AD FALCON API Manual

Step Definitions

Javad Ghorbani

March 26, 2026

Contents

1 Step Definitions **3**
1.1 Syntax . . . 3
1.1.1 Section header . . . 3
1.1.2 Section format . . . 3
1.1.3 Step header rules . . . 3
1.1.4 Parameter line rules . . . 4
1.2 Step graph validation . . . 4
1.3 Key reference . . . 4
1.3.1 Core time control (step-scoped) . . . 4
1.3.2 Solver (step-scoped) . . . 5
1.3.3 Simulation mode and time integration (step-scoped) . . . 5
1.3.4 Output control (step-scoped) . . . 7
1.3.5 Automatic increment control (step-scoped) . . . 8
1.3.6 Explicit/IMEX mass solve options (step-scoped) . . . 9
1.3.7 Element activation (“birth”) options (step-scoped) . . . 10
1.3.8 Postprocessing and GiD controls (analysis-scoped unless noted) . . . 12
1.3.9 Additional flags (mixed scope) . . . 13
1.4 Examples . . . 15
1.4.1 Example 1: Basic fixed-increment step . . . 15
1.4.2 Example 2: Implicit dynamics with generalized-α via `RhoInf` . . . 15
1.4.3 Example 3: Explicit dynamics with mass-solve controls . . . 16

1 Step Definitions

Each `@Step` block defines a simulation phase (time control, solver settings, time integration, output schedule, and selected advanced options). Steps are defined inside the `% Step Definitions` section of the input file.

1.1 Syntax

1.1.1 Section header

FALCON treats section names as case-insensitive and whitespace-insensitive, so these headers are equivalent:

```
% Step Definitions
%StepDefinitions
% step_definitions
% Step-Definitions
```

1.1.2 Section format

```
% Step Definitions
@Step <StepID>
  @@<Key>: <Value...>
  @@<Key>: <Value...>
  ...
@Step <StepID>
  ...
%%%
```

1.1.3 Step header rules

- A step begins with a line whose first token is `@Step` (one or more leading `@` characters are accepted).
- The step ID is the next token and must be an integer; an optional trailing `:` is allowed (so `@Step 1:` works).
- These are equivalent:
  - `@Step 1`
  - `@Step 1:`
  - `@Step: 1`

1.1.4 Parameter line rules

- Inside a step block, each setting is a single line beginning with @ (typically @@), followed by a **key token that ends in :**.
- You may omit the trailing :; FALCON accepts both `@@Key: ...` and `@@Key ...`.
- Step keys are **case-insensitive**.
- Boolean values are parsed loosely (e.g., `Yes/No`, `True/False`, `1/0` are accepted case-insensitively).
- Unrecognized step keys are ignored (typos can be silently ignored).

1.2 Step graph validation

After all steps are parsed, FALCON validates the step graph:

- Step IDs must be unique.
- Exactly one step must have `StartStep: 0` (the root step).
- Every nonzero `StartStep` must reference an existing step.
- A step cannot reference itself.
- Cycles in `StartStep` dependencies are rejected.
- Once a step uses `UL: Yes`, later steps cannot revert to small-deformation.
- `TimeIntegration: Explicit` and `TimeIntegration: IMEX` are supported only with `SimMode: Dynamic`.

1.3 Key reference

Some keys are **step-scoped** (apply only to that `@Step`). Others are **analysis-scoped** (set a mesh/global option; the last value read wins).

1.3.1 Core time control (step-scoped)

Key	Type	Default	Notes
`StartStep:`	int	0	Parent step ID. Exactly one step must use 0 (root step); other steps must reference a prior step.

Key	Type	Default	Notes
`StepTime:`	double	`0.0`	Physical duration of the step. Must be `> 0`.
`NumberSteps:`	int	`1`	Fixed-increment mode: number of substeps (Δt = `StepTime/NumberSteps`).

1.3.2 Solver (step-scoped)

Key	Type	Default	Notes
`SolverType:`	enum	`KSPBICG`	Allowed: `Direct`, `KSPBICG`.
`PreconditionerType:`	enum	`PCILU`	Must be compatible with `SolverType`. Examples: `KSPBICG` → `PCILU`/`PCNONE`.
`MaxIterations:`	int	`100`	Maximum Newton iterations per substep.
`UseModifiedNewton:`	bool	`No`	If `Yes`, reuse the stiffness matrix between some Newton iterations.
`ModifiedNewtonFrequency:`	int	`1`	When `UseModifiedNewton: Yes`, rebuild stiffness every `ModifiedNewtonFrequency` iterations.

1.3.3 Simulation mode and time integration (step-scoped)

Key	Type	Default	Notes
`SimMode:`	enum	`Dynamic`	Allowed: `Dynamic`, `Static`, `Consolidation`. `Consolidation` is allowed only for coupled / fully-coupled analyses.
`Time Integration:` / `TimeIntegrator:`	enum	`Implicit`	Allowed: `Implicit`, `Explicit` (alias `Explicit Lumped`), `IMEX` (alias `Imex`). `Explicit/IMEX` require `SimMode: Dynamic`.
`Theta:`	double	`0.6`	Flow θ-method parameter in `[0.5, 1.0]`. Alias: `FlowTheta:`.
`Enforce Backward:`	bool	`No`	If `Yes` in `Sim Mode: Consolidation`, forces `Theta=1.0` (backward Euler) for flow.

Key	Type	Default	Notes
`RhoInf:`	double	*(unset)*	Generalized-α spectral radius at infinity $\rho\infty\in$[0, 1]. If provided, FALCON maps it to algorithmic parameters and updates the Newmark-like coefficients. If omitted, implicit dynamics uses the classic Newmark setting (α_f=α_m=0).
`AlphaF:` / `AlphaM:`	—	—	Not allowed. These keys are rejected; use `RhoInf:` instead.

1.3.4 Output control (step-scoped)

Key	Type	Default	Notes
`OutputTypes:`	list	*(empty)*	Space/comma/semicolon-separated list of output variables. Built-in output types are matched case-insensitively; UMAT custom variable names should match the registered spelling. See Output Variables.

Key	Type	Default	Notes
`OutputInterval:`	int	—	Legacy shorthand for step-based output: sets `OutputControlType=ByStep` and `OutputControlValue=<interval>`.
`OutputControlType:`	enum	`ByStep`	Allowed: `ByStep`, `ByTime` (case-sensitive).
`OutputControlValue:`	double	`1.0`	For `ByStep`: output every N substeps. For `ByTime`: output every Δt of analysis time.

1.3.5 Automatic increment control (step-scoped)

Enable and tune automatic substepping per step. For full guidance, see Automatic time incrementation.

Key	Type	Default	Notes
`ModernAutoInc:`	bool	`No`	Alias: `AutomaticStepControl:`.
`InitialStepIncrement:`	double	`1.0`	Initial Δt guess (used by auto-substepping).
`MinTimeStep:`	double	`1.0e-6`	Lower bound for Δt.
`MaxTimeStep:`	double	`1.0`	Upper bound for Δt (must be > `MinTimeStep`).
`ErrorTarget:`	double	`1.0e-2`	Target convergence indicator (used for adaptive stepping/retries).

Key	Type	Default	Notes
`MaxSubStep:` / `Maximum Substeps:`	int	`100000`	Safety cap on number of sub-steps within the step.
`MaxRetry:`	int	`30`	Maximum retries after a failed sub-step.
`RetryTol Relaxation:`	bool	`No`	If `Yes`, allows temporary relaxation of the target during retries.
`FnormDamping:`	double	`0.0`	Exponential smoothing of the reported residual indicator (0 = no smoothing, 1 = heavily smoothed).

The following 5 advanced parameters are supported for user tuning (see Automatic time incrementation):

- `HardSafeFactor:`
- `GrowthCooldownMax:`
- `AdjustClampMax:`
- `GrowthCapLimit:`
- `SmoothingHistoricalWeight:`

Other internal auto-increment tuning keys exist but are not intended for user configuration; if provided, they are ignored with a warning.

1.3.6 Explicit/IMEX mass solve options (step-scoped)

These apply when `TimeIntegration: Explicit` or `TimeIntegration: IMEX`.

Key	Type	Default	Notes
`MassSolve:`	enum	`LumpedDiagonal`	Aliases: `Explicit MassSolve:`, `Displacement MassSolve:`. Allowed: `Lumped Diagonal` (aliases: `Diagonal`, `Lumped`), `Row Sum`, `Diagonal Consistent`, `Direct`, `CG`.
`MassSolveTol:`	double	`1e-10`	Alias: `Explicit MassSolveTol:`.
`MassSolveMaxIters:`	int	`500`	Alias: `Explicit MassSolveMaxIters:`.

1.3.7 Element activation (“birth”) options (step-scoped)

These options apply on steps that activate elements (see the element activation section in the manual).

Key	Type	Default	Notes
`ElementBirthStressFree:`	bool	`No`	Alias: `Element Activation StressFree:`. If `Yes`, newly activated elements are reset to stress-free state at activation.

Key	Type	Default	Notes
`ElementBirthInitializeUMAT:`	bool	`No`	Aliases: `ElementActivationInitializeUMAT:`, `ElementBirthInitializeCustomVariables:`, `ElementActivationInitializeCustomVariables:`. If `Yes`, UMAT custom-variable initialization is executed for newly activated elements.
`ElementBirthResetCustomVariables:`	bool	`No`	Aliases: `ElementActivationResetCustomVariables:`, `ElementBirthResetCustomState:`, `ElementActivationResetCustomState:`. If `Yes`, custom state variables are zeroed for newly activated elements.

Key	Type	Default	Notes
`ElementBirthRampTime:`	double	`0.0`	Alias: `ElementActivationRampTime:`. If `>0`, stiffness/internal-force contributions are ramped from 0→1 over this duration.

1.3.8 Postprocessing and GiD controls (analysis-scoped unless noted)

Key	Scope	Default	Notes
`PostprocessTool:`	analysis	`GiD`	Aliases: `PostProcessor:`, `VisualizationTool:`, `Postprocess:`, `PostProcess:`, `PostprocessOutputs:`. Accepts one-or-more outputs (comma/space/semicolon separated). Supported: `GiD`, `ParaView` (alias `VTK`), `GenericXDMF` (aliases: `XDMF`, `HDF5`).
`GIDOutputFormat:`	analysis	`Binary`	Values: `Ascii`/`Text` or `Binary` (currently falls back to ASCII output).

Key	Scope	Default	Notes
`GIDOmitInactiveElements:`	analysis	`No`	Aliases: `GIDOmitInactiveElementsInMesh:`, `GID_OmitInactiveElements:`, `GID_OmitInactiveElementsInMesh:`.
`GIDSplitPerStep:`	analysis	`No`	Aliases: `GIDSplitOnEachStep:`, `GIDSplitFilesPerStep:` (and underscore variants).
`GIDSplitOnElementActivity:`	analysis	`No`	Aliases: `GIDSplitFilesOnElementActivity:` (and underscore variants).
`GIDStartNewFiles:`	step	`No`	Aliases: `GIDStartNewSegment:`, `GIDNewFiles:`, `GIDNewFile:` (and underscore variants). Starts a fresh GiD segment at the beginning of *this* step.
`GIDExcludeElements:`	analysis	*(empty)*	Aliases: `GIDSkipElements:` (and underscore variants). Use `NONE`/`OFF` to clear, `ALL` for all elements, or a list/ranges (e.g. `1 2 10-20`).

1.3.9 Additional flags (mixed scope)

Key	Scope	Default	Notes
`Gravity:`	analysis	*(existing mesh value)*	Sets the gravitational constant used in hydraulic-head calculations; must be >0.
`UL:`	step	`No`	Updated-Lagrangian (large deformation). Once enabled in any step, later steps cannot revert.
`ResetInternalForce:`	step	`No`	At end of step, carry over internal force as reference and reset load history (useful for staged construction).
`EnforceElasticFlag:`	step	*(inactive)*	`All` or element IDs/ranges. Marks specified elements as elastic for this step (plasticity disabled).
`Geostatic:`	step	`No`	Legacy flag (parsed and stored). Use dedicated geostatic workflows described elsewhere in the manual. Alias: `BodyForce:`.

Key	Scope	Default	Notes
`Contact Mechanics:`	step	`No`	Legacy flag (parsed and stored). Contact activation is controlled by contact-pair step ranges rather than this toggle.

1.4 Examples

1.4.1 Example 1: Basic fixed-increment step

```
% Step Definitions
@Step 1:
  @@StartStep: 0
  @@StepTime: 100000
  @@NumberSteps: 100
  @@SolverType: Direct
  @@OutputControlType: ByStep
  @@OutputControlValue: 10
  @@OutputTypes: Displacement EffStress
%%%
```

1.4.2 Example 2: Implicit dynamics with generalized-α via `RhoInf`

```
@Step 2:
  @@StartStep: 1
  @@StepTime: 1.0
  @@NumberSteps: 2000
  @@SimMode: Dynamic
  @@TimeIntegration: Implicit
  @@RhoInf: 0.7
  @@OutputInterval: 100
  @@OutputTypes: Displacement EffStress
%%%
```

1.4.3 Example 3: Explicit dynamics with mass-solve controls

```
@Step 3:
  @@StartStep: 2
  @@StepTime: 0.5
  @@SimMode: Dynamic
  @@TimeIntegration: Explicit
  @@InitialStepIncrement: 1e-4
  @@MinTimeStep: 1e-6
  @@MaxTimeStep: 1e-3
  @@MassSolve: LumpedDiagonal
  @@OutputControlType: ByTime
  @@OutputControlValue: 0.01
  @@OutputTypes: Displacement
%%%
```