SPYForge: SPY Suite Control-Plane Orchestration

Nickname **SPYForge** · Engine repo GraphForge (martialsystems/graphforge) · Martial Systems LLC · Technical note · Not investment advice · Not a forecast model

Verification export: 2026-08-02T06:14:26Z · mode=simulate · graphforge 0.3.1 · result **PASS 13/13**

Reproduction (simulate)

```
cd ~/spy_vol_forecast && PYTHONPATH=. python spyforge/scripts/export_verification.py
```

→ **PASS 13/13** (product-law cases; no market I/O, no registry mutation)

1. Executive summary

Result first: pure simulate mode reports **PASS 13/13** product-law cases. That is mechanical verification of control flow, not forecast MAE.

Main takeaway. GraphForge is a standalone framework to **encode operational product law as typed state graphs** and **mechanically verify** that research/ops pipelines respect those laws (any workflow). **SPYForge** is the **application folder inside the SPY research repo** (`spy_vol_forecast/spyforge/`) that applies GraphForge to this suite.

GraphForge (product). Typed state-graph runtime: channels, reducers, conditional edges, cycles, checkpoints, interrupts. Standalone engine with a small dependency surface — domain apps pin it and run fail-closed product-law verification. Repo: martialsystems/graphforge. Optional LLM nodes (any OpenAI-compatible API) are off the critical path for verification. Does not train models or claim forecast skill.

SPYForge (this application). Encodes SPY product law: architecture freeze, candidate-only retrain, PSI / golden / shadow promotion, multi-agent write isolation. Illegal transitions become failing cases. Weekday production remains run_pipeline.py / Actions until an explicit cutover.

Result. Simulate only (no market I/O, no registry mutation): **PASS 13/13**. PASS means product-law assertions held, not that MAE beat baselines.

Figure 1. Standalone framework + SPY application

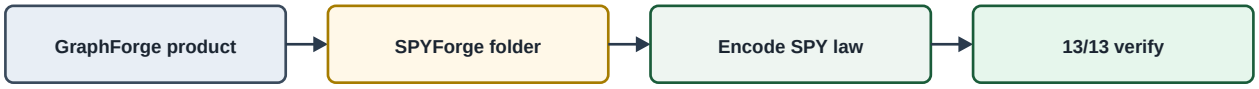

Figure 1. Framework repo is domain-agnostic; SPY packaging and cases live under spyforge/.

2. Architecture

SPY suite control flow is modeled as a small catalog of compiled graphs. Each graph has a state schema (channels + reducers), nodes that return partial updates, and fixed or conditional edges. The runtime is a worklist over a frontier; after each node it checkpoints state and next nodes for resume.

Catalog id	Mirrors	Role
`spy.vol.daily_pipeline`	scripts/run_pipeline.py	Daily ops under freeze
`spy.vol.promotion_gate`	src/promotion_gate.py	PSI → golden → shadow
`spy.protocol.multi_agent`	docs/AGENT_PROTOCOL.md	Write-set ACL
`spy.suite.health`	SPY_SUITE_END_GOAL.md	Module honesty cards
`spy.module.research_lifecycle`	MODULE_EVAL_PATTERN.md	PASS / CLEAN_FAIL process

Relationship to production. Weekday automation remains `run_pipeline.py` / GitHub Actions. SPYForge is the executable specification and test harness first. Live adapters exist behind flags (SPY_VOL_FORECAST_ROOT) but are not the default path. Cutover requires measured parity and a written decision; this note does not authorize cutover.

Simulate guarantees. Default graphs use simulate helpers: no Yahoo/network ingest, no joblib registry writes, no promotion file mutation. That is what the public board reports. Live mode is operator-only and out of scope for the 13/13 table.

Runtime superstep. One worklist step: pop frontier node, run node function, apply channel reducers to the partial update, resolve fixed/conditional edges, write a checkpoint (state + next frontier). Interrupt nodes pause with next re-queued for resume. recursion_limit bounds runaway cycles.

Figure 2. Orchestration superstep (engine)

Figure 2. Shared by all SPYForge graphs; suite law lives in topology, not in the loop.

Figure 3. Daily pipeline linear prefix (always, through predict)

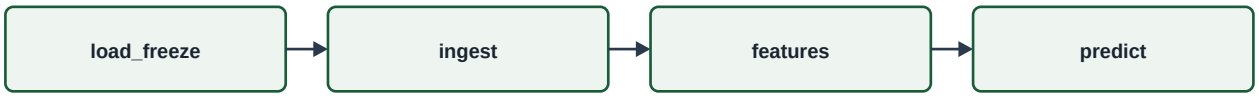

Figure 3. Always-run prefix under freeze. Next step is monitor (Figure 4).

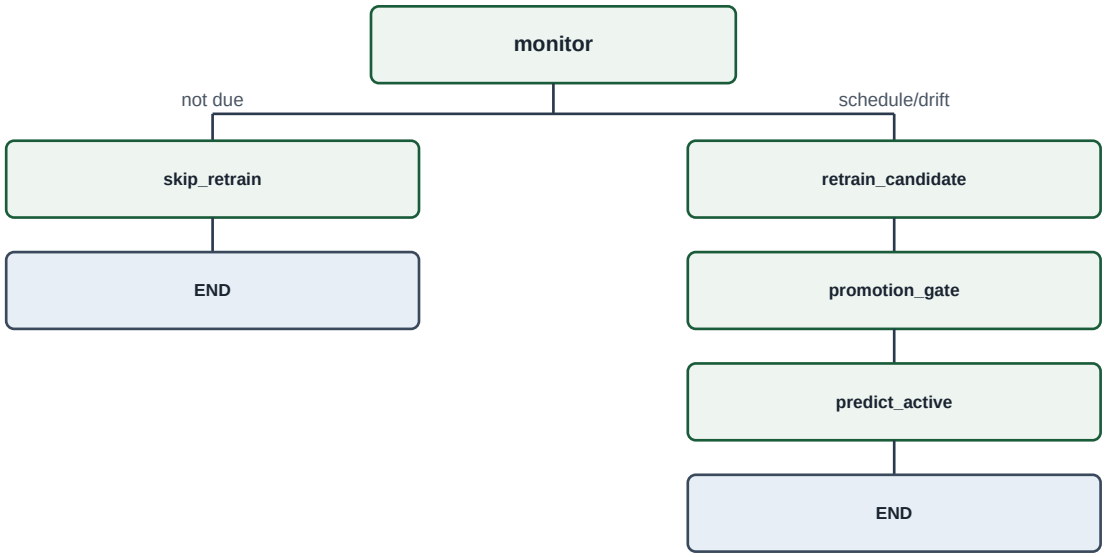

Figure 4. After monitor: skip, or candidate-only retrain then gate then re-predict. One monitor box only (not repeated from Figure 3).

3. Product-law invariants

These are operational-risk rules, not model-training details. Verification cases and state transitions below are the enforceability proof.

- Ops graphs read freezes; they do not rewrite architecture freezes.
- Retrain path sets set_active=False (candidate only).
- PSI CRITICAL blocks promote; unfreeze remains false.
- Site role cannot write frozen_best_config.
- Research PASS does not set public_ok without a separate publish step.

3.1 Freeze is read-only on the daily path

Node `load_freeze` loads a snapshot into channel `freeze` (merge_dicts). No later node in `spy.vol.daily_pipeline` writes freeze files. Channel `unfreeze` is forced false on freeze load, retrain, and gate. **Cases:** daily_skip_retrain, daily_retrain_candidate_only (both require unfreeze=False). **Transition:** start → load_freeze{unfreeze:false, freeze:snapshot} → … → terminal; freeze artifact path never appears as a write target.

3.2 Retrain is candidate-only

When monitor latches should_retrain (or force_retrain), edge monitor→retrain_candidate. Node always emits set_active=False and a candidate_model_id. Activation is only possible later via promotion_gate, never inside retrain. **Case:** daily_retrain_candidate_only (set_active=False, candidate present). **Transition:** monitor{should_retrain:true} → retrain_candidate{set_active:false, candidate_id} → promotion_gate → predict_active.

3.3 PSI CRITICAL blocks promote; no unfreeze

Graph spy.vol.promotion_gate: psi → (psi_ok? golden : block_critical). CRITICAL sets psi_ok=false, routes to block_critical → keep_active, active_model_id stays production, unfreeze=false. SAFE/CAUTION continue; golden or shadow failure also keep_active. **Cases:** psi_safe, psi_caution, psi_critical, golden_fail_blocks. **Transition (CRITICAL):** psi{psi_tier:CRITICAL, psi_ok:false} → block_critical{decision:keep_active, unfreeze:false} → END.

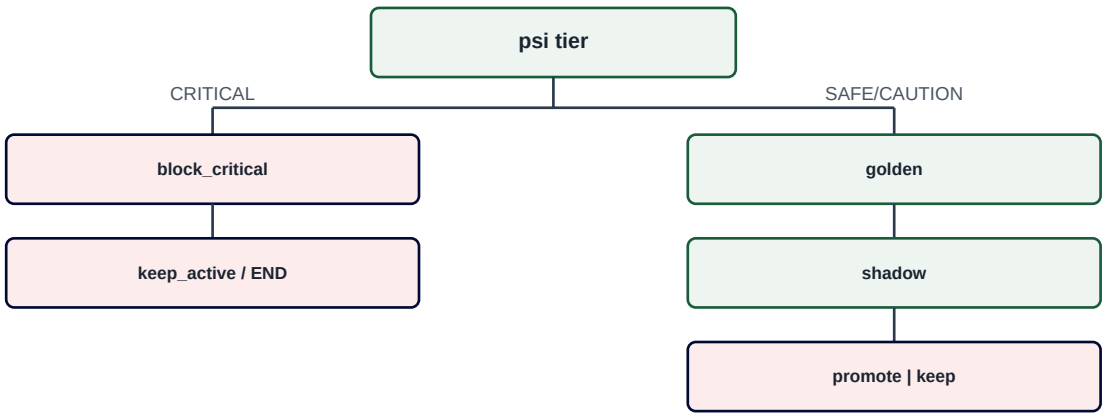

Figure 5. Promotion short-circuit: CRITICAL never reaches golden/shadow/promote.

3.4 Site cannot write freeze

Graph spy.protocol.multi_agent: authorize checks role ACL. frozen_best_config is forbidden for site (and for release). Training may write freeze only if ladder_status=complete. **Cases:** site_cannot_write_freeze (rejected); training_freeze_needs_complete (rejected on partial); training_freeze_on_complete (written). **Transition (site):** authorize{role:site, requested_write:frozen_best_config} → write_allowed:false → reject → END.

3.5 Research PASS is not auto-public

Lifecycle graph may set freeze_decision=PASS while public_ok remains false. Clean fail sets CLEAN_FAIL. Suite health keeps direction as clean_fail, not live. **Cases:** research_pass_not_auto_public, research_clean_fail, suite_vol_live_direction_fail.

Operational reading: a broken invariant is a production incident waiting to happen (silent active swap, provisional site freeze, auto-promote in a regime shift). SPYForge turns those into failing unit cases before weekday Actions is rewritten.

4. Verification results

Harness: `python spyforge/scripts/export_verification.py`. Mode simulate. Headline: **13/13 PASS**, all_pass=True.

Scope of the 13/13 table: these are **SPY product-law cases** (pipeline, promotion, multi-agent ACL, suite honesty, research lifecycle, topology export). Separately, the GraphForge engine package runs its own unit/smoke suite (`python scripts/sanity_engine.py` in `~/graphforge`); those engine tests also pass and are not double-counted here.

What PASS means: the case assertion on final channel state (or topology check) held. It does not mean live MAE skill, profitable trading, or that production Actions was invoked. It means the graph would not take a state transition that violates the encoded rule under the simulated inputs.

Case	OK	Detail	PASS means
`daily_skip_retrain`	PASS	`status=completed_no_retrain unfreeze=False`	No retrain path; unfreeze false; status completed_no_retrain.
`daily_retrain_candidate_only`	PASS	`set_active=False candidate=lgbm_sim_candidate`	Retrain emits candidate; set_active stays false.
`psi_safe`	PASS	`tier=SAFE promoted=True decision=promote`	PSI SAFE can promote when golden+shadow clear; unfreeze false.
`psi_caution`	PASS	`tier=CAUTION promoted=True decision=promote`	PSI CAUTION still continues gates; promote if clear; unfreeze false.
`psi_critical`	PASS	`tier=CRITICAL promoted=False decision=keep_active`	PSI CRITICAL keeps production active; no promote.
`golden_fail_blocks`	PASS	`decision=keep_active`	Golden fail forces keep_active even if PSI safe.
`site_cannot_write_freeze`	PASS	`role=site forbidden write=frozen_best_config`	Site role rejected on frozen_best_config.
`training_freeze_needs_complete`	PASS	`freeze requires ladder_status=complete`	Training freeze denied unless ladder complete.
`training_freeze_on_complete`	PASS	`frozen_best_config`	Training freeze allowed only when ladder complete.
`suite_vol_live_direction_fail`	PASS	`suite=VOL_LIVE_OK failed=['direction_intraday']`	Vol live card; direction remains clean_fail.
`research_pass_not_auto_public`	PASS	`decision=PASS public_ok=False`	PASS freeze decision does not set public_ok.
`research_clean_fail`	PASS	`CLEAN_FAIL`	Failed scorecard to CLEAN_FAIL.
`daily_mermaid`	PASS	`chars=600`	Daily graph Mermaid export contains required nodes.

5. Design decisions and trade-offs

Typed graphs vs free-form agents. A free-form agent can invent tool calls and rewrite freezes. SPYForge forbids that topology: freeze writes are ACL-gated; ops nodes cannot emit unknown channels. The graph is the policy surface.

Typed graphs vs more if-statements. run_pipeline.py already has correct order. Graphs make illegal reorderings and missing short-circuits visible in topology tests, Mermaid exports, and case IDs. They do not replace the production script on day one.

Why freeze is read-only on the daily path. Architecture freeze is a deliberate research decision (FREEZE_DECISION). Monthly ops may retrain weights as candidates; they must not reopen the feature/model ladder by editing freeze from the pipeline agent.

Why site cannot write freeze. AGENT_PROTOCOL splits training (metrics/freeze on complete) from site (scaffold/provisional export). Collapsing those roles is how provisional boards become false production claims.

Why simulate is the public board. Public verification must be reproducible without secrets, Yahoo rate limits, or mutating live registries. Live skill stays on the vol board and predictions ledger.

Small standalone runtime. GraphForge stays a thin control-plane kernel (typed state, reducers, checkpoints, product-law gates). Domain packaging and verification live in SPYForge. Avoid coupling research ops to a large external agent-framework dependency tree.

Choice	Accepted cost	Rejected alternative
Simulate default	Public board is not live skill	Board mutates registry for demo
Candidate-only retrain node	Extra gate hop before active	set_active=True inside retrain
Site freeze forbidden	Site needs training for freezes	Shared write path for all roles
Standalone runtime	Own small control-plane kernel	Heavy external agent stack for one ACL

6. Appendix

6.1 Key channels (daily pipeline, abbreviated)

Channel	Reducer	Role
`freeze`	merge_dicts	Read-only freeze snapshot
`should_retrain`	binary_or latch	Schedule/drift flag
`set_active`	last_value	Must stay false on retrain
`unfreeze`	last_value	Always false in ops graphs
`events`	operator_add	Append-only audit
`candidate_model_id`	last_value	Timestamped candidate

6.2 Topology diagrams (rendered)

Visual topology for the daily pipeline and promotion gate. Same structure as the Mermaid sources in 6.3; rendered as figures for portfolio review.

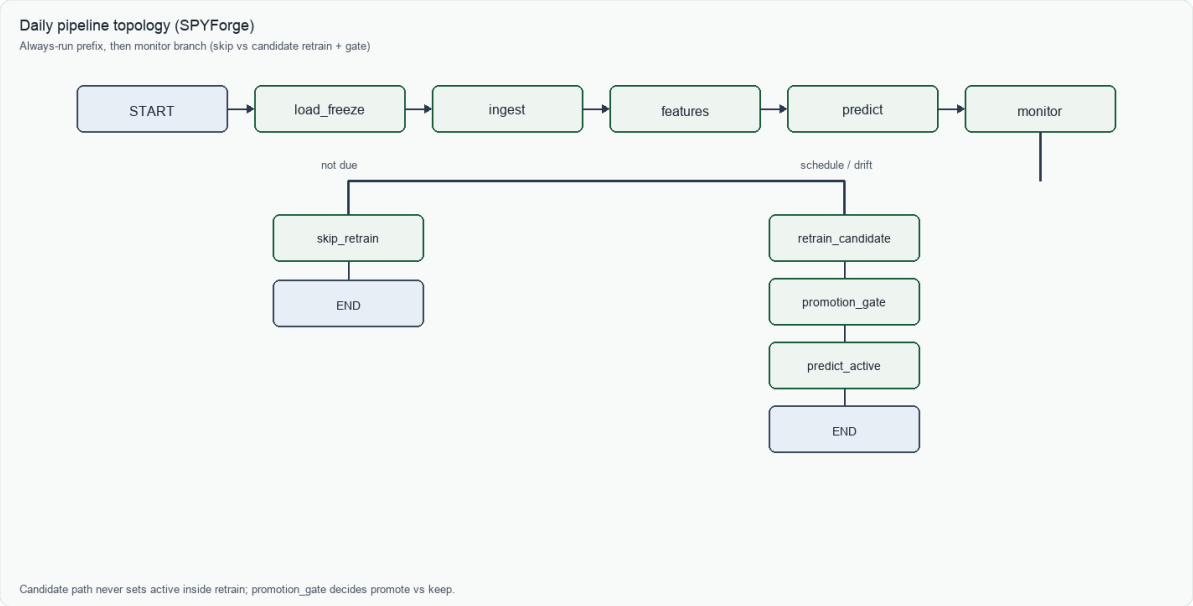

Figure A1. Daily pipeline topology (prefix + retrain branch).

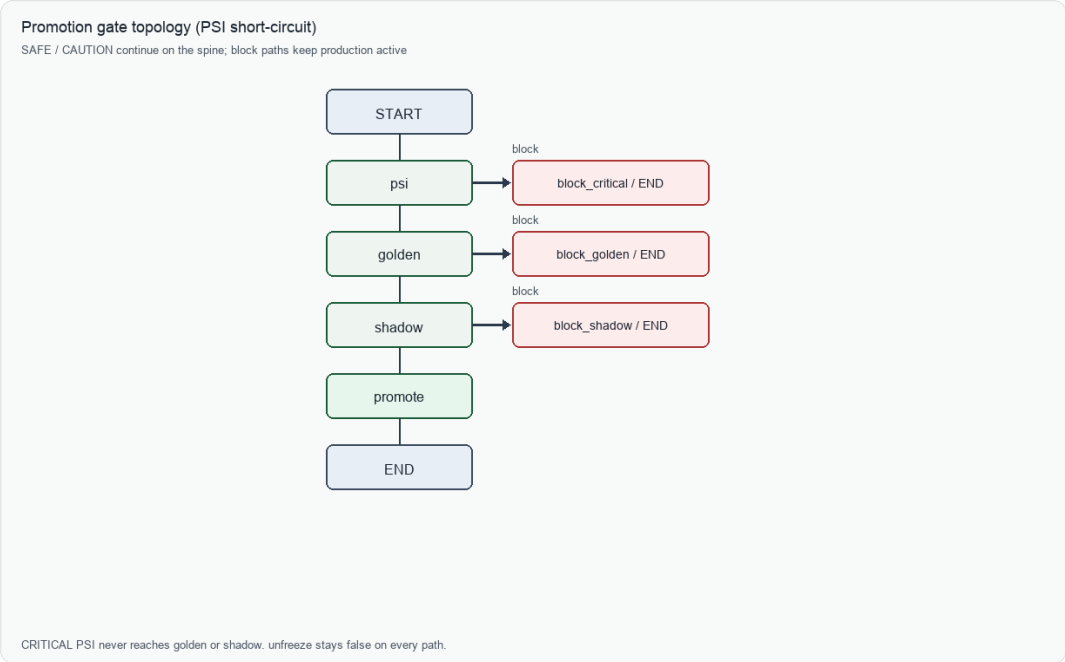

Figure A2. Promotion gate topology with PSI / golden / shadow short-circuits.

6.3 Mermaid source (for editors / copy-paste)

```
flowchart TD
  START --> load_freeze --> ingest --> features --> predict --> monitor
  monitor -->|retrain| retrain_candidate --> promotion_gate --> predict_active --> END
  monitor -->|skip| skip_retrain --> END
```

```
flowchart TD
  START --> psi
  psi -->|continue| golden -->|continue| shadow -->|continue| promote --> END
  psi -->|block| block_critical --> END
  golden -->|block| block_golden --> END
  shadow -->|block| block_shadow --> END
```

6.4 Full reproduce commands

```
cd ~/spy_vol_forecast
export PYTHONPATH=.
python spyforge/scripts/export_verification.py
python spyforge/scripts/sanity_spyforge.py
python spyforge/scripts/build_spyforge_note_pdf.py
# engine (standalone):
cd ~/graphforge && PYTHONPATH=src:. python scripts/sanity_engine.py
```

Public board: /spy-graphforge.html · PDF: /archive/spyforge-control-plane.pdf · Suite pointer: spy_vol_forecast/docs/GRAPHFORGE_CONTROL_PLANE.md · Repo: https://github.com/martialsystems/graphforge