData(
      bytearray(bytes.fromhex("AABBCCDD"))
   )
   print(f"encrypted: {len(output)} bytes, header: {len(header)} bytes")
```


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

This example retrieves the FlashRunner public key and encrypts a byte payload using FR_EncryptData(). The returned header and output can be sent to FlashRunner via #DYNMEMSETHEADER and #DYNMEMSET.

Example 4: Dynamic Memory Session-based Encryption

```
from fr_comm import FlashRunner

with FlashRunner("LAN", "192.168.1.100:1234") as flashrunner:
    flashrunner.FR_GetPublicKey()

    # Open session and send header to FlashRunner once
    header = flashrunner.FR_DynMemBeginEncryption()
    flashrunner.FR_SendCommand(f"#DYNMEMSETHEADER {bytes(header).hex().upper()}")
    flashrunner.FR_GetAnswer(1500)

    # Encrypt and send multiple blocks
    for data in [b"\x01\x02\x03\x04", b"\x05\x06\x07\x08"]:
        output = flashrunner.FR_DynMemEncryptData(bytearray(data))
        flashrunner.FR_SendCommand(f"#DYNMEMSET {bytes(output).hex().upper()}")
        flashrunner.FR_GetAnswer(1500)

    flashrunner.FR_DynMemResetEncryption()
```

Example 5: AMPlus Module Integration

```
from fr_comm import FlashRunner, FR_FileType, ProgressHandler

class MyProgress(ProgressHandler):
    def showTransferProgress(self, progress: int):
        print(f"transfer: {progress}%")

with FlashRunner("LAN", "192.168.1.100:1234") as flashrunner:
    # Send a command to AMPlus port 1 and read its answer
    flashrunner.FR_SendCommandToAMPlus(1, "GETENGSTATUS")
    answer = flashrunner.FR_GetAnswer(1500)
    print(answer)
```


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

```
# Upload a file to AMPlus port 2
progress = MyProgress()
flashrunner.FR_SendFileToAMPlus(
    r"C:\FlashRunner\app.frb", FR_FileType.FRB, 2, progress
)

# Download a file from AMPlus port 2
flashrunner.FR_GetFileFromAMPlus(
    "app.frb", r"C:\FlashRunner\dest", FR_FileType.FRB, 2, progress
)
```

Example 6: Real-Time Logger

```
from fr_comm import FR_Logger

logger = FR_Logger("LAN", "192.168.1.100:1235")
try:
    for _ in range(20):
        text = logger.read()
        if text:
            print(text, end="")
finally:
    del logger
```

Example 7: Real-Time Log Stream (LAN)

Capture the FlashRunner log live while a project runs, then read it back when the stream is stopped. No file is written. Real-time streaming is LAN only; on a USB connection use the log file download (Example 8) instead.

```
from fr_comm import FlashRunner, LogStreamOptions

# Optional filters; an empty LogStreamOptions() keeps every channel and level.
options = LogStreamOptions()
options.channels = [1, 55]   # worker channel 1 + master (55); None = all
options.min_level = 1        # 1=verbose, 2=info, 3=warning, 4=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

```
with FlashRunner("LAN", "192.168.1.100:1234") as flashrunner:
   stream = flashrunner.FR_StartLogStream(options)    # start before the activity
   flashrunner.FR_RunProject("demo.prj", [1], 120000)
   result = flashrunner.FR_StopLogStream(stream)      # returns the captured log

print(f"{result.lines_returned} lines, {result.bytes_returned} bytes")
print(result.data)
```

Example 8: Download the Log File (LAN or USB)

Download the log file the FlashRunner stores on its filesystem, filtered and optionally split per channel. The raw download is preserved untouched. Works over both LAN and USB.

```
from fr_comm import FlashRunner, LogFetchOptions

options = LogFetchOptions()
options.log_dir = r"C:\FlashRunner\Logs"   # required; a session subfolder is created here
options.channels = [1, 55]                 # None = all channels
options.min_level = 1                      # 1=verbose, 2=info, 3=warning, 4=error
options.split_by_channel = True            # also write one file per channel

with FlashRunner("LAN", "192.168.1.100:1234") as flashrunner:
   result = flashrunner.FR_FetchLogFile(options)

print(f"session: {result.session_dir}")
print(f"log:     {result.total_file} ({result.lines_written} lines)")
for path in result.channel_files:
   print(f"channel file: {path}")
```

Customer Example Script

A ready-to-use example script is provided with:

- scripts/wheel_customer_example.py

This script is designed as a customer integration reference and should be executed in an environment where the wheel has already been installed.

Useful commands:

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

```
python wheel_customer_example.py version

python wheel_customer_example.py send-command --port LAN --settings 192.168.1.100:1234 --command "#GETENGSTATUS"

python wheel_customer_example.py send-file --port COM3 --settings 115200 --src C:\FlashRunner\app.frb --file-type FRB

python wheel_customer_example.py run-project --port LAN --settings 192.168.1.100:1234 --name demo.prj --channels 1,2,3 --timeout 120000 --with-callback

python wheel_customer_example.py encrypt --port LAN --settings 192.168.1.100:1234 --hex "AA BB CC DD"

python wheel_customer_example.py dynmem-encrypt --port LAN --settings 192.168.1.100:1234 --hex-blocks "AABBCCDD" "EEFF0011"

python wheel_customer_example.py send-command-amplus --port LAN --settings 192.168.1.100:1234 --amplus-port 1 --command "GETENGSTATUS"

python wheel_customer_example.py send-file-amplus --port LAN --settings 192.168.1.100:1234 --amplus-port 1 --src C:\FlashRunner\app.frb --file-type FRB

python wheel_customer_example.py logger --logger-settings 192.168.1.100:1235 --seconds 10

python wheel_customer_example.py version

python wheel_customer_example.py send-command --port LAN --settings 192.168.1.100:1234 --command "#GETENGSTATUS"

python wheel_customer_example.py send-file --port COM3 --settings 115200 --src C:\FlashRunner\app.frb --file-type FRB

python wheel_customer_example.py run-project --port LAN --settings 192.168.1.100:1234 --name demo.prj --channels 1,2,3 --timeout 120000

python wheel_customer_example.py encrypt --port LAN --settings 192.168.1.100:1234 --hex "AA BB CC DD"

python wheel_customer_example.py logger --logger-settings 192.168.1.100:1235 --seconds 10
```


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

```
python wheel_customer_example.py log-stream --port LAN --settings 192.168.1.100:1234 --name demo.prj --channels 1,2,3 --filter-channels 1,3,55 --min-level 2

python wheel_customer_example.py log-stream-multi --port LAN --settings 192.168.1.100:1234 --name demo.prj --channels 1,2,3 --channel-a 1 --channel-b 55

python wheel_customer_example.py log-fetch --port LAN --settings 192.168.1.100:1234 --out-dir .\out --name demo.prj --channels 1 –split

python wheel_customer_example.py log-fetch --port COM3 --settings 115200 --out-dir .\out --filter-channels 1,55 --min-level 2
```

For logger integrations, the example script uses explicit object lifetime management rather than a `with` block, because the public `FR_Logger` API only exposes construction and `read()`.

Error Handling Recommendations

Customer applications should always catch `FR_Comm_Exception` around device operations.

Recommended approach:

- validate connection parameters before calling the library
- keep command and file operations inside `try/except`
- log both the exception message and `exc.code` when available
- use reasonable timeout values for command and project execution calls

Troubleshooting

import fr_comm fails

Possible causes:

- the wheel is not installed in the active Python environment
- the wrong Python interpreter is being used

Recommended checks:

```
python -m pip show fr-comm-python
python -c "import fr_comm; print(fr_comm.__file__)"
```


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

Cannot open communication port

Possible causes:

- wrong serial port name
- invalid baud rate
- USB adapter or device not connected correctly

Recommended checks:

- verify the exact serial port name
- confirm that the baud rate is one of the supported values

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 `#GETENGSTATUS`
- increase the timeout value
- verify that the LAN endpoint or USB port is the correct one

FR_EncryptData() fails

Possible causes:

- `FR_GetPublicKey()` was not called first
- the device did not return a valid public key

Recommended checks:

- call `FR_GetPublicKey()` in the same session before `FR_EncryptData()`
- test the `get-public-key` example first

Logger does not produce output

Possible causes:

- wrong LAN logger port
- no logger traffic available at that moment
- device logger not enabled or not connected to the requested endpoint

Recommended checks:

- verify the logger endpoint independently from the command channel
- try a longer logger read interval
- confirm the logger port configured on the FlashRunner side

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