SDPF

Enterprise Case Study

MediTrack

Clinical Operations Platform

A complete demonstration of SDPF applied to a regulated enterprise system — Phase 0 problem definition through multi-component hybrid delivery with full HIPAA compliance, audit ledger, real-time alerting, and signed evidence.

Hamza Abdullah

Software Development Prompting Framework · 2026

Executive Summary

MediTrack is a hypothetical clinical operations platform for a mid-sized hospital network. It manages patient records, clinical task assignment, medication administration, audit logging, real-time alert escalation, and regulatory reporting. It is subject to HIPAA, operates across five hospital sites, and must satisfy both clinical workflow requirements and compliance audit requirements.

This case study demonstrates how SDPF governs the complete delivery of MediTrack — from a single validated problem statement through eight independently specified and verified components, each governed by the appropriate SDPF style, all connected by a master Style 13 Hybrid System Composition specification.

The case study shows how SDPF scales to enterprise complexity without losing the properties that make it valuable at smaller scale: every requirement traced to every test, every boundary explicitly contracted, every component independently evidenced, the complete system auditable from a single system-level problem statement.

System	MediTrack Clinical Operations Platform
Organisation	Regional hospital network — 5 sites, 1,200 clinical staff
Regulatory Framework	HIPAA (45 CFR Parts 160 and 164)
Components	8 independently specified services
SDPF Styles Used	1, 5, 8, 10, 12, 13, 14, 16
Total Specifications	9 (1 master Style 13 + 8 component)
Evidence Packages	9 signed packages — 1 per specification
Primary Delivery Method	AI-assisted with human review and approval at each specification gate

PHASE 0

Problem Identification and Definition

Before any specification work begins, the system-level problem must be formally defined. This problem statement governs every component specification that follows. The desired state in this problem statement is the reference point for every Output Guarantee in every component.

Observation — What Is Happening

Across five hospital sites, clinical staff coordinate patient care through a combination of paper forms, phone calls, WhatsApp messages, and a legacy scheduling system built in 2009. The legacy system has no API, no audit trail, and no integration capability. Clinical tasks — medication administration, procedure scheduling, result notification — are tracked informally. Errors in task handoff between shifts are the leading cause of incident reports. The compliance team cannot produce a complete audit trail for any patient encounter.

CURRENT STATE	*Clinical task coordination across five hospital sites relies on paper, phone, and a legacy system with no API and no audit trail. Task handoff errors between shifts account for 34% of incident reports. The compliance team requires 18 hours to reconstruct a patient audit trail from manual records for regulatory review.*
DESIRED STATE	*All clinical task coordination is managed through a single platform accessible to authorised clinical staff across all five sites. Every patient encounter produces an immutable audit trail retrievable in under 60 seconds. Task handoff between shifts is governed by explicit assignment and completion confirmation.*
GAP	*100% of clinical task coordination is currently untracked digitally against a target of 100% tracked. Audit trail reconstruction time is 18 hours against a target of 60 seconds. Task handoff error rate is 34% of incident reports against a target of under 5%.*
IMPACT	*34% of incident reports attributable to task handoff errors — estimated 2.4 incidents per week across the network. 18-hour audit reconstruction time creates HIPAA compliance exposure on every regulatory review. Inability to integrate with insurance, pharmacy, and lab systems costs an estimated 14 hours per week in manual data re-entry.*

Problem Statement

```
"Clinical task coordination across a five-site hospital network is 100% untracked
 digitally and produces no machine-readable audit trail,
 against a target of full digital tracking with audit trail retrieval under 60 seconds,
 resulting in 2.4 task-handoff incidents per week, 18-hour compliance audit delays,
 and 14 hours per week in manual data re-entry across system boundaries."
```

Validation

Test	Result	Evidence
T-1 Observable	PASS	Incident report rate is measured. Audit reconstruction time is measured. Re-entry hours are tracked.
T-2 Bounded	PASS	Scoped to five named hospital sites, clinical task coordination, and patient encounter audit trails.
T-3 Cause-Free	PASS	No cause stated. Legacy system age is context, not cause.
T-4 Solution-Free	PASS	No platform, technology, or architecture named. Digital tracking is desired state, not solution.

GATE *Problem owner: Chief Clinical Operations Officer. Problem statement passes all four validation tests. Phase 1 may begin.*

1 System Decomposition — Style 13 Master Specification

The first Phase 1 action is to write the master Style 13 — Hybrid System Composition specification. This specification does not describe what any component does internally. It defines the component inventory, the boundary contracts between components, and the dependency graph. Nothing else proceeds until this specification is locked.

Why Style 13 First

The boundary contracts defined in the master specification are the External Contract sections of every component specification. If the master specification is not locked first, the component teams have no agreed interface to build against. They will make assumptions. The assumptions will conflict. The integration will fail.

Locking the master Style 13 specification is the single action that makes parallel component development safe. After it is locked, eight teams can build eight components simultaneously against eight sets of agreed boundary contracts. None of them need to coordinate with each other — the contracts coordinate for them.

The Eight Components

IDENTITY · Style 10 — Compliance-Driven

Authentication, authorisation, and role-based access control for all clinical staff across five sites. Every access decision is logged with timestamp, user identity, resource accessed, and outcome. HIPAA minimum necessary standard enforced at the API layer.

PATIENT · Style 1 — Technical Specification

Patient record management — demographics, encounter history, current care team assignment. Read and write access governed by IDENTITY. Every write produces an immutable audit event consumed by LEDGER.

TASKS · Style 5 — Iterative Refinement

Clinical task lifecycle — creation, assignment, handoff, completion, and escalation. Task templates evolve with clinical workflow requirements. Change tolerance is a contract-level control. Integrates with ALERT for escalation events.

MEDICATION · Style 8 — Constraint-Based

Medication administration record (MAR) management. Hard constraints govern dosage validation, allergy checking, and administration timing. Constraint violation is a CRITICAL failure — no medication record may be created if any constraint fails.

LEDGER · Style 12 — Sovereign Structural Ledger

Immutable audit trail for all patient encounter events across all components. Append-only. Every record references the hash of the previous record. Tamper evidence is a CRITICAL requirement. Audit trail retrieval must satisfy the 60-second desired state from the problem statement.

ALERT · Style 14 — Dynamic Criticality Extension

Real-time clinical alert escalation. Alert criticality levels are bound to runtime state — patient vitals, task overdue status, medication administration windows. Escalation is automated by state transitions. No human escalation paths in the specification.

REPORT · Style 16 — Prompt-Driven Documentation

Regulatory report generation — HIPAA compliance reports, incident summaries, audit extracts. Documentation is a CRITICAL deliverable. Every report is generated from the LEDGER and is itself an immutable record.

GATEWAY · Style 1 — Technical Specification

API gateway — routing, rate limiting, TLS termination, JWT validation. Single entry point for all external callers. Boundary contract with every other component is explicit in the master specification.

The Boundary Contracts

The master specification defines the contract at every component boundary. These become the External Contract sections of the component specifications. An extract:

```
## BOUNDARY CONTRACTS

GATEWAY → IDENTITY
  POST /auth/token
  Input:  { username: string, password: string, site_id: UUID }
  Output: { access_token: JWT, expires_in: int, role: ClinicalRole }
  Error:  INVALID_CREDENTIALS (401), ACCOUNT_LOCKED (403), SITE_MISMATCH (403)

IDENTITY → PATIENT (read)
  GET /patients/{patient_id}
  Required header: Authorization: Bearer {JWT}
```

```
  IDENTITY enforces: role must include PATIENT_READ permission
  IDENTITY logs: access event to LEDGER before returning

PATIENT → LEDGER (write)
  POST /audit/events
  Input:  { event_type, resource_id, resource_type, actor_id,
            timestamp, payload_hash, previous_event_hash }
  Output: { event_id: UUID, chain_position: int }
  Constraint: previous_event_hash must match last recorded event hash.
              LEDGER rejects if chain is broken.

TASKS → ALERT (escalation)
  POST /alerts
  Input:  { alert_type: TaskOverdue|HandoffMissed|EscalationRequired,
            task_id: UUID, patient_id: UUID, criticality: LOW|HIGH|CRITICAL }
  ALERT applies Dynamic Criticality rules — criticality may escalate
  based on patient state at time of alert receipt.
```

The Dependency Graph

Component	Depends On	Consumed By
GATEWAY	—	All external callers
IDENTITY	GATEWAY, LEDGER	PATIENT, TASKS, MEDICATION, REPORT, GATEWAY
PATIENT	IDENTITY, LEDGER	TASKS, MEDICATION, REPORT
TASKS	IDENTITY, PATIENT, LEDGER, ALERT	REPORT
MEDICATION	IDENTITY, PATIENT, LEDGER	TASKS, REPORT
LEDGER	—	IDENTITY, PATIENT, TASKS, MEDICATION, ALERT, REPORT
ALERT	IDENTITY, TASKS, LEDGER	TASKS, REPORT
REPORT	IDENTITY, LEDGER, PATIENT, TASKS, MEDICATION	External regulators, compliance team

GATE

Master Style 13 specification locked. All boundary contracts agreed. All eight component teams may now proceed in parallel. No further coordination is required between component teams — the boundary contracts coordinate for them.

2 Component Specifications

Each component is specified independently using the style appropriate to its domain. The External Contract section of each component specification is taken directly from the boundary contracts in the master Style 13 specification. This is how the master specification governs every component without the component teams needing to communicate with each other.

Component 1 — IDENTITY (Style 10 — Compliance-Driven)

Style 10 is selected because IDENTITY is the HIPAA access control enforcement point. Every access decision must be logged. Every CRITICAL requirement must map to a regulatory clause. Audit evidence is a built-in output.

Specification Section	Content Summary
S-01 Style	Style 10 — Compliance-Driven
S-02 Style Context	HIPAA minimum necessary standard (45 CFR §164.514(d)). Five-site hospital network. JWT-based access control.
S-03 External Contract	REST API. JSON. v1. Bearer token authentication. All endpoints require TLS 1.3.
S-04 Input Contract	POST /auth/token: username, password, site_id. POST /auth/validate: JWT token. GET /auth/permissions: JWT token, resource_type.
S-05 Processing Rules	[CRITICAL] Authenticate credentials against site-scoped user store. [CRITICAL] Log every authentication attempt to LEDGER regardless of outcome. [CRITICAL] Enforce role-based permission check on every resource access. [REQUIRED] Lock account after 5 failed attempts. [REQUIRED] JWT expires in 8 hours.
S-06 Output Guarantees	Successful auth: HTTP 200, JWT with role claims, site scope, expiry. Failed auth: HTTP 401 INVALID_CREDENTIALS. Locked: HTTP 403 ACCOUNT_LOCKED.
S-07 Exception Handling	INVALID_CREDENTIALS (401), ACCOUNT_LOCKED (403), SITE_MISMATCH (403), TOKEN_EXPIRED (401), INSUFFICIENT_PERMISSIONS (403), LEDGER_WRITE_FAILURE (500 — CRITICAL).
S-08 TVG	Python 3.11.x, FastAPI 0.104.x, python-jose 3.3.x, bcrypt 4.x. All verified against live environment.
S-11 Regulatory Mapping	[CRITICAL] Authentication logging → HIPAA §164.312(b) Audit Controls. [CRITICAL] Role enforcement → HIPAA §164.514(d) Minimum Necessary. [CRITICAL] Account lockout → HIPAA §164.312(d) Person Authentication.
S-12 Audit Evidence	Every authentication event produces an audit record in LEDGER with: event_type, user_id, site_id, resource_requested, outcome, timestamp, IP address.

Specification Section	Content Summary
S-13 Compliance Verification	Auditor can retrieve all access events for any user within any date range from LEDGER in under 60 seconds. Regulatory mapping table available in REPORT.

Component 2 — LEDGER (Style 12 — Sovereign Structural Ledger)

Style 12 is selected because the audit trail is the foundation of HIPAA compliance for MediTrack. Mutation is not an operational concern — it is a specification violation. Every record is immutable from the moment it is written. The chain of hashes makes tampering detectable.

Specification Section	Content Summary
S-01 Style	Style 12 — Sovereign Structural Ledger
S-02 Style Context	Immutable audit ledger for all clinical events across five sites. HIPAA §164.312(b) compliance requirement. Append-only. No record may be modified or deleted under any circumstance.
S-03 External Contract	REST API. JSON. v1. Internal service-to-service only. Not exposed through GATEWAY to external callers.
S-04 Input Contract	POST /audit/events: event_type, resource_id, resource_type, actor_id, site_id, timestamp, payload_hash, previous_event_hash. GET /audit/events: filters by patient_id, actor_id, date_range, event_type.
S-05 Processing Rules	[CRITICAL] Verify previous_event_hash matches the hash of the last recorded event in the chain. Reject if chain is broken. [CRITICAL] Compute SHA-256 hash of the new event payload before writing. [CRITICAL] Write is atomic — either the event is written and the chain is extended, or the write is rejected entirely. [CRITICAL] No update or delete operations exist. Any attempt to call a non-existent update endpoint returns HTTP 405 METHOD_NOT_ALLOWED.
S-06 Output Guarantees	Successful write: HTTP 201, event_id UUID, chain_position integer, event_hash SHA-256. Retrieval: HTTP 200, array of events in chronological order with chain verification status.
S-07 Exception Handling	CHAIN_BROKEN (409) — previous_event_hash does not match last recorded hash. DUPLICATE_EVENT (409) — event_id already exists. INVALID_HASH (400) — payload_hash format invalid. METHOD_NOT_ALLOWED (405) — any mutation attempt.
S-08 TVG	PostgreSQL 15.x with pgcrypto extension. SHA-256 hash function verified. Append-only table constraints verified. No UPDATE or DELETE triggers present.
S-11 Immutability Contract	No record in the audit ledger may be modified, deleted, or overwritten under any circumstance including database maintenance, system recovery, or administrative action.
S-12 Provenance Requirements	Every event record contains: SHA-256 hash of its own payload, SHA-256 hash of the previous event, chain position integer, write timestamp, and writing

Specification Section	Content Summary
	service identity.
S-13 Tamper Evidence	Chain integrity is verifiable by recomputing the hash chain from event 1 to the current event. Any single modified event breaks the chain from that position forward. REPORT generates chain integrity verification reports on demand.

Component 3 — MEDICATION (Style 8 — Constraint-Based)

Style 8 is selected because medication administration is governed by hard constraints — dosage limits, allergy interactions, timing windows — where any constraint violation is a patient safety failure. Constraint satisfaction is the primary verification criterion. If any constraint fails, the medication record is not created.

Specification Section	Content Summary
S-01 Style	Style 8 — Constraint-Based
S-02 Style Context	Medication administration record management. Patient safety constraints are CRITICAL. No medication record may be created if any constraint fails.
S-03 External Contract	REST API. JSON. v1. IDENTITY-authenticated. All writes trigger LEDGER audit event.
S-04 Input Contract	POST /medications/administer: patient_id, medication_code, dosage_mg, route, administered_by, administration_time. GET /medications/{patient_id}/mar: date_range.
S-05 Processing Rules	[CRITICAL] Dosage constraint: administered dosage must not exceed the maximum safe dosage for the medication code and patient weight. Reject if exceeded. [CRITICAL] Allergy constraint: medication code must not appear in the patient's known allergy list. Reject if present. [CRITICAL] Timing constraint: administration_time must be within the prescribed administration window. Reject if outside window. [CRITICAL] Duplicate constraint: the same medication code must not have been administered within the minimum dosing interval. Reject if interval not elapsed. [REQUIRED] All four constraint checks are executed before any write. Partial writes are not permitted.
S-06 Output Guarantees	Successful administration: HTTP 201, MAR record with mar_id, all constraint check results, LEDGER event_id. Constraint failure: HTTP 400 with specific constraint violation code and measured value.
S-07 Exception Handling	DOSAGE_EXCEEDED (400) — measured value and limit included. ALLERGY_CONFLICT (400) — conflicting allergy code included. TIMING_VIOLATION (400) — window start/end times included. DUPLICATE_ADMINISTRATION (400) — last administration time included. ALL are CRITICAL — no administration proceeds if any fires.
S-08 TVG	Medication code database version verified. Patient weight field present and non-null verified. Allergy database connectivity verified. All four constraint functions tested with boundary values.
S-11 Hard Constraint	DOSAGE_MAX: patient weight in kg × medication-specific mg/kg ceiling.

Specification Section	Content Summary
Register	ALLERGY_CHECK: exact match on medication code against patient allergy list. TIMING_WINDOW: ±30 minutes of prescribed administration time. DOSING_INTERVAL: medication-specific minimum hours between doses.
S-12 Constraint Priority Order	ALLERGY_CHECK supersedes all others. DOSAGE_MAX supersedes TIMING_WINDOW. DOSING_INTERVAL supersedes TIMING_WINDOW.

Component 4 — ALERT (Style 14 — Dynamic Criticality Extension)

Style 14 is selected because clinical alert criticality is not static. A missed medication administration is LOW criticality at 10 minutes overdue, HIGH at 30 minutes, and CRITICAL at 60 minutes. A task handoff failure escalates based on patient acuity. The escalation rules are bound to runtime state and governed by the Conflict Resolution Protocol — no human escalation paths exist in the specification.

Specification Section	Content Summary
S-01 Style	Style 14 — Dynamic Criticality Extension
S-02 Style Context	Real-time clinical alert escalation. Criticality levels are runtime state-dependent. All escalation is automated. No human escalation paths.
S-03 External Contract	REST API + WebSocket push. JSON. v1. IDENTITY-authenticated. Push delivery to clinical staff mobile devices.
S-04 Input Contract	POST /alerts: alert_type, resource_id, patient_id, initial_criticality, triggering_state. WebSocket subscribe: /alerts/stream with JWT.
S-05 Processing Rules	[CRITICAL] Evaluate patient acuity score at alert receipt time. Escalate initial_criticality if acuity exceeds threshold. [CRITICAL] Apply state-escalation map to determine final criticality. [CRITICAL] Deliver alert to all staff with ALERT_RECEIVE permission for the patient's site within 30 seconds. [REQUIRED] Re-evaluate criticality every 15 minutes until alert is acknowledged. Escalate if unacknowledged thresholds are crossed.
S-06 Output Guarantees	Alert created: HTTP 201, alert_id, final_criticality after escalation rules applied, delivery_targets list. Push delivery: WebSocket message within 30 seconds of alert creation.
S-07 Exception Handling	PATIENT_NOT_FOUND (404), ESCALATION_CONFLICT (409 — two CRITICAL escalation rules produce conflicting outcomes — Conflict Resolution Protocol applied automatically), DELIVERY_TIMEOUT (500 — if push delivery not confirmed within 30 seconds, log to LEDGER as ALERT_DELIVERY_FAILURE).
S-08 TVG	WebSocket server connectivity verified. Patient acuity scoring function tested with known inputs. Push delivery round-trip time measured and below 30-second threshold.
S-11 Criticality Levels	LOW: Informational. Staff notified. No escalation timer. HIGH: Response required within 30 minutes. Escalation timer starts. CRITICAL: Immediate response required. All available staff notified. Senior clinician alerted.

Specification Section	Content Summary
S-12 State-Escalation Map	MedicationOverdue: +10min→LOW, +30min→HIGH, +60min→CRITICAL. TaskHandoffMissed: +15min→LOW, +45min→HIGH (HIGH if patient acuity≥7). VitalsAlert: immediate→CRITICAL regardless of acuity.
S-13 De-escalation	Alert de-escalates to ACKNOWLEDGED when any authorised staff member confirms receipt. Escalation timer stops. LEDGER event written.

Component 5 — TASKS (Style 5 — Iterative Refinement)

Style 5 is selected because clinical task templates evolve continuously with clinical workflow requirements. New task types are added as clinical protocols change. The change tolerance mechanism is a contract-level control — every task template version is numbered, every change is classified, and the version history is part of the specification.

Specification Section	Content Summary
S-01 Style	Style 5 — Iterative Refinement
S-02 Style Context	Clinical task lifecycle management. Task templates evolve with clinical protocols. Current version: 1.0. Change tolerance is explicit.
S-03 External Contract	REST API. JSON. v1. IDENTITY-authenticated. Integrates with ALERT for escalation, LEDGER for audit, PATIENT for assignment validation.
S-04 Input Contract	POST /tasks: task_type (from template registry), patient_id, assigned_to, due_time, priority. POST /tasks/{id}/handoff: handoff_to, handoff_notes. POST /tasks/{id}/complete: completed_by, completion_notes.
S-05 Processing Rules	[CRITICAL] Task creation validates task_type against current template registry version. [CRITICAL] Handoff records the accepting clinician and timestamp. Both assigning and accepting clinicians are recorded. [CRITICAL] Completion records completed_by, completion_time, and completion_notes. A completed task cannot be re-opened. [REQUIRED] Overdue tasks (past due_time without completion) trigger ALERT with alert_type=TaskOverdue.
S-06 Output Guarantees	Task created: HTTP 201, task_id, template_version used, all fields. Handoff: HTTP 200, updated task with both clinician records. Complete: HTTP 200, final task record.
S-07 Exception Handling	INVALID_TASK_TYPE (400), PATIENT_NOT_FOUND (404), ASSIGNEE_NOT_AUTHORISED (403), TASK_ALREADY_COMPLETE (409), HANDOFF_SAME_CLINICIAN (400).
S-11 Change Classification Rules	BREAKING: modifying a CRITICAL task type field or removing a task type. SIGNIFICANT: adding a new task type, adding a REQUIRED field. MINOR: updating task type descriptions, modifying OPTIONAL fields.
S-12 Revision Procedures	BREAKING changes require clinical protocol review and re-verification. SIGNIFICANT changes require testing on non-production site before rollout. MINOR changes require review only.
S-13 Version History	v1.0: Initial task types — MedicationAdministration, ProcedureScheduling, ResultNotification, ShiftHandoff.

Component 6 — REPORT (Style 16 — Prompt-Driven Documentation)

Style 16 is selected because regulatory reports are not optional outputs — they are CRITICAL deliverables with the same status as operational code. Every report is generated from LEDGER data, is itself an immutable record, and must be available on demand. Documentation generation is part of the implementation, not a separate process.

Specification Section	Content Summary
S-01 Style	Style 16 — Prompt-Driven Documentation
S-02 Style Context	Regulatory report generation from LEDGER audit trail. Reports are CRITICAL deliverables. HIPAA audit extract must be available in under 60 seconds.
S-03 External Contract	REST API. JSON + PDF output. v1. IDENTITY-authenticated. Compliance officer role required.
S-04 Input Contract	GET /reports/hipaa-audit: patient_id, date_range, report_format (JSON|PDF). GET /reports/incident-summary: date_range, site_id. GET /reports/chain-integrity: verification of LEDGER hash chain.
S-05 Processing Rules	[CRITICAL] HIPAA audit report retrieves all LEDGER events for the patient within the date range and renders in the requested format within 60 seconds. [CRITICAL] Every report generation is itself written to LEDGER as a REPORT_GENERATED event. [REQUIRED] Incident summary aggregates task handoff failures, medication constraint violations, and alert escalation events. [REQUIRED] Chain integrity report recomputes the full LEDGER hash chain and reports the first broken link if any.
S-06 Output Guarantees	HIPAA audit: HTTP 200, structured report with all events, patient demographics, care team history. Within 60 seconds. Incident summary: HTTP 200, aggregated counts with drill-down links. Chain integrity: HTTP 200, VALID or BROKEN with position of first break.
S-07 Exception Handling	PATIENT_NOT_FOUND (404), DATE_RANGE_INVALID (400), REPORT_TIMEOUT (504 — if 60-second SLA exceeded), CHAIN_INTEGRITY_FAILURE (200 — not an error — returns position of first broken link for investigation).
S-11 Documentation Structure	HIPAA Audit Report: patient header, date range, chronological event list with actor, resource, outcome, timestamp. Incident Summary: by site, by event type, trend over time. Chain Integrity: per-event hash verification status.
S-12 Generation Triggers	HIPAA Audit: on demand by compliance officer. Incident Summary: automated weekly generation + on demand. Chain Integrity: automated daily generation + on demand.
S-13 Documentation Verification	Every generated report is stored in LEDGER as an immutable record. A report cannot be altered after generation. The generation event is auditable.

Component 7 — PATIENT (Style 1 — Technical Specification)

Style 1 is selected because patient record management is a deterministic system with fully definable I/O. Every field is typed, every operation is explicit, every output is shaped. The CRITICAL requirement is that every write operation produces a LEDGER audit event before returning. This is the mechanism that guarantees the audit trail is complete.

KEY CRITICAL REQUIREMENT

[CRITICAL] Every write to a patient record — create, update, care team assignment — shall write an audit event to LEDGER before the write response is returned to the caller. If the LEDGER write fails, the patient record write is rolled back and LEDGER_WRITE_FAILURE is returned. Under no circumstance does a patient record change without a corresponding audit event.

Component 8 — GATEWAY (Style 1 — Technical Specification)

Style 1 is selected because the API gateway is a deterministic routing, validation, and rate-limiting component with fully definable behaviour. Its CRITICAL requirement is that no request reaches any internal component without passing JWT validation through IDENTITY. The gateway is stateless and has no business logic.

KEY CRITICAL REQUIREMENT

[CRITICAL] Every inbound request shall be validated against IDENTITY before routing. A request that fails JWT validation shall be rejected at the gateway with HTTP 401. The internal component shall never receive an unauthenticated request.

3 Delivery, Verification, and Evidence

Parallel Delivery

After the master Style 13 specification is locked, all eight component teams proceed in parallel. Each team follows the full SDPF Phase 1 lifecycle for their component: write specification, complete TVG, lock specification, generate tests, lock tests, generate implementation, run verification gate, export evidence.

No team needs to wait for another team. The boundary contracts in the master specification define exactly what each component must accept and return. As long as each component satisfies its boundary contracts, integration will succeed.

Integration Verification — Gate G3

After all eight components achieve VERIFIED status individually, integration verification begins. This is Gate G3 in the five-gate verification hierarchy. Integration verification tests the boundary contracts defined in the master specification — not the internal behaviour of any component, which is already verified by the component's own evidence package.

Boundary	Integration Test
GATEWAY → IDENTITY	Unauthenticated request is rejected at gateway. Authenticated request is forwarded with JWT intact.
IDENTITY → LEDGER	Every authentication attempt produces a LEDGER event. LEDGER event chain is unbroken after 100 authentication operations.
PATIENT → LEDGER	Patient record write produces LEDGER event. LEDGER write failure causes patient write rollback.
MEDICATION → PATIENT	Medication administration retrieves current patient allergies and weight from PATIENT before constraint evaluation.
TASKS → ALERT	Overdue task triggers ALERT within 30 seconds. ALERT criticality is correctly escalated based on patient acuity.
ALERT → LEDGER	Every alert creation and escalation produces a LEDGER event.
REPORT → LEDGER	HIPAA audit report for a patient with 100 events is returned within 60 seconds. Every event in the report matches the LEDGER record exactly.

The Evidence Structure

MediTrack produces nine signed evidence packages — one for the master Style 13 specification and one for each of the eight component specifications. Together they constitute the complete audit record for the system.

Evidence Package	Contains	Signs
MEDITRACK-MASTER	System problem statement. Complete component inventory. All boundary contracts. Dependency graph.	The problem the entire system solves and the architecture that solves it.
MEDITRACK-IDENTITY	HIPAA regulatory mapping for all CRITICAL requirements. Authentication audit trail evidence.	Every access decision is governed by a formally verified HIPAA-compliant specification.
MEDITRACK-LEDGER	Immutability contract. Chain integrity proof. Append-only constraint verification.	The audit trail is immutable, tamper-evident, and formally specified.
MEDITRACK-MEDICATION	All four constraint registers. Constraint priority order. Boundary value test results.	No medication record was created without passing all four patient safety constraints.
MEDITRACK-ALERT	State-escalation map. Criticality level definitions. Delivery timing verification.	Alert escalation is automated, not discretionary, and formally specified.
MEDITRACK-TASKS	Task template registry v1.0. Change classification rules. Handoff verification.	Task handoff is formally governed with both parties recorded.
MEDITRACK-REPORT	Report generation specifications. 60-second SLA verification. Chain integrity report.	Regulatory reports are generated from immutable data and are themselves immutable.
MEDITRACK-PATIENT	Audit write atomicity verification. Care team traceability.	Every patient record change has a corresponding LEDGER audit event.
MEDITRACK-GATEWAY	JWT validation verification. Unauthenticated rejection verification.	No unauthenticated request reaches any internal component.

What the Evidence Answers

An auditor reviewing MediTrack for HIPAA compliance can answer every relevant question from the evidence packages without asking the development team:

Auditor Question	Evidence Package	Specific Field
What problem was this system built to solve?	MEDITRACK-MASTER	problem_statement
Does every access decision satisfy HIPAA minimum necessary?	MEDITRACK-IDENTITY	S-11 Regulatory Mapping, verification field
Is the audit trail immutable and tamper-evident?	MEDITRACK-LEDGER	S-11 Immutability Contract, chain_integrity check
Were medication safety constraints formally specified and verified?	MEDITRACK-MEDICATION	S-11 Hard Constraint Register, traceability field
Can a patient audit trail be retrieved in under 60 seconds?	MEDITRACK-REPORT	S-05 Processing Rules, VER check for SLA
Has any evidence package been altered since it was produced?	All packages	provenance.signature — verify HMAC-SHA256

4 What This Case Study Demonstrates

Problem First — Everything Derives From It

The system-level problem statement — 100% of clinical task coordination untracked, 18-hour audit reconstruction, 34% of incidents from handoff errors — is referenced in every component specification. The LEDGER's 60-second retrieval requirement comes directly from the problem statement. The MEDICATION component's CRITICAL constraints come directly from the patient safety gap. The REPORT component's documentation-as-CRITICAL-deliverable comes directly from the compliance exposure named in the impact statement.

Without Phase 0, each component team would have defined their own implicit success criteria. The LEDGER team might have targeted 5 minutes for audit retrieval. The MEDICATION team might have treated constraint violations as warnings rather than blocking errors. The REPORT team might have treated reports as optional outputs. With Phase 0, every component derives its CRITICAL requirements from the same source of truth.

Each Style Does Exactly the Right Thing

The power of the style system is visible here. The same framework produces eight completely different specifications because eight different domains require eight different semantic emphases:

- IDENTITY uses Style 10 because regulatory mapping is not an annotation — it is the contract. Every CRITICAL requirement is a HIPAA clause.
- LEDGER uses Style 12 because mutation is not a failure mode — it is a specification violation. The style makes this explicit at the level of the contract.
- MEDICATION uses Style 8 because the constraints are not validation rules — they are primary requirements. Constraint satisfaction is what the component is for.
- ALERT uses Style 14 because criticality is not a property of the alert — it is a function of runtime state. The style formalises this before implementation begins.
- TASKS uses Style 5 because the task templates will change. Change tolerance is not an assumption — it is a contract-level control.
- REPORT uses Style 16 because the reports are not outputs — they are deliverables with the same status as the code that generates them.

Parallel Development Without Coordination

Eight teams built eight components without needing to coordinate with each other after the master specification was locked. This is the most commercially significant property of Style 13. In conventional enterprise development, parallel teams need constant coordination because

their interfaces are informal and drift. In SDPF, the interfaces are locked contracts. Drift is not possible without invoking the change protocol, which is visible, classified, and recorded.

The integration verification passed first time on all boundary contracts. Not because the teams were exceptional developers. Because the contracts were exact before anyone wrote code.

The Audit Record Is the System's History

The nine signed evidence packages are not documentation that describes MediTrack. They are a cryptographically protected record of what MediTrack was specified to do, what was verified, and what was built — in that order, with the chain of custody intact.

A regulator reviewing MediTrack in five years, after the original development team has left, can read the evidence packages and know exactly what every component was required to do, which regulatory clauses each requirement satisfies, and whether any evidence package was altered after it was signed. This is not possible with any other development methodology. It is a structural property of SDPF.

THE DEMONSTRATION

MediTrack is a complex, regulated, multi-component enterprise system. SDPF governed its complete delivery — from a single validated problem statement through eight independently specified and verified components — without losing traceability, without losing audit evidence, and without requiring coordination between component teams after the master specification was locked. This is what SDPF at enterprise scale looks like.

Problem first. Specification second. Facts before execution. Verification always.

SDPF Enterprise Case Study — MediTrack Clinical Operations Platform

Hamza Abdullah · 2026