NHOS Track™ Health Intelligence Console:

A Privacy-First Architecture for Local-First Health Data Integration, Temporal Analysis, Relationship Detection, and Provenance-Aware Health Intelligence

NHOS Labs, Inc. | Privacy-first health intelligence. Evidence-informed innovation.
Version 1.0 | Unites States. | August 2026 | www.nhos.health/track

Table of Contents

Abstract

NHOS Track™ Health Intelligence Console presents a privacy-oriented health intelligence architecture for transforming heterogeneous personal health records into structured, traceable, and interpretable health information without requiring a centralized user account or remote storage of personal health data.

The architecture introduces a structured processing pipeline that proceeds through: health record generation by NHOS tools; data adaptation; canonical entity resolution; typed canonical observations; temporal analysis; relationship detection; health-intelligence interpretation; dashboard presentation;

and provenance tracing. The system is implemented around the NIME™ Intelligence Pipeline, which provides the normalization, entity-resolution, analytical, relationship, and provenance infrastructure.

A central architectural objective is preservation of data integrity while enabling higher-level analysis. Source records are retained and associated with canonical representations rather than destructively transformed. Observations retain metadata including entity identity, value, unit, timestamp, source, and provenance information, enabling dashboard-level information to remain traceable to its originating record.

The architecture also separates relationship detection from health-intelligence interpretation. Temporal relationships between health entities are detected through a Relationship Matrix, while resulting patterns are presented using observational language such as "was associated with," "coincided with," "recorded data suggests," and "observed association." This distinction is intended to reduce the risk of presenting observational relationships as causal conclusions.

The resulting system establishes four architectural pillars: Data Integrity, Intelligence, Transparency, and Privacy. Together, these principles provide a foundation for a local-first health tracking system capable of integrating personal health observations while maintaining traceability and user control.

Keywords

NHOS Track™, NIME™, health intelligence, digital health, health informatics, local-first architecture, privacy-preserving computing, personal health data, entity resolution, temporal analysis, provenance, health data integration, canonical observations, relationship matrix, observational insights, health data architecture

1. Introduction

1.1 The Problem of Heterogeneous Health Data

Personal health information is frequently generated across multiple applications, tracking tools, devices, and user-entered records. These records may differ substantially in naming conventions, units, structures, timestamps, and storage mechanisms.

A weight record may be represented as weight, body_weight, or weightKg. Blood-pressure records may contain separate systolic and diastolic values. Sleep information may be represented as duration, start and end times, or a combination of multiple observations. Exercise, nutrition, symptoms, and medication records similarly exhibit heterogeneous structures.

The result is a common architectural problem: health information may exist in multiple structured and semi-structured forms without a unified semantic representation.

1.2 The NHOS Track™ Approach

NHOS Track™ addresses this problem through a canonical health-data architecture designed around the NIME™ Intelligence Pipeline. Rather than treating a dashboard as the primary architectural layer, NHOS Track™ treats the dashboard as the final presentation layer of a larger information-processing system.

The architecture follows the conceptual progression:

This distinction is fundamental to the system design. The dashboard is not itself the intelligence engine. It is the user-facing console through which the outputs of a structured health-information pipeline become accessible.

1.3 Scope and Intended Audience

This publication documents the architectural design, implementation principles, and analytical framework underlying the NHOS Track™ Health Intelligence Console. It is intended for:

- • Health informatics researchers
- • Digital health architects
- • Privacy-preserving computing practitioners
- • Health data integration specialists
- • NHOS ecosystem developers
- • Academic reviewers

2. System Objectives and Design Principles

2.1 Primary Objectives

The architecture was designed around six primary objectives:

- • Integrate heterogeneous health records from multiple local sources.
- • Resolve heterogeneous representations into canonical health entities.
- • Preserve original source records and their provenance.
- • Enable temporal analysis across multiple analytical periods.
- • Detect relationships among health entities without implying causality.
- • Present resulting information through a privacy-first, user-controlled interface.

2.2 The Four Architectural Pillars

2.2.1 Data Integrity

Health observations should retain their relationship to their original source records. This is achieved through:

- canonical entities;
- typed observations;
- source attribution;
- preservation of original records;
- structured timestamps;
- unit-aware representations;
- provenance metadata.

The canonical model should improve interoperability without destroying source fidelity.

2.2.2 Intelligence

The system should transform structured observations into meaningful temporal and relational information. Intelligence is provided through:

- temporal analysis;
- cross-entity relationship detection;
- Relationship Matrix processing;
- longitudinal views;
- human-readable observational insights.

The intelligence layer is positioned above the data normalization layer rather than being embedded directly inside individual tracking tools.

2.2.3 Transparency

Users should be able to understand where dashboard-level information originated. Transparency is established through:

- visible analytical context;
- Relationship Matrix outputs;
- Health Timeline;
- provenance tracing;
- explicit source attribution;
- qualified observational language.

The user should be able to distinguish between recorded information and system-generated interpretation.

2.2.4 Privacy

Personal health data should remain under local user control whenever the architecture permits. The production console is positioned as:

• local-first;
• privacy-first;
• no-account-required;
• offline-capable;
• based on locally available health data.

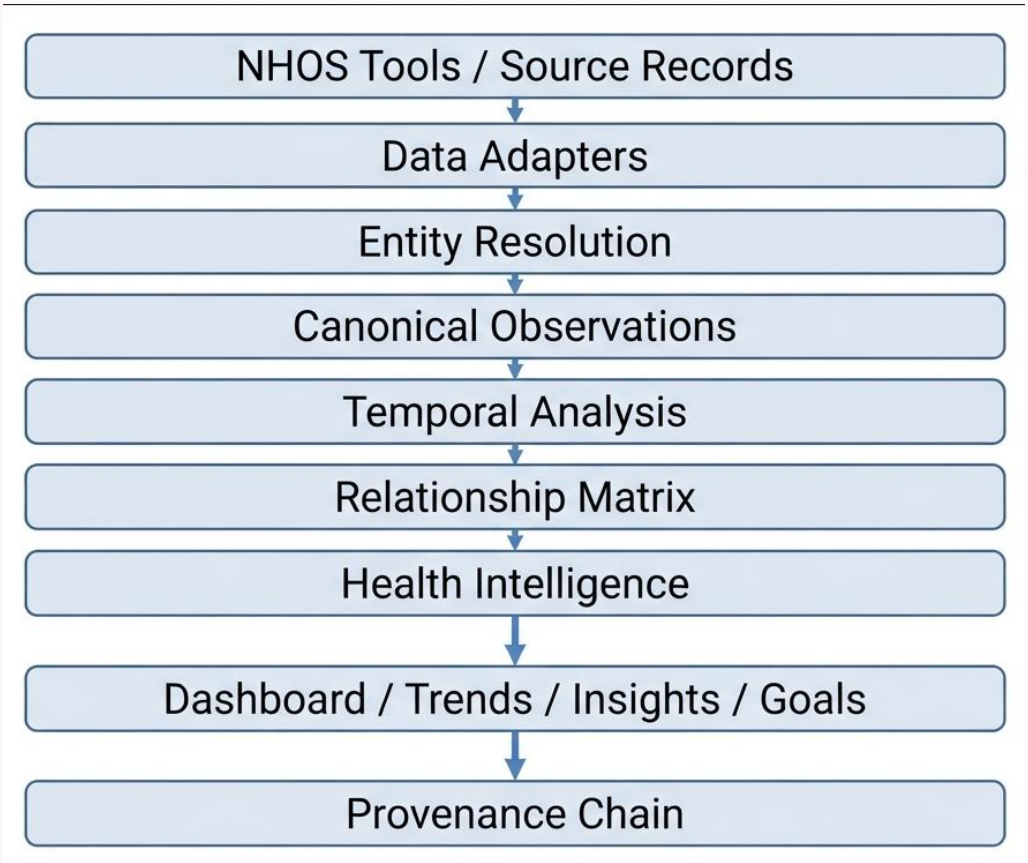
2.3 Design Principles

- **Non-Destructive Processing:** Source records are never altered or destroyed.
- **Provenance Preservation:** Every derived value must be traceable to its source.
- **Observational Language:** Insights use qualified, non-causal language.
- **Local-First:** Processing occurs locally whenever possible.
- **Extensibility:** The architecture supports new entities and data sources.

3. NHOS Track™ System Architecture

3.1 Architectural Overview

The NHOS Track™ architecture consists of the following principal stages:



3.2 Processing Pipeline

The architecture is intentionally sequential while allowing provenance metadata to persist throughout the pipeline. The implementation identifies itself as Health Intelligence Console · NIME™ v2.0 and exposes six analytical periods: 7D, 30D, 90D, 6M, 1Y, and All.

3.3 Component Interaction

Each component in the pipeline has a distinct responsibility:

Component	Responsibility
Source Records	Original health observations
Data Adapters	Interface with storage mechanisms
Entity Resolution	Map source fields to canonical entities
Canonical Observations	Standardized, typed representations
Temporal Analysis	Time-aware processing
Relationship Matrix	Cross-entity association detection
Health Intelligence	Qualified insight generation
Dashboard	Presentation layer
Provenance	Traceability infrastructure

3.4 Data Flow

Data flows unidirectionally through the pipeline. Each stage produces output that becomes input for the next stage. This separation of concerns enables independent testing, clear responsibility boundaries, replaceable components, and auditability.

4. NIME™ Intelligence Pipeline

4.1 Overview and Purpose

NIME™ provides the underlying intelligence architecture for NHOS Track™. Its principal responsibilities are:

• health-entity identification;
• data normalization;
• source adaptation;
• canonical representation;
• temporal organization;
• relationship detection;
• provenance preservation;
• health-intelligence generation.

4.2 Pipeline Components

Component	Description
Entity Registry	Canonical vocabulary definitions
Data Adapters	Source-specific interfaces
Entity Resolution	Semantic mapping
Canonical Observations	Standardized representations
Temporal Analysis	Time-aware processing
Relationship Matrix	Association detection
Health Intelligence	Insight generation
Provenance	Traceability infrastructure

4.3 Separation of Concerns

The separation between the NIME™ infrastructure and the NHOS Track™ product experience is intentional.

• NHOS Track™ represents the user-facing health intelligence product.
• NIME™ represents the underlying intelligence pipeline.

This separation allows the same architectural concepts to potentially support additional NHOS applications and analytical experiences without requiring the user-facing console to become the intelligence engine itself.

4.4 Architectural Boundaries

The NIME™ pipeline operates strictly within local processing boundaries. No personal health data is transmitted to external servers. The pipeline can function entirely offline.

5. NIME™ Entity Registry

5.1 Registry Architecture

The NIME™ Entity Registry serves as the semantic foundation of the entire intelligence pipeline. It is not merely a lookup table but a formal vocabulary that enables heterogeneous health records to be resolved into consistent analytical entities.

```
ENTITY REGISTRY ENTRY STRUCTURE:{
  id: "nhos.entity.v1.[entity-name]",
  domain: "[health-domain]",
  label: "[display-name]",
  icon: "[ui-icon-reference]",
  unit: "[canonical-unit]",
  patterns: ["field1", "field2", ...],
  cssClass: "[visual-identity]",
  dotClass: "[timeline-identity]"
}
```

5.2 Canonical Entity Definitions

Entity ID	Domain	Label	Canonical Unit	Pattern Keywords
weight	body_composition	Weight	kg	weight, body_weight, mass, weight_kg, weight_lbs
bmi	body_composition	BMI	kg/m²	bmi, body_mass_index
heart_rate	cardiovascular	Heart Rate	BPM	heart_rate, heartrate, pulse, hr, resting_pulse
blood_pressure	cardiovascular	Blood Pressure	mmHg	blood_pressure, bloodpressure, bp, systolic, diastolic

sleep	recovery	Sleep Duration	hours	sleep, sleep_duration, hours_sleep
steps	fitness	Steps	steps	steps, step_count, daily_steps
exercise	fitness	Exercise	minutes	exercise, workout, activity
water	nutrition	Water Intake	L	water, hydration, water_intake
calories	nutrition	Calories	kcal	calories, calorie_intake
symptom	symptoms	Symptom	severity	symptom, symptoms, pain, headache, fatigue
medication	medications	Medication	dose	medication, medicine, drug, prescription
mood	mental	Mood	scale	mood, stress, anxiety

5.3 Domain Classification

Each entity belongs to a health domain, enabling domain-level aggregation and analysis:

Domain	Entities
body_composition	weight, bmi
cardiovascular	heart_rate, blood_pressure
recovery	sleep
fitness	steps, exercise

nutrition	water, calories
symptoms	symptom
medications	medication
mental	mood

This domain classification enables the Relationship Matrix to identify cross-domain associations (e.g., cardiovascular ↔ sleep, nutrition ↔ body_composition).

5.4 Entity Resolution Patterns

Entity resolution may use field-name and structural pattern recognition to identify semantically equivalent source representations. For example:

- weight
- body_weight
- weight_kg
- weightKg

may be resolved to the canonical entity: WEIGHT. The objective is not merely normalization of field names. It is establishment of a stable semantic identity that downstream analytical components can use consistently.

5.5 Registry Extensibility

New entities can be added without code changes:

```
NHOS_ENTITY_REGISTRY["new_entity"] = {
  id: "nhos.entity.v1.new_entity",
  domain: "new_domain",
  label: "New Entity",
  icon: "bi-new-icon",
  unit: "unit",
  patterns: ["field1", "field2"],
  cssClass: "stat-new",
  dotClass: "timeline-dot-new"
};
```

6. Canonical Observation Model

6.1 Observation Structure

```
CANONICAL OBSERVATION:{
  observationId: string (unique identifier)
  entityId: string (references Entity Registry)
  entityKey: string (entity lookup key)
  domain: string (health domain)
  value: number (normalized numeric value)
  unit: string (canonical unit)
  unitSource: "source" | "entity_registry"
  timestamp: ISO 8601 datetime
  source: string (data source identifier)
  sourceRecordId: string (original record reference)
  originalRecord: object (immutable source preservation)
  schemaVersion: string
  quality: "raw" | "verified" | "calculated"
  confidence: number (0.0 - 1.0)
}
```

6.2 Preservation Principle

The canonical observation shall preserve the original record immutably. No destructive transformation shall occur. This principle ensures that:

- Data lineage remains intact
- Auditing and debugging remain possible
- Re-processing can occur without data loss
- Users can verify dashboard values against source records

6.3 Quality Tiers

Quality Level	Description	Use Case
raw	Direct from source, no validation	Initial ingestion
verified	Cross-referenced with multiple sources	Confidence enhancement
calculated	Derived from multiple observations	Aggregated metrics

6.4 Schema Versioning

The canonical observation model includes a schema version field to support future evolution. As the architecture matures, new versions may add fields or modify interpretations while maintaining backward compatibility with existing observations.

7. Data Adapter and Entity-Resolution Architecture

7.1 Data Adapter Layer

The Data Adapter layer provides the interface between heterogeneous source records and the canonical NHOS representation.

The production console currently identifies local health records from mechanisms including:

- IndexedDB;
- localStorage;
- NHOS tracking tools;
- other compatible local source records.

The live implementation explicitly describes its local discovery process as scanning IndexedDB and localStorage for health records.

7.2 Source Discovery

```
Data Adapter Configuration:{
  sourceName: string
  entityMap: { sourceType: canonicalEntityId }
  extractor: function(record) { ... }
}
```

The adapter layer is responsible for discovering records while retaining their source identity.

7.3 Entity Resolution

Entity resolution establishes a critical separation: Source representation ≠ canonical representation. The source record remains the original record, while the canonical representation provides a standardized analytical structure.

7.4 Normalization Process

- Source Discovery: Scan available storage mechanisms
- Record Extraction: Retrieve individual health records
- Field Mapping: Identify semantically equivalent fields
- Entity Resolution: Map to canonical entity IDs
- Value Normalization: Convert to canonical units

• Observation Creation: Build canonical observation objects

8. Temporal Analysis Framework

8.1 Analytical Periods

Health information is inherently temporal. A single measurement may have limited interpretive value without knowing when it occurred and how it relates to other observations over time. NHOS Track™ therefore provides six analytical periods:

Period	Symbol	Days	Use Case
7 Days	7D	7	Recent trends, immediate patterns
30 Days	30D	30	Monthly patterns, habit formation
90 Days	90D	90	Seasonal trends, progress evaluation
6 Months	6M	180	Semiannual health patterns
1 Year	1Y	365	Annual health review
All	ALL	∞	Complete historical analysis

8.2 Temporal Operations

• Trend Analysis: Linear and non-linear pattern detection
• Aggregation: Sum, average, min, max, count, standard deviation
• Comparison: Period-over-period change detection
• Sequencing: Chronological event ordering

8.3 Cutoff Calculation

```
function getCutoff(period, referenceDate):
  if period == "all":
    return epoch_start
  days = period_to_days[period]
```

```
  return referenceDate - (days * 24 * 60 * 60 * 1000)
```

8.4 Longitudinal Analysis

Temporal filtering provides the basis for trend analysis, observation aggregation, event sequencing, cross-entity comparison, relationship detection, and longitudinal visualization. This design allows the same underlying observations to support different analytical horizons without requiring separate datasets.

9. Relationship Matrix

9.1 Relationship Detection Algorithm

The Relationship Matrix represents the architecture's cross-entity analytical layer. Rather than analyzing each health entity in isolation, the system examines temporal relationships among entities.

```
RELATIONSHIP DETECTION:

For each pair of entities (A, B):
  Identify observations from A and B occurring within a
  24-hour temporal window.

  Match observations using a one-to-one temporal association
  rule so that an observation is not counted multiple times.

  matched_observations = number of valid temporal matches

  Temporal Association Strength =
      matched_observations /
      (|A.observations| + |B.observations| - matched_observations)

  If Temporal Association Strength > 0:
    record relationship(
      A,
      B,
      strength,
      matched_observations
    )

The Relationship Matrix score is a normalized temporal co-occurrence metric defined by the
NHOS Track™ architecture. It is not a Pearson correlation coefficient, causal effect
estimate, clinical risk score, diagnostic measure, or evidence of causation. The score
quantifies the degree of temporal overlap observed between recorded entities within the
defined analytical window.
```

9.2 Relationship Types

Type	Description	Example
temporal_proximity	Co-occurrence within the defined 24-hour analytical window	Sleep ↔ Heart Rate
sequential_dependency	Chronological ordering in which one recorded event precedes another	Exercise → Sleep

Potential future relationship types such as direct_correlation and inverse_correlation require additional statistical methods and are not represented by the current Temporal Association Strength score.

9.3 Strength Thresholds

Strength	Category	Interpretation
> 0.7	Strong	Consistent co-occurrence
0.4 - 0.7	Moderate	Regular co-occurrence
0.1 - 0.4	Weak	Occasional co-occurrence
< 0.1	Negligible	No meaningful relationship

9.4 Temporal Proximity Analysis

The Relationship Matrix is deliberately separated from the interpretation layer. This distinction is important because detecting an association is not equivalent to establishing causation. The production console therefore treats relationship detection as an analytical output that can subsequently be translated into carefully qualified health-intelligence observations.

9.5 Statistical Interpretation Boundary

The Relationship Matrix is an architectural association-detection mechanism rather than a general-purpose statistical inference framework. Its temporal co-occurrence scores identify patterns for subsequent interpretation but do not establish statistical significance, causality, clinical relevance, or predictive validity. Future versions may incorporate formal statistical methods where appropriate.

10. Health Intelligence and Observational Insight Generation

10.1 Insight Structure

```
INSIGHT:{
  title: string (relationship label)
  description: string (observational statement)
  evidence: string (supporting data)
  strength: number (0.0 - 1.0)
  timeWindow: string (period identifier)
  qualifier: string (interpretation framing)
}
```

10.2 Language Constraints

The architecture intentionally uses observational language rather than deterministic medical or causal language.

- Allowed Language:

 - "was associated with"

 - "coincided with"

 - "recorded data suggests"

 - "observed association"

 - "during this period"

 - "in the recorded data"

- Prohibited Language:

 - "caused"

 - "resulted in"

 - "proves"

 - "diagnoses"

 - "permanently"

 - "always"

10.3 Insight Examples

Pattern	Qualified Insight
Sleep ↔ Heart Rate	"On days following 7+ hours of sleep, resting heart rate was observed to be lower, based on recorded data."
Exercise ↔ Sleep	"Exercise days were associated with longer sleep duration during this period, as recorded in your data."
Nutrition ↔ Weight	"Weight changes coincided with changes in recorded calorie intake, suggesting a potential relationship."
Symptoms ↔ Sleep	"Symptom severity was observed to correlate with changes in sleep duration during this period."

10.4 Interpretation Boundaries

The intended interpretation is observational rather than causal. The architecture should not transform a temporal association into a clinical causal claim. This separation provides an important safety boundary:

```
Observed Data → Analytical Relationship → Qualified Observation
```

rather than:

```
Observed Data → Medical Causal Claim
```

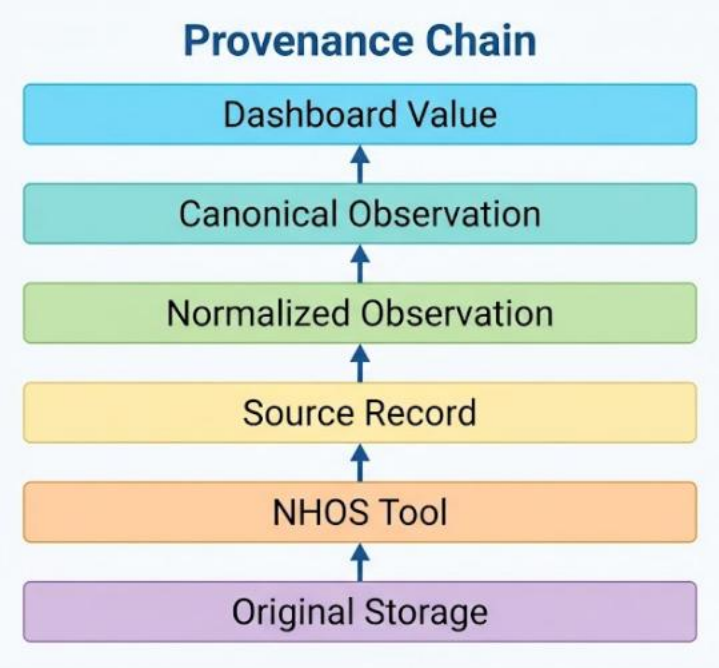
11. Provenance and Data Lineage

11.1 Provenance Structure

```
PROVENANCE:{
  source: string (source identifier)
  sourceId: string (original record ID)
  entityKey: string (canonical entity)
```

```
  observationId: string (canonical observation ID)
  timestamp: ISO 8601 datetime
}
```

11.2 Provenance Chain



11.3 Traceability Objectives

Every dashboard value shall be traceable to its originating source record through a documented provenance chain. A dashboard value should therefore have an identifiable relationship to:

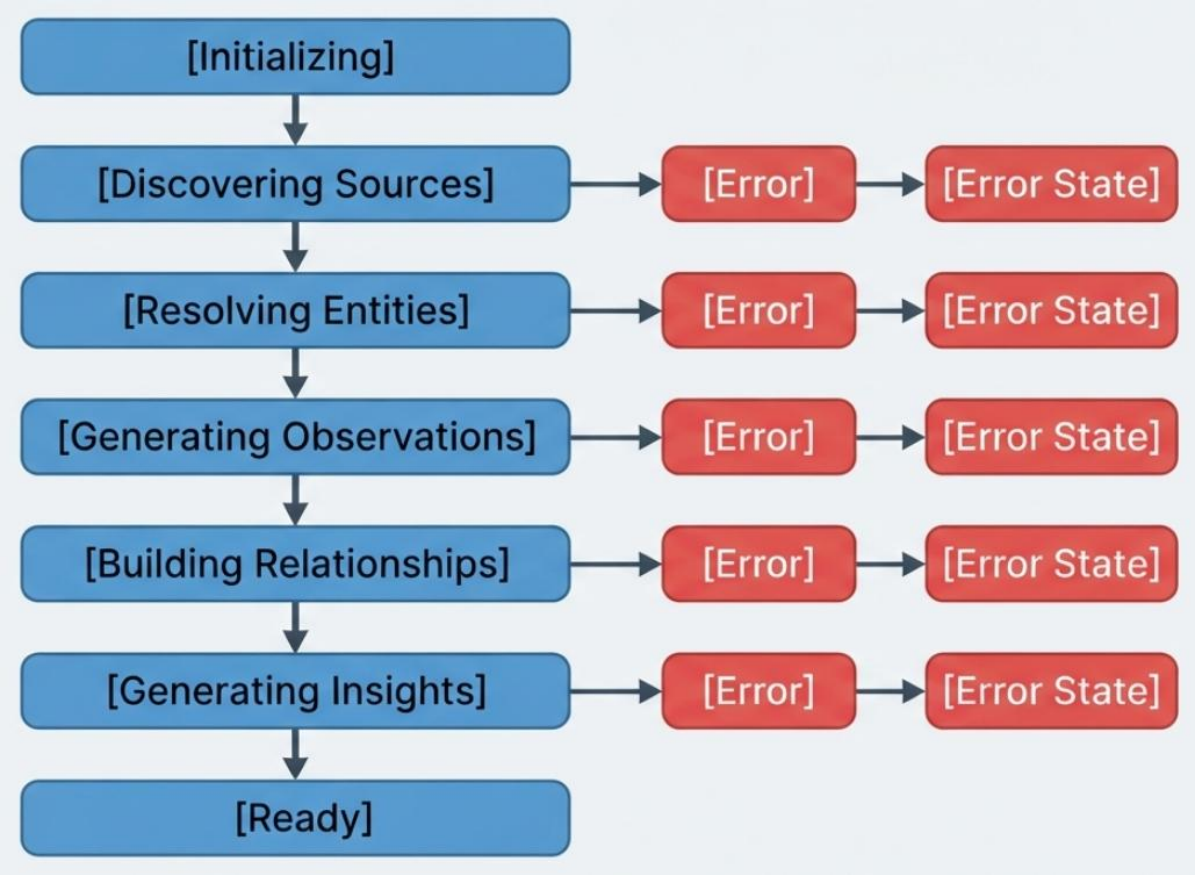
- its originating source record;
- its canonical entity representation;
- its analytical context;
- its resulting dashboard presentation.

11.4 Auditability

This approach provides an architectural foundation for auditability, debugging, data-quality assessment, user transparency, regulatory compliance, and reproducibility.

12. State Architecture

12.1 State Transitions



12.2 State Definitions

State	Description	User Experience
Initializing	Pipeline is starting	"NIME™ is connecting to your local health data…"
Discovering	Scanning data sources	"Scanning IndexedDB and localStorage for health records."
Resolving	Entity resolution	"Resolving canonical entities from source records..."

Generating	Observation creation	"Building canonical observations..."
Ready	Complete	Full dashboard with data
Empty	No data found	"No health observations yet. Start tracking your health to build your personal health timeline."
Error	Access failure	"Unable to access local health data. Your existing records have not been modified."

12.3 Error Handling

The architecture includes explicit error states for each pipeline stage. Errors in any stage prevent progression to subsequent stages. The system preserves any data that was successfully processed before the error occurred.

12.4 User Experience Mapping

This state architecture is important because an absence of displayed data should not automatically be interpreted as an absence of health records. The distinction between "no data" and "data not yet discovered" is critical for user trust.

13. User Interface and Console Architecture

13.1 Component Hierarchy

```
Health Intelligence Console
```

13.2 Visual Design System

The console uses a consistent visual language based on color (domain-specific color coding), typography (clean, readable fonts), spacing (consistent padding and margins), cards (rounded corners with shadow depth), and icons (semantic iconography).

13.3 Color Scheme — Domain Identity

Domain	Color	Hex	CSS Variable
Cardiovascular	Deep Red	#e53935	--card-cardiovascular
Weight	Teal	#00897b	--card-weight
Sleep	Indigo	#5c6bc0	--card-sleep
Fitness	Green	#43a047	--card-fitness
Nutrition	Amber	#ff8f00	--card-nutrition
Symptoms	Deep Orange	#e64a19	--card-symptoms
Medications	Purple	#7b1fa2	--card-medications
Mental	Blue	#3949ab	--card-mental
General	Slate	#546e7a	--card-general
Discovery	Deep Blue	#283593	--card-discovery
Relationship	Teal	#00695c	--card-relationship
Provenance	Brown	#6d4c41	--card-provenance
Timeline	Dark Slate	#37474f	--card-timeline
Insight	Deep Purple	#4a148c	--card-insight

13.4 Responsive Design

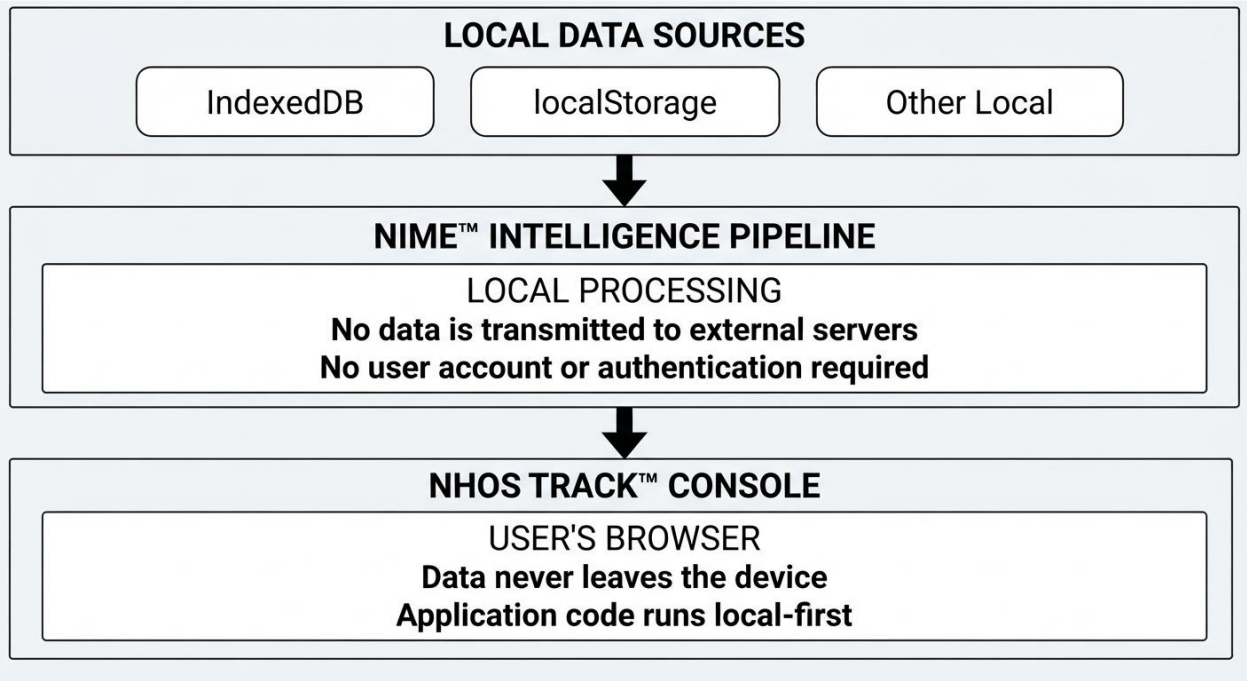
The console is designed for all screen sizes: Desktop (full-width layout), Tablet (adaptive grid), and Mobile (stacked layout with touch-optimized controls).

13.5 Quick Actions

The console provides a 2-column grid of Quick Actions: Vitals Track, Weight Log, Sleep Record, Exercise Log, Nutrition Track, Symptoms Check, Water Log, Medication Track, Wellness Track, Goals Manage.

14. Privacy and Security Architecture

14.1 Data Flow Architecture



14.2 Privacy Principles

Principle	Implementation
Data Ownership	User retains all data
Data Location	Data never leaves the device
Authentication	Not required
Account Requirement	None
Analytics	None (no tracking)
External Services	None required
Offline Capability	Full functionality available

14.3 Security Boundaries

The architecture shall not transmit personal health data across network boundaries unless explicitly configured to do so by the user.

14.4 Local-First Design

The NHOS Track™ Health Intelligence Console is designed according to a **local-first processing model**. Personal health data are processed within the user's browser using locally available storage mechanisms, including IndexedDB and localStorage. The architecture does not require transmission of personal health data to external servers for core dashboard operations, including entity resolution, temporal analysis, relationship detection, health intelligence generation, and provenance processing.

Accordingly, the console can provide its core functionality without requiring a centralized health-data repository, user account, or server-side health-data processing. Any future capability involving external data transfer should be explicitly initiated and controlled by the user.

Privacy Boundary: Local-first processing is an architectural property of the console and should not be interpreted as a guarantee against all forms of data exposure. Browser storage, device security, operating-system access, browser extensions, backups, application code, and other third-party software may affect the security and privacy of locally stored information.

The design principle can therefore be summarized as:

Compute locally when possible; transmit only when necessary.

This architecture provides several potential advantages:

- reduced dependence on centralized services;
- reduced exposure of personal health information through server-side processing;
- support for offline-capable operation;
- reduced account-management requirements; and
- greater user control over personal health data.

Important Note: Local-first architecture does not, by itself, guarantee complete privacy or security. The privacy characteristics of the console depend not only on its software architecture but also on the security of the user's browser, device, operating system, extensions, backups, and surrounding computing environment. Accordingly, the privacy claims presented in this publication should be understood as **architectural properties and design objectives, rather than absolute guarantees of security or privacy**.

15. Performance Architecture

15.1 Lazy Loading

• Charts render only when data is available

- Entity registry updates incrementally
- Timeline shows most recent events first
- Resource-intensive operations are deferred until needed

15.2 Data Efficiency

- Historical data filtered by analytical period
- Observations normalized to reduce redundancy
- Aggregated summaries computed on demand
- Minimal memory footprint

15.3 Mobile Optimization

- Responsive design for all screen sizes
- Touch-optimized interactions
- Reduced memory footprint on mobile devices
- Graceful degradation on lower-performance devices

15.4 Caching Strategy

- Intermediate results cached when appropriate
- Chart data cached to avoid re-rendering
- Entity registry cached for quick access

16. Extensibility and Integration Architecture

16.1 Entity Registry Extensibility

New entities can be added without code changes (see Section 5.5).

16.2 Data Adapter Extensibility

New data sources can be registered:

```
SOURCE_ADAPTERS["new_tool"] = {
  entityMap: { ... },
  extractor: function(record) { ... }
};
```

16.3 Insight Extensibility

New insight patterns can be added:

```
function generateInsights(entities, observations) {
```

```
  // Pattern detection logic
  // Return array of insight objects
}
```

16.4 Plugin Architecture

Future versions may support a formal plugin architecture for custom entity resolvers, specialized data adapters, domain-specific insights, and export/import formats.

17. Comparison with Conventional Health-Tracking Architectures

17.1 Feature Comparison

Feature	NHOS Track™	Conventional Health Apps
Data Ownership	User	Vendor
Account Required	No	Yes
Offline Capable	Yes	Rarely
Privacy-First	Yes	Variable
Entity Resolution	Yes	No
Temporal Analysis	Yes	Basic
Relationship Detection	Yes	No
Provenance Tracking	Yes	No
Observational Insights	Yes	Yes (often causal)
Local Processing	Yes	No
Open Architecture	Yes	No

17.2 Architectural Differences

Conventional health-tracking architectures typically store data in vendor-controlled clouds, require user accounts, process data centrally, display isolated metrics, offer limited cross-entity analysis, and provide minimal provenance information.

NHOS Track™ distinguishes itself through local-first data storage, no account requirement, local processing, cross-entity analysis, comprehensive provenance, and observational language.

17.3 Privacy Implications

The architectural differences have significant privacy implications: personal health data is designed to remain on the user's device, the architecture does not require centralized vendor access to personal health information, exposure associated with centralized health-data storage is reduced, and the user retains control over their own data.

18. Limitations and Architectural Boundaries

18.1 Observational Scope

NHOS Track™ is an observational health-information architecture. Its analytical outputs should not be interpreted as medical diagnoses, treatment recommendations, or proof of causality. Temporal association does not establish causation.

18.2 Data Quality Dependencies

The quality of the resulting intelligence is dependent upon availability of source records, completeness of observations, consistency of timestamps, accuracy of recorded values, correctness of entity resolution, appropriateness of analytical windows, and quality of relationship detection. Sparse, incomplete, inconsistent, or incorrectly classified data may reduce analytical reliability.

18.3 Clinical Limitations

The architecture is not intended for clinical diagnosis, treatment recommendation, medication management, emergency response, or medical decision-making.

18.4 Technical Boundaries

The architecture operates within browser storage limits, client-side processing constraints, single-device data isolation, and no external integration (unless explicitly configured).

19. Future Development Roadmap

19.1 Phase 1: Core Architecture (Implemented)

- Entity Registry
- Data Adapters
- Entity Resolution
- Canonical Observations
- Temporal Analysis
- Relationship Matrix
- Health Intelligence
- Provenance Chain

19.2 Phase 2: Enhanced Analysis (Planned)

- Advanced statistical correlation
- Longitudinal trend prediction
- Pattern recognition with machine learning
- Custom insight rules
- Anomaly detection

19.3 Phase 3: Ecosystem Integration (Planned)

- Standardized data export (FHIR, HL7)
- API for third-party tools
- Plugin architecture for custom adapters
- Data import from external sources

19.4 Phase 4: Clinical Research Support (Planned)

- De-identified data export for research
- Study cohort matching
- Observational study tooling
- Research data quality scoring

20. Conclusion

20.1 Summary of Contribution

NHOS Track™ Health Intelligence Console establishes a structured architecture for transforming heterogeneous personal health records into a privacy-first, provenance-aware health intelligence environment.

The architecture is organized around a clear progression:

20.2 Architectural Significance

- **Privacy and intelligence are compatible.** Local-first processing can support sophisticated health intelligence without centralized data collection.
- **Provenance is a feature, not a burden.** Tracing dashboard values to their sources enhances user trust and system accountability.
- **Relationship detection can be separated from causal claims.** Observational language maintains scientific integrity while providing value.
- **Entity resolution enables integration.** A canonical vocabulary transforms heterogeneous records into a unified analytical model.
- **Temporal analysis reveals patterns.** Time-aware processing transforms isolated observations into meaningful trends.

20.3 Final Remarks

The architecture treats personal health intelligence not simply as a visualization problem, but as a problem of data integration, semantic normalization, temporal reasoning, relationship detection, transparent interpretation, and provenance-preserving presentation. NHOS Track™ therefore represents an architectural approach to local-first health intelligence in which the user's health information remains connected to its origin while becoming progressively more useful for longitudinal understanding.

21. User Experience Architecture — Timeline and Data Management

21.1 Collapsible Timeline Architecture

The Health Timeline provides chronological context for the analytical system. However, as the user's longitudinal dataset grows, an infinitely long timeline would become overwhelming and degrade the

user experience. To address this, the timeline implements a collapsible architecture with state persistence.

21.1.1 Interaction Model

The timeline uses a chevron-based disclosure control:

```
▶  Health Timeline — 24 observations     (Collapsed)
▼ Health Timeline — 24 observations     (Expanded)
```

The interaction follows standard UI conventions: clicking the header or chevron expands the timeline, clicking again collapses it, a compact indicator displays the total event count when collapsed, underlying observations remain intact (UI collapse, not data deletion), and the timeline remains chronologically ordered when expanded.

21.1.2 State Persistence

The collapsed/expanded state is preserved locally to maintain user preference across sessions:

```
// Save timeline state
localStorage.setItem('nhos_timeline_expanded', String(isExpanded));
// Restore timeline state
var savedState = localStorage.getItem('nhos_timeline_expanded');
```

This ensures that users who prefer a compact view are not required to collapse the timeline after every page load.

21.1.3 Architectural Benefits

Benefit	Description
Scalability	Prevents the console from becoming an enormous page as the dataset grows
User Control	Users control how much timeline context they want to see
State Preservation	User preferences persist across sessions
Performance	Reduces DOM rendering overhead for collapsed content
Information Density	Compact view shows count without overwhelming the user

21.2 Data Management Architecture

The architecture includes a dedicated Data Management section that gives users control over their locally stored health information. This section provides two primary functions: Export and Clear All Data.

21.2.1 Export Functionality

The Export feature allows users to download all their health data as a structured JSON file:

```
EXPORT STRUCTURE:{
  exportedAt: ISO 8601 datetime
  version: string
  source: "NHOS Track™ Health Intelligence Console"
  totalObservations: number
  entities: { ... }           // All canonical entities
  observations: [ ... ]       // All canonical observations
  provenance: [ ... ]         // All provenance records
}
```

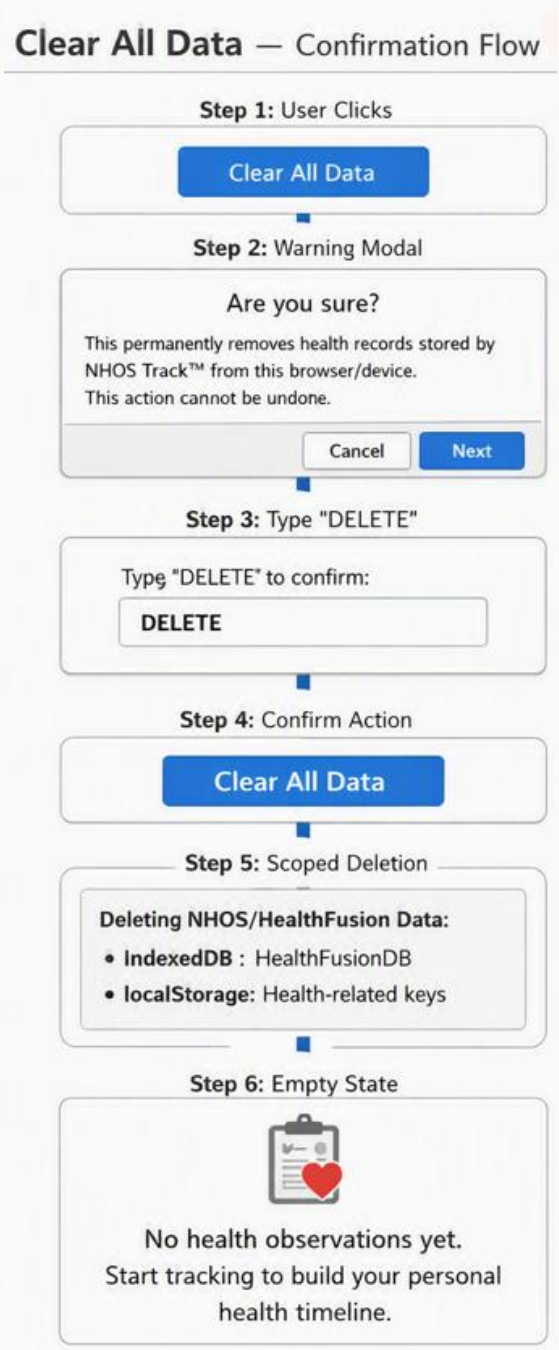
The export provides complete data portability, backup capability before destructive actions, JSON format for interoperability, and a timestamped filename for version tracking.

21.2.2 Clear All Data — Confirmation Flow

The "Clear All Data" operation includes a multi-step confirmation process to prevent accidental data loss:

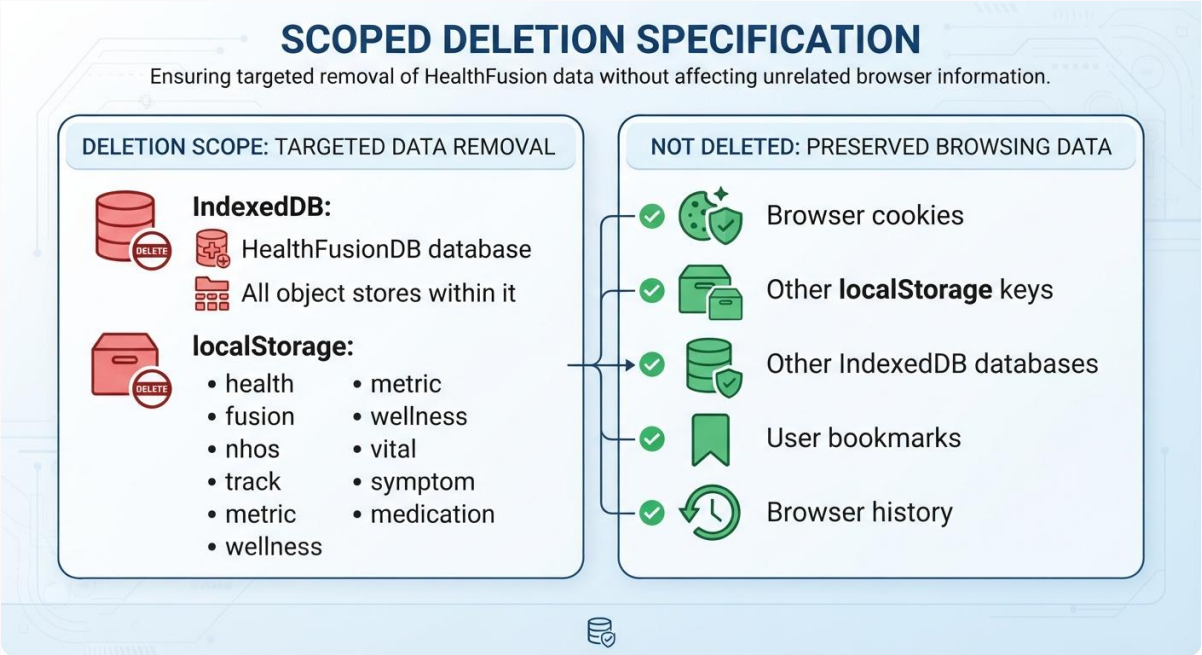
```

```



21.2.3 Scoped Deletion Specification

The deletion operation is carefully scoped to avoid affecting unrelated browser data:



21.2.4 Architectural Benefits

Benefit	Description
Data Ownership	Users can permanently remove their data
User Autonomy	Complete lifecycle control over personal health information
Privacy	Users can delete all traces of their health data from the device
Safety	Multi-step confirmation prevents accidental deletion
Export Option	Users can backup before clearing
Scoped Deletion	Only NHOS-related data is removed

21.3 Component Hierarchy Update

The UI component hierarchy now includes the Data Management section as illustrated in the system architecture overview.

21.4 State Architecture Update

The state architecture now includes data management and export/clear lifecycle transitions.

21.5 Privacy and Data Lifecycle

The Data Management architecture establishes a complete data lifecycle model from creation, organization, analysis, presentation, export, through deletion.

21.6 Implementation Notes

Component	Implementation Detail
Timeline Collapsible	CSS max-height transition with collapsed/expanded classes
State Persistence	localStorage.setItem('nhos_timeline_expanded', ...)
Export Format	JSON with observations, entities, provenance
Clear Scoping	Targets only HealthFusionDB and health-related localStorage keys
Confirmation	Input validation requiring exact match of "DELETE"
Empty State	Shows "No health observations yet" with Start Tracking button

21.7 Summary

The Collapsible Timeline and Data Management features represent architectural improvements that address the complete user experience lifecycle:

1. **Timeline Collapsibility** solves the longitudinal scalability problem, preventing the console from becoming an overwhelmingly long page as the user accumulates health observations.

2. **Data Management** solves the data lifecycle and user autonomy problem, providing both export capabilities and secure, confirmed deletion.

3. **Together**, these features establish the console as a complete health intelligence environment where users can:

• View their health data (Timeline)
• Control how much they see (Collapsible)

• Export their data (Portability)
• Delete their data (Autonomy)
• These features complement the four architectural pillars:

4. **Data Integrity**: Preserved through non-destructive operations

5. **Intelligence**: Maintained through temporal analysis

6. **Transparency**: Enhanced through export capability

7. **Privacy:** Strengthened through user-controlled deletion

Appendix A — Entity Registry Specification

A.1 Registry Entry Structure

```
ENTITY REGISTRY ENTRY:{
  id: string (canonical entity identifier)
  domain: string (health domain)
  label: string (display name)
  icon: string (UI icon reference)
  unit: string (canonical unit)
  patterns: [string] (field name patterns)
  cssClass: string (visual identity class)
  dotClass: string (timeline identity class)
}
```

A.2 Canonical Entity Definitions Table

Entity ID	Domain	Label	Canonical Unit	Pattern Keywords
weight	body_composition	Weight	kg	weight, body_weight, mass, weight_kg, weight_lbs
bmi	body_composition	BMI	kg/m²	bmi, body_mass_index
heart_rate	cardiovascular	Heart Rate	BPM	heart_rate, heartrate, pulse, hr, resting_pulse

blood_pressure	cardiovascular	Blood Pressure	mmHg	blood_pressure, bloodpressure, bp, systolic, diastolic
sleep	recovery	Sleep Duration	hours	sleep, sleep_duration, hours_sleep
steps	fitness	Steps	steps	steps, step_count, daily_steps
exercise	fitness	Exercise	minutes	exercise, workout, activity
water	nutrition	Water Intake	L	water, hydration, water_intake
calories	nutrition	Calories	kcal	calories, calorie_intake
symptom	symptoms	Symptom	severity	symptom, symptoms, pain, headache, fatigue
medication	medications	Medication	dose	medication, medicine, drug, prescription
mood	mental	Mood	scale	mood, stress, anxiety

A.3 Domain Classification Table

Domain	Entities
body_composition	weight, bmi
cardiovascular	heart_rate, blood_pressure
recovery	sleep

fitness	steps, exercise
nutrition	water, calories
symptoms	symptom
medications	medication
mental	mood

Appendix B — Canonical Observation Schema

B.1 Observation Structure

```
CANONICAL OBSERVATION:{
  observationId: string (unique identifier)
  entityId: string (references Entity Registry)
  entityKey: string (entity lookup key)
  domain: string (health domain)
  value: number (normalized numeric value)
  unit: string (canonical unit)
  unitSource: "source" | "entity_registry"
  timestamp: ISO 8601 datetime
  source: string (data source identifier)
  sourceRecordId: string (original record reference)
  originalRecord: object (immutable source preservation)
  schemaVersion: string
  quality: "raw" | "verified" | "calculated"
  confidence: number (0.0 - 1.0)
}
```

B.2 Quality Tiers

Quality Level	Description	Use Case
raw	Direct from source, no validation	Initial ingestion
verified	Cross-referenced with multiple sources	Confidence enhancement
calculated	Derived from multiple	Aggregated metrics

	observations

B.3 Schema Versioning

Version	Changes
1.0	Initial schema

Appendix C — Relationship Detection Specification

C.1 Detection Algorithm

```
RELATIONSHIP DETECTION:

For each pair of entities (A, B):

  Identify observations from A and B occurring within a

  24-hour temporal window.

  Match observations using a one-to-one temporal association

  rule so that an observation is not counted multiple times.

  matched_observations = number of valid temporal matches

  Temporal Association Strength =

    matched_observations /

    (|A.observations| + |B.observations| - matched_observations)

  If Temporal Association Strength > 0:

    record relationship(

      A,

      B,

      strength,

      matched_observations
```

```
    )

The Relationship Matrix score is a normalized temporal co-occurrence metric defined by the
NHOS Track™ architecture. It is not a Pearson correlation coefficient, causal effect
estimate, clinical risk score, diagnostic measure, or evidence of causation. The score
quantifies the degree of temporal overlap observed between recorded entities within the
defined analytical window.
```

C.2 Relationship Types

Type	Description	Example
temporal_proximity	Co-occurrence within the defined 24-hour analytical window	Sleep ↔ Heart Rate
sequential_dependency	Chronological ordering in which one recorded event precedes another	Exercise → Sleep

Potential future relationship types such as direct_correlation and inverse_correlation require additional statistical methods and are not represented by the current Temporal Association Strength score.

C.3 Strength Thresholds

Strength	Category	Interpretation
> 0.7	Strong	Consistent co-occurrence
0.4 - 0.7	Moderate	Regular co-occurrence
0.1 - 0.4	Weak	Occasional co-occurrence
< 0.1	Negligible	No meaningful relationship

Appendix D — Insight Language Constraints

D.1 Allowed Language

- "was associated with"
- "coincided with"

- "recorded data suggests"
- "observed association"
- "during this period"
- "in the recorded data"
- "based on recorded data"
- "suggests a potential relationship"

D.2 Prohibited Language

- "caused"
- "resulted in"
- "proves"
- "diagnoses"
- "permanently"
- "always"
- "will improve"
- "will prevent"
- "will cure"

D.3 Insight Examples

Pattern	Qualified Insight
Sleep ↔ Heart Rate	"On days following 7+ hours of sleep, resting heart rate was observed to be lower, based on recorded data."
Exercise ↔ Sleep	"Exercise days were associated with longer sleep duration during this period, as recorded in your data."
Nutrition ↔ Weight	"Weight changes coincided with changes in recorded calorie intake, suggesting a potential relationship."
Symptoms ↔ Sleep	"Symptom severity was observed to correlate with changes in sleep duration during this period."

Appendix E — Version History

Version	Date	Changes
1.0	2026	Initial publication

Title: *NHOS Track™ Health Intelligence Console: A Privacy-First Architecture for Local-First Health Data Integration, Temporal Analysis, Relationship Detection, and Provenance-Aware Health Intelligence*

Creator: NHOS Labs, Inc.

Resource Type: Technical Report

Version: 1.0

Publication Year: 2026

Publisher: NHOS Labs, Inc.

Website: https://nhos.health

NHOS Track: https://nhos.health/track

NHOS Research Registry: https://nhos.health/research