C# .NET Library
Quick Start Guide
AN00191

Application Note

Revision 1.1 – Jul 2026

SMH Technologies®

UNIVERSAL PRODUCTION IN-SYSTEM PROGRAMMING

Copyright © 2019 SMH Technologies
DC11842

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# .NET library in a customer environment.

The library provides a managed API to communicate with FlashRunner devices for:

- 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 is a .NET DLL assembly:

```
FR_COMM.dll
System.IO.Ports.dll
```

System.IO.Ports.dll is included in this package as a runtime dependency required only by .NET Framework 4.x consumers and COM registration tools (RegAsm.exe). It must be present in the same folder as FR_COMM.dll in those scenarios.
Consumers running on .NET Core / .NET 5 or later do not require this file, as System.IO.Ports is part of the modern .NET runtime. If you consume FR_COMM as a NuGet package, this file is resolved automatically by the package manager.

Supported Environments

- Assembly target: .NET Standard 2.0
- Compatible host runtimes: .NET Framework 4.6.1 or later, .NET 5.0 or later
- Operating systems:
  - Windows

The assembly targets AnyCPU and can be referenced from both x86 and x64 host processes on Windows.

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

Add a reference to FR_COMM.dll in your project.

In Visual Studio: right-click the project References node, select Add Reference, then Browse to FR_COMM.dll.

In an SDK-style .csproj file, add the following item group:

```
<ItemGroup>
  <Reference Include="FR_COMM">
    <HintPath>path\to\FR_COMM.dll</HintPath>
  </Reference>
</ItemGroup>
```

Add a using directive at the top of each source file that calls the library:

```
using FR_COMM;
```

COM Registration

COM registration is only required for unmanaged COM clients such as VB6 or 32-bit VBA applications. .NET consumers do not need this step.

Before running any registration command, ensure FR_COMM.dll and System.IO.Ports.dll are in the same folder.

Register for 64-bit COM clients (run as Administrator):

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\RegAsm.exe "FR_COMM.dll"
/tlb:"FR_COMM.tlb" /codebase /nologo
```


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

To additionally support 32-bit COM clients (VB6, 32-bit VBA), run the following as Administrator:

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe "FR_COMM.dll" /codebase
/nologo
```

To unregister (run as Administrator):

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\RegAsm.exe "FR_COMM.dll" /unregister
/nologo

C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe "FR_COMM.dll" /unregister
/nologo
```

Installation Verification

Verify that the assembly loads and the version is available:

```
using FR_COMM;

ComManager cm = new ComManager();
Console.WriteLine(cm.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 types from the FR_COMM namespace:

- `ComManager`
- `FR_Logger`
- `FR_COMM_ERRORS`
- `TransferProgressHandler`
- `ProjectExecutionHandler`
- `LogStreamOptions`
- `LogStreamResult`
- `LogFetchOptions`
- `LogFetchResult`

API Summary

API	Purpose	Typical use
new ComManager()	Create the library entry point	One instance per application or test
ComManager.FR_OpenCommunication()	Open a LAN or USB communication session	Required before any other operation
ComManager.FR_CloseCommunication()	Close and release a communication session	Always call in a finally block
ComManager.FR_SendCommand()	Send one fully-prefixed command to FlashRunner	Request status or trigger a device action
ComManager.FR_SendCommand_Ex()	Send one command to an explicit set of channels	Multi-channel targeted commands
ComManager.FR_GetAnswer()	Read the answer for the last sent command	Retrieve command output with timeout
ComManager.FR_SendFile()	Upload one file to FlashRunner	Transfer .frb, .prj, .lic, .log, .lib files
ComManager.FR_GetFile()	Download one file from FlashRunner	Retrieve files from device storage

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

ComManager.FR_RunProject()	Execute a project on one or more channels	Multi-channel project execution
ComManager.FR_GetPublicKey()	Retrieve the FlashRunner RSA public key	Required before any encryption call
ComManager.FR_EncryptData()	Encrypt one payload using the device key	Legacy single-shot dynamic memory encryption
ComManager.FR_DynMemBeginEncryption()	Open a dynamic memory encryption session	Required before FR_DynMemEncryptData()
ComManager.FR_DynMemEncryptData()	Encrypt one block within the active session	Sequential encryption with CBC chaining
ComManager.FR_DynMemResetEncryption()	Close the active dynamic memory session	Explicit session teardown
ComManager.FR_GetDllVersion()	Read the library version string	Installation and runtime validation
ComManager.FR_GetLastErrorMessage()	Read the last error description	Call after any non-RET_OK return code
ComManager.FR_HexToAsciiStream()	Convert a byte array to a hex string	Formatting encrypted or binary payloads
ComManager.FR_GetLogger()	Open a real-time LAN logger connection	Read logger output over LAN
ComManager.FR_DisposeLogger()	Close and dispose a logger connection	Always call when logger is no longer needed
ComManager.FR_SendFileToAMPlus()	Upload one file to a specific AM+ module	Transfer files to AM+ ports 1-8
ComManager.FR_GetFileFromAMPlus()	Download one file from a specific AM+ module	Retrieve files from AM+ ports 1-8
ComManager.FR_SendCommandToAMPlus()	Send one command to a specific AM+ module	Direct command to AM+ modules; read response with FR_GetAnswer()

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.Read()	Read one available logger chunk as a string	Polling logger output
ComManager.FR_StartLogStream()	Start a real-time, filtered log capture (LAN only)	Capture a run's log into memory
ComManager.FR_StopLogStream()	Stop a log capture and return the accumulated data	Retrieve the filtered log captured live
ComManager.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():

- `port = "LAN"`
- `settings = "<ip>:<tcp_port>"`

Example:

```
cm.FR_OpenCommunication(out handle, "LAN", "192.168.1.100:1234");
```

USB or Serial Connection

Pass the following arguments to FR_OpenCommunication():

- `port = "COMx" (e.g. "COM3", "COM10")`
- `settings = baud rate as a string`

Examples:

```
cm.FR_OpenCommunication(out handle, "COM3", "115200");
cm.FR_OpenCommunication(out handle, "COM3", "3000000");
```

Supported baud rates are:

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

- 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_ERRORS

Enumeration returned by all ComManager methods.

Key values:

- RET_OK: the call succeeded
- any other value: an error occurred

After any non-RET_OK return, call FR_GetLastErrorMessage() for a human-readable description.

Typical usage:

```
FR_COMM_ERRORS rc = cm.FR_SendCommand(handle, "#GETENGSTATUS");
if (rc != FR_COMM_ERRORS.RET_OK)
   Console.WriteLine("Error: " + cm.FR_GetLastErrorMessage());
```

TransferProgressHandler

Delegate used for file transfer progress callbacks.

Signature:

```
delegate void TransferProgressHandler(int progress);
```

Example:

```
TransferProgressHandler progress = (p) => Console.WriteLine("transfer: " + p + "%");
```

ProjectExecutionHandler

Delegate used to receive per-channel project execution results.

Signature:

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

```
delegate void ProjectExecutionHandler(int channel, bool result);
```

Parameters:

- channel: 1-based channel number that completed
- result: true if the channel executed successfully, false otherwise

Example:

```
ProjectExecutionHandler onDone = (ch, ok) =>
   Console.WriteLine("Channel " + ch + ": " + (ok ? "PASS" : "FAIL"));
```

ComManager

Main library class. Manages one or more communication handles.

Constructor:

```
ComManager cm = new ComManager();
```

A single ComManager instance can hold multiple simultaneous handles.
Each handle corresponds to one independent FlashRunner connection.

FR_OpenCommunication()

```
FR_COMM_ERRORS FR_OpenCommunication(
   out object handle,
   string port,
   string settings);
```

Parameters:

- port: "LAN" for TCP, or "COMx" for USB/serial
- settings: for LAN, "<ip>:<port>"; for USB, the baud rate as a string
- handle: output handle to pass to subsequent calls

Behavior:

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

- opens the communication channel and queries FlashRunner active channels
- returns RET_OK and a valid handle on success
- returns an error and sets null handle on failure

FR_CloseCommunication()

```
FR_COMM_ERRORS FR_CloseCommunication(object handle);
```

Closes and removes the handle. Always call this in a finally block.

FR_SendCommand()

```
FR_COMM_ERRORS FR_SendCommand(object handle, string command);
```

Parameter:

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

Postcondition:

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

FR_SendCommand_Ex()

```
FR_COMM_ERRORS FR_SendCommand_Ex(
   object handle,
   int[] target_channels,
   string command);
```

Parameters:

- target_channels: array of 1-based channel numbers, e.g. new int[] { 1, 2, 3 }
- 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

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_GetAnswer()

```
FR_COMM_ERRORS FR_GetAnswer(
   object handle,
   out string answer,
   int timeout_ms);
```

Parameters:

- answer: output string containing the decoded device response
- timeout_ms: maximum wait time in milliseconds

Precondition:

- FR_SendCommand() or FR_SendCommand_Ex() must have been called on the same handle

FR_SendFile()

```
FR_COMM_ERRORS FR_SendFile(
   object handle,
   string src_filename,
   string dst_path,
   TransferProgressHandler progress);
```

Parameters:

- src_filename: full local path of the file to send
- dst_path: destination directory on FlashRunner — one of "FRB", "PRJ", "LIC", "LOG", "LIB"
- progress: progress delegate, or null to disable callbacks

Preconditions:

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

FR_GetFile()

```
FR_COMM_ERRORS FR_GetFile(
   object handle,
   string src_filename,
   string dst_path,
   string file_type,
   TransferProgressHandler 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

Parameters:

- src_filename: name of the file stored on FlashRunner
- dst_path: local destination directory
- file_type: file type string — one of "FRB", "PRJ", "LIC", "LOG", "LIB"
- progress: progress delegate, 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_RunProject()

```
FR_COMM_ERRORS FR_RunProject(
   object handle,
   string project_name,
   int[] channels,
   int timeout_ms,
   ProjectExecutionHandler callback);
```

Parameters:

- project_name: project file name including the .prj extension
- channels: array of 1-based channel numbers, e.g. new int[] { 1, 2, 3 }
- timeout_ms: maximum time to wait for all channels to complete
- callback: ProjectExecutionHandler called 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_GetPublicKey()

```
FR_COMM_ERRORS FR_GetPublicKey(object 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_ERRORS FR_EncryptData(
   object handle,
   byte[] input,
   out byte[] output,
   out byte[] header);
```

Parameters:

- input: payload bytes to encrypt
- output: AES-256-CBC encrypted payload
- header: RSA-encrypted AES key + IV material for device-side decryption

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 method automatically resets any active DynMem session

Notes:

- output is the encrypted payload
- header contains the encrypted key material required by the device-side decryption flow

FR_DynMemBeginEncryption()

```
FR_COMM_ERRORS FR_DynMemBeginEncryption(
   object handle,
   out byte[] header);
```

Parameters:

- header: 384-byte RSA-encrypted session key material to send to FlashRunner via #DYNMEMSETHEADER

Precondition:

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

Postcondition:

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

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_DynMemEncryptData()

```
FR_COMM_ERRORS FR_DynMemEncryptData(
   object handle,
   byte[] input,
   out byte[] output);
```

Parameters:

- input: data block to encrypt
- output: AES-256-CBC encrypted block

Precondition:

- FR_DynMemBeginEncryption() must have been called first

Behavior:

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

FR_DynMemResetEncryption()

```
FR_COMM_ERRORS FR_DynMemResetEncryption(object handle);
```

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

FR_GetDllVersion()

```
string FR_GetDllVersion();
```

Returns the assembly version string in X.Y.Z.W format.

FR_GetLastErrorMessage()

```
string FR_GetLastErrorMessage();
```

Returns a human-readable description of the last error. The message is cleared after each call. Call this immediately after any non-RET_OK return value to retrieve the error detail.

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()

```
string FR_HexToAsciiStream(byte[] input);
```

Converts a byte array to a lowercase hexadecimal string. Useful for formatting encrypted payloads or binary data.

FR_SendFileToAMPlus()

```
FR_COMM_ERRORS FR_SendFileToAMPlus(
   object handle,
   string src_filename,
   string file_type,
   int amplusPort,
   TransferProgressHandler callback);
```

Parameters:

- src_filename: full local path of the file to send
- file_type: destination directory on AM+ — one of "FRB", "PRJ", "LIC", "LOG", "LIB"
- amplusPort: AM+ USB port number (1-8)
- callback: progress delegate, or null to disable callbacks

Preconditions:

- src_filename must exist and must not be empty
- the handle must be open
- amplusPort must be in the range 1-8

FR_GetFileFromAMPlus()

```
FR_COMM_ERRORS FR_GetFileFromAMPlus(
   object handle,
   string src_filename,
   string dst_path,
   string file_type,
   int amplusPort,
   TransferProgressHandler callback);
```

Parameters:

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

- src_filename: name of the file stored on the AM+
- dst_path: local destination directory
- file_type: source directory on AM+ — one of "FRB", "PRJ", "LIC", "LOG", "LIB"
- amplusPort: AM+ USB port number (1-8)
- callback: progress delegate, 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+
- amplusPort must be in the range 1-8

FR_SendCommandToAMPlus()

```
FR_COMM_ERRORS FR_SendCommandToAMPlus(
   object handle,
   int amplusPort,
   string command);
```

Parameters:

- amplusPort: AM+ USB port number (1-8)
- command: command string (e.g. "FSLS2 FRB 0 1000", "FSRM FRB file.frb")

Behavior:

- sends the command to the specified AM+ module
- call FR_GetAnswer() on the same handle to retrieve the response
- amplusPort must be in the range 1-8

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_Logger.Read): a live, unfiltered 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 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, even 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

- 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_ERRORS FR_GetLogger(
   string ip_address,
   out FR_Logger logger);
```

Parameters:
- ip_address: logger endpoint in the form "<ip>:<port>" (e.g. "192.168.1.100:1235")
- logger: output FR_Logger instance

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

FR_DisposeLogger()

```
FR_COMM_ERRORS FR_DisposeLogger(FR_Logger logger);
```

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

FR_Logger.Read()

```
FR_COMM_ERRORS Read(
   out string buff,
   out int readBytes);
```

Parameters:
- buff: decoded UTF-8 string containing available logger output
- readBytes: number of bytes read (0 if no data was available at the time of the call)

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

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()

```
FR_COMM_ERRORS FR_StartLogStream(
   object handle,
   LogStreamOptions options,
   out object 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 LogStreamOptions); pass a new LogStreamOptions() to keep the defaults
- streamHandle: output opaque handle to pass to FR_StopLogStream()

Behavior:

- real-time streaming is supported on LAN only; a USB handle returns RET_ERR_LOG_CONNECT_FAILED
- 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

FR_StopLogStream()

```
FR_COMM_ERRORS FR_StopLogStream(
   object streamHandle,
   out LogStreamResult result);
```

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

- streamHandle: the handle returned by FR_StartLogStream()
- result: output LogStreamResult containing the captured data and counters

Behavior:

- pass the stream handle returned by FR_StartLogStream(), not the command handle
- stopping an unknown or already-stopped handle returns RET_ERR_LOG_NOT_ACTIVE

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_FetchLogFile()

```
FR_COMM_ERRORS FR_FetchLogFile(
   object handle,
   LogFetchOptions options,
   out 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 LogFetchOptions); LogDir is required
- result: output LogFetchResult describing the files produced

Behavior:

- a session folder is created under LogDir, named “SN<serial> - Date YYYY-MM-DD - Time HH-mm-ss” unless SessionName is set
- the raw download is kept under the “raw” subfolder; the filtered log is written as “log.txt”
- when SplitByChannel 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(). Public fields:

- int[] Channels: channel whitelist; only listed channels are kept. null (default) keeps all. Valid ids: 1-32 worker, 55 master, 56-63 AM+.
- int MinLevel: minimum log level to keep (1 = verbose, 2 = info, 3 = warning, 4 = error). Default 1 (keep everything).
- int MaxBufferBytes: 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(). Public fields:

- string Data: the accumulated filtered log as UTF-8 text, with lines separated by newline characters.
- long LinesReturned: number of log lines contained in Data.
- long BytesReturned: size of Data in bytes.
- long LinesDropped: lines discarded because the buffer reached MaxBufferBytes (oldest first).
- long BytesDropped: bytes discarded for the same reason.
- long MalformedLines: lines that could not be parsed; still included raw in Data.

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

LogFetchOptions

Options passed to FR_FetchLogFile(). Public fields:

- string LogDir: parent directory where the session folder is created (required; created if missing).
- int[] Channels: channel whitelist applied to the downloaded log; null (default) keeps all.
- int MinLevel: minimum log level to keep (1-4). Default 1.
- bool SplitByChannel: when true, also produces one file per channel. Default false.
- string SessionName: full custom session folder name, used verbatim; null (default) auto-generates “SN<serial> - Date ... - Time ...”. When set, FolderPrefix is ignored.
- string FolderPrefix: text prepended to the auto-generated folder name (e.g. “BatchA - ”). Ignored when SessionName is set.
- string FilePrefix: text prepended to every generated file name (e.g. “BatchA_”).
- bool IncludeBackupFile: when true, also downloads the rotated log.txt.1 (if present) and prepends it with a separator. Default false.

SessionName, FolderPrefix and FilePrefix must be valid name components: no path separators and none of the characters : * ? " < > |.

LogFetchResult

Returned by FR_FetchLogFile(). Public fields:

- string SessionDir: full path of the session folder created.
- string RawFile: full path of the raw, unmodified log.txt (under the raw subfolder).
- string RawBackupFile: full path of the raw log.txt.1, or null if not requested or not present.
- string TotalFile: full path of the combined, filtered log.txt.
- string[] ChannelFiles: per-channel files (log channel N.txt and log master.txt); empty when SplitByChannel was false.
- long LinesWritten: number of valid, filter-passing lines written.
- long MalformedLines: 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 application around the library:

1. Add a reference to FR_COMM.dll and verify that new ComManager() succeeds.
2. Verify that FR_GetDllVersion() returns the expected version string.
3. Configure the correct LAN or USB connection parameters.
4. Start from a minimal script that calls FR_OpenCommunication(), sends "#GETENGSTATUS", and reads the answer. 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 ProjectExecutionHandler to receive per-channel results if needed.
7. Add FR_GetLogger only for LAN-based logger endpoints. Logger and command handles are independent — open them separately.
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. Check all return values. Call FR_GetLastErrorMessage() for error details. Always close handles in finally blocks.

Integration Examples

Minimal Integration Example

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

```
using FR_COMM;

static string ReadEngineStatus(string ip)
{
   ComManager cm = new ComManager();
   object handle = null;
   string answer = null;

   FR_COMM_ERRORS rc = cm.FR_OpenCommunication(out handle, "LAN", ip);
   if (rc != FR_COMM_ERRORS.RET_OK)
      return null;

   try
   {
      rc = cm.FR_SendCommand(handle, "#GETENGSTATUS");
      if (rc == FR_COMM_ERRORS.RET_OK)
         cm.FR_GetAnswer(handle, out answer, 1500);
   }
```


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

```
   finally
   {
      cm.FR_CloseCommunication(handle);
   }

   return answer;
}
```

Example 1: Command Execution

```
using FR_COMM;

ComManager cm = new ComManager();
object handle = null;
string answer;
FR_COMM_ERRORS rc;

rc = cm.FR_OpenCommunication(out handle, "LAN", "192.168.1.100:1234");
if (rc != FR_COMM_ERRORS.RET_OK)
{
   Console.WriteLine("Open failed: " + cm.FR_GetLastErrorMessage());
   return;
}

try
{
   rc = cm.FR_SendCommand(handle, "#GETENGSTATUS");
   if (rc == FR_COMM_ERRORS.RET_OK)
   {
      rc = cm.FR_GetAnswer(handle, out answer, 1500);
      if (rc == FR_COMM_ERRORS.RET_OK)
         Console.WriteLine(answer);
   }
}
finally
{
   cm.FR_CloseCommunication(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

Example 2: File Transfer and Project Execution

```
using FR_COMM;

ComManager cm = new ComManager();
object handle = null;
FR_COMM_ERRORS rc;

TransferProgressHandler progress =
   (p) => Console.WriteLine("transfer: " + p + "%");

ProjectExecutionHandler onDone =
   (ch, ok) => Console.WriteLine("ch" + ch + ": " + (ok ? "PASS" : "FAIL"));

rc = cm.FR_OpenCommunication(out handle, "LAN", "192.168.1.100:1234");
if (rc != FR_COMM_ERRORS.RET_OK) { return; }

try
{
   cm.FR_SendFile(handle, @"C:\FlashRunner\app.frb", "FRB", progress);
   cm.FR_GetFile(handle, "app.frb", @"C:\FlashRunner", "FRB", progress);
   cm.FR_RunProject(handle, "demo.prj", new int[] { 1, 2, 3 }, 120000, onDone);
}
finally
{
   cm.FR_CloseCommunication(handle);
}
```

Example 3: Dynamic Memory – Single-shot Encryption

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

```
using FR_COMM;

ComManager cm = new ComManager();
object handle = null;
byte[] output, header;
FR_COMM_ERRORS rc;

rc = cm.FR_OpenCommunication(out handle, "LAN", "192.168.1.100:1234");
```


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

```
if (rc != FR_COMM_ERRORS.RET_OK) { return; }

try
{
   cm.FR_GetPublicKey(handle);
   byte[] input = new byte[] { 0xAA, 0xBB, 0xCC, 0xDD };
   rc = cm.FR_EncryptData(handle, input, out output, out header);
   if (rc == FR_COMM_ERRORS.RET_OK)
   {
      Console.WriteLine("output: " + output.Length + " bytes");
      Console.WriteLine("header: " + header.Length + " bytes");
   }
}
finally
{
   cm.FR_CloseCommunication(handle);
}
```

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.
- 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

This approach uses the FR_DynMem* methods. The header is generated and sent once; multiple data blocks are encrypted sequentially with CBC chaining.

```
using FR_COMM;

ComManager cm = new ComManager();
object handle = null;
byte[] header, block1, block2;
FR_COMM_ERRORS rc;
```


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

```
rc = cm.FR_OpenCommunication(out handle, "LAN", "192.168.1.100:1234");
if (rc != FR_COMM_ERRORS.RET_OK) { return; }

try
{
   cm.FR_GetPublicKey(handle);

   rc = cm.FR_DynMemBeginEncryption(handle, out header);
   if (rc != FR_COMM_ERRORS.RET_OK) { return; }
   // Send header to FlashRunner: #1*DYNMEMSETHEADER <header_hex>
   string headerHex = cm.FR_HexToAsciiStream(header);
   cm.FR_SendCommand(handle, "#1*DYNMEMSETHEADER " + headerHex);

   cm.FR_DynMemEncryptData(handle, new byte[] { 0xAA, 0xBB, 0xCC, 0xDD }, out block1);
   cm.FR_DynMemEncryptData(handle, new byte[] { 0x11, 0x22, 0x33, 0x44 }, out block2);

   cm.FR_DynMemResetEncryption(handle);
}
finally
{
   cm.FR_CloseCommunication(handle);
}
```

Example 5: Real-Time Logger

```
using FR_COMM;

ComManager cm = new ComManager();
FR_Logger logger = null;
FR_COMM_ERRORS rc;

rc = cm.FR_GetLogger("192.168.1.100:1235", out logger);
if (rc != FR_COMM_ERRORS.RET_OK)
{
   Console.WriteLine("Logger open failed: " + cm.FR_GetLastErrorMessage());
   return;
}

try
{
   for (int i = 0; i < 20; i++)
```


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

```
   {
      string buff;
      int readBytes;
      rc = logger.Read(out buff, out readBytes);
      if (rc == FR_COMM_ERRORS.RET_OK && readBytes > 0)
         Console.Write(buff);
      System.Threading.Thread.Sleep(100);
   }
}
finally
{
   cm.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.

```
using FR_COMM;
ComManager cm = new ComManager();
object handle = null;
object stream = null;
FR_COMM_ERRORS rc;
rc = cm.FR_OpenCommunication(out handle, "LAN", "192.168.1.100:1234");
if (rc != FR_COMM_ERRORS.RET_OK) { return; }
try
{
   LogStreamOptions options = new LogStreamOptions();
   options.MinLevel = 1;              // keep everything
   options.Channels = new int[] { 1 };   // channel 1 only (null = all)
   rc = cm.FR_StartLogStream(handle, options, out stream);
   if (rc != FR_COMM_ERRORS.RET_OK) { return; }

   cm.FR_RunProject(handle, "demo.prj", new int[] { 1 }, 120000, null);

   LogStreamResult result;
   cm.FR_StopLogStream(stream, out result);
   Console.WriteLine(result.Data);
   Console.WriteLine(result.LinesReturned + " lines, " + result.LinesDropped + " dropped");
}
finally
```


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

```
{
   cm.FR_CloseCommunication(handle);
}
```

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.

```
using FR_COMM;
ComManager cm = new ComManager();
object handle = null;
FR_COMM_ERRORS rc;
rc = cm.FR_OpenCommunication(out handle, "LAN", "192.168.1.100:1234");
if (rc != FR_COMM_ERRORS.RET_OK) { return; }
try
{
   LogFetchOptions options = new LogFetchOptions();
   options.LogDir = @"C:\FlashRunner\Logs";   // required
   options.MinLevel = 1;
   options.Channels = new int[] { 1, 55 };    // worker 1 + master (null = all)
   options.SplitByChannel = true;
   options.IncludeBackupFile = true;         // also fetch log.txt.1 if present

   LogFetchResult result;
   rc = cm.FR_FetchLogFile(handle, options, out result);
   if (rc == FR_COMM_ERRORS.RET_OK)
   {
      Console.WriteLine("Session: " + result.SessionDir);
      Console.WriteLine("Total:   " + result.TotalFile);
      Console.WriteLine(result.LinesWritten + " lines written");
   }
}
finally
{
   cm.FR_CloseCommunication(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

Error Handling Recommendations

All ComManager methods return FR_COMM_ERRORS. Check every return value and call FR_GetLastErrorMessage() for details when the return is not RET_OK.

Recommended approach:

- validate connection parameters before calling FR_OpenCommunication()
- always open handles in a try block and close them in finally
- check return values for all file and command operations
- call FR_GetLastErrorMessage() immediately after a non-RET_OK return — the message is cleared on the next call
- use reasonable timeout values for FR_GetAnswer() and FR_RunProject()

Pattern:

```
FR_COMM_ERRORS rc = cm.FR_OpenCommunication(out handle, port, settings);
if (rc != FR_COMM_ERRORS.RET_OK)
{
   Console.WriteLine("Error [" + rc + "]: " + cm.FR_GetLastErrorMessage());
   return;
}
```

Troubleshooting

Assembly not found at runtime

Possible causes:

- FR_COMM.dll is not in the application output directory
- the HintPath in the .csproj is incorrect

Recommended checks:

- verify that FR_COMM.dll is in the same folder as the application executable
- check that the Reference HintPath points to the correct DLL location
- rebuild the project and confirm that the DLL is copied to the output folder

Cannot open communication port

Possible causes:

- wrong serial port name or IP address
- invalid baud rate

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 not powered or not connected correctly

Recommended checks:

- verify the exact COM port name or LAN endpoint
- confirm that the baud rate is one of the supported values
- call FR_GetLastErrorMessage() to read the specific error code description

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 COM port is correct

FR_EncryptData() or FR_DynMemBeginEncryption() 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 any encryption call
- confirm that FR_GetPublicKey() returned RET_OK

Logger does not produce output

Possible causes:

- wrong logger TCP 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 polling interval between Read() calls
- 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

Log stream or fetch fails

Possible causes:

- FR_StartLogStream was called on a USB handle (real-time streaming is LAN only)
- LogDir is missing or cannot be created (FR_FetchLogFile)
- invalid options: MinLevel outside 1-4, an invalid channel id, MaxBufferBytes 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 LogDir before calling FR_FetchLogFile
- keep MinLevel in 1-4, MaxBufferBytes >= 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

COM registration fails

Possible causes:

- System.IO.Ports.dll is not in the same folder as FR_COMM.dll
- the registration command was not run as Administrator

Recommended checks:

- verify that System.IO.Ports.dll is alongside FR_COMM.dll before running RegAsm

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