HITRUST-ready as of 2026-05-20

Behavioral-health Medicaid RCM, rebuilt from the canonical model up.

One platform. Six brands. Zero stored procedures. A modern multi-tenant SaaS that replaces legacy AdminConsole / ClientConsole / Client Central stacks at roughly **one-tenth the cost** of running the same workload.

See it vs. legacy

Try the brand switcher ↑ — every brand colour swap is live.

OPERATING ECONOMICS, BY THE NUMBERS

Receipts, not adjectives.

$1.5–1.8k

Production Azure spend / month

vs. $5k–$20k for the legacy on-prem equivalent

234

feat + fix commits

50 business days · ~4.7/day · 2026-04-01 → 2026-05-20

6

Brands shipping today

New brand onboarding: 1 day

9

First-class program categories

BH · I/DD · ABA · ADC · ICF · Home Care · OTP · PRTF · CCBHC

THE ARCHITECTURAL BETS

Three commitments that change the economics.

01

No stored procedures.

Every business rule — claim scrub, modifier injection, group-size adjustment, COB cascade, denial categorisation, EVV validation — lives in TypeScript services with content-addressed YAML rule artifacts. Eight rule-set kinds span the whole pipeline.

The customer ships the change; engineering does not.

02

EDI as a service, not per-customer plumbing.

A single edi-gateway centralises 270/271, 276/277, 278, 837P/I, 835, 999, TA1, and 834 across every tenant. Routing rules are declarative; replay is a UI button; companion-guide validation is live-previewable from a Monaco editor.

"We lost a batch" **becomes a click, not an incident.**

03

Federation as a first-class integration model.

EMR and EHR partners register themselves via the federated SSO trust registry: RFC 7519 JWTs, RFC 7517 JWKs, staged → active → superseded → revoked key lifecycle. Revoke propagates across every running instance in under 30 seconds.

One self-service registration replaces a custom integration project.

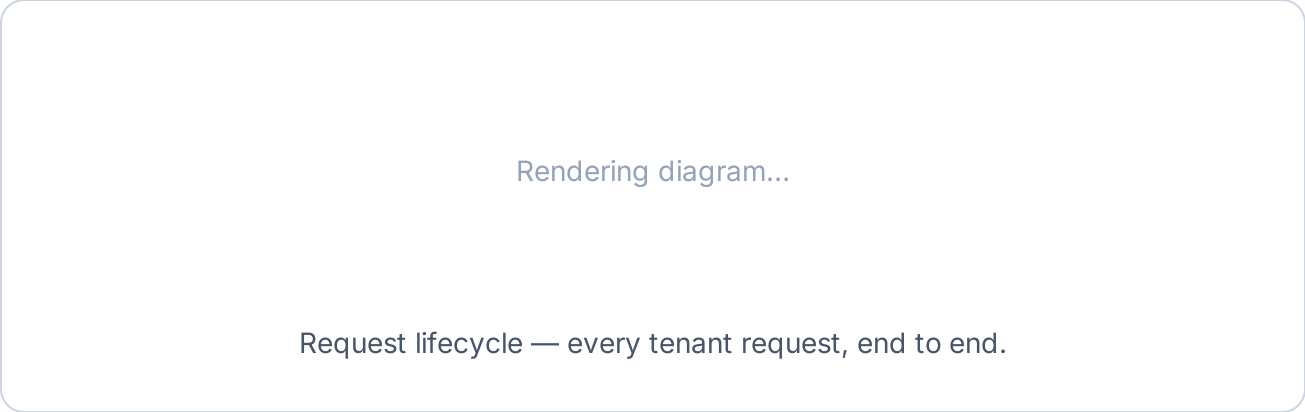
A SINGLE REQUEST, END TO END

Physical multi-tenancy. One database per customer.

There is no row-level security. Tenant isolation is **physical** — one PostgreSQL database per customer, connection strings sealed in Azure Key Vault, never persisted in the master DB.

Routing is subdomain-based, brand selection follows the URL, and tenant resolution happens at the API edge before any business code runs.

> **Why physical isolation**
>
> Leaking a tenant token cannot grant access to platform routes; leaking a platform token cannot grant access to any tenant's data without an explicit, audited impersonation event.



Request lifecycle — every tenant request, end to end.

> “ It is structurally cheaper, structurally faster, and structurally more flexible than any legacy platform — because the architectural choices remove the categories of work that drive legacy cost. ”
>
> COMPETITIVE DIFFERENTIATION — EXECUTIVE OVERVIEW

Where to next?

Three paths through the dossier — pick the one that matches what you want to verify.

DEEP DIVE

Capabilities

The rules engine, EDI engine, master data, security model, configuration surface — page by page.

→

SIDE-BY-SIDE

vs. Legacy

Every AdminConsole / ClientConsole / Client Central surface mapped to its modern equivalent — and the capability that's new.

→

TAKE IT WITH YOU

Executive brief

Download the full 14-document dossier as a single PDF. Glossary, references, and contact channel included.

→

THE STRATEGIC THESIS

Three bets that change the economics.

Legacy behavioural-health RCM platforms accumulated complexity for two decades — per-customer database stored procedures, per-payer T-SQL scrubbers, brittle ETL between vendors. The new platform inverts every one of those constraints. Three architectural commitments make it work.

THE ARCHITECTURE BET · 01

A canonical model, no stored procedures, AI-first development.

Every business rule — claim scrub, modifier injection, group-size adjustment, COB cascade, denial categorisation, EVV validation — lives in TypeScript services with content-addressed YAML rule artifacts. Eight rule-set kinds span the whole pipeline.

A

Content-addressed artifacts

Every rule version has a hash, a provenance trail, and an immutable snapshot. A rule that's *published* in production today is the same artifact you'd see if you re-built the system from scratch tomorrow — bit-for-bit.

B

Scope precedence

Rules match against a `7+1` dimensional scope — org, site, facility, billing-entity, payer, program, service-line, plus state. Specificity scoring + `precedence_rank` tie-break make the winning rule **explainable**, not buried in a stored procedure.

C

Effective dating everywhere

Rules carry `effective_from` and `effective_to` ranges. A fee-schedule change scheduled for Q3 doesn't require a code deploy; an end-of-year payer policy update is a YAML change dated for January 1.

What this unlocks

The customer ships the change; engineering does not. A new state's modifier policy, a payer's revised companion guide, a program-specific denial categorisation — each becomes a configuration action with a draft / review / approve / publish lifecycle. No back-office tickets. No code deploys. No "wait for the next release."

> “ No stored procedures. No code deploys. No back-office tickets. ”

RULES ENGINES & CONFIGURATION

THE ECONOMIC BET · 02

Operating leverage that doesn't grow with customer count.

The legacy model is one DBA-supported deployment per customer, one custom integration project per partner, one bespoke report per executive ask. The new platform changes the unit economics — and the headcount they imply.

$1.5–1.8k

Production Azure spend per month

Legacy on-prem equivalent: $5K–$20K/month per customer

2–3

AI development engineers

The full long-term support headcount, not a per-customer ratio

234

Production commits

In 50 business days · ~4.7 per day · 2026-04-01 → 2026-05-20

Three numbers point at the same thing — **the cost structure is structurally different.**

Production Azure spend is bounded by the platform's resource footprint, not by the customer count. Adding a customer adds a tenant database (a few

dollars/month of Postgres storage) and a Front Door host (cents/month), not a server, not a license, not a DBA on the support rotation.

The development model is the same. Adding a state's modifier policy isn't a sprint — it's a configuration commit. The 234-commit cadence isn't a sprint either. *It's the steady-state.*

> **What this unlocks**
>
> A behavioural-health billing organisation that runs the legacy stack on $5K–$20K/month per customer can replace the entire platform layer for roughly **one-tenth the cost** — while shipping changes daily instead of quarterly.

THE COMPLIANCE BET · 03

HITRUST-ready, native MFA, federation as a first-class integration model.

Security and compliance posture is built into the platform, not bolted on. The controls a HITRUST auditor would expect — they're already there, with audit evidence.

Native MFA — TOTP + WebAuthn

Phishing-resistant authentication enforceable per-tenant. FIDO MDS3 sync gates trusted authenticator models. Counter-regression detection catches credential-clone attacks. Step-up MFA on sensitive operations with a 300-second freshness window.

Federation, not custom integrations

EMR / EHR partners register themselves: RFC 7519 JWT, RFC 7517 JWK, staged → active → superseded → revoked key lifecycle. Revoke propagates across every running instance in **under 30 seconds** via a 30-second LRU revoke cache.

Physical tenant isolation

No row-level security. One PostgreSQL database per customer, connection strings sealed in Azure Key Vault, never persisted in the master DB. Leaking a tenant token cannot grant access to platform routes; leaking a platform token cannot grant access to any tenant's data without an explicit, audited impersonation event.

BCDR with RTO ≤ 1h, RPO ≤ 5 min

Production Postgres on ZR-HA with 35-day backup retention; geo-redundant storage (RAGZRS); a DR-by-flag posture so the passive region only costs $1.1K/month when armed. The DR drill is dated and the runbook is current.

HITRUST-ready as of 2026-05-20

The platform maps cleanly to the HITRUST CSF identity, access, cryptography, audit-logging, and BCDR control families. Evidence for each control lives in code + Terraform + the audit log — not in a spreadsheet that gets re-built every audit cycle.

> “ It is structurally cheaper, structurally faster, and structurally more flexible than any legacy platform — because the architectural choices remove the categories of work that drive legacy cost. ”

COMPETITIVE DIFFERENTIATION

KEEP READING

Verify the thesis.

The three bets above only matter if they hold up under scrutiny. Three paths through the dossier — pick the one that matches what you want to verify.

SIDE-BY-SIDE

vs. Legacy

Every AdminConsole, ClientConsole, Client Central, and SuperAdmin surface mapped to its modern equivalent. →

DEEP DIVE

Capabilities

The rules engine, EDI engine, master data, security model, and configuration surface — page by page. →

SHORT FORM

For executives

The 10-minute read: TCO, cadence, risk reduction. Enough to forward with confidence. →

40 pairings · 4 categories

SIDE-BY-SIDE

vs. Legacy.

Every legacy AdminConsole, ClientConsole, SuperAdmin, and Client Central surface, mapped to its modern equivalent — and the capability that's new. This is the page to forward to anyone who's lived inside the Millin stack for a decade.

The architectural premise

The biggest architectural difference is the disappearance of T-SQL stored procedures as the carrier of business rules. Every recurring legacy "sproc category" — claim scrubbing, modifier injection, group-size adjustment, COB cascade, denial categorisation, state-specific EVV — has a configuration-surface counterpart on the new platform. **The customer ships the change; engineering does not.**

Search 40 pairings — try “modifier”, “NPPES”, “replay”...

All 40 | AdminConsole / SuperAdmin 11 | ClientConsole / Client Central 13 | Specialized utilities 8 | Stored procedures 8

CATEGORY · 01

AdminConsole / SuperAdmin → /platform/*

Platform-operator surfaces — the things only the platform team can do. Legacy stacks scattered these across half a dozen tools and a lot of DBA-driven SQL. The new platform consolidates them into a single platform-JWT-gated console.

LEGACY

~~Provision new tenant DB (DBA-driven scripts)~~

→

MEDSUITE

Multi-step provisioning wizard

`/platform/tenants/new`

DIFFERENTIATOR

Region + Postgres-server placement is previewed before commit; tenant DEK + Key Vault secret + initial admin user are seeded in one async job.

LEGACY

~~Move tenant between Postgres servers (manual SQL)~~

→

MEDSUITE

UI-driven tenant move with capacity preview

`/platform/tenants/:id/move`

DIFFERENTIATOR

Capacity headroom on the target server is shown alongside the move; audit lands in master + tenant simultaneously.

LEGACY

~~Tenant inventory + status~~

→

MEDSUITE

Filterable tenant directory

`/platform/tenants`

DIFFERENTIATOR

Filters cover active / read-only / suspended / off-boarding; region + brand + Postgres-server columns visible without leaving the page.

LEGACY

~~Postgres server inventory + capacity tracking~~

→

MEDSUITE

DB-server inventory with headroom view

`/platform/db-servers`

DIFFERENTIATOR

Tenant-per-server count and headroom visible at a glance; no T-SQL queries against system tables required.

LEGACY

~~NPPES refresh (custom scripts per server)~~

→

MEDSUITE

Scheduled NPPES refresh with URL-template editor

`/platform/nppes`

DIFFERENTIATOR

Monthly full + weekly delta cadence is platform-managed; URL templates editable without a redeploy; manual ingest available per run.

LEGACY

~~Environment configuration (sprawling INI/config files)~~

→

MEDSUITE

Read-only env-config viewer

`/platform/env-config`

DIFFERENTIATOR

Source of truth lives in Terraform + Azure Key Vault; the surface is a viewer, not an editor — no config drift between Portal clicks and IaC.

LEGACY

~~Federation partner registration (engineer task per partner)~~

→

MEDSUITE

Federation partner registry with key lifecycle

`/platform/federation`

DIFFERENTIATOR

JWT keypairs follow staged → active → superseded → revoked. Up to two active keys per partner during rollover; revoke propagates platform-wide in < 30 s.

LEGACY

~~Tenant impersonation (database-level access, audit-light)~~

→

MEDSUITE

Time-bounded, audited tenant impersonation

`/platform/tenants/:id/impersonate`

DIFFERENTIATOR

Live banner in the impersonated session; audit lands in both master and tenant; TTL enforced server-side.

LEGACY

~~Cross-tenant audit (per-tenant log file aggregation)~~

→

MEDSUITE

Single canonical platform audit log

`/platform/audit`

DIFFERENTIATOR

Every platform-JWT action recorded; cross-tenant queries are first-class; retention class + legal-hold flag per row.

LEGACY

~~Operator MFA (third-party VPN-tied)~~

→

MEDSUITE

Native operator MFA

`/platform/account/security`

DIFFERENTIATOR

TOTP + WebAuthn + recovery codes; phishing-resistant policy enforceable; FIDO MDS3 sync gates trusted authenticator models.

LEGACY

~~Cross-tenant alerting (per-DB job scheduler)~~

→

MEDSUITE

App Insights + Action Group + KQL

DIFFERENTIATOR

Azure-native, not custom: nine production KQL alert rules ship with the platform; alerts route to email / Slack / PagerDuty via Action Group.

CATEGORY · 02

ClientConsole / Client Central → /admin/*

Tenant-operator surfaces — the things customer ops + billing leaders need to run their own organisation. Legacy stacks put dozens of forms behind a stored-procedure backend; the new platform exposes each domain as a configuration surface.

LEGACY

~~User management (custom roles table per customer)~~

→

MEDSUITE

Users + roles + RBAC matrix

`/admin/users`

DIFFERENTIATOR

RBAC matrix editor; per-user effective-scope drawer; audit drawer per row; every change captures actor + before/after.

LEGACY

~~Provider directory (manual NPPES re-keying)~~

→

MEDSUITE

Provider directory with NPPES enrichment + discrepancy queue

`/admin/providers`

DIFFERENTIATOR

EMR↔NPPES discrepancies detected at ingest; suppression list per row; taxonomies and endpoints kept in sync via the platform refresh schedule.

LEGACY

~~Organisation structure (free-text fields)~~

→

MEDSUITE

Sites / facilities / billing-entities as rules-engine dimensions

`/admin/sites`

DIFFERENTIATOR

Each org-structure entity becomes a scope dimension in the rules engine; no more "we forgot to update the dropdown" bugs.

LEGACY

~~Payer setup (per-payer ad-hoc tables)~~

→

MEDSUITE

Payer directory with Overview / Aliases / Contracts / Programs / Submission tabs

`/admin/payers`

DIFFERENTIATOR

A single source of truth per payer; submission capabilities (270/271/276/277/278/837P/I) enforced at config time, not discovered in production.

LEGACY

~~Contract management (PDF + spreadsheet)~~

→

MEDSUITE

Contracts registry with 30/60/90-day expiry buckets

`/admin/contracts`

DIFFERENTIATOR

Deep-link to the payer + program scope; expiry alerts surface contracts before they lapse.

LEGACY

~~Fee schedules (CSV imports against a tightly-coupled procedure table)~~

→

MEDSUITE

Fee-schedule grid with import + per-row validation

`/admin/fee-schedules`

DIFFERENTIATOR

UI grid; CSV import; per-row validation against the master procedure catalog; effective-dating + draft → published lifecycle.

LEGACY

~~Scrubber rules (T-SQL stored procedures per payer)~~

→

MEDSUITE

PRE_SUBMIT_VALIDATE rules in a Monaco YAML editor

`/admin/rules/PRE_SUBMIT_VALIDATE`

DIFFERENTIATOR

Content-addressed artifacts; draft → in-review → approved → published lifecycle; simulator paste-a-claim → winning rule + also-rans.

LEGACY

~~Modifier rules (procedural triggers)~~

→

MEDSUITE

Modifier injection + validation + simulator

`/admin/modifiers`

DIFFERENTIATOR

Five trigger families (credential / POS / time / telehealth / group) shown side-by-side; simulator returns winning + losing rules to make precedence explicit.

LEGACY

~~State-specific configs (forks per state in T-SQL)~~

→

MEDSUITE

State directory + state rate codes aggregator

`/admin/states`

DIFFERENTIATOR

Read aggregator that deep-links into the rules / fee-schedules / contracts scoped to the state — no more grepping the database for `OH_`.

LEGACY

~~EMR ingestion setup (per-customer SSIS/Talend pipelines)~~

→

MEDSUITE

Ingestion sources + feeds + mappings with diff/dry-run

`/admin/ingestion`

DIFFERENTIATOR

Mappings are YAML-versioned; visual editor + Monaco; dry-run + diff before publish.

LEGACY

~~Audit log (read-only DB grep)~~

→

MEDSUITE

Audit log viewer with legal-hold + retention class

`/admin/audit`

DIFFERENTIATOR

Retention class assigned per row; legal-hold flag prevents deletion; cross-references the actor and the affected resource.

LEGACY

~~MFA enforcement (per-customer customisation request)~~

→

MEDSUITE

Per-tenant MFA policy editor

`/admin/security/mfa-policy`

DIFFERENTIATOR

TOTP + WebAuthn factors configurable; phishing-resistant requirement toggleable; enforcement window dated.

LEGACY

~~Federated SSO setup (engineering task per integration)~~

→

MEDSUITE

Self-service partner federation config

`/admin/federation`

DIFFERENTIATOR

Standards-based JWT (RFC 7519) + JWK (RFC 7517); customer-side keypair management; 30 s revoke propagation across instances.

CATEGORY · 03

Specialized utilities → modern surfaces

Legacy stacks shipped a constellation of single-purpose utilities: a Trading Partner Tool, a Companion Guide editor, an EDI Replay Tool, and a stack of T-SQL stored procedures for every recurring need. The new platform folds these into the rules engine + EDI operator console.

LEGACY

~~EDI Trading Partner Tool (Windows desktop app)~~

→

MEDSUITE

Trading-partner directory with capability matrix

`/trading-partners`

DIFFERENTIATOR

Per-partner capability matrix (270/271/276/277/278/834/835/837P/I/999/TA1); credentials sealed in Key Vault; test-connection per row.

LEGACY

~~Companion Guide editor (Word docs + tribal knowledge)~~

→

MEDSUITE

Companion-guide editor with live preview

`/companion-guides`

DIFFERENTIATOR

Monaco editor; live preview pane re-assembles a sample 837 / 270 as you type; rule scope per partner + transaction type.

LEGACY

~~Routing-rule SQL workbench~~

→

MEDSUITE

Routing-rule simulator + explainer

`/routing`

DIFFERENTIATOR

Precedence simulator returns the winning + the losing rules; paste a claim, see the resolution.

LEGACY

~~Replay / retry SQL scripts~~

→

MEDSUITE

Replay surface with DLQ + exponential backoff

`/replay`

DIFFERENTIATOR

"We lost a batch" becomes a UI click. DLQ visibility; idempotent re-submit; audit per replay.

LEGACY

~~Per-customer NPPES refresh scripts~~

→

MEDSUITE

Platform NPPES URL-template editor

`/platform/nppes`

DIFFERENTIATOR

NPPES loaded once for the platform, not once per customer. URL templates editable without a redeploy.

LEGACY

~~Per-customer mapping ETL (SSIS / Talend / custom scripts)~~

→

MEDSUITE

Ingestion mappings as versioned YAML

`/admin/ingestion/mappings`

DIFFERENTIATOR

Diff between versions; dry-run before publish; rollback by re-publishing a prior version.

LEGACY

~~Custom T-SQL claim scrubbers (one per state × payer)~~

→

MEDSUITE

Eight rule-set kinds + content-addressed YAML

`/admin/rules`

DIFFERENTIATOR

INGEST / CHARGE_VALIDATE / CHARGE_DERIVE / CLAIM_BUILD_GROUPING / PRE_SUBMIT_VALIDATE / COMPANION_GUIDE_VALIDATE / POSTING / EVV_CONFIG. Each artifact is content-addressed and effective-dated.

LEGACY

~~Custom denial auto-correction stored procedures~~

→

MEDSUITE

Auto-correction handler registry + per-CARC success rates

DIFFERENTIATOR

Materialised view exposes success rate per CARC; new handlers are TypeScript functions registered against a CARC, not stored procedures.

CATEGORY · 04

Stored procedures → declarative, configurable rules

The biggest architectural difference is the disappearance of T-SQL stored procedures as the carrier of business rules. Every recurring legacy "sproc category" has a configuration surface counterpart on the new platform.

LEGACY

~~usp_ScrubClaim_<Payer> stored procedures~~

→

MEDSUITE

PRE_SUBMIT_VALIDATE YAML rule artifacts

DIFFERENTIATOR

Per-payer scrubbers become per-payer rules with effective dating and a simulator.

LEGACY

~~Modifier_Required lookup table + procedural code~~

→

MEDSUITE

modifier_validation rule kind + simulator

DIFFERENTIATOR

Per-payer / per-state requirements become rules; simulator returns the winning rule.

LEGACY

~~Per-payer trigger procedures (credential / POS / etc.)~~

→

MEDSUITE

Five-trigger modifier injection with simulator

DIFFERENTIATOR

Five trigger families exposed as configuration; precedence is explicit, not buried.

LEGACY

~~One stored procedure per state~~

→

MEDSUITE

State-scoped rule artifacts + state navigator UI

DIFFERENTIATOR

States become a scope dimension; the navigator deep-links the rules / fee schedules / contracts.

LEGACY

~~X12_Override table + 837 assembler tweaks~~

→

MEDSUITE

Per-segment rules + companion-guide live preview

DIFFERENTIATOR

Segment-level overrides expressed as rules; the live preview surfaces the resulting 837.

LEGACY

~~T-SQL WHERE-clause ladders~~

→

MEDSUITE

Declarative routing table + explainable resolver

DIFFERENTIATOR

Resolution is explainable: every match returns the winning + losing rules and the specificity score that broke the tie.

LEGACY

~~Custom group-calculation procedures~~

→

MEDSUITE

group-billing canonical domain + rules engine

DIFFERENTIATOR

Group-size bands + supervisor rules + triggers are first-class canonical entities, not sproc state.

LEGACY

~~Per-state EVV check procedures~~

→

MEDSUITE

EVV_CONFIG rule kind + state EVV YAML

DIFFERENTIATOR

EVV requirements expressed as configuration; state-by-state rollouts are a YAML change.

> “ The legacy model is one custom integration project per partner; the new model is one self-service partner registration per partner. ”

COMPETITIVE DIFFERENTIATION

WHERE TO NEXT

Drill into the new surfaces.

The comparisons above point at configuration surfaces, rules engines, and EDI flows. Each has a dedicated capability page.

RULES ENGINE

Configuration & Rules

Eight rule-set kinds, content-addressed artifacts, scope precedence, lifecycle state machine, simulator. →

EDI

EDI Engine

Every X12 transaction, every routing rule, every replay capability. Trading-partner directory + companion-guide editor + live preview. →

APP

RCM Application

Every tenant workflow and Configuration tab. 22 admin surfaces, 8 platform surfaces, 12 EDI operator surfaces. →

THE SME LANDING ZONE

Capabilities, deeply.

Seven sub-pages designed for the people who will scrutinise whether the platform can actually run their state and their programs. Each carries the diagrams, surface inventories, and verbatim claims that matter for that decision.

TENANT WORKFLOWS

RCM Application

Every tenant-facing surface: eligibility, claims, members, group sessions, encounters, receivables, authorizations, denials, charges. 22 admin Configuration tabs.

→

X12 AS A SERVICE

EDI Engine

Centralized EDI: 270/271, 276/277, 278, 834, 835, 837P/I, 999, TA1. Trading-partner directory, companion guides, routing rules, replay.

→

NO STORED PROCEDURES

Configuration & Rules

Eight rule-set kinds, content-addressed YAML artifacts, 7+1 scope precedence, effective dating, lifecycle state machine, paste-a-claim simulator.

→

CANONICAL CATALOGS

Master Data

NPPES, CPT, HCPCS, ICD-10, POS, TOB, Revenue, NCCI, CARC/RARC, modifiers, state rate codes, payer registry, Transparency in Coverage.

→

BUILT IN, NOT BOLTED ON

Security & HITRUST

TOTP + WebAuthn MFA, federated SSO with key lifecycle, physical tenant isolation, PHI sealed in Key Vault, BCDR with RTO ≤ 1h.

→

SIX BRANDS TODAY

White-Label & Tenancy

Hostname-routed brand selection, physical multi-tenancy, federated integration. New brands onboard in a single day.

→

THE OPERATIONAL MODEL

AI-First Development

Every line of production code was authored by AI agents under a structured roadmap/workplan/session protocol. 234 commits in 50 business days.

→

> “ Everything an analyst would have asked back-office engineers to do in the legacy system is here. ”

RCM APPLICATION FEATURES

OR ZOOM OUT

Choose your verification path.

If you'd rather see the head-to-head against legacy first, or look at the program × state coverage, both paths lead back to the same capability detail.

SIDE-BY-SIDE

vs. Legacy →

40 legacy → modern pairings across 4 categories.

COVERAGE MATRIX

Program coverage →

9 program categories × deeply-modeled states.

SME RECAP

For RCM experts →

The deep-form summary in one read.

CAPABILITY · APPLICATION SURFACE

The RCM application.

Forty-two operator surfaces across three console roles — tenant admin, platform operator, and EDI operator. Every action a back-office engineer would have done in the legacy system is here, as a UI surface with audit, scope, and a draft/publish lifecycle.

What this app eliminates

Every "open a ticket → engineer writes T-SQL → DBA promotes to production → wait for next release" loop. The platform turns the most common engineer-mediated changes into authenticated, audited, scope-aware UI actions with a lifecycle.

THE SURFACE INVENTORY

38 operator surfaces.

Grouped by role. The tenant operator (customer-side billing leadership), the platform operator (the team running the platform), and the EDI operator (the people running the X12 plumbing) each have their own console — gated by a separate JWT scope and a separate auth flow.

/admin/* 22 | /platform/* 8 | EDI Gateway operator console 8

Inside-the-tenant configuration. Every customer-side ops, billing, and compliance workflow has a UI here.

`/admin/users`

Users

`/admin/roles`

Roles

`/admin/rbac`

RBAC matrix

`/admin/security/mfa-policy`

MFA policy

`/admin/federation`

Federation partners

`/admin/account/security`

Account security

`/admin/providers`

Select a surface on the left to inspect what it configures.

REAL-TIME, NOT POLL

The application is event-driven by default.

The legacy model is polling — refresh, query, refresh, query. The new platform streams events to the browser via Server-Sent Events. Every meaningful change in the system pushes to subscribed clients within ~1 second.

50-event ring buffer

Recent activity is held in a per-tab ring buffer; navigating away and back replays without re-fetching.

Unread counter + toast routing

Unread counters per surface; toast routing by severity. The ops team sees what changed, when, and what it affected.

1s → 30s exponential reconnect

The SSE stream reconnects with exponential backoff on transient network failure — no thundering herd, no infinite spinner.

> “ Everything an analyst would have asked back-office engineers to do in the legacy system is here. ”

RCM APPLICATION FEATURES

KEEP GOING

The application surface sits on top of three engines.

ENGINE

Configuration & Rules

What runs behind the /admin/rules/* and /admin/modifiers surfaces. →

EDI

EDI Engine

The operator console under apps/edi-app — trading partners, routing, replay. →

TRUST

Security & HITRUST

The auth, MFA, and federation models that gate every operator surface. →

CAPABILITY · X12 AS A SERVICE

EDI as a service, not per-customer plumbing.

Legacy stacks treat EDI as per-customer plumbing — a trading-partner project per customer, a stored procedure per payer, a script per recovery. The new platform centralises EDI behind canonical events and surfaces every operation as a configuration action.

TRANSACTIONS

Every X12 transaction the platform speaks.

Twelve transactions cover the eligibility / authorization / claim / remittance lifecycle plus the acknowledgments. Each is implemented end-to-end — not deferred to a future release.

All 12 | Outbound 4 | Inbound 7 | Both 1

270 — OUTBOUND
Eligibility / Benefit Inquiry

271 — INBOUND
Eligibility / Benefit Response

276 — OUTBOUND
Claim Status Inquiry

277 — INBOUND
Claim Status Response

277CA — INBOUND
Claim Acknowledgment (with errors)

278 — BIDIRECTIONAL
Services Review (Prior Authorization)

834 — INBOUND
Benefit Enrollment & Maintenance

835 — INBOUND
Electronic Remittance Advice

837P — OUTBOUND
Professional Claim

837I — OUTBOUND
Institutional Claim (UB-04)

999 ↙ INBOUND

Functional Acknowledgment

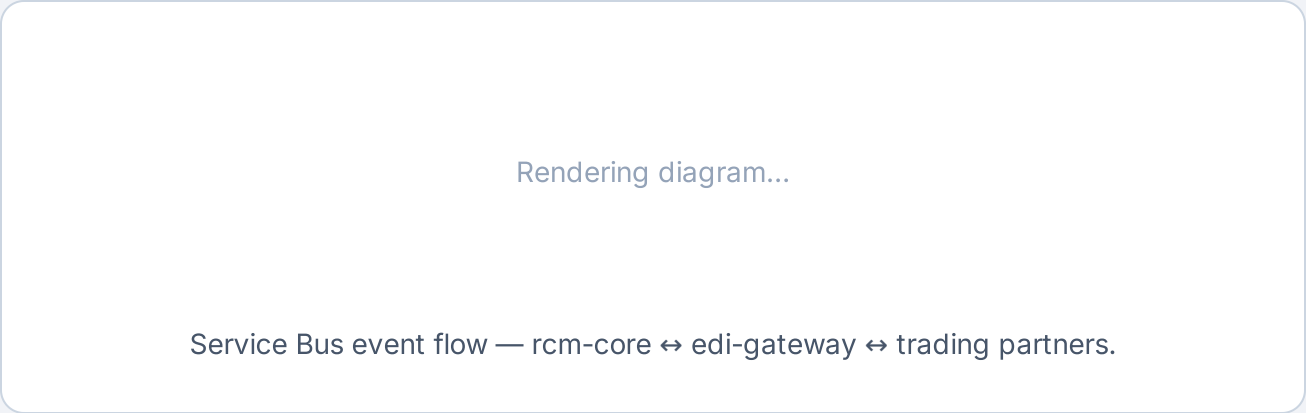
TA1 ↙ INBOUND

Interchange Acknowledgment

SERVICE BUS EVENT FLOW

Canonical events in, X12 envelopes out.

The application never builds X12. It publishes canonical events to rcm.* Service Bus topics; consumers in the EDI gateway translate to 837 / 270 / 276 / 278 and dispatch by routing rule. Inbound X12 is parsed back into canonical events and re-published.



Service Bus event flow — rcm-core ↔ edi-gateway ↔ trading partners.

ROUTING

Declarative routing, not buried T-SQL.

A new payer's EDI capability is enforced at config time, not discovered in production. Routing rules match against state, payer, payer-type, service-line, claim-type, transaction-type, dispatch-mode, and effective dates.

Each routing rule has a precedence score and an effective date. The `/routing` simulator returns the winning rule **and the losing rules** for any given inbound claim — so "why did this claim go to clearinghouse A instead of B?" is a click, not an investigation.

Transport modes are first-class: SFTP, REST, AS2, MFT, and a manual fall-back for trading partners that still require an email attachment. Credentials never traverse the application surface — they live in Azure Key Vault and only the edi-gateway worker sees them at dispatch time.

Transport modes supported

`SFTP`	Per-partner credentials in Key Vault
`REST`	OAuth2 / bearer / mTLS
`AS2`	Trading-partner AS2 IDs + signing certs
`MFT`	Managed file transfer hand-off
`Manual`	Operator-driven for legacy partners

Routing dimensions

`state` `payer` `payer type` `service line` `claim type` `txn type` `dispatch mode` `effective dates`

> “ This makes 'we lost a batch' a UI click instead of an incident. ”
>
> EDI GATEWAY FEATURES

REPLAY + COMPANION GUIDES

The two features that change EDI operations.

Replay as a UI button

Dead-letter queue visibility per consumer + exponential backoff + idempotent re-submit. "We lost a batch" used to be an incident with a war-room call and a SQL recovery script. Now it's a row in the replay surface with a button.

- DLQ visibility per consumer
- Audit log per replay
- Idempotent re-submit (won't duplicate)
- Per-row failure reason + retry count

Companion guides, live-previewable

The companion-guide editor pairs a Monaco YAML pane with a live preview that re-assembles a sample 837 or 270 as you type. Rules are scoped per partner + transaction type; a bad edit fails the validator *before* production.

- Monaco editor with YAML schema
- Sample 837 / 270 preview pane
- Per-partner + per-transaction scope
- Draft / approve / publish lifecycle

The operator difference

A legacy stored-procedure system has no such introspection — operators have to read T-SQL to understand routing or recover from a failed batch. The new platform turns both into self-service surfaces.

KEEP GOING

EDI sits at the seam of rules + master data.

RULES

Configuration & Rules

COMPANION_GUIDE_VALIDATE + PRE_SUBMIT_VALIDATE run inside the EDI flow. →

REFERENCE

Master Data

Payer registry, companion guides, NPPES — all referenced during EDI assembly. →

TRUST

Security & HITRUST

Trading-partner credentials sealed in Key Vault — never on the application surface. →

CAPABILITY · CANONICAL CATALOGS

The master-data spine.

The canonical model is the single internal vocabulary. Master-data catalogs are loaded once for the platform, then projected onto every tenant. Translation lives only at the edge — there is no canonical-shim layer.

TOPOLOGY

What lives in master, what lives in tenant.

Two-database design: a single master DB holds platform-wide reference data (NPPES, code catalogs, payer registry, companion guides); each tenant has its own database for charges, claims, ledger, rules, audit. Connection strings sealed in Key Vault.

Why this matters

The legacy model loads NPPES (~10 million providers) *per customer*. Some installations do it nightly; some quarterly; all of them spend a lot of money on storage and compute they didn't need to. On the new platform, NPPES is loaded once. Adding a customer adds a tenant database, not a duplicate copy of the federal provider registry.

THE PIPELINE

Inbound artifact → ledger entry, end to end.

Every claim follows the same canonical pipeline. The rule kinds that run at each stage are annotated on the edges — this is the same diagram that drives the application's claim-status UI.

CATALOGS

The platform's reference library.

Twenty-plus catalogs, grouped by status. The live ones are loaded today; the partials are seeded for behavioural-health Medicaid and on the active master-data roadmap; the rest are sequenced behind concrete differentiators.

Search 25 catalogs — try “NPPES”, “quarterly”, “NCCI”...

All 25 | Live 11 | Partial 3 | Roadmap 11

Showing **25** of 25 catalogs.

LIVE

NPPES NPI Registry

Monthly full + weekly delta

~10M providers + taxonomies + addresses + endpoints. Master DB `npi.*`.

LIVE

NPPES URL templates

Per env

Editable without a redeploy via `/platform/nppes`.

LIVE

CARC / CAGC / RARC

As-needed (CAQH CORE)

Adjustment + remark codes; master `rcm_reference.code_set_adjustment_*`.

LIVE

POS — Place of Service

Rare

Seeded; master `rcm_reference.code_set_place_of_service`.

LIVE

TOB — Type of Bill

Rare

UB-04 type-of-bill codes; master `rcm_reference.ub04_code`.

LIVE

Revenue codes

Rare

Seeded; revenue × procedure compatibility on the roadmap.

LIVE

Modifier catalog

Quarterly

HA-HZ + state U1-UD; master `rcm_reference.code_set_modifier`.

LIVE

State rate codes

Per-state amendments

Ohio + multi-state seed; master `rcm_master.state_rate_code`.

LIVE

Companion guides + routing

Per-payer amendments

Master `rcm_master.companion_guide` + `companion_guide_rule`.

LIVE

Commercial payer registry

Per-amendment

UHC, Aetna, Cigna, Anthem, BCBS Federation, Humana, Centene.

LIVE

Group-billing rules

Per-payer

Group-size rate adjustments, supervision rules, session triggers.

PARTIAL P0

NCCI PTP + MUE (BH subset)

Quarterly (CMS)

BH-seeded today; full CMS NCCI replaces $5K-$15K/yr/customer commercial scrubbers.

PARTIAL

CPT / HCPCS catalog

Quarterly

Schema exists; ingester active on the roadmap.

PARTIAL

ICD-10 catalog

Annual (Oct 1)

Schema exists; annual ingester on the roadmap.

ROADMAP P0

HHS OIG LEIE Exclusions

Monthly full + daily delta

~75K active exclusions; pre-claim provider check.

ROADMAP P0

State Medicaid fee schedules

Quarterly-annual

Per-state fee-schedule pipeline (PA pilot under way).

ROADMAP P1

Transparency in Coverage rates

Monthly (~5 TB JSON per major payer)

Customer-facing rate-renegotiation advisory — no incumbent BH-Medicaid platform offers anything close.

ROADMAP P1

NUCC Provider Taxonomy

Semi-annual (Jan + Jul)

Taxonomy refresh + reconciliation against NPPES endpoints.

ROADMAP P1

SAM.gov Exclusions

Daily (API)

Federal exclusion enrichment.

ROADMAP P1

State 1915(c) HCBS Waiver service defs

Per-waiver renewal

Codified waiver service definitions for I/DD + ABA + Adult Day.

ROADMAP P1

Customer-derived service scope

Daily per tenant

Pre-req for the TiC storage filter; bounds the rate-feed footprint per customer.

ROADMAP P2

State Medicaid exclusion lists

Monthly

Per-state exclusion lists alongside LEIE and SAM.gov.

ROADMAP P2

SAMHSA OTP Directory

Quarterly

Cross-reference OTP authorisations against the federal directory.

ROADMAP P3

CMS LCD/NCD coverage database

Continuous

Local + national coverage determinations.

ROADMAP P3

DEA controlled substances

Annual

Practitioner cross-reference for OTP + SUD.

> “ NPPES is loaded once. Not once per customer. ”
>
> MASTER DATA & CANONICAL MODEL

KEEP GOING

Master data is the substrate beneath the rules engine.

ENGINE

Configuration & Rules

The rules engine looks up every catalog entry the platform uses. →

EDI

EDI Engine

Payer registry + companion guides drive every X12 dispatch. →

COVERAGE

Program coverage

How catalog seeding translates into 9 first-class program categories. →

CAPABILITY · ENGINE

The rules engine.

No stored procedures. No code deploys. No back-office tickets. Three architectural commitments — content-addressed artifacts, scope precedence, effective dating — turn every business-rule change into a configuration action with a lifecycle.

THE EIGHT RULE-SET KINDS

A rule for every stage of the pipeline.

Each rule kind is its own YAML schema, its own editor surface, and its own simulator. Together they cover ingestion through posting plus EVV — every category of work that used to live in a stored procedure.

INGESTION

INGEST

Validate + enrich inbound EMR / SFTP feeds before they become canonical ServiceEvents.

CHARGE ENTRY

CHARGE_VALIDATE

Validate charge entries against payer, program, and credential constraints at the point of capture.

CHARGE ENRICHMENT

CHARGE_DERIVE

Derive units, supervision flags, group-size adjustments, and modifiers from canonical service events.

CLAIM ASSEMBLY

CLAIM_BUILD_GROUPING

Group charges into claims by payer, date, and submission rules; respect per-payer grouping policies.

PRE-SUBMISSION SCRUB

PRE_SUBMIT_VALIDATE

Final claim scrub before EDI dispatch — the rule kind that replaces per-payer T-SQL scrubbers.

X12 ASSEMBLY

COMPANION_GUIDE_VALIDATE

Validate generated 837 / 270 against the trading-partner companion guide before dispatch.

ERA / 835 POSTING

POSTING

Categorise denials, distribute responsibility, and post remittances against the ledger.

EVV COMPLIANCE

EVV_CONFIG

State-by-state Electronic Visit Verification configuration — required for Home Care and I/DD.

LIFECYCLE

Draft → review → publish — auditable end to end.

Every rule artifact follows the same lifecycle. The diagram below is the literal state machine; transitions are audited; published artifacts are content-addressed and effective-dated.

SCOPE PRECEDENCE

Which rule wins? It's explainable.

Rules match against a 7+1 dimensional scope — org, site, facility, billing entity, payer, program, service line, plus state. Specificity scoring + a precedence_rank tie-break make the resolution deterministic and auditable.

The legacy approach was a T-SQL `WHERE` ladder buried inside a stored procedure. Diagnosing "why did this claim block?" meant reading SQL — and trusting that the ladder did what the author meant.

On the new platform, resolution is a pipeline: filter by effective date, filter by rule kind, match candidates on the 7+1 dimensions, score by specificity, tie-break by `precedence_rank`. Every match returns the **winning rule plus the also-rans** — so explaining a block isn't an investigation, it's a click.

> **What this unlocks**
>
> "Why did this claim block?" answered without reading code. The simulator on `/admin/rules/PRE_SUBMIT_VALIDATE` takes a sample claim and returns the resolution, the specificity score, and the precedence tie-break — in real time.

Scope resolution algorithm — deterministic + auditable.

> “ The customer ships the change; engineering does not. ”
>
> RULES ENGINES & CONFIGURATION

WHAT THIS REPLACES

Every category of stored procedure has a configuration surface.

The hardest part of replacing a legacy RCM platform isn't the application — it's the dozens of T-SQL stored procedures encoding business rules. Each one maps cleanly to a rule kind or configuration surface.

Per-payer claim scrubbers

~~`usp_ScrubClaim_<Payer>` stored procedures~~

PRE_SUBMIT_VALIDATE rule artifacts with effective dating and a paste-a-claim simulator.

Modifier requirements

~~`Modifier_Required` lookup table + procedural code~~

modifier_validation rule kind with a simulator that returns the winning rule and the losing rules.

Per-payer trigger procedures

~~credential / POS / time / telehealth / group T-SQL triggers~~

5-trigger modifier injection with explicit precedence — no more buried logic.

Per-state procedures

~~One stored procedure per state, per behaviour~~

State-scoped rule artifacts deep-linked from the state navigator UI.

837 segment overrides

~~`X12_Override` table + 837 assembler tweaks~~

Per-segment rules + live preview from the companion-guide editor.

State-specific EVV checks

~~Per-state EVV check procedures~~

EVV_CONFIG rule kind with state-by-state YAML — rolling out a new state is a configuration commit.

KEEP GOING

The rules engine sits in the middle of two other capabilities.

UPSTREAM

EDI Engine

Where COMPANION_GUIDE_VALIDATE artifacts run and where 835s come in to trigger POSTING rules. →

REFERENCE

Master Data

The canonical catalogs the rules engine looks up — providers, payers, codes, state rate codes. →

UI SURFACE

RCM Application

The /admin/rules/:kind editor + simulator surface that every rule kind exposes. →

CAPABILITY · TRUST

Security, built in.

HITRUST-ready as of 2026-05-20. Security and compliance posture is built into the platform, not bolted on. The controls a HITRUST auditor would expect — identity, access, cryptography, audit, BCDR — they're already here, with evidence.

< 30s

Revoke propagation across instances

30s LRU revoke cache; bound by the cache TTL, not by deploy cycles

RTO ≤ 1h

Recovery time objective

RPO ≤ 5 min; DR-by-flag posture in centralus

35d

Postgres backup retention (prod)

GRS storage; 14d staging; 7d dev

OWASP 2.1

WAF rule set

DefaultRuleSet 2.1 + Bot Manager 1.0; 1000 req/min/IP rate limit

MULTI-FACTOR AUTHENTICATION

Native TOTP + WebAuthn — phishing-resistant when policy demands it.

Both platform operators and tenant users authenticate via the same engine: password + factor + (optional) step-up. WebAuthn counter-regression detection catches credential-clone attacks; the platform-side enrolment cache expires aggressively.

TOTP follows RFC 6238 with a 30-second skew window. Recovery codes are 10 single-use codes, bcrypt-hashed at cost 12, deleted on use.

WebAuthn uses the FIDO MDS3 metadata service to gate which authenticator models are trusted. A credential whose signature counter regresses below the stored value triggers a security event — a clone-attack indicator.

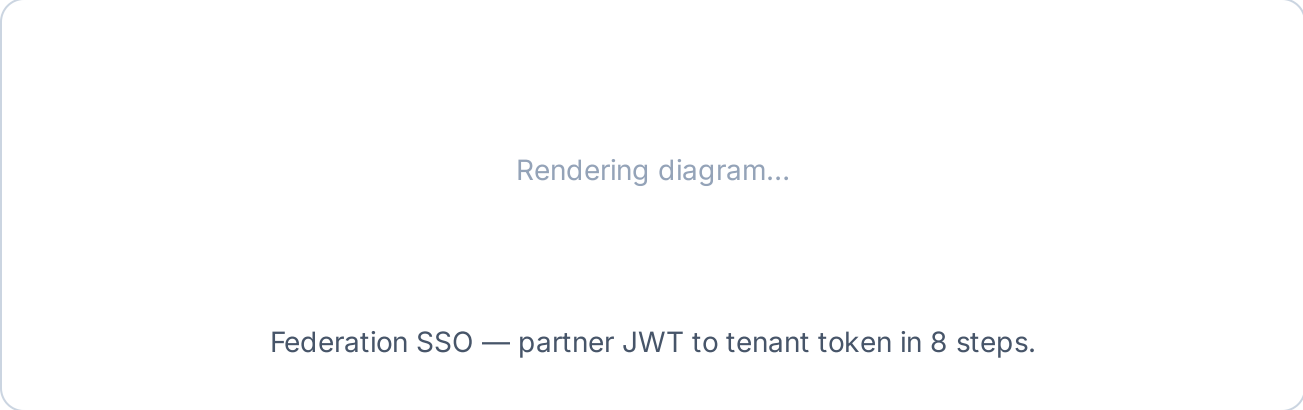
Step-up MFA applies to sensitive operations (impersonation, federation key management). The `mfaVerifiedAt` freshness window is 300 seconds by default; outside the window the user re-asserts their factor.

> **JWT TTLs**
>
> Tenant JWT 1h · Platform JWT 30m · mfa_pending 5m · mfa_enrollment 10m · refresh token 24h (revocable).

FEDERATED SSO

Standards-based federation — RFC 7519 JWT, RFC 7517 JWK.

EMR / EHR partners register themselves via the federated SSO trust registry. Per-partner signing keys follow a staged → active → superseded → revoked lifecycle. Revoke propagates across every running rcm-core instance in under 30 seconds via the LRU revoke cache.



Federation SSO — partner JWT to tenant token in 8 steps.

TENANT ISOLATION

Physical, not logical.

No row-level security. One PostgreSQL database per customer, connection strings sealed in Key Vault, never persisted in the master DB. The isolation surface is architectural — not enforced by application code that could regress.

Master DB

Tenant directory + Postgres-server inventory + master-data catalogs. No PHI.

Tenant DB (per customer)

Members, charges, claims, remittances, ledger, rules, audit. PHI lives here.

Key Vault

Sealed tenant connection strings + DEKs. Read at API edge only; never logged.

Why this matters for auditors

Leaking a tenant token cannot grant access to platform routes; leaking a platform token cannot grant access to any tenant's data without an explicit, audited impersonation event.

> “ Security and compliance posture is built into the platform, not bolted on. ”
>
> SECURITY, COMPLIANCE & HITRUST

BCDR

Disaster recovery is a flag, not a project.

The DR posture is dormant by default and armed by configuration. Passive region in centralus costs about $1.1K/month when armed and zero when not — but the wiring is in place so the flag-flip is a configuration change, not an engineering effort.

RPO ≤ 5 min

Recovery point objective. Postgres ZR-HA + RAGZRS storage carry the last few minutes of transactions.

RTO ≤ 1h

Recovery time objective. The DR drill is dated 2026-04-26 with a current runbook.

PHI DEK rotation

Quarterly key rotation. Next rotation window: 2026-07-01 at 02:00 UTC.

Front Door TLS ≥ 1.2

WAF + Bot Manager + rate limit gate every endpoint. /api/* limited to 1000 req/min per IP.

KEEP GOING

Security is the substrate every other capability sits on.

TENANCY

White-label & Tenancy

Hostname-routed brand selection + per-tenant security policy. →

ENGINE

Configuration & Rules

The lifecycle that governs every rule artifact — also audited. →

SHORT FORM

For executives

Risk-reduction narrative without the implementation details. →

CAPABILITY · TENANCY

One application, six brands, physical isolation.

The active brand follows the URL. The active tenant follows the URL. The browser experience is fully branded; the database is fully isolated; the integration model is federated by default.

6

Brands shipping today

MedSuite, Millin, Electrobills, Brittco, Accumedic, EMR-Bear

1 day

New brand onboarding

Theme TS file + favicon + logo; CI smoke; production

0

Per-tenant brand setting

Brand selection is fully automatic based on hostname

HOW IT WORKS

Brand selection is fully automatic, based on hostname.

No per-tenant brand setting. No login-time choice. The browser request lands at the platform's edge, the host header is inspected against the brand registry, and the resolved brand drives the entire UI before any application code runs.

1

Request hits Azure Front Door

The browser request lands at Front Door with the customer-facing hostname (e.g., `billing.partner-x.app`).

2

Hostname resolved to brand

`resolveBrand()` matches the hostname against each brand's `domainRoots` array. The first match wins; localhost falls back to a localStorage override.

3

Theme applied pre-paint

An inline script in the document head rewrites `--color-*` CSS custom properties before the first paint, so there is no flash-of-wrong-brand.

4

Application boots branded

React's `BrandProvider` reads the resolved brand from the document; the entire SPA — login, sidebar, surfaces — is branded from the very first frame.

TRY IT

Switch brands live.

The header dropdown on every page of this site swaps the active brand in real time across the entire experience. Every component reads from the same CSS-variable contract; the swap is a single set of writes to the document root.

Live demonstration

The brand switcher in the top-right of this page is the same registry that drives the product. Pick MedSuite, Millin, Electrobills, Brittco, Accumedic, or EMR-Bear — the change is immediate, persists across pages, and survives a refresh.

MULTI-TENANCY

Physical isolation. One PostgreSQL database per customer.

There is no row-level security. Tenant resolution happens at the API edge before any business code runs; the connection string is fetched from Azure Key Vault per request, decrypted, and used to open a tenant-scoped pool. The master DB never holds tenant credentials.

Subdomain-routed tenancy

`tenant.medsuite.app` resolves to a tenant id in the master DB; the rest of the stack reads only from that tenant's database.

Key Vault-sealed credentials

Connection strings live exclusively in Azure Key Vault. The master DB stores only the tenant directory and the Key Vault reference — not the secret itself.

Cross-app handoff

Federation between the RCM app and partner EMRs uses a 30-second TTL handoff token, replay-protected by jti uniqueness.

> “ Brand selection is fully automatic based on hostname. There is no per-tenant brand setting; the brand follows the URL. ”

WHITE-LABELING & MULTI-TENANCY

KEEP GOING

White-label sits at the seam of the application and the trust model.

TRUST

Security & HITRUST

How federation + MFA make multi-tenancy trustworthy. →

APP

RCM Application

The branded surface every tenant operator works inside. →

CADENCE

AI-First Development

Why a new brand takes a day, not a quarter. →

CAPABILITY · OPERATING MODEL

AI-built today. AI-embedded next.

"AI-First" on this platform is not a marketing claim — it is the operational model. There is no team of human engineers writing code with AI as a sidekick. Humans write the roadmap and review the pull requests; the code itself is written by AI agents following a structured protocol.

234

feat + fix commits

2026-04-01 → 2026-05-20 · 50 business days · ~4.7/day

42

Daily prompt logs

2026-01-21 → 2026-05-20 · every AI session captured

5 in 8h

MFA platform shipped

Five ADM_02 sessions on 2026-05-14 — full MFA platform

6 in 21m

White-label brands

Four RCM_02 sessions on the same day, 23:25 → 23:46

PHASE ONE

The platform is AI-built today.

A human owns the strategic roadmap. AI agents work each item to a Definition of Done — implementation, tests, security review, accessibility check, and commit. The cadence is not a sprint. It is the steady-state.

Structured protocol

Every change trajectory is the same: `roadmap` → `workplan` → `session` → `commit` → `memory note`. The protocol is codified in the repository's `.claude/CLAUDE.md`.

Per-session commits

Each session ends with a single commit — scoped, descriptive, with a `Co-Authored-By` tag on the AI agent that shipped the work. The commit message is the change log.

Mandatory review gates

Security checklist + accessibility checklist before "feature complete". The checklists are baked into the repo conventions — not policy, code.

Memory across sessions

Each shipped item generates a structured memory note that future sessions reference; the "context window" is the repository plus a curated index of prior decisions.

The cadence example, 2026-05-14

Five MFA sessions and a full white-label theming system shipped in a single day. The MFA platform — platform admin TOTP, tenant TOTP, platform WebAuthn, tenant WebAuthn parity, step-up MFA — landed across five

sequential commits between 14:26 and 22:35. The four white-label sessions landed between 23:25 and 23:46. **The cadence is not a sprint. It is the steady-state.**

VERIFIED CADENCE · 2026-04-01 → 2026-05-20

234 commits in 50 business days.

Hover or focus any day for the breakdown. Highlighted days sit above the noise; everything between is the steady-state cadence — ~4.7 commits per business day.

TOTAL	DAYS	AVG / DAY	PEAK
380	37	10.3	40

Hover a bar to see the date + commit count. Peaks carry a short note explaining what landed.

PHASE TWO

The platform is AI-embedded next.

The same agentic primitives that build the platform will run inside it. A natural-language assistant for analysts; an AI Healthcare Analyst that drafts payer-policy rule changes from a PDF; an autonomous denial-resolution agent that proposes corrections, runs them dry, and submits for human approval.

Analyst copilot

Natural-language queries against tenant data, scoped to the user's RBAC. The output is structured rows + a re-runnable query — not a one-off result.

Healthcare Analyst Agent

PDF of a payer's revised policy in; YAML draft rule + dry-run feedback loop + human approval out. The agent never publishes; the human ships.

Denial-resolution agent

Per-CARC success rates drive a closed-loop agent that proposes corrections, runs them against a sample, and queues the approved ones for re-submission.

Substrate, not bolt-on

These features sit on the same canonical model and rules engine as the rest of the platform. The agents read and write through the same surfaces a human operator uses.

> “ The competitive moat is the substrate, not any single AI feature. ”
>
> AI-FIRST DEVELOPMENT

KEEP GOING

The AI-first model is what makes the rest of the platform deliverable.

SHORT FORM

For executives

What the operational model means for staffing and TCO. →

SUBSTRATE

Configuration & Rules

The rule lifecycle the Healthcare Analyst Agent will write into. →

CADENCE

White-Label & Tenancy

How four sessions shipped six brands in one evening. →

PROGRAM COVERAGE

Nine first-class program categories.

Program-type coverage is canonical-by-design, not per-program code. Each category is modelled in the canonical layer; the differences are in seed data — codes, modifiers, rate bands, EVV configuration — not in application code.

The architectural premise

No second-tier or third-tier program is gated by code. The rules engine, master-data catalogs, and configuration UI are the extension path. Adding CCBHC = add a `canonical_program` row + a state-scoped YAML rule set. No release. No code review. No re-deploy.

DEEP-MODELLED STATES

Four states modelled deeply; five with rate-code seed.

Deep modelling means pre-submit scrubbers, companion guides, modifier injection, group-size rate bands, and EVV configuration are seeded for the state. Rate-code seed means the master state-rate-code table is populated and ready for fee-schedule pricing.

DEEP MODELLING

4 states

OH NY NJ PA

Pre-submit + companion guide + modifier injection + group-size + EVV config. The four states the platform has been hardened against in production.

RATE-CODE SEED

5 states

TX CA NY FL PA

State rate codes loaded into the master DB. Fee-schedule pipeline wires up next; Pennsylvania is the announced pilot.

INTERACTIVE MATRIX

Program × state at a glance.

The matrix below renders the full coverage surface. Click any cell to inspect the seeded codes, default unit, auth model, EVV posture, and notes for that program × state combination.

INTERACTIVE · 9 × 7 MATRIX

Program × state coverage.

Click any cell to inspect the seeded codes and configuration depth for that program in that state. Tap the legend for the visual language.

DEEP CELLS	PARTIAL CELLS	GENERAL CELLS
23	1	14

Deep · Partial · General · Extension

PROGRAM	OH	NY	NJ	PA	TX	CA	FL
Behavioral Health (BH · IOP · PHP) MENTAL_HEALTH	✓	✓	✓	✓			
I/DD (HCBS Waivers) IDD	✓	✓	/	✓	✓	✓	✓
ABA — Applied Behaviour Analysis ABA	✓	✓	✓	✓			
Adult Day Care ADULT_DAY	✓	✓	✓	✓			
ICF / ICF-IID ICF	·	·	·	·	·	·	·

PROGRAM	OH	NY	NJ	PA	TX	CA	FL
Home Care · HCBS · Personal Care HOMECARE	✓	✓	✓	✓			
OTP / SUD — MAT enforced OTP	✓						
PRTF — Psychiatric Residential RESIDENTIAL	·	·	·	·	·	·	·
CCBHC — Certified Community BH Clinic (extension via config)							

PER-PROGRAM DETAIL

What's seeded, in full.

The same data the matrix renders, laid out as per-program cards for skim-readability.

HEAVY

EVV optional

Behavioral Health (BH · IOP · PHP)

MENTAL_HEALTH

UNIT

varies

AUTH MODEL

varies

SEEDED CODES

11 BH codes (H2019, H0015, 90837, 96151-96155, T1017, H0036, H0038); IOP/PHP revenue 0905/0906/0912/0913.

NOTES

Pre-submit + companion guides seeded for OH / NY / NJ / PA.

OH NY NJ PA

FIRST-CLASS

EVV required

I/DD (HCBS Waivers)

IDD

UNIT

15-MIN

AUTH MODEL

DATE_RANGE

SEEDED CODES

H2016, H2017, T2021, H2014, T2020. Max 10 per group.

NOTES

Ohio rate codes deeply modeled; multi-state TX/CA/NY/FL/PA rate-code seed.

OH TX CA NY FL PA

HEAVY

EVV optional

ABA — Applied Behaviour Analysis

ABA

UNIT

15-MIN

AUTH MODEL

UNIT

SEEDED CODES

97151-97158, H2019; HO/HN credential injection.

NOTES

Most-developed program. Modifier injection + group-size rate bands (1-4=100%, 5-7=95%, 8+=80%).

OH NY NJ PA

FIRST-CLASS

EVV required

Adult Day Care

ADULT_DAY

UNIT

DAY

AUTH MODEL

PER_DIEM

SEEDED CODES

2 HCPCS today; expansion plan covers S5101/S5102/T2025/T2026/T2027.

NOTES

Taxonomy 251C00000X anchor; EVV required for personal care components.

OH NY NJ PA

FIRST-CLASS

EVV n/a

ICF / ICF-IID

ICF

UNIT

DAY

AUTH MODEL

PER_DIEM

SEEDED CODES

Revenue 0100-0160, LOA 0180/0183, subacute, BH 1001-1009. UB-04 master.

NOTES

Full institutional billing with condition/occurrence/value codes; occurrence code 74 anchor.

OH NY NJ PA

HEAVY

EVV required

Home Care · HCBS · Personal Care

HOMECARE

UNIT

HOUR

AUTH MODEL

VISIT

SEEDED CODES

27 codes — broadest seed of any vertical (S5125, S5126, S5130, S5131, T1019, G0156, G0299, companion services).

NOTES

OH EVV config + global EVV.

OH NY NJ PA

FIRST-CLASS

EVV optional

OTP / SUD — MAT enforced

OTP

UNIT

DAY

AUTH MODEL

EPISODE

SEEDED CODES

7 codes (H0020, H0033, H0004, H0005, J0571, J0572, H2033).

NOTES

MAT NDC enforcement; OH SUD residential includes +14-day PA.

OH

FIRST-CLASS

EVV n/a

PRTF — Psychiatric Residential

RESIDENTIAL

UNIT

DAY

AUTH MODEL

PER_DIEM

SEEDED CODES

Revenue 1001-1009 seeded.

NOTES

First-class via UB-04 master; state-specific seed pending.

EXTENSION PATH

EVV —

CCBHC — Certified Community BH Clinic

`(extension via config)`

UNIT

—

AUTH MODEL

—

SEEDED CODES

Not seeded.

NOTES

Extension path: add a `canonical_program` row + state-scoped YAML rule set; no code change.

> “ Program-type coverage is canonical-by-design, not per-program code. ”
>
> PROGRAM COVERAGE

EXTENSION PATH

Adding a new program is a row + a YAML.

The platform's extension path for a new program category — say, an emerging waiver service definition — is mechanical. There is no code path that says “if program = X”.

1. **Add a `canonical_program` row**
   One row in the master DB defines the new program key, its display name, and its service-line classification.

2. **Seed the codes**
   HCPCS / CPT codes + state rate codes go into the existing master catalogs — the same tables every other program reads.

3. **Author the rule artifacts**
   State-scoped YAML for the rule kinds that matter — typically PRE_SUBMIT_VALIDATE, CHARGE_DERIVE, and (if applicable) EVV_CONFIG.

4. **Publish**
   Draft → in-review → approved → published. No code change. No release cycle. No DBA.

KEEP GOING

Coverage is downstream of the canonical model.

SUBSTRATE

Master Data

The canonical model + catalogs that program coverage rides on. →

ENGINE

Configuration & Rules

The eight rule kinds that encode per-state, per-program behaviour. →

SME RECAP

For RCM experts

The deep-form summary in one read — for the people who'll scrutinize this. →

FOR EXECUTIVES · 10-MIN READ

The strategic case, in three frames.

Cost, cadence, risk. If a CEO of a behavioural-health billing organisation has ten minutes, these are the three frames that matter. The technical detail is elsewhere on the site.

FRAME 01 · COST

Structurally cheaper — by an order of magnitude.

The legacy stack model is roughly $5K-$20K per month per customer on the platform layer alone, plus DBA support, plus license costs, plus the per-customer integration overhead that grows linearly with the customer count. The new platform's cost structure is different in kind, not in degree.

$1.5–1.8k

Production / month

Azure spend at the platform's resource footprint, not per-customer

~$40

Dev environment / month

Idle dev environment cost

~$300–400

Staging / month

Hybrid tier; co-tenants the DevTest subscription

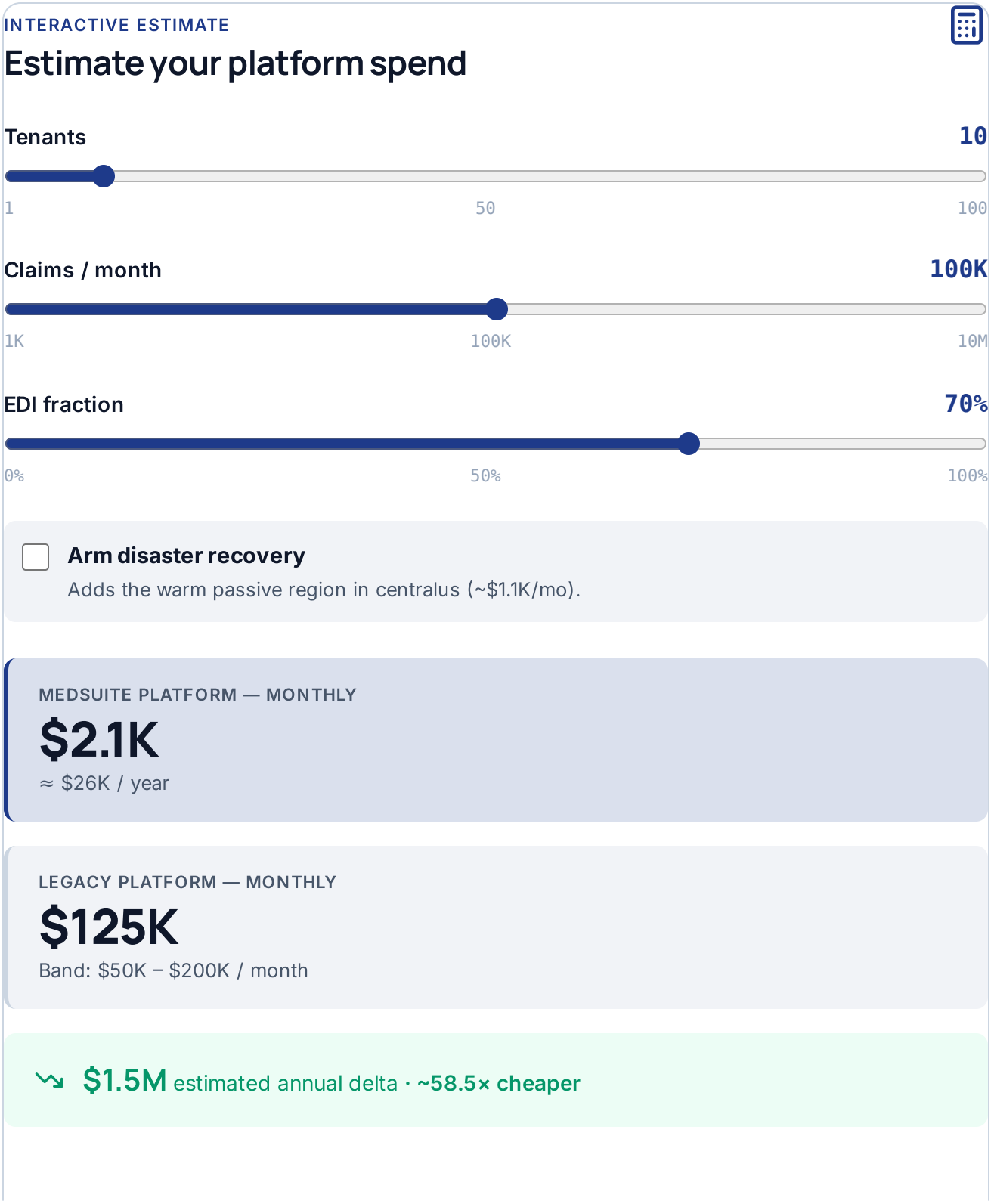
+$1.1k

DR flag-flip / month

Warm passive in centralus; armed only when needed

The cost is bounded by the platform's resource footprint — not by the customer count. Adding a customer means adding a tenant database (a few dollars/month) and a Front Door host (cents/month). It doesn't mean adding a server, a license, a DBA on the rotation, or a vendor SKU.

The numbers above are real production figures. They're not projections, and they're not best-case. The DR posture is dormant-by-default; the passive region adds ~$1.1K/month only when armed — but the wiring is in place so arming is a flag-flip, not an engineering effort.



Model uses the $1.5K base platform spend + per-tenant + per-claim overhead from the platform's infrastructure brief, against the $5K-$20K/month/customer legacy band documented in the competitive differentiation section. Real engagements vary; this is an order-of-magnitude estimate, not a quote.

FRAME 02 · CADENCE

Structurally faster — daily releases as the steady state.

The relevant cadence metric isn't how many features ship in a quarter. It's how long a customer waits between asking for a change and getting it. On this platform, that interval is hours, not months.

234

Production commits

2026-04-01 → 2026-05-20 · 50 business days · ~4.7/day

2–3

AI dev engineers

Long-term support headcount, not a per-customer ratio

1 day

New brand onboarding

Theme + favicon + smoke test + production

5 in 8h

MFA platform shipped

Five sequential commits on 2026-05-14

The cadence is enabled by the operating model: AI agents do the implementation under a structured roadmap → workplan → session → commit protocol; humans own strategy and review. The platform is *AI-built today*; the same primitives will power *AI-embedded* features (analyst copilot, Healthcare Analyst Agent, autonomous denial resolution) next.

The competitive moat is the substrate, not any single AI feature. Competitors will eventually ship a copilot. They cannot easily ship an operating model.

> “ The cadence is not a sprint. It is the steady-state. ”

EXECUTIVE SUMMARY

FRAME 03 · RISK

Structurally safer — HITRUST-ready, native MFA, federation.

The third frame is the one that lands with the CIO / CFO / general counsel. Security and compliance posture are part of the platform — not a checklist a vendor promises to satisfy.

HITRUST-ready as of 2026-05-20

The control families an auditor expects — identity, access, cryptography, audit logging, BCDR — map cleanly. Evidence lives in code + Terraform + the audit log.

Physical tenant isolation

One PostgreSQL database per customer. No row-level security to regress. Leaking a tenant token cannot grant access to platform routes, and vice versa.

Native MFA — TOTP + WebAuthn

Phishing-resistant policy enforceable per tenant. WebAuthn counter-regression detection catches credential-clone attacks. Step-up MFA on sensitive operations.

Federation as the integration model

Partner EMRs register themselves with standards-based JWT. Revoke propagates platform-wide in under 30 seconds.

BCDR — RTO ≤ 1h, RPO ≤ 5 min

DR drill artifact dated. Passive region in centralus; flag-flip, not an engineering effort.

PHI handling sealed

Per-tenant DEKs in Azure Key Vault. Quarterly rotation cadence. Connection strings never persisted in the master DB.

The honest summary

The platform is **structurally cheaper, structurally faster, and structurally more flexible than any legacy alternative** — because the architectural choices remove the categories of work that drive legacy cost. The ten-minute read is: cost is ~10× lower, cadence is daily, security posture is HITRUST-aligned.

KEEP GOING

Where to forward from here.

SIDE-BY-SIDE

vs. Legacy

40 legacy → modern pairings — the page for the VP of Revenue Cycle. →

SME RECAP

For RCM experts

Deep-form summary for billing directors, compliance officers, EDI managers. →

BRIEF

Resources

The full dossier + glossary + contact channel. →

FOR RCM EXPERTS · THE DEEP-FORM SUMMARY

The case, as you'd want to test it.

If you've spent a career inside legacy AdminConsole / ClientConsole tooling, the questions are very specific: how do rules work, how do EDI flows recover, what's seeded for my state. This page is the short index into the answers, with deep links to the surfaces.

What this page is for

A single index for the SME audience. Each section is a one-paragraph answer + a deep link to the surface that holds the evidence. **If a claim here doesn't hold up under scrutiny, the deep link is where to dispute it.**

How is a claim scrubbed without stored procedures?

PRE_SUBMIT_VALIDATE rule artifacts. Content- addressed YAML, scoped against the 7+1 dimensional matrix (org, site, facility, billing entity, payer, program, service line, state), with effective dating + a paste-a-claim simulator that returns the winning rule and the also-rans. The legacy `usp_ScrubClaim_<Payer>` procedure becomes a YAML file you can diff.

See the rules engine →

How does the platform recover from a lost batch?

Replay surface with DLQ + exponential backoff + idempotent re-submit. Per-row failure reason, retry count, audit log per replay. "We lost a batch" used to be an incident; on the new platform it's a row in a table with a button.

See the EDI engine →

What modifier policy can the customer actually own?

Five-trigger modifier injection (credential, POS, time, telehealth, group) plus a **modifier validation** rule kind with its own simulator. Per-payer precedence is explicit, not buried in T-SQL. The state-rate-code and HA-HZ / U1-UD modifier catalogues are master-side and quarterly-refreshed.

See modifier rules →

Can the EDI engine actually do my partner?

Trading-partner directory with a capability matrix per partner. 270/271, 276/277, 277CA, 278, 834, 835, 837P, 837I, 999, TA1 — each capability enforced at config time. Credentials live in Key Vault, not on the application surface. Companion guides are Monaco YAML + live 837/270 preview.

See the transaction inventory →

Is my state and my program already configurable?

Nine first-class program categories; deep seed for OH, NY, NJ, PA; rate-code seed for TX, CA, NY, FL, PA. Codes, modifiers, group-size rate bands, EVV configuration are all configuration-layer, not application-layer. Adding a category = one canonical_program row + state-scoped YAML.

See program coverage →

What happens when CMS releases a new HCPCS or ICD-10?

Quarterly + annual ingestion harness on the master-data roadmap. HCPCS Level II quarterly, ICD-10 annually, POS / TOB / Revenue / modifier catalogs already live. NCCI PTP + MUE seeded for BH today, full CMS on the roadmap (which removes the $5K-$15K/year/customer commercial-scrubber spend).

See master data →

How is denial-resolution different?

Auto-correction handler registry + per-CARC success rate views. A materialised view exposes success rate per CARC; new handlers are TypeScript functions registered against a CARC, not stored procedures. Phase two will close the loop with an autonomous agent that proposes corrections, dry-runs them, and queues human-approved re-submits.

See AI-first development →

How does ingestion handle a per-EMR mapping change?

Versioned YAML mappings + visual editor + Monaco + diff + dry-run. Per-customer SSIS / Talend pipelines become per-customer YAML files in `/admin/ingestion/mappings`. Roll back by re-publishing a prior version; no service window.

See the application surfaces →

> “ Everything an analyst would have asked back-office engineers to do in the legacy system is here. ”

RCM APPLICATION FEATURES

THE HONEST SUMMARY

It does the work you have today, with surfaces you'd actually use.

None of the legacy categories of work go away — claim scrubbing, modifier injection, EDI dispatch, denial categorisation, EVV compliance, fee schedule maintenance. What changes is who does the work and how it's expressed.

What goes away

- T-SQL stored procedures as the carrier of business rules
- Per-customer custom mapping ETL pipelines
- Per-payer scrubber engineering tickets
- Tribal-knowledge companion-guide edits
- "We lost a batch" war-room calls
- NPPES per-customer storage + refresh scripts
- Per-state stored-procedure forks
- Engineering tickets for SSO integrations

What it becomes

- Content-addressed YAML rule artifacts with a simulator
- Versioned ingestion mappings with diff + dry-run
- Configuration surfaces with draft/approve/publish lifecycles
- Monaco editor + live X12 preview per partner
- Replay surface with DLQ + idempotent retry
- Platform-wide NPPES loaded once + monthly refresh
- State-scoped rule artifacts with explainable precedence
- Self-service federation with key lifecycle

KEEP GOING

Three paths through the dossier.

ALL SEVEN

Capabilities

The full capability surface — seven sub-pages, each with diagrams + inventories. →

SIDE-BY-SIDE

vs. Legacy

40 legacy → modern pairings to forward to the CFO with confidence. →

BRIEF

Resources

Full dossier + glossary + contact channel. →