The Search for Logical Models and the place of the openEHR Reference Model

Sam Heard, Seref Arikan April 2026

Contents

The search for a comprehensive set of logical models for health care has a long history. My experience began as a key investigator in the Good European Health Record project in the early 90s. It became clear during this project that 1) an all-inclusive single model of health data was not achievable, and 2) the data specifications had to meet the needs of the people who wanted to use the data.

I have always been interested in a single life-long health record that could grow over time and be available throughout a person's life. Clearly it would not be appropriate to base this on a particular technology, terminology or other feature that is bound to evolve over time. This is considered to be the role of a logical model. Let us consider then what logical models are and why people are interested in them.

What is a logical model?

A logical model is a structured representation of data, showing entities, attributes, and relationships, independent of any specific database technology. It acts as a blueprint that defines how data is organized conceptually and ensures consistency, integrity, and clarity before moving to physical implementation.

Key Characteristics of a Logical Model

- ***Technology-independent***: Unlike physical models, logical models don't depend on a particular database system.
- ***Entities***: Represent real-world objects or concepts (e.g., Customer, Product, Order).
- ***Attributes***: Describe properties of entities (e.g., Customer Name, Order Date).
- ***Relationships***: Define how entities connect (e.g., Customer places Order).
- ***Normalization***: Organizes data to reduce redundancy and improve integrity.

- ***Semantic layer***: Provides a business-friendly view of data, bridging raw data and meaningful insights.

Why Logical Models Matter

Standards bodies like the Centre for European Standards (CEN) have been creating logical models for more than 30 years in an attempt to standardise health information.

Logical models potentially offer:

- ***Blueprint for databases***: Guides the transition from conceptual ideas to physical database design.
- ***Business alignment***: Ensures data structures reflect business rules and requirements.
- ***Data integrity***: Prevents anomalies (like duplication or inconsistent updates).
- ***Scalability***: Supports analytical applications by offering a consistent framework across systems.

Table 1: Comparison of data models

Model type	Focus	Example use case	Dependency on technology
Conceptual	High-level business concepts	Define what data exists (e.g., "Students enroll in Courses")	No
Logical	Detailed structure of entities, attributes, relationships	Define how data relates (e.g., "Student has StudentID, Course has CourseCode")	No
Physical	Actual implementation in a DBMS	Create tables, indexes, constraints in SQL Server, Oracle, etc.	Yes

A Simple Example

Imagine designing a IT system for a university, using an SQL database. The different models developed might include:

- Conceptual model: Students, Courses, Professors.
- Logical model:
  - Entity: Student → Attributes: StudentID (PK), Name, Email
  - Entity: Course → Attributes: CourseCode (PK), Title, Credits

    - Relationship: Student enrolls in Course (many-to-many)
- Physical model: Actual SQL tables with primary/foreign keys, indexes, and constraints.

In short, a logical model is the bridge between abstract business concepts and concrete database implementation, ensuring clarity, consistency, and scalability.

openEHR Reference Model and Logical Models

The design philosophy of openEHR is to provide a single logical model. The openEHR reference model can be frustrating for developers who want to see how to implement the specifications. What is the role of archetypes? The combination of a logical reference model (to ensure consistent representation of recurrent concepts) and archetypes (to define how to represent clinical data using the reference model) is considered one of the most elegant aspects of the design.

Let's consider how the reference model and archetypes relate to the idea of logical models.

First, the **Reference Model (RM)** defines the *core entities, attributes, and relationships* (e.g., Composition, Entry, AuditInfo).[1] This highly specified model captures data and relationships that:

- Ensure data provenance, integrity, and consistency across implementations; and
- Is technology-agnostic: it doesn't care if the backend is relational, document-based, or graph-oriented.

This is essentially the **logical model layer**, specifying *what data is and how it relates*, independent of physical storage.

Second, the **Archetypes** which are built on top of the RM and constrain and specialize the generic structures into **clinical concepts** (e.g., "Blood Pressure Measurement," "Medication Order"). Archetypes are:

- Reusable, shareable definitions that map clinical semantics onto the logical backbone; and
- Acting as **domain-specific logical models**, expressing how to use the RM entities to represent real-world healthcare data, and requiring no change to the underlying database implementation.

[1] See table on page 16

How do these two models relate to Logical Models?

- In database design, a **logical model** defines entities, attributes, and relationships without tying them to a physical schema.
- In openEHR:
    - The **Reference Model** is the *foundational logical model* — the abstract schema of health record data. This provides all the entities and attributes required to ensure the integrity of a life-long health record.
    - **Archetypes** are *applied logical models* — they specify how to represent particular clinical concepts using the RM, and can be used to capture, query and validate data. These provide the changing, context specific, evolving and often critical data specifications for personal health care – at times fractal in nature.
- Both layers remain **agnostic to technology and DBMS**, just like a logical model in traditional data modeling.

Table 2: Analogous equivalence in standard data modelling

Layer in openEHR	Equivalent in Data Modelling	Role
Reference Model	Logical Data Model	Defines abstract entities, attributes, relationships necessary for data integrity and provenance of a life-long health record
Archetypes	Domain-specific constraints expressions independent of the RM implementation	Constrains the logical RM model to provide the diverse and ever evolving clinical concepts necessary for health care
Physical Schema	Physical data model	Tables, indexes, Json docs of the reference model

There are a number of features of this two-level modelling that are a good fit for personal health data, which is almost infinitely complex and certainly fractal in nature. These include as key features:

- **Interoperability**: Logical models (like openEHR RM) ensure that data can be exchanged across systems without loss of meaning.
- **Flexibility**: Archetypes allow clinical communities to define semantics without changing the underlying logical backbone.

- **Future-proofing**: Because both RM and archetypes are DB-agnostic, they can be implemented in SQL, NoSQL, or even distributed ledger systems without redesign.

In summary, openEHR's Reference Model provides a generic, technology-independent information model for representing health record data. It defines the structural and lifecycle semantics of clinical information but does not represent specific clinical concepts.

Archetypes specialise the Reference Model by expressing domain-specific constraints that define clinical concepts such as observations, diagnoses, or procedures. Terminology bindings within archetypes further define the meaning of coded data.

In this openEHR architecture, the Reference Model enables structural interoperability between systems, while archetypes and terminology bindings enable semantic interoperability of clinical data.

A detailed description of the openEHR RM as a logical model

Introduction

The openEHR Reference Model (RM) is a foundational, formal information model that defines the structure, semantics, and behaviour of electronic health record (EHR) data. It is designed to enable semantic interoperability, robust data provenance, and clinical integrity across healthcare systems and over time. The RM achieves this by providing a stable, extensible set of information models and data types that serve as the backbone for all openEHR-based systems. Through its integration with archetypes and templates, the RM supports the consistent, high-fidelity representation of clinical data, facilitating safe data exchange, querying, and long-term record-keeping.

This section provides a comprehensive overview of the openEHR RM, detailing its key entities, attributes, and packages. It explains the role and utility of each major component—such as Composition, Entry, Section, Cluster, Element, and others—in the context of a longitudinal EHR. There is an emphasis on how the RM supports data provenance, clinical integrity, and semantic interoperability, and how it interacts with archetypes and templates to enable consistent clinical data representation across systems.

Overview of the openEHR Reference Model

Purpose and Scope

The openEHR RM defines a logical, interoperable architecture for EHRs. Its primary goals are to:

- Enable consistent representation of clinical data across systems and over time.
- Support semantic interoperability by providing a stable, well-defined set of structures and data types.
- Ensure data provenance and clinical integrity through robust versioning, audit trails, and validation mechanisms.
- Facilitate integration with archetypes and templates, allowing domain experts to define and constrain clinical content without altering the underlying technical model.

Package Structure

The openEHR RM documentation is organized into several interrelated packages, each responsible for a specific aspect of the EHR:

- **ehr**: Defines the top-level EHR structure, including access control, status, compositions, folders, and contributions.
- **composition**: Specifies the COMPOSITION class, the primary container for clinical content.
- **content**: Contains navigation (SECTION) and entry (ENTRY and its subtypes) packages.
- **data_structures**: Provides generic data structures such as ITEM_TREE, ITEM_LIST, ITEM_TABLE, and HISTORY.
- **data_types**: Defines core data types (DV_TEXT, DV_CODED_TEXT, DV_QUANTITY, etc.).
- **common**: Contains shared concepts like LOCATABLE, versioning, and participation.
- **support**: Provides supporting types such as identifiers and URIs.
- **ehr_extract**: Defines the EHR Extract model for data exchange.
- **integration**: Supports integration with legacy and external systems.
- **(demographic**: Models parties, roles, and demographic entities but is not included in this document as this package is not required for openEHR health record implementations).

Core RM Entities: Structure and Longitudinal Utility

The following Entities are the major components and provide the structure and organisation of health related data, which itself conforms to archetypes.

EHR

The EHR object is the root container for all health record content for a patient. It references all compositions, folders, status, access control, and contributions. The EHR object supports longitudinal record-keeping by maintaining a complete, versioned history of all clinical and administrative data for a subject. It is the central access point for querying, auditing, and managing the patient's health information.

EHR-Level Structures: EHR_STATUS, EHR_ACCESS, Folder, Director

- **EHR_STATUS**: Contains metadata about the EHR, such as subject, queryability, and modifiability.
- **EHR_ACCESS**: Defines access control policies and settings.
- **FOLDER**: Organizes compositions into a directory structure, supporting thematic or temporal grouping.
- **Directory**: Optional hierarchical folder structure for advanced organization.

These structures support privacy, security, and efficient retrieval of clinical content.

Composition

A COMPOSITION represents a unit of information committed to the EHR, typically corresponding to a clinical document or event (e.g., a discharge summary, progress note, or lab report). Each composition is versioned, allowing for full audit trails and rollback. Compositions are categorized as:

- **Persistent**: Represent long-term patient state (e.g., problem list, medication list).
- **Event**: Capture time-specific clinical encounters.
- **Episodic**: Support episode-based care (e.g., pregnancy episode).

Compositions encapsulate context (EVENT_CONTEXT), content (Sections and Entries), and metadata (language, territory, category, composer).

Event Context and Clinical Context

The EVENT_CONTEXT class captures the context of a clinical event, generally a composition, including:

- **start_time, end_time**: Time interval of the event.
- **location**: Physical location.

- **setting**: Clinical setting (coded).
- **health_care_facility**: Facility where the event occurred.
- **participations**: List of parties involved.

EVENT_CONTEXT enables understanding of the circumstances surrounding clinical data, supporting provenance and contextualization.

Section

A SECTION is an organizational structure within a composition, used to group entries under logical headings (e.g., "History", "Examination", "Medications"). Sections can be nested, allowing for hierarchical organization of clinical content. While sections themselves do not carry clinical meaning, they provide structure for consistent documentation and facilitate navigation and querying over time.

Entry (and subtypes)

The ENTRY class is the abstract superclass for all clinical statements in the EHR. It is further specialized into several subtypes, each representing a distinct type of clinical information:

- **OBSERVATION**: Records measurements, findings, and test results and data with specific timing. It utilises a time-series structure (see History below) to support repeated measurements or data streaming (e.g., vital signs, lab results).
- **EVALUATION**: Captures clinical assessments, diagnoses, and opinions. Represents the outcome of clinical reasoning. The timing is drawn from the date in the archetyped data (e.g. date of onset) or from the composition.
- **INSTRUCTION**: Specifies intended future actions, such as orders, prescriptions, or care plans. Supports workflow and care planning.
- **ACTION**: Documents actions that have been performed, which may be in response to instructions (e.g., medication administration, procedures).
- **ADMIN_ENTRY**: Records administrative or non-clinical data (e.g., admission details, social services information).

Each ENTRY subtype supports longitudinal tracking of clinical reasoning, interventions, and administrative events, enabling a comprehensive view of patient care over time.

Item Structures: Element, Cluster, Item_Tree, Item_List, Item_Table

Item structures provide the building blocks for organizing clinical data within entries:

- **ELEMENT**: The atomic unit of data, holding a single value (e.g., blood pressure reading). Supports null_flavour for missing data semantics.

- **CLUSTER**: Groups related elements or nested clusters, enabling modular and reusable data structures (e.g., a device cluster containing make and model elements).
- ITEM_TREE: Represents hierarchical data, allowing for complex nested structures and is used in archetypes at the root of all data collections, although any structure is allowed in the reference model. The following ITEM_LIST and ITEM_TABLE are not used widely as they limit evolution of the data.
- ITEM_LIST: Represents ordered lists of elements (e.g., problem lists).
- ITEM_TABLE: Models tabular data with named columns and rows (e.g., lab result panels).

These structures enable flexible, archetypable organisation of clinical information, supporting both simple and complex data patterns.

History and Time-Series Modelling: History, Event, Point_Event, Interval_Event

The HISTORY structure models time-series data, essential for representing longitudinal observations such as vital signs or glucose tolerance tests. It contains:

- **origin**: The starting point of the time series.
- **events**: A list of EVENT instances, each representing a data point.

EVENT is the abstract superclass for time-stamped data points, with two main subtypes:

- **POINT_EVENT**: Represents an instantaneous observation (e.g., a single blood pressure reading).
- **INTERVAL_EVENT**: Represents data aggregated over a time interval (e.g., average blood pressure over 5 minutes), with attributes for width (duration), math_function (e.g., mean, max, min, change etc.), and sample_count.

The HISTORY structure supports both periodic and aperiodic sampling, efficient representation of fine-grained device data, and inclusion of summary data. It enables detailed temporal modeling of clinical phenomena, supporting longitudinal analysis and trend detection.

Data Types in the RM

The RM defines a comprehensive set of data types, all inheriting from the abstract DATA_VALUE class. Key data types include:

- **DV_TEXT**: Free text with optional language and informal encoding.
- **DV_CODED_TEXT**: Text with associated coded terminology, supporting terminology bindings and semantic interoperability. As this is a subclass of

DV_TEXT, it enables any text to be formally coded when appropriate, linking the text and code from the terminology source.
- **DV_QUANTITY**: Numeric values with units (e.g., 120 mmHg).
- **DV_DATE_TIME**: Date and time values, supporting temporal precision.
- **DV_MULTIMEDIA**: Encapsulated multimedia content (e.g., images, audio).
- **DV_BOOLEAN, DV_COUNT, DV_PROPORTION, DV_ORDINAL**: Support for boolean, count, ratio, and ordinal data (name: value pair e.g. Apgar scoring).

These data types enable precise, semantically meaningful representation of clinical information, supporting both human readability and machine processability.

Versioning, Change-control, and Audit

The RM implements a robust versioning and change-control mechanism to ensure data provenance, integrity, and traceability:

- **VERSIONED_OBJECT<T>**: A container for all versions of a top-level object (e.g., Composition).
- **VERSION<T>**: Represents a single version, with audit details and digital signature.
- **CONTRIBUTION**: Groups one or more versions committed together, representing a logical change-set.
- **AUDIT_DETAILS**: Records metadata about each commit (who, when, what, why).
- **ATTESTATION**: Supports legal attestation and digital signing of content.

This mechanism ensures that all changes are indelible (no physical deletion), fully auditable, and reconstructible, supporting medico-legal requirements and forensic examination of past states.

Feeder Audit and Provenance

The FEEDER_AUDIT class captures metadata about the origin of data imported from external systems. It includes:

- **originating_system_audit**: Audit details from the source system.
- **feeder_system_audit**: Audit details from any intermediate system.
- **original_content**: Optional inclusion of the original content.

FEEDER_AUDIT enables traceability, trust, and integration of data from non-openEHR sources, supporting data provenance and duplicate detection.

Locatable, Identifiers, and Paths

The LOCATABLE class is the base for all archetypable objects in the RM. It provides:

- **uid**: Unique identifier for the object.

- **archetype_node_id**: Identifier linking the object to its archetype definition.
- **name**: Human-readable name.
- **archetype_details**: Metadata about the archetype and template used.
- **feeder_audit**: Provenance information for imported data.

LOCATABLE supports path-based querying using Xpath-compatible syntax such as openEHR's AQL, enabling precise referencing and navigation of any node within the EHR. This is essential for semantic querying, linking data, data extraction, and interoperability.

Interaction with Archetypes and Templates

openEHR employs a two-level modeling approach involving use of two models:

1. **Reference Model** (RM): Provides a stable, technical foundation for data structures and types.
2. **Archetype Model** (AM): Allows domain experts to define reusable, constraint-based models (archetypes) for clinical concepts (e.g., blood pressure, lab result).

The actual clinical data is specified as archetypes, which are statements of how to use the reference model to express specific data instances. The archetypes, often expressed as ADL, conform to the archetype model.

Templates combine multiple archetypes to define forms, documents, or messages for specific use cases (e.g., discharge summary, lab report).

Archetype and Template Binding

- Archetypes constrain RM structures, specifying allowed data elements, value sets, and terminology bindings.
- Templates further constrain archetypes for particular contexts, setting occurrence constraints, default values, and hiding irrelevant elements.

At runtime, archetype_node_id and archetype_details in LOCATABLE instances link data nodes to their generating archetypes and templates. This enables:

- **Validation**: Ensuring data conforms to clinical and technical constraints.
- **Semantic querying**: Using archetype paths and codes for precise data retrieval.
- **Consistent modification**: Ensuring updates respect original constraints.

Archetypes and templates are stored separately from data, supporting reuse and independent evolution of clinical models and technical infrastructure.

Common features

- Archetype-root points: ENTRY subtypes (OBSERVATION, EVALUATION, etc.) are typical root points for archetypes.
- Templates: Used to define structures for common documents (e.g., discharge summaries, lab results, care plans).
    - Example structures: OGTT (oral glucose tolerance test), asthma management plan, multi-drug therapy.

Templates can be viewed and edited using tools such as the openEHR Template Designer, and numerous real-world examples are available in the openEHR Clinical Knowledge Manager (CKM).

Semantic Interoperability

Terminology Binding

The RM and archetypes support semantic interoperability through:

- DV_CODED_TEXT and CODE_PHRASE: Allowing data values to be coded using standard terminologies (e.g., SNOMED CT, LOINC, ICD).
- Archetype term definitions: Each archetype may contain an internal terminology for node definitions and value sets which can be directly linked to multiple terminologies through:
    - External bindings: Archetypes can bind internal codes to external terminologies, enabling mapping and querying across systems.

This approach ensures that clinical data retains its meaning across different systems, languages, and contexts, supporting consistent querying, data sharing, and decision support.

Path-Based Querying

Every data point can be accessed directly using path based queries.

- LOCATABLE paths: providing Xpath-compatible syntax enables precise referencing of any node.
- Archetype paths: Combine RM attribute names and archetype codes for semantic navigation.
- AQL (Archetype Query Language): Supports expressive, semantically rich queries over archetyped data.

Semantic interoperability is further enhanced by the use of archetype constraints, which enforce consistent use of codes, value sets, and data structures across systems and allow tight data validation in forms or at the time of committing the data.

Clinical Integrity and Data Quality

The quality of data is increasingly important with more automatic processing of health data through AI and other mechanisms. It is worth summarising what openEHR offers in this domain.

Data Validation and Constraints

- **Archetype constraints**: Define permissible values, structures, and terminologies for each data element.
- **Templates**: Further constrain archetypes for specific contexts, ensuring only relevant data is captured.
- **Null_flavour**: Indicates reasons for missing data (e.g., unknown, not applicable, masked), supporting accurate interpretation and data quality assessment.
- **Data validation**: Implementations in all technologies are able to validate data against archetype and template constraints at data entry and modification time.

Attestation and Audit

Clinical accountability and medico-legal provenance is important for longitudinal health records. openEHR provides many features to support this, as well as versioning of all contributions to the health record.

- **ATTESTATION**: Supports digital signing and legal attestation of content, ensuring accountability and trust.
- **AUDIT_DETAILS**: Records who made changes, when, and why, supporting traceability and medico-legal compliance.

These mechanisms ensure that clinical data is accurate, complete, and trustworthy, supporting safe patient care and regulatory compliance.

Interoperability and Exchange

With the future of health care in mind, openEHR provides a means of extracting all or part of a health record and providing it to a new health record instance, while maintaining the provenance of the data, as well as the state of each health record instance at any point in time. This is considered to be critical for health care professionals.

EHR Extract Model

The EHR Extract model enables standardized exchange of health record content between systems, supporting:

- Full, simplified, and synchronization extracts: For ad hoc queries, batch transfers, migrations, and system synchronization.
- Compatibility with external standards: Such as ISO 13606, HL7 CDA, and FHIR.

- Preservation of versioning and audit trails: Ensuring data integrity and provenance during exchange.

The extract model uses X_VERSIONED_OBJECT and related classes to serialize versioned content for lossless transmission, supporting a wide range of interoperability scenarios.

Integration with Other Standards

- GENERIC_ENTRY: Supports integration with legacy and non-openEHR systems.
- Terminology bindings: Enable mapping to external vocabularies and value sets.
- Canonical and simplified serializations: Support XML, JSON, and other formats for data exchange and integration.

These features ensure that openEHR-based systems can interoperate with a diverse ecosystem of health IT solutions, supporting data liquidity and patient mobility.

Knowledge Engineering and Ontologies

There are many specialists who understand a great deal about the structure and organisation of knowledge, and the meaning of the words we use and the data we collect. Thomas Beale brought an ontological perspective to the design of the openEHR reference model – which represented the consistent features of health data, provenance, timing, language, and many other aspects.

Archetypes., although designed to work with any (and multiple) terminologies, were firmly based on clinical requirements and needed no semantic basis, at least at the outset, as clinicians already shared this very large data space that was common; straddling language, culture, time and space. Organising the archetypes to maximise interoperability and reuse was the challenge and it is a challenge that persists. Some data collections, such as laboratory results, were so well established that following the widespread pattern and using LOINC terms aided implementation.

The great benefits that openEHR offers are consistency of representation provided by the reference model and the community of practice to create the archetypes which together provide the context for the words and phrases in terminologies, and limit the set to a small number in many situations. All this makes meaning manageable; and automatic processing more resilient.

Conclusion

The openEHR Reference Model provides a robust, semantically rich, and longitudinally consistent framework for representing electronic health records. Its modular architecture, comprehensive set of entities and data types, and integration with archetypes and templates enable high-quality, interoperable clinical data across systems and over time. Through its support for versioning, audit, provenance, and semantic interoperability, the RM ensures clinical integrity, data quality, and safe data exchange in diverse healthcare environment.

The RM is explicitly designed to support longitudinal health records, enabling comprehensive tracking of patient data over time:

- **Persistent Compositions**: Maintain single sources of truth for long-term patient state (e.g., problem list, medication list).
- **Event Compositions**: Capture discrete clinical encounters or events.
- **Episodic Compositions**: Support episode-based care (e.g., pregnancy, hospital admission).
- **HISTORY and EVENT structures**: Model time-series data, supporting trend analysis and temporal reasoning.
- **Versioning and audit**: Enable reconstruction of past informational states, supporting medico-legal requirements and forensic analysis.
- **Folders and directories**: Organize data thematically or temporally for efficient retrieval and navigation.

This longitudinal utility is critical for supporting chronic disease management, population health analytics, and continuity of care across organizational boundaries.

By separating the technical foundation (RM) from clinical content modeling (archetypes and templates), openEHR empowers domain experts to define and evolve clinical models independently of technical infrastructure, fostering innovation, adaptability, and future-proof health information systems.

Appendix A: Major Reference Model classes, purpose and use

Entity	Main Attributes	Purpose	Longitudinal Utility
The features of the entire longitudinal EHR and how it is organised			
EHR	system_id, ehr_id, contributions, ehr_status, ehr_access, compositions, folders	Top-level container for a patient's health record	Provides the longitudinal anchor for all clinical data; ensures continuity across systems and time
EHR status	is_modifiable, is_queryable, subject, other_details	Defines modifiability and queryability of the EHR	Controls lifecycle and accessibility of the record over time
EHR Access	settings, policies	Defines access control policies for the EHR	Supports longitudinal governance and security of patient data
Folder	name, items	Logical grouping of Compositions	Supports organization of longitudinal record across episodes or domains
The components of the EHR			
Composition	language, territory, category, context, composer, content	Top-level container for clinical content; represents a single clinical encounter or document	Anchors entries to time, place, and author; provides audit trail and context for longitudinal record
Section	name, further sections or entries	Organizational grouping within a Composition	Supports hierarchical structuring of content across encounters; aids navigation and retrieval and human readability
Entry	language, encoding, subject, provider, other_participations, data (ItemStructure)	Abstract superclass for all clinical and administrative statements	Provides consistent framework for observations, evaluations, instructions, and actions across time
- Observation	data (history), state, protocol	Captures measured or observed data	Supports time-series data (e.g., vitals, labs) enabling longitudinal tracking and trend analysis

Entity	Main Attributes	Purpose	Longitudinal Utility
--History	origin (time of first event), events or serial measurements of fixed interval	Time-structured series of data points	Enables single point in time or longitudinal representation of repeated measures (e.g., lab series). Longitudinal measures include maximum value, minimum value, average, range, or sum.
---Event (abstract)	time, data and state information	Single point in a History series. Events are either point in time or an interval of time	Captures discrete longitudinal data points within a time series
---Point event	time, data and state information	Represents a single point in time	Captures instantaneous observations
---Interval event	time, width, math_function, data	Represents an interval of time	Captures aggregated or interval-based observations
-Evaluation	data, protocol	Represents clinical interpretation, diagnosis, or assessment	Preserves evolving clinical judgments and assessments over time
-Instruction	narrative, activities, workflow plan, expiry_time, wf_definition	Prescribes actions to be carried out	Captures intended care plans and orders; supports continuity of care across encounters
-Action	description, time, instruction details, ism_transition (state transition such as from prescribed to dispensed for a medication)	Documents actions performed in response to instructions	Provides record of what was actually done; supports audit and outcome tracking
-Admin Entry	data (item_structure)	Records administrative data	Supports non-clinical longitudinal data
Cluster	items, archetype_node_id	Structure for nested or complex data - may be archetyped for reuse in other Entries	Supports detailed modeling of panels, imaging, or structured measurements across time

Entity	Main Attributes	Purpose	Longitudinal Utility
Element	name, value	Leaf node holding a single data value	Basic unit of clinical data; enables fine-grained longitudinal comparison (e.g., systolic BP values)
The features which allow the health record to be coherent, safe and useful			
AuditInfo	committer, time_committed, change_type	Metadata for provenance and versioning	Ensures traceability, accountability, and integrity across the evolving longitudinal record
Party	identifiers, name, roles	Represents people or organizations involved	Supports continuity by linking data to patients, clinicians, institutions across time
Locatable	archetype_node_id, name	Base class for all RM structures	Provides consistent identification and semantic anchoring across longitudinal data and enables a single data point to be queried and linked
Version	uid, lifecycle_state, contribution	Represents a version of an RM object	Supports version control and audit trail across longitudinal record, independent of the system it may be stored within
Contribution	uid, versions, audit	Groups versions committed together	Ensures atomic commits and provenance across longitudinal updates
Archetyped	archetype_id, template_id	Links RM structures to archetypes/templates	Provides semantic binding for longitudinal consistency
Link	meaning, type, target	Represents relationships between RM objects	Supports semantic connections across longitudinal record
Attestation	attester, proof, reason, items	Legal attestation of content	Supports medico-legal integrity