AD FALCON API Manual

Command-Line Paths & Working Directory

Javad Ghorbani

March 26, 2026

Contents

1 Command-Line Paths & Working Directory

This page explains how FALCON resolves **input/output paths** and the **project/work directory** mechanism. This applies to all launch methods:

- Standalone runs (terminal)
- GUI-launched runs
- Other third-party launchers or integrations (e.g., GiD)

1.1 Syntax

1.1.1 Command-Line Arguments

FALCON accepts the following command-line options:

Option	Description
`-p, --project <dir>`	Base project/work directory (used for resolving relative paths)
`-i, --input <file>`	FEM input file path (absolute or relative to `--project`)
`-e, --env <file>`	Explicit path to `.env` file (optional)
`--gid-mesh <file>`	**GiD Postprocess export** mesh output path (`.post.msh`). (The `gid` option name is kept for compatibility.)
`--gid-results <file>`	**GiD Postprocess export** results output path (`.post.res`). (The `gid` option name is kept for compatibility.)
`--vtk-dir <dir>`	**ParaView/VTK export** output directory (used when `PostprocessTool: ParaView` / `VTK`)
`--xdmf-dir <dir>`	XDMF output directory (used when `PostprocessTool: GenericXDMF` / `XDMF` / `HDF5`)
`--mini-root <dir>`	Approved root containing registered standalone mini tools
`--list-mini-tools`	List registered mini tools under the approved root and exit
`--mini-tool <id>`	Run a registered mini tool by id
`--mini-input <file>`	Optional input file to stage into the mini tool run directory
`--no-env`	Ignore `.env` files entirely
`-h, --help`	Display usage information

Positional Input File: If no `--input` option is provided, the first non-option argument is treated as the input file path.

1.1.2 Examples

```
# Specify project directory and input file
falcon --project /path/to/model --input fem_data.txt

# Use input file path (project directory inferred from input file location)
falcon --input /path/to/model/fem_data.txt

# Positional input file argument
falcon /path/to/model/fem_data.txt

# Override GiD Postprocess export paths explicitly
falcon --project /path/to/model --input fem_data.txt --gid-mesh
results.post.msh --gid-results results.post.res

# ParaView/VTK export: write `.vtk` frames into a directory
falcon --project /path/to/model --input fem_data.txt --vtk-dir vtk_out

# Generic XDMF output: write `falcon.xdmf` + binary payload files into a
directory
falcon --project /path/to/model --input fem_data.txt --xdmf-dir xdmf_out

# List registered mini tools
falcon --mini-root /path/to/mini_tools --list-mini-tools

# Run a registered mini tool with an input file
falcon --mini-root /path/to/mini_tools --mini-tool Mit_s1_3 --mini-input
/path/to/input.txt

# Disable .env file usage
falcon --no-env --input fem_data.txt
```

1.2 Path Resolution Order

FALCON resolves paths in the following priority order (highest to lowest):

1.2.1 Project/Work Directory

The project/work directory is used as the base for resolving relative paths. It is determined by:

1. `--project <dir>` command-line argument
2. `ROOT_PATH` from `.env` file (if `.env` is enabled and found)
3. Directory containing the input file (if `--input` or positional input provided)
4. Current working directory (where the solver was launched)

1.2.2 Input File Path

1. `--input <file>` command-line argument
2. `FEM_DATA_RELATIVE_PATH` from `.env` file (resolved relative to project/work directory)
3. `fem_data.txt` in the project/work directory

Path Resolution Rules: - Relative paths in command-line arguments are resolved relative to the resolved project/work directory - Relative paths in `FALCON.env` are resolved relative to the resolved project/work directory (which is often the same as `ROOT_PATH` when `ROOT_PATH` is used to choose that directory) - Absolute paths are used as-is

1.2.3 GiD Postprocess export paths (`.post.msh` / `.post.res`)

These files are the standard GiD Postprocess export format written by FALCON:

- The CLI options use the `gid` prefix because these files are commonly opened in **GiD** (kept for compatibility).
- You can still set these paths even if you plan to parse the files in external scripts.

Mesh Output Path (`.post.msh`)

1. `--gid-mesh <file>` command-line argument
2. `GID_MESH_RELATIVE_PATH` from `.env` file (resolved relative to project/work directory)
3. `<input_stem>.post.msh` in the project/work directory

Results Output Path (`.post.res`)

1. `--gid-results <file>` command-line argument
2. `GID_RESULTS_RELATIVE_PATH` from `.env` file (resolved relative to project/work directory)
3. `<input_stem>.post.res` in the project/work directory

Default Naming: - `<input_stem>` is derived from the input filename without extension (e.g., `fem_data.txt` → `fem_data`) - If the input filename has no stem or is empty, the default stem is `falcon` - Example: `fem_data.txt` → `fem_data.post.msh` and `fem_data.post.res`

Fallback Behavior: - If a specified output path is not writable, FALCON falls back to the default and prints a warning - If the default path is also not writable, FALCON throws an error

1.2.4 ParaView/VTK export output directory

1. `--vtk-dir <dir>` command-line argument
2. `VTK_OUTPUT_DIR_RELATIVE_PATH` from `.env` file (resolved relative to project/work directory)

Notes:

- VTK output is written only when the input file selects `PostprocessTool: ParaView` (alias `VTK`).
- If no VTK output directory is provided, VTK output is skipped.

1.2.5 Generic XDMF export output directory

1. `--xdmf-dir <dir>` command-line argument
2. `XDMF_OUTPUT_DIR_RELATIVE_PATH` from `.env` file (resolved relative to project/work directory)
3. Default: `<input_stem>_xdmf` in the project/work directory

Notes:

- XDMF output is written only when the input file selects `PostprocessTool: GenericXDMF` (aliases: `XDMF`, `HDF5`).
- The directory contains `falcon.xdmf` plus binary payload files (RAW `.bin` by default, or HDF5 if enabled in the build). See Postprocessing Outputs.
- If no XDMF output directory is provided/usable, XDMF output is skipped.

1.3 Registered Mini Tools

FALCON can also launch approved standalone executables such as single-point constitutive-model drivers, benchmark reproduction programs, or other small helper codes. This path is intentionally restricted and does not allow arbitrary executable paths on the command line.

1.3.1 Safety model

- Only tools under an approved root are visible to the CLI
- Each tool must contain a `FALCON.mini` manifest
- The manifest must point to an executable using a relative path
- FALCON canonicalizes the executable path and rejects anything that escapes the approved root
- The tool is launched **without a shell**
- The run happens inside an isolated `mini_runs/<tool>_<timestamp>/` directory under the project/work directory
- If `--mini-input` is provided, the input file is copied into that isolated run directory before launch

1.3.2 Mini-tool root resolution order

The approved root is resolved in the following order:

1. `--mini-root <dir>`
2. `FALCON_MINI_TOOLS_ROOT` from `FALCON.env`
3. `FALCON_MINI_TOOLS_ROOT` from the process environment
4. `mini_tools/` under the resolved project/work directory

If the resolved root does not exist, mini-tool commands fail with an error.

1.3.3 Mini-tool manifest format

Each registered tool directory must contain a file named `FALCON.mini`.

Minimal example:

```
Executable=mits1_main
```

Supported keys:

Key	Required	Meaning
`Executable`	Yes	Relative path to the executable from the tool directory
`DisplayName`	No	Friendly name shown by `--list-mini-tools`
`Args`	No	Extra default arguments appended before any staged input argument
`StagedInputName`	No	Filename to use when copying `--mini-input` into the run directory
`InputMode`	No	`path_arg` (default) or `none`
`PassInputArg`	No	Backward-compatible boolean alias for `InputMode`

1.3.4 Example: MIT-S1 standalone driver

Suppose your approved root is:

```
/path/to/UMATLIB_FALCON/falcon_minis
```

and the MIT-S1 standalone driver lives in:

```
/path/to/UMATLIB_FALCON/falcon_minis/Mit_s1_3
```

Then place this manifest in `Mit_s1_3/FALCON.mini`:

```
DisplayName=MIT-S1 main driver
Executable=mits1_main
StagedInputName=input.txt
```

```
InputMode=path_arg
```

After that:

```
falcon --mini-root /path/to/UMATLIB_FALCON/falcon_minis --list-mini-tools
falcon --mini-root /path/to/UMATLIB_FALCON/falcon_minis --mini-tool Mit_s1_3
--mini-input /path/to/input.txt
```

The second command stages the input in a fresh run folder and calls the tool as:

```
mits1_main input.txt
```

from inside that isolated run directory.

1.4 Optional .env File (Backward Compatible)

If a `FALCON.env` file is present and `--no-env` is not specified, FALCON will use it to override default filenames/locations. This is optional and intended for backwards compatibility.

1.4.1 .env File Search Order

1. Path specified by `--env <file>` command-line argument
2. `FALCON.env` in the project directory (if `--project` specified)
3. `FALCON.env` in the input file directory (if `--input` or positional input provided)
4. `FALCON.env` in the current working directory

The first `.env` file found is loaded; subsequent candidates are not checked.

1.4.2 .env File Format

The `.env` file uses key-value pairs (one per line):

```
ROOT_PATH=/path/to/model
FEM_DATA_RELATIVE_PATH=fem_data.txt
GID_MESH_RELATIVE_PATH=fem_data.post.msh
GID_RESULTS_RELATIVE_PATH=fem_data.post.res
VTK_OUTPUT_DIR_RELATIVE_PATH=vtk_out
XDMF_OUTPUT_DIR_RELATIVE_PATH=xdmf_out
FALCON_MINI_TOOLS_ROOT=/path/to/mini_tools
```

Notes: - `ROOT_PATH` is one way to define the project/work directory when `--project` is not provided - `FEM_DATA_RELATIVE_PATH`, `GID_MESH_RELATIVE_PATH`, `GID_RESULTS_RELATIVE_PATH`, `VTK_OUTPUT_DIR_RELATIVE_PATH`, and `XDMF_OUTPUT_DIR_RELATIVE_`

PATH may be relative to ROOT_PATH or absolute paths - FALCON_MINI_TOOLS_ROOT may be absolute or relative to the resolved project/work directory - If ROOT_PATH is not set (or is not used), relative paths are resolved relative to the project/work directory determined by the command-line and input file location - The .env file is only loaded if --no-env is not specified

1.5 Default Behavior

If paths are not explicitly configured, FALCON uses the following defaults:

- **Input file**: `fem_data.txt` in the project/work directory
- **Mesh output**: `<input_stem>.post.msh` in the project/work directory
- **Results output**: `<input_stem>.post.res` in the project/work directory

1.6 Other File Paths in Input File

For file paths referenced inside the input file (include files, external tables, restart files, UMAT file paths, output CSVs), FALCON follows these rules:

- Most **relative paths** are resolved relative to the project/work directory
- **Include directives** are resolved relative to the directory of the file that contains the include first, then fall back to the project/work directory
- If an **absolute path** is given, it is used as-is
- If a path cannot be found or written, FALCON prints a warning and may attempt a safe fallback

Common postprocessing-related paths defined inside the input file include the output files for targeted CSV outputs (for example `% ReactionForceSum` and `% DOFOutput`); these follow the same path-resolution rules above.

1.7 Related Pages

- GiD postprocessing: see GiD Postprocess export
- Include files and modular inputs: Include & Modular Input Files