



TRINITY NUCLEAR • TRINITY SENTINEL

TECHNICAL ARCHITECTURE

How the system is built, and why it is built that way.



One process

Flask, server-rendered;
no client framework

One file

SQLite store;
nothing to administer

Three libraries

plus the window or
the server; that is all

Two shapes

Windows desktop, or a
Linux service behind your
proxy

One process, one file, three libraries.

There is no application server to administer, no message queue and no cloud component. The whole shape of the system fits on one page — which is the point, because a reviewer has to be able to hold it in their head.

SHAPE OF THE SYSTEM

LAYER	IMPLEMENTATION	NOTES
Window	pywebview on WebView2 (desktop)	Ships with Windows 10/11. Falls back to a browser if absent. Absent by design on the server.
Front end	nginx with TLS (server)	Authenticates, sets X-Remote-User from its own proof, blanks any identity a client sent.
Server	waitress (shared server)	Bounded worker pool and connection limit; pure Python. The desktop uses the built-in server.
Interface	Flask, server-rendered HTML	No client framework, no build step, no JavaScript bundle. CSRF token injected into every form and fetch.
Engine	Pure Python, standard library	CPM with forward and backward pass, DCMA, EVMS, Monte Carlo, snapshots.
Store	SQLite, one file	Per-person workspace keyed by session; every version of every record kept; one writer at a time to the change log.
Model	Local runtime, localhost, optional	Application is complete without it.
Speech	Kokoro, a child process, optional	Read-aloud in four voices. Opens no socket; a person installs it, Sentinel never downloads it.

Runtime dependencies in full: Flask, openpyxl, python-docx; pywebview for the desktop window; waitress on the shared server. That is the entire third-party surface, and every shipped file is listed with its checksum.

INGEST — FOUR FORMATS, ONE MODEL

Every reader normalizes into one internal structure, so nothing downstream knows or cares which format a schedule arrived in. Every reader also carries the schedule's own data date when the file declares one.

FORMAT	READER	CARRIES
P6 XER	xer_parser	Activities, logic, WBS, calendars, activity codes, last_recalc_date. Drops baselines and resources — a limitation of the format.
P6 PMXML	pmxml	Everything XER carries, plus baselines, resource and cost assignments, the WBS hierarchy and DataDate.
MS Project	mspd	Activities, logic, calendars, extended attributes, StatusDate.
CSV	csv_import	Any tabular schedule; extra columns become activity codes. No data date — inferred or set.

WHERE THE FIDELITY ACTUALLY COMES FROM

P6 stores several finish dates per activity — planned, early, actual. Sentinel applies P6's own precedence rule (actual for complete, then early, then planned) rather than comparing against one field, and reports which it measured against. A fidelity metric that quietly measures nothing is worse than one that fails.

The engine.

CALENDAR-AWARE CRITICAL PATH

Sentinel does not schedule in calendar days. Each activity is scheduled on its own P6 calendar — working week plus holiday exceptions, parsed from the source file — and a duration of D working days finishes on the Dth working day of that calendar. FS, SS, FF and SF are honored with lags converted in working time.

Two passes. The plan pass reproduces P6's stored dates and is what the agreement figure measures — it is byte-identical to the pass that scored 706 of 706. The **stated pass** holds every actual, forecasts remaining work from the data date, records per activity where the remaining figure came from, and runs a backward pass from the Must Finish By date and every finish- or start-no-later-than constraint. Float is therefore signed; negative float is real, and the interface says which pass produced each date.

Hours-per-day is resolved per activity — its own calendar, then the project calendar, then an administrator setting, then a default — and the interface states which applied. A synthesized calendar leaves the field empty rather than assuming eight, so declared is always distinguishable from assumed.

WHY THIS IS NOT A DETAIL

P6 stores durations and lags in hours. On a 4×10 or 9/80 calendar an eight-hour assumption makes every duration read 25% long and every one written back 25% short — silently. v4.27 found exactly this on the write path; hours now resolve per activity.

MODULES

MODULE	RESPONSIBILITY
dcma	The CPM itself — plan pass, stated forecast, backward pass — and the DCMA 14-point assessment.
evms · variance · varlife · budgetlog	EIA-748 earned value per control account and work package; variance narratives; the VAR lifecycle; the budget log, its identity held at every entry.
risk · riskcost · riskreg	Monte Carlo schedule risk (P10/50/80/90, drivers); the cost of schedule risk against contingency; the DOE G 413.3-7A-style register with import/export.
snapshots · history · period	The record of every read; two reads compared, a cause for every move; period series.
compare · generator · estate	Schedule-to-schedule diff producing proposals; the template library and Combine; the declared-link estate pass and its driving path.
program · programgantt · resources · budget · riskquant	The program pages — review, Gantt, resources, funding, register. None writes a date.
edits · actions · import_builder	Proposals, the approval gate with separation of duties, and the scoped P6 import workbook.
venues · evidence748 · opsmeasures · endpoint · visuals	Profiles and the 413.3C gates; the EIA-748-E map; operations; the end milestone and its driver; charts.
report · docs · archive · evidence · knowledge	The report (HTML, PDF, Word); the document register; the period archive; evidence packages; the knowledge pack.
auth · csrf · workspace · presence · backup	Identity (desktop, proxy); cross-site refusal; the per-person workspace; presence; chain-verified backups.

Scaling to a large estate.

Two different scaling questions, with different answers. The engine is untouched by any of this — what changed is where the roll-up gets its numbers, and who is looking at it.

	BEFORE V4.26	NOW
Portfolio or program page	11.6 min, every load	25–55 ms
First read of a new estate	11.6 min	6.8 min, once, in the background
After one schedule is updated	11.6 min	0.7 s
After a normal week — 50 updated	11.6 min	19 s

Finding out what is there is one directory listing. Reading what changed is the expensive part, and it is skipped entirely for any file whose size and modification time are unchanged. Freshness is size and mtime, deliberately not a content hash — hashing means reading every byte of every file, which is the exact cost being avoided, and on a UNC share it is catastrophic.

The derived cache holds 0.8 MB for a 291 MB estate, is keyed on the data date as well as the file, and can be dropped at any moment — the next scan rebuilds it. Nothing is silently omitted: a schedule not yet read comes back as pending and is stated on the page.

ENGINE THROUGHPUT — FULL CPM AND DCMA PASS

ACTIVITIES	FULL ASSESSMENT	PAGE LOAD, WARM
184	0.06 s	instant
10,120	4.2 s	< 0.5 s
20,240	8.7 s	< 0.5 s

MANY PEOPLE — ONE SERVER

Before v4.60 the application held one process-global state: one person switching schedules switched everyone. State is now a per-person **workspace** — installation keys shared, person keys (open project, scope, scenario, working set) per browser session, a system bucket for automated reads. Every audited act bumps a project revision and other sessions re-hydrate on their next request; a person's last schedule is remembered per user. The compute cache is an eight-entry LRU. An automated two-user test (39 checks) proves isolation, freshness without restart, that A cannot approve A's own proposal, and attribution proposed-by / decided-by. **Concurrency is measured in the lab:** a real server process driven by 10, 25 and 50 simulated people at once, with a second process writing the change log on a timer — 17 of 17 checks at each, no cross-talk, chain intact. It found two defects and two scale problems the first time it ran; all four are fixed. The target hardware's own numbers are still to be measured on it.

No locks. Nobody is ever locked out of a schedule. Each person sees who else is on it; each form sends only the fields that person changed; a save that would replace something someone else changed after the page was opened stops and shows both versions. 51 checks cover it.

A MASTER SCHEDULE ACROSS THE ESTATE

Sentinel provides one bar per schedule on a shared timeline, drill-down within any schedule, Combine to build the summary project from its sources, and — from links a person declared — the driving path across projects with estate float per anchor. Since v4.75 the program pages sit over those links: one row per project ranked by what needs attention, the program Gantt with a slip scenario that writes nothing, same-named resources overlaid by month against a declared capacity, and the funding profile by fiscal year. It does not infer inter-project logic or level a resource; and the import of a combined schedule back into P6 is gated on the write-side test. Those are a defined program of work rather than a configuration.

The governed cycle.

This is the architecture that matters most for nuclear work. The sequence is fixed: Read → Compute → Draft → STOP → Decide → Import.

Descriptive changes apply on save

An activity name, a note. Moves no date and no dollar, so it applies and is recorded. Requiring approval for a typo is ceremony without protection.

Scheduling changes become proposals

Duration, dates, percent complete, logic. Changes what the engine computes, so it goes to the gate with an impact preview computed against a throwaway copy.

WHAT THE GATE ENFORCES

Nothing that alters a schedule executes on its own. Ever.

Separation of duties: the proposer cannot approve. On a shared server the decision is stamped with the deciding person's own identity. Refusals are written to the change log.

Approved changes apply to Sentinel's working copy — the source file on disk is never modified.

Export produces a scoped workbook containing only approved fields, for P6's own Import Wizard.

Revert takes changes back out of the working copy and keeps them on record — who proposed, who approved, who reverted.

SCENARIOS

Recovery options can be modeled in parallel and compared on finish, health, negative float and worst slip. Promoting one applies only its own changes, and the alternatives are kept as the record of what was considered.

THE CHANGE LOG

Hash-chained: every event links to the one before it. Rows cannot be removed without the application reporting its own log as tampered — so deletion and purge deliberately never touch it, and the chain still verifies after either. Refusals of every kind are entries: a rejected proposal, a refused self-approval, a wrong passphrase, a request without an identity, a cross-site request. Beside it, every version of every governed record is kept, and a sealed evidence package's manifest fingerprint is written into the chain.

THE SHARED-SERVER PERIMETER

CONTROL	WHERE IT LIVES
Only the front end can reach Sentinel	Service binds 127.0.0.1; the unit's IPAddressAllow=localhost keeps it there.
Identity comes from the front end or not at all	--behind-proxy requires one identity header on every request; anything else is a 401, logged. Serving a network interface without the front end is refused at start (v4.72).
A client cannot name itself	nginx sets X-Remote-User from its own proof and blanks the alternatives — a spoofed header was verified overwritten.
Cross-site requests are refused	Per-session token in every form and fetch; Origin/Referer must match; SameSite=Lax, HttpOnly. Every refusal is a readable page and a log entry.
The process writes three places	Store, output folder, backup folder — ProtectSystem=strict with ReadWritePaths. Daily backup timer verifies the chain first, and every sealed change-log entry writes its chain head to the system journal, which the service cannot edit (v4.72).

Verification.

The verification tools ship inside the package, and one command — `tools/run_battery.py` — runs them all against a copy of the store and writes a V&V record carrying the tested code’s fingerprint and, since v4.73, a requirement-coverage table: 80 stated requirements, each traced to the checks that prove it, each with that run’s outcome. Each build ships its own record, so the numbers below are reproducible by anyone holding the build.

TOOL	WHAT IT PROVES (V5.25)
<code>forecast_acceptance.py</code>	68 hand-worked checks across 11 cases — CPI, SPI, EAC, float and the statused forecast worked on paper first, then asked of the engine.
<code>status • pop • records • evidence • gantt</code>	44, 44, 24, 34 and 21 checks: statusing rules; a cause for every move between periods; nothing a person wrote destroyed; a changed sealed package caught.
<code>route_sweep.py</code>	Renders every GET route plus every schedule through every major page, in process. 67 routes + 39 schedules × 12 pages, both themes, clean.
<code>program • program_gantt • resources • budget • riskquant</code>	75, 57, 55, 59 and 53 checks: a null stays a null; the Gantt writes nothing; capacity declared, never inferred; phasing totals to BAC and EAC.
<code>var_lifecycle • budget_structure • records_disposition • access_review</code>	55, 68, 48 and 38 checks (v4.86–96): every VAR state a person’s dated act; the budget identity held at every entry; a hold blocking disposition, a disposition leaving a certificate; the access review.
<code>dcma_thresholds • risk_spread • assistant_pack • report_numbers</code>	41, 41, 33 and 306 checks (v4.83–98): thresholds recorded with every score; the spread stated everywhere; a tampered pack never loaded; every report figure identical across page, HTML, Word and PDF.
<code>two_user • collab • concurrency • csrf</code>	39 checks between two people; 51 without locks; 17 at 10, 25 and 50 people on a real server; 19 of the CSRF layer.
<code>contrast_sweep.py • xss_sweep.py</code>	WCAG contrast on the composited color of every text element on 28 pages outside the navigation rail, whose layered gradient the sweep cannot resolve (6,400 elements), clean. Stored-XSS payloads planted on 15 surfaces, a hostile schedule file among them, and hunted across 93 pages: none render.
<code>security • route_inventory • injection • advisories</code>	77 checks of the v4.72–84 controls on the real server; every one of 148 POST routes gated and audited or reviewed, a Viewer changing nothing (12); hostile schedule names unable to move the assistant beyond the asker’s role (26); every pinned dependency queried against the OSV advisory feed.
<code>notices • demo_estate • demo_fill • remove_demo</code>	29, 26, 79 and 28 checks: a copyright header on every file; the demos built byte-identical, and taken out for a clean install with a site’s schedules untouched.
<code>assistant_ui • edits • day • newsched • audit</code>	55, 29, 26, 46 and 11 checks (v5.06–12): changes drafted in conversation, never approved from a chat; no claim of work not done; on 25 schedules her figures equal the report’s.
<code>hands • gates • evidence748 • ops • endpoint • visuals</code>	71, 36, 54, 47, 43, 50 checks (v5.14–20): clicks drafted for Confirm, forbidden acts refused; no Critical Decision approved, no EVMS called certified; finishes at the end milestone.
<code>validate_real_schedule.py</code>	Runs on the machine holding a real client file and prints aggregate metrics only. 37 of 37 finishes on a genuine site export.

What a passing battery means.

WHAT A 200 DOES NOT PROVE

A route sweep proves nothing renders an error. It does not prove anything renders right — two chart defects and a page of invisible text have passed every status check in this project's history. The acceptance suite is what proves computation, and it is still being extended. Screenshots of changed views are part of the release, not an optional extra.

WHAT THE RECORD IS FOR

Every build ships the record of the battery that verified it, with the fingerprint of the code it ran against. A reviewer holding the build can re-run the same command and compare. A record that does not match the installed code says so on the Admin page — the application compares the two at start.

REPRODUCE IT YOURSELF

- 1 Open the build**
 Every release is a folder. Beside the application sits `verification/`, holding the V&V record for this exact code — the checks, their outcomes, and the requirement-coverage table.
- 2 Run the battery**
`python tools\run_battery.py` copies the store, runs every shipped check against the copy, and writes a new record carrying the fingerprint of the code it ran against. Nothing in your store is touched.
- 3 Compare**
 Two records for the same fingerprint should agree line by line. Where they do not, the difference is the finding — and it is yours to raise, with the evidence already in hand.



Trinity Nuclear Management Group, Inc.

Project controls for DOE and defense nuclear work — Primavera P6, EVMS, DCMA schedule assurance and risk.

CONTACT

Jeffery S. Gammon, President & CEO
 jgammon@trinitynuclear.ai
 505-205-1307

ONLINE

TrinityNuclear.ai
 trinitynuclear.ai/sentinel

REGISTRATION

UEI JXLHYSWM1AR9
 CAGE 22FY1 · SAM.gov Active
 NAICS 541330 (primary)

CLEARANCE & BUILD

DOE Q cleared
 Trinity Sentinel v5.25